



Co-funded by
the European Union

Гриценко К. Г., Яценко В. В.

СИСТЕМНИЙ АНАЛІЗ ТА ПРОГРАМНА ІНЖЕНЕРІЯ: ЄВРОПЕЙСЬКА ПЕРСПЕКТИВА

Навчальний посібник



Hrytsenko K. H., Yatsenko V. V.

SYSTEM ANALYSIS AND SOFTWARE ENGINEERING: A EUROPEAN PERSPECTIVE

Textbook

Міністерство освіти і науки України
Сумський державний університет

Гриценко К. Г., Яценко В. В.

СИСТЕМНИЙ АНАЛІЗ ТА ПРОГРАМНА ІНЖЕНЕРІЯ: ЄВРОПЕЙСЬКА ПЕРСПЕКТИВА

Навчальний посібник

Рекомендовано вченою радою Сумського державного університету

Суми
Сумський державний університет
2026

УДК 330.46:004.4(4)

Авторський колектив:

К. Г. Гриценко, кандидат технічних наук (СумДУ);

В. В. Яценко, кандидат технічних наук (СумДУ)

Рецензенти:

О. Ю. Чигрин – докторка економічних наук, професорка, доцентка кафедри маркетингу Сумського державного університету (м. Суми);

С. В. Агаджанова – кандидатка технічних наук, доцентка, доцентка кафедри кібернетики та інформатики Сумського національного аграрного університету (м. Суми)

Рекомендовано до видання

вченою радою Сумського державного університету

як навчальний посібник

(протокол № 2 від 10 вересня 2026 року)

Гриценко К. Г.

Системний аналіз та програмна інженерія : європейська перспектива : навчальний посібник / К. Г. Гриценко, В. В. Яценко. – Суми : Сумський державний університет, 2026. – 183 с.

Навчальний посібник призначений для оволодіння теоретичними та практичними навичками, які необхідні майбутнім фахівцям для вивчення дисципліни «Системний аналіз та економічна кібернетика» (або іншої за назвою та подібної за змістом). Обсяг та перелік тем запропонованого навчального посібника забезпечує потреби спеціальності С1 «Економіка та міжнародні економічні відносини (спеціалізація Економіка)» (освітня програма «Економічна кібернетика»), а також спеціальності F2 «Інженерія програмного забезпечення». Для кожного розділу наведено перелік контрольних запитань, що сприяє кращому засвоєнню навчального матеріалу з однойменної дисципліни. Окремо наведено перелік тестових завдань та інструкції щодо виконання практичних робіт.

Навчальний посібник підготовлено в рамках проекту Еразмус+ напрямку Жан Моне (Модуль) 101127435 – EU_DIGIT «Розвиток цифрових компетентностей фахівців цифрової економіки та бізнес-лідерів: кращі європейські практики для України». Фінансується Європейським Союзом. Однак висловлені погляди та думки належать лише авторам і не обов'язково відображають погляди Європейського Союзу чи Європейського виконавчого агентства з освіти та культури (EACEA). Ні Європейський Союз, ні EACEA не можуть нести відповідальність за них.

Для здобувачів спеціальностей С1 «Економіка та міжнародні економічні відносини (спеціалізація Економіка)», F2 «Інженерія програмного забезпечення» освітнього ступеня «бакалавр» усіх форм здобуття вищої освіти.

УДК 330.46:004.4(4)

© Сумський державний університет, 2026

ЗМІСТ

ПЕРЕДМОВА	7
РОЗДІЛ 1 СИСТЕМНИЙ АНАЛІЗ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	9
1.1 Поняття системного аналізу	9
1.2 Розуміння бізнес-проблеми	11
1.3 Визначення системних вимог, обмежень та інформаційних потреб	13
1.4 Створення звіту про системний аналіз	14
1.5 Життєвий цикл розроблення програмного забезпечення	18
1.6 Гнучке розроблення програмного забезпечення ..	29
1.7 Європейський цифровий регуляторний простір: виклики для системного аналізу та програмної інженерії	38
Контрольні запитання	42
РОЗДІЛ 2 ІДЕНТИФІКАЦІЯ ВИМОГ КОРИСТУВАЧІВ ТА ОСНОВИ ПРОЄКТУВАННЯ СИСТЕМ	43
2.1 Визначення бізнес-проблем.....	43
2.2 Методи ідентифікації вимог користувачів.....	44
2.3 Загальні принципи та дорожня карта проєктування програмних систем.....	49
2.4 Дизайн, орієнтований на користувача	53
2.5 Нормативне регулювання подолання цифрової нерівності в ЄС.....	57
Контрольні запитання	62
РОЗДІЛ 3 ОСНОВИ ПРОЄКТУВАННЯ МОБІЛЬНИХ ПРОГРАМНИХ СИСТЕМ	64
3.1 Мобільна економіка та її вплив на бізнес	64
3.2 Проєктування мобільних програмних систем	70
3.3 Аналіз ринку та валідація бізнес-ідеї	75
3.4 Основи UI/UX дизайну для бізнес-застосунків	80
3.5 Процес створення прототипу: від схеми до інтерактивного макету	84
3.6 Основи нативного розроблення для Android	89

3.7 Основи крос-платформного розроблення	98
3.8 Взаємодія з сервером.....	101
Контрольні запитання	105
РОЗДІЛ 4 ПІДВИЩЕННЯ	
КОНКУРЕНТОСПРОМОЖНОСТІ МОБІЛЬНИХ	
ПРОГРАМНИХ СИСТЕМ	106
4.1 Моделі монетизації мобільних програмних систем.....	106
4.2 Ключові показники ефективності та мобільна аналітика	110
4.3 Стратегія виходу на ринок та оптимізація сторінки мобільного застосунку в магазинах застосунків.....	114
4.4 Підтримка програмного продукту після запуску	118
Контрольні запитання	122
РОЗДІЛ 5 АНАЛІТИКА ДАНИХ	123
5.1 Процес аналізу даних	123
5.2 Правила конфіденційності даних у ЄС	129
5.3 Основи бізнес-аналітики.....	139
Контрольні запитання	141
ТЕСТОВІ ЗАВДАННЯ	143
ПРАКТИЧНА РОБОТА 1	157
ПРАКТИЧНА РОБОТА 2.....	162
ПРАКТИЧНА РОБОТА 3.....	163
ПРАКТИЧНА РОБОТА 4.....	164
ПРАКТИЧНА РОБОТА 5.....	165
ПРАКТИЧНА РОБОТА 6.....	172
ПРАКТИЧНА РОБОТА 7.....	173
ПРАКТИЧНА РОБОТА 8.....	174
ПРАКТИЧНА РОБОТА 9.....	175
ПРАКТИЧНА РОБОТА 10.....	176
ПРАКТИЧНА РОБОТА 11.....	177
ПРАКТИЧНА РОБОТА 12.....	178
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ	179

ПЕРЕДМОВА

Стрімкий розвиток цифрових технологій, поширення великих даних і штучного інтелекту суттєво трансформують сучасне суспільство й економіку. У цих умовах особливого значення набуває підготовка фахівців, здатних здійснювати системний аналіз і проєктування сучасних інформаційних систем, розробляти якісне програмне забезпечення відповідно до міжнародних стандартів і найкращих європейських практик.

Європейський Союз послідовно формує єдиний цифровий простір, у якому ключова роль відводиться розвитку цифрових компетентностей, стандартизації програмної інженерії, забезпеченню кібербезпеки, захисту персональних даних, впровадженню технологій штучного інтелекту та розвитку інноваційної екосистеми. У зв'язку з євроінтеграційним курсом України актуальним є наближення змісту освітніх програм до європейських підходів, рекомендацій та професійних стандартів.

Навчальний посібник «Системний аналіз та програмна інженерія: європейська перспектива» підготовлено з метою комплексного висвітлення сучасних теоретичних засад і практичних аспектів системного аналізу та програмної інженерії в контексті європейських підходів до створення, супроводу й розвитку програмного забезпечення. Особливу увагу приділено інтеграції принципів системного мислення із сучасними методологіями розроблення програмного забезпечення, проєктування мобільних програмних систем, управління вимогами та життєвим циклом програмного забезпечення.

У посібнику систематизовано сучасні концепції системного аналізу, інструменти програмної інженерії, проєктування мобільних програмних систем, аналітики даних, європейські стандарти, моделі життєвого циклу

програмного забезпечення, принципи гнучкої розробки та управління проєктами.

Видання побудоване відповідно до компетентнісного підходу та принципів студентоцентрованого навчання. Теоретичний матеріал доповнено прикладами, контрольними запитаннями, тестовими завданнями та інструкціями для виконання практичних робіт, що сприяє формуванню професійних компетентностей, розвитку аналітичного мислення та набуттю практичних навичок розроблення сучасного програмного забезпечення.

Матеріал посібника відповідає сучасним тенденціям розвитку інформаційних технологій, рекомендаціям Європейського простору вищої освіти та може бути використаний під час вивчення дисциплін, пов'язаних із системним аналізом, програмною інженерією, інформаційними системами, цифровою економікою та управлінням проєктами інформатизації.

Навчальний посібник призначений для здобувачів закладів вищої освіти спеціальності С1 «Економіка та міжнародні економічні відносини (спеціалізація Економіка)» (освітня програма «Економічна кібернетика»), а також спеціальності F2 «Інженерія програмного забезпечення». Видання стане у пригоді викладачам, аспірантам, науковцям, ІТ-фахівцям, системним аналітикам, розробникам програмного забезпечення та всім, хто цікавиться сучасними підходами до аналізу, проєктування та створення інформаційних систем в умовах цифрової трансформації та європейської інтеграції України.

Автори сподіваються, що запропонований матеріал сприятиме формуванню сучасного професійного світогляду, розвитку системного мислення, опануванню інструментів програмної інженерії та підготовці конкурентоспроможних фахівців, здатних ефективно працювати в європейському цифровому середовищі.

РОЗДІЛ 1

СИСТЕМНИЙ АНАЛІЗ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття системного аналізу

Системним аналізом називається поетапний процес оцінювання інформаційних систем (ІС) шляхом розуміння потреб бізнесу. ІС поєднують людей, процеси, дані, апаратне та програмне забезпечення (ПЗ), необхідні для збору, зберігання, оброблення, аналізу та розповсюдження даних. Прикладами ІС є системи оброблення замовлень, управління запасами, управління людськими ресурсами та управління продажами. Наприклад, робочий процес (workflow) електронного рецепта включає такі ключові етапи: створення лікарем, отримання повідомлення пацієнтом та видачу ліків в аптеці. Така система діє для всіх рецептурних препаратів та державної програми «Доступні ліки» в Україні.

На рис. 1.1 наведено робочий процес електронного рецепта, що пов'язує лікаря (Provider), який виписує електронний рецепт (E-prescription), з фармацевтом (Pharmacist), який видає ліки пацієнту (Patient).

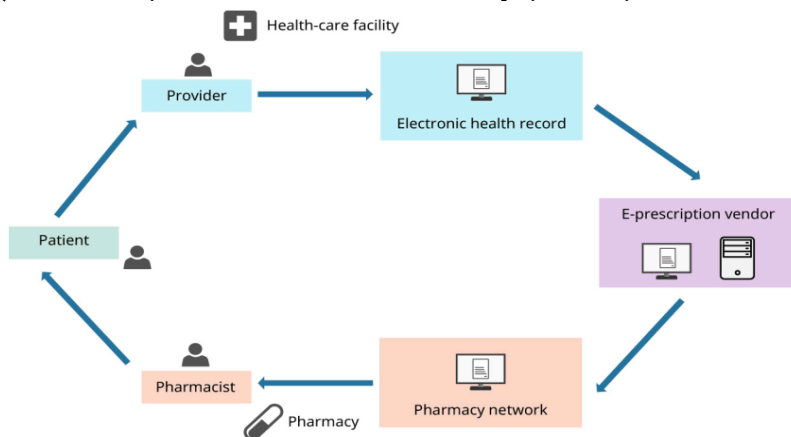


Рисунок 1.1 – Робочий процес електронного рецепта

Робочий процес електронного рецепту включає кілька систематичних кроків. Лікар, що ставить діагноз пацієнту та виписує рецепт на ліки, вносить медичний запис (Electronic health record) в електронну систему охорони здоров'я (ЕСОЗ) (E-prescription vendor). Остання надає інформацію про електронний рецепт постачальнику ліків (аптечна мережа) (Pharmacy network), що в свою чергу повідомляє про замовлення персонал аптеки, який, видає ліки пацієнту.

У даному сценарії відбувається проміжна взаємодія лікаря та постачальника ліків із ЕСОЗ, щоб персонал аптеки міг задовольнити запит лікаря на ліки для пацієнта, та працює багато систем, які з'єднують користувачів (медичний та фармацевтичний персонал), зокрема їх апаратне та програмне забезпечення.

Компанії використовують системний аналіз для оцінювання своїх робочих процесів, визначення шляхів покращень та створення ефективних ІС. Компанії встановлюють пріоритети проєктів інформатизації, використовуючи різні критерії, методи, інструменти та інформацію від підрозділів, щоб узгодити проєкти інформатизації з бізнес-цілями компанії.

Системний аналітик – це фахівець, основною функцією якого є застосування системного аналізу для підтримки ІС компанії та вирішення проблем, пов'язаних із використанням ІС. Системних аналітиків часто розглядають як посередників між ІТ-підрозділами та бізнес-підрозділами компанії, оскільки вони допомагають зрозуміти, як саме ІС використовується для підтримки бізнес-функцій компанії, ідентифікувати бізнес-проблеми та сфери, що потребують удосконалення. Хоча системні аналітики найчастіше працюють у ІТ-підрозділах, вони також можуть працювати в різних сферах бізнесу, таких як операційні підрозділи, фінанси, маркетинг, продажі або управління персоналом.

Системний аналіз включає такі основні компоненти (рис. 1.2): розуміння поточної бізнес-проблеми; визначення системних вимог, обмежень та інформаційних потреб; створення звіту про системний аналіз.

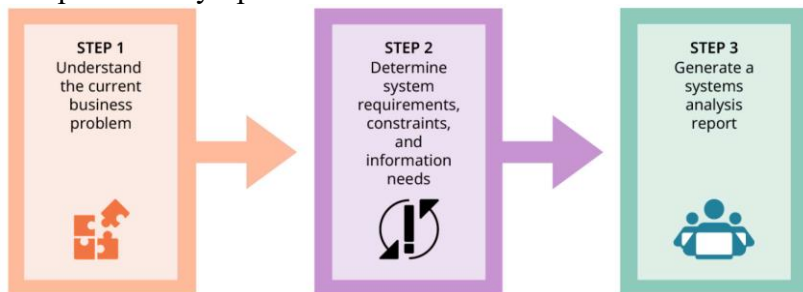


Рисунок 1.2 – Основні компоненти системного аналізу

1.2 Розуміння бізнес-проблеми

Під час ідентифікації бізнес-проблеми збирається та аналізується детальна інформація про ІС компанії за допомогою різноманітних інструментів і методів. Щоб зрозуміти бізнес-проблему, системний аналітик спочатку визначає зацікавлені сторони (стейкхолдерів), які мають особисту стурбованість щодо бізнес-рішення. Стейкхолдери можуть бути внутрішніми або зовнішніми по відношенню до компанії та можуть включати громаду, державні установи, співробітників, клієнтів, інвесторів, постачальників або профспілки та асоціації.

Після визначення стейкхолдерів системний аналітик починає збирати інформацію про ІС, щоб краще зрозуміти її функціональність та будь-які проблеми, з якими вона наразі стикається. Ця інформація може включати: призначення ІС; вхідні та вихідні дані; функціональні можливості; кількість користувачів і рівні доступу; простоту використання; доступність та проблеми, пов'язані з використанням. Загальні методи, що використовуються для збору цієї інформації або даних, включають такі.

Огляд системної документації. Системна документація використовується для розуміння функціонування ІС, а також служить довідковою інформацією. Вона може включати посібники користувача, програмний код, описи архітектури та дизайну ІС, а також відповіді на часто задавані питання (FAQ). Вивчаючи системну документацію, системний аналітик може виявити зміни, внесені до ІС з моменту її першого використання в компанії. Він може виявити, що існуюча системна документація може бути неактуальною, неточною та неповною. У такому випадку системний аналітик працює над оновленням документації для забезпечення її актуальності, точності та повноти. Збір інформації також включає вивчення документів, пов'язаних з політикою компанії, та нормативних правил, що формують потреби ІС.

Опитування. Це поширена форма збору інформації. Опитування зазвичай займають мінімум часу та можуть залучати різноманітних стейкхолдерів. Опитування дозволяє великій кількості користувачів дати відповіді на питання щодо функціонування ІС та надати системному аналітику уявлення про специфіку її використання.

Інтерв'ю. Системний аналітик проводить інтерв'ю з різними користувачами ІС та представниками різних ролей в ІС (адміністратор, звичайний користувач, привілейований користувач, модератор тощо). Щоб підготуватися до цих інтерв'ю, системний аналітик повинен визначити цілі інтерв'ю, сформулювати питання інтерв'ю, задокументувати результати та оцінити відповіді на питання інтерв'ю. Інтерв'ю можуть бути структурованими (тобто використовувати заздалегідь визначені питання інтерв'ю) або неструктурованими (тобто не передбачати встановленого формату або заздалегідь визначених питань).

Спостереження. Спостереження за ІС зазвичай включають відстеження того, як користувачі виконують

свої функції за допомогою ІС та як система реагує на це. Спостереження можуть бути пасивними (наприклад, спостереження за тим, як користувачі виконують свої дії в ІС) або активними (наприклад, взаємодія з користувачами та запитання про їхню діяльність). Для вимірювання спостережень використовуються метрики та вебаналітика. Вхідні та вихідні дані ІС спостерігаються, а потім документуються для подальшого аналізу.

Аналіз даних. Після збору системної документації вона очищується, сортується та стискається, щоб можна було на її основі зробити висновки, які допоможуть у вирішенні бізнес-проблем. Аналіз даних – це ітеративний процес із використанням якісних і кількісних (статистичних) методів аналізу, під час якого системному аналітику може знадобитися вдосконалити процес збору даних, щоб отримати глибше розуміння представлених даних.

1.3 Визначення системних вимог, обмежень та інформаційних потреб

Системні вимоги – це можливості, які стейкхолдери вимагають від створення ІС, що відповідає їхнім бізнес-потреbam. Системні вимоги можна класифікувати як функціональні, нефункціональні, бізнес-, технічні, нормативні та пов'язані з продуктивністю. Для збору цих вимог системний аналітик співпрацює з стейкхолдерами. Далі системні вимоги розподіляються за категоріями та отримують пріоритети на основі критеріїв, важливих для бізнесу.

Доступ до системних даних про функціональність ІС є ключовим компонентом вимог користувачів. Один із прикладів того, як системні дані слугують ключовим компонентом вимог користувачів, можна знайти в історії дзвінків до служби підтримки клієнтів у центрах оброблення даних. Характеристики дзвінка (тобто

тривалість, дата та час, причина дзвінка, статус вирішення проблеми) можна використовувати для досягнення різних бізнес-цілей, включаючи навчання користувачів, покращення послуг та мотивацію співробітників. Дата та час, пов'язані з дзвінком, можуть вказувати на низку проблем, таких як пропускна здатність або проблеми з обробленням інформації, що може призвести до змін у навчанні кінцевих користувачів, необхідних покращень ІС або виявлення проблеми, специфічної для певної групи користувачів. Системні вимоги потім переглядаються, оцінюються та остаточно узгоджуються з урахуванням інтересів стейкхолдерів.

Сьогодні алгоритми та штучний інтелект (ШІ) керують інструментами, що використовуються в системному аналізі та програмній інженерії. Це може призвести до етичних проблем, підриваючи довіру користувачів. Наприклад, системи відстеження кандидатів на роботу часто надають перевагу словам у резюме, що більше асоціюються з чоловіками. Процес ціннісної інженерії (Values-Based Engineering, VbE) допомагає вирішити ці проблеми шляхом встановлення пріоритетів та інтеграції цінностей у процес розробки вимог. VbE застосовується як до функціональних, так і до нефункціональних вимог.

1.4 Створення звіту про системний аналіз

Системний аналітик протягом усього робочого процесу надає зворотний зв'язок стейкхолдерам, а фінальний крок включає створення звіту про системний аналіз. Цей звіт містить рекомендовані рішення певної проблеми ІС (наприклад, зміни у бізнес-процесах компанії та вдосконалення ІС). Звіт про системний аналіз часто має форму техніко-економічного обґрунтування бізнес-ідеї та містить такі основні розділи.

Визначення проблеми. Містить обговорення виявлених соціальних, економічних та екологічних проблем. Для обґрунтування доцільності вирішення виявлених проблем використовуються галузеві дані, висновки з маркетингових досліджень та результати емпіричних досліджень.

Рішення (бізнес-ідея/продукт/послуга). Містить обговорення бізнес-можливостей, які можна створити для сталого вирішення соціальних, економічних та екологічних проблем, виявлених у громаді та суспільстві. Запропоновані бізнес-рішення потрібно чітко викласти.

Ринкова обґрунтованість. Містить аналіз динаміки ринку, галузевих тенденцій, сегментів ринку, конкуренції, стратегії сегментів ринку, плану дистрибуції, стратегії маркетингового просування/ціноутворення у зв'язку з запропонованою бізнес-ідеєю. У цьому розділі необхідно обґрунтувати запропоновану бізнес-ідею, надавши відповіді на низку ключових питань: чи підтверджують результати аналізу галузі, ринкових тенденцій і споживчих уподобань доцільність її реалізації; чи було здійснено аналіз конкурентного середовища; яким є план розвитку бізнес-ідеї; який цільовий сегмент ринку визначено для її впровадження; а також які маркетингові, цінові та дистрибуційні стратегії передбачено для забезпечення її успішної реалізації.

Технічна обґрунтованість. Містить фактичні докази того, що запропонована бізнес-ідея технічно можлива, і ви маєте технічні знання, навички та здібності, необхідні для створення цінності та надання продуктів і послуг. У цьому розділі необхідно охарактеризувати операційну складову бізнес-ідеї, надавши відповіді на такі ключові питання: чи володіє підприємство висококваліфікованими фахівцями, здатними забезпечити розроблення та надання запропонованих продуктів і послуг; чи має воно відповідну

матеріально-технічну базу, включаючи офісні та виробничі приміщення, необхідні для створення, надання й отримання цінності від цих продуктів і послуг; а також яким є операційний план, що визначає процеси їх розроблення, впровадження та надання споживачам.

Обґрунтованість бізнес-моделі. Містить опис того, як ваш бізнес прагне створювати цінність, продаючи продукти, послуги та рішення. У цьому розділі необхідно представити характеристику бізнес-моделі, надавши відповіді на такі ключові питання: якою є розроблена діаграма бізнес-моделі (Business Model Canvas, BMC); чи підтверджує вона спроможність підприємства створювати цінність для цільового ринку у вигляді продуктів, послуг або комплексних рішень; чи забезпечує бізнес-модель ефективне доведення цієї цінності до споживачів; а також чи передбачає вона наявність декількох джерел формування доходів, що сприяють фінансовій стійкості та розвитку бізнесу.

Обґрунтованість структури управління. У цьому розділі необхідно охарактеризувати організаційне забезпечення реалізації бізнес-ідеї, надавши відповіді на такі ключові питання: якою є організаційна структура компанії, зокрема система управління та розподілу відповідальності; які кадрові потреби існують у короткостроковій і довгостроковій перспективі; якими компетентностями повинні володіти працівники для ефективного виконання своїх функцій; яким є план управління об'єктами інтелектуальної власності, включаючи торговельну марку, комерційне найменування, патенти, авторські права та комерційну таємницю; яким чином організовано управління мінімально життєздатним продуктом (Minimum Viable Product, MVP); а також чи розроблено діаграму Ганта для планування бізнес-операцій із визначенням основних завдань і термінів їх виконання.

Фінансова обґрунтованість. Забезпечує ясність щодо витрат та вигод від інвестування в вашу бізнес-ідею, а також її довгострокову стійкість з точки зору прибутковості. У цьому розділі необхідно обґрунтувати економічну доцільність реалізації бізнес-ідеї, надавши відповіді на такі ключові питання: чи існує на цільовому ринку незадоволений попит або дисбаланс між попитом і пропозицією щодо запропонованого продукту чи послуги; якими є основні фінансово-економічні показники проєкту, зокрема точка беззбитковості, термін окупності інвестицій та рівень рентабельності інвестицій.

Регуляторна обґрунтованість. Містить опис нормативних вимог, які має виконати ваш бізнес, а також опис наявного сприятливого середовища (PESTEL-аналіз) для процвітання запропонованої бізнес-ідеї. У цьому розділі необхідно проаналізувати зовнішнє середовище функціонування бізнес-ідеї, надавши відповіді на такі ключові питання: які нормативно-правові акти регулюють відповідну сферу діяльності; наскільки економічне та політичне середовище країни сприяє реалізації запропонованої бізнес-ідеї; яким є вплив соціальних та екологічних чинників на її впровадження; чи забезпечує технологічне середовище необхідні умови для її успішної реалізації; а також чи відповідає чинне правове середовище вимогам щодо створення, розвитку та функціонування запропонованого бізнесу.

Обґрунтованість стратегії виходу. Забезпечує ясність щодо стратегічного плану компанії щодо стратегії масштабування, ліквідації, запобігання операційним збоям, зниження ризиків та отримання прибутку у разі успіху бізнесу. У цьому розділі необхідно визначити підходи до завершального етапу реалізації бізнес-ідеї та управління можливими ризиками, надавши відповіді на такі ключові питання: якою є стратегія виходу з бізнесу, зокрема через

реалізацію внутрішніх або зовнішніх стратегій зростання; а також якою є політика ліквідації підприємства, спрямована на мінімізацію наслідків операційних невдач і ризику банкрутства.

1.5 Життєвий цикл розроблення програмного забезпечення

Проектування програмної системи – це організаційний підхід, метою якого є покращення існуючої програмної системи або розроблення нової. Цей процес включає визначення даних, інтерфейсів та загальної архітектури програмної системи на основі вимог користувача та визначення того, як елементи програмної системи взаємодіють один з одним для створення надійної, стійкої та добре документованої програмної системи.

Перш ніж розробляти проєкт, компанії потрібно вирішити, чи розробляти нову програмну систему, чи вдосконалювати існуючу програмну систему шляхом аутсорсингу, чи використовувати власні ресурси. Внутрішнє розроблення програмної системи зазвичай включає проектування даних, інтерфейсів та процесів.

Життєвий цикл (ЖЦ) розроблення програмного забезпечення (ПЗ) являє собою сукупність послідовних етапів, які охоплюють процес створення ПЗ – від формування бізнес-ідеї та її трансформації у програмний код до впровадження в бізнес-процеси, експлуатації та подальшого супроводу й розвитку. ЖЦ розроблення ПЗ (Software Development Life Cycle, SDLC) – це структура, яка визначає етапи розроблення ПЗ, від його створення до виведення з експлуатації, надаючи план завдань, які необхідно виконати для кожного етапу ЖЦ (рис. 1.3).

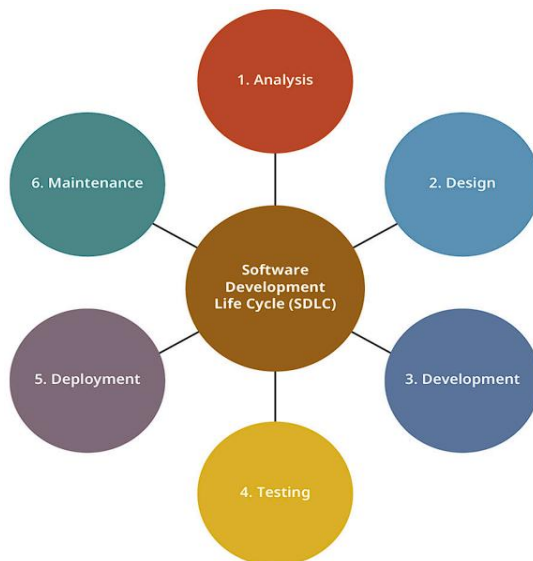


Рисунок 1.3 – Життєвий цикл розроблення ПЗ

Дотримання рекомендованого процесу SDLC веде до створення якісного ПЗ.

Аналіз вимог (Analysis). На цьому етапі збирається інформація, необхідна для розроблення ПЗ, його потенційного дизайну та програмного коду. До цієї діяльності зазвичай долучені системні аналітики, які формують вимоги користувачів або аспекти програмного рішення, зазначені стейкхолдерами. Системні аналітики обов'язково обговорюють кожен аспект вимог із стейкхолдерами, щоб усунути неоднозначність та покращити ясність і розуміння.

Проектування (Design). На етапі проектування вимоги використовуються як вхідні дані (специфікації проекту). Специфікації проекту створюються, переглядаються стейкхолдерами та затверджуються для розроблення. Розробник використовує цю інформацію, щоб оцінити

підготовлене ПЗ відповідно до вимог кінцевих користувачів.

Програмування (Development). Етап програмування іноді називають кодуванням. Розробник створює програмну систему на мові програмування. Цей етап є найдовшим у життєвому циклі SDLC.

Тестування і налагодження (Testing). Після завершення етапу розроблення починається тестування проекту для перевірки функцій програмної системи на відповідність вимогам та виявлення помилок і дефектів у коді. Розробники виправляють виявлені проблеми, оновлюючи компоненти програмної системи для повторного тестування.

Розгортання (Deployment). На цьому етапі продукт випускається для використання стейкхолдерами, його супроводжує документація щодо використання продукту.

Експлуатація, супровід і підтримка (Maintenance). Етап супроводу відбувається після розгортання програмного продукту та його функціонування у виробничому середовищі, коли можуть знадобитися періодичні виправлення та оновлення для покращення зручності використання продукту. Протягом періоду супроводу відбуваються вдосконалення продукту.

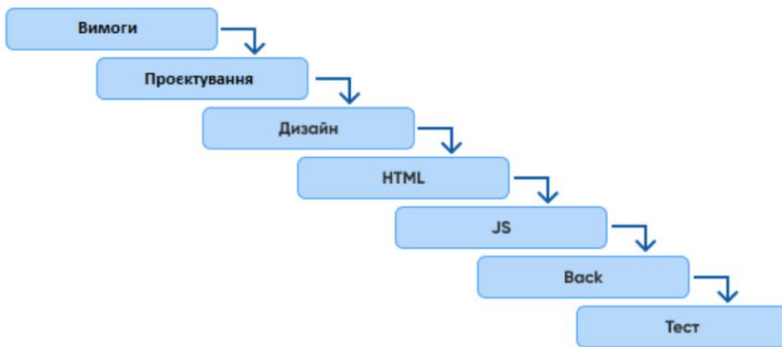
Процес розроблення та впровадження ПЗ може відрізнятися залежно від особливостей проходження етапів його ЖЦ. Для впорядкування цих процесів були розроблені моделі ЖЦ, які є спрощеним і узагальненим відображенням підходів до створення та розвитку ПЗ. До основних моделей належать каскадна (водоспадна, Waterfall), ітераційна, інкрементна, спіральна.

Узагальнюючи, усі моделі ЖЦ розробки ПЗ можна класифікувати на дві основні групи: **послідовні**, що передбачають поетапне виконання робіт, та **ітераційні**, які

ґрунтуються на багаторазовому повторенні циклів розроблення з поступовим удосконаленням ПЗ.

Каскадна модель (Waterfall) базується на принципі послідовного виконання етапів ЖЦ розробки ПЗ. Кожний наступний етап розпочинається лише після повного завершення попереднього, що забезпечує чітку залежність між стадіями та формує безперервний поступальний процес розроблення (рис.1.4).

Як працює зсередини



Як бачить клієнт



Рисунок 1.4 – Каскадна модель

Хоча каскадна модель передбачає послідовне виконання етапів ЖЦ, окремі незалежні роботи можуть здійснюватися паралельно. Проте результати таких робіт інтегруються лише на наступних стадіях розроблення, а не в межах одного етапу.

У межах каскадної моделі команди розробників, відповідальні за різні етапи ЖЦ, працюють відносно автономно, при цьому кожна з них виконує чітко визначений комплекс завдань. До основних недоліків такого підходу належать отримання завершеного

результату лише після проходження всіх етапів розроблення, а також складність своєчасного виявлення й усунення помилок. У разі виявлення дефектів на пізніх стадіях повернення до попередніх етапів є трудомістким, оскільки наслідки помилок часто проявляються лише після завершення значної частини робіт.

З позиції замовника каскадна модель є зрозумілою та логічною, оскільки всі етапи виконуються у визначеній послідовності: після завершення проєктування здійснюється програмування, далі – тестування, а потім впровадження продукту. Такий підхід забезпечує чіткий контроль за виконанням робіт, оскільки кожний етап завершується контрольними точками, що дає змогу оцінити ступінь готовності проєкту та дотримання встановлених термінів. Саме тому каскадна модель широко використовується у великих проєктах, реалізація яких може тривати роками.

Разом із тим уявлення про простоту каскадної моделі є певною мірою умовним. Це пояснюється тим, що замовник здебільшого не бере безпосередньої участі у процесі розроблення, а оцінює результати лише на визначених контрольних етапах або після завершення проєкту. Насправді управління такою моделлю є досить складним, особливо за умов виникнення нових вимог або непередбачених ризиків. У разі суттєвих змін до вимог або появи критичних ризиків виникає потреба у перегляді структури проєкту, коригуванні планів робіт і перегляді календарних графіків.

На відміну від каскадної моделі, **ітераційна модель** передбачає поділ проєкту на окремі ітерації, кожна з яких охоплює повний цикл виконання основних етапів ЖЦ розроблення ПЗ (рис. 1.5).

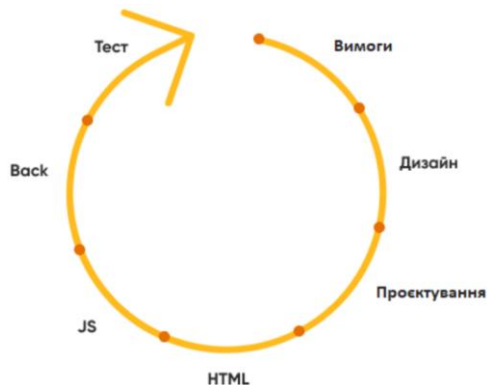


Рисунок 1.5 – Ітераційна модель

Кожна ітерація є завершеним етапом розроблення, результатом якого виступає працездатна версія ПЗ. Сукупність усіх ітерацій забезпечує створення кінцевого продукту, що повністю відповідає визначеним вимогам.

Принцип функціонування ітераційної моделі можна пояснити на простому прикладі. Якщо необхідно виготовити стіл для вітальні, то на першій ітерації створюється його базова конструкція – стільниця та ніжки. На наступній ітерації здійснюється фарбування виробу, а на завершальній – його остаточне оформлення, наприклад комплектування стільцями. Таким чином, після кожного етапу отримується результат, який уже має практичну цінність, хоча й може мати певні функціональні обмеження.

Аналогічний підхід використовується і під час створення ПЗ. На кожній ітерації виконується робота над тим самим програмним продуктом, який поступово доповнюється новими функціональними можливостями, удосконалюється та наближається до остаточної версії. Одночасно можуть уточнюватися вимоги до ПЗ, тому кожна ітерація потенційно може стати як проміжним, так і завершальним етапом розроблення.

Ітераційний підхід є ефективним засобом управління невизначеністю, оскільки дозволяє поступово перевіряти прийняті технічні, організаційні та маркетингові рішення ще на ранніх стадіях проєкту. Це сприяє своєчасному виявленню недоліків, зменшенню ризику невдалого завершення проєкту, раціональному використанню бюджету, а також забезпечує узгоджене розуміння вимог і результатів між замовником та командою розробників. Крім того, на відміну від каскадної моделі, ітераційний підхід допускає завершення проєкту після будь-якої ітерації за умови досягнення необхідного рівня функціональності.

До основних різновидів ітераційної моделі належать **спіральна** та **інкрементна** моделі ЖЦ.

У **спіральній моделі** життєвий цикл організований у вигляді послідовності витків спіралі, кожен з яких охоплює повний цикл основних процесів розроблення, зокрема аналіз вимог, проєктування, програмування, тестування та оцінювання отриманих результатів (рис.1.6).

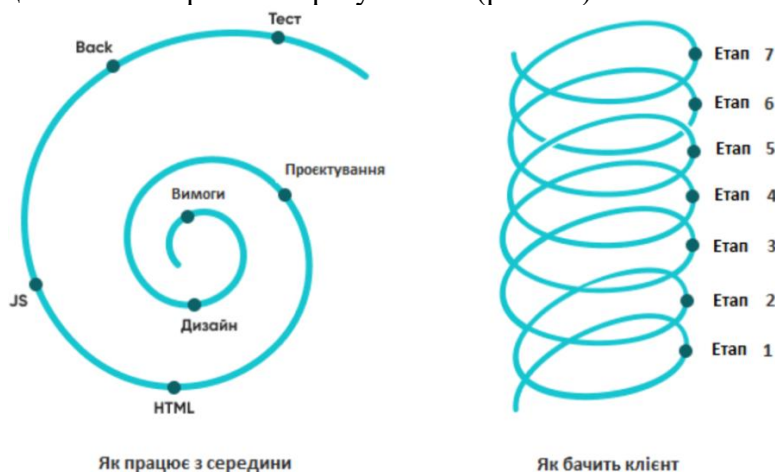


Рисунок 1.6 – Спіральна модель

Спіральна модель повністю відповідає своїй назві: процес розроблення організовується у вигляді послідовних

витків спіралі, кожен з яких наближає проєкт до досягнення кінцевої мети. На кожному витку можуть виконуватися різні процеси та використовуватися різні підходи залежно від поточних потреб проєкту, однак результати кожної ітерації є складовою загального процесу створення програмного продукту.

В основі **інкрементної моделі** лежить принцип поступового розширення функціональних можливостей ПЗ шляхом послідовного додавання нових модулів і компонентів. Сам термін *increment* означає «приріст» або «збільшення», що відображає поетапне нарощування функціональності програмної системи. На відміну від каскадної моделі, де програмний продукт фактично існує лише у двох станах — «у процесі розроблення» та «готовий до використання», інкрементний підхід передбачає створення серії проміжних версій продукту.

Кожне завершення ітерації супроводжується випуском нової версії ПЗ, яка позначається відповідним номером (наприклад, v1.0, v2.0, v3.0). Зміна основного номера версії зазвичай свідчить про суттєві зміни архітектури або функціональних можливостей системи. Якщо ж виникає потреба оперативно виправити помилки або додати незначні вдосконалення без зміни базової архітектури, випускаються проміжні версії, наприклад v1.1, v1.2, v1.3, або релізи нижчого рівня – v1.1.1, v1.1.2 тощо. Послідовне накопичення таких удосконалень у підсумку приводить до випуску нової основної версії ПЗ.

Однією з найвідоміших ітераційних технологій розроблення ПЗ є Rational Unified Process (RUP), що вважається однією з найрозвиненіших методологій програмної інженерії та широко використовується як корпоративний стандарт під час реалізації складних програмних проєктів.

Ключовим принципом RUP є ітераційність. Відповідно до цього принципу розроблення програмної системи здійснюється у вигляді серії короткотривалих мініпроектів тривалістю, як правило, від двох до шести тижнів. Кожна ітерація охоплює повний цикл робіт, включаючи аналіз вимог, проектування, програмування, тестування, інтеграцію та отримання працездатної версії програмної системи. Подальший розвиток ПЗ відбувається шляхом послідовного розширення його функціональності, інтеграції нових модулів із наявним ядром системи, постійного отримання зворотного зв'язку та внесення необхідних змін за результатами кожної ітерації.

Відповідно до методології RUP ЖЦ розроблення ПЗ поділяється на окремі цикли розроблення, кожен із яких завершується створенням нової версії або покоління ПЗ (рис.1.7).

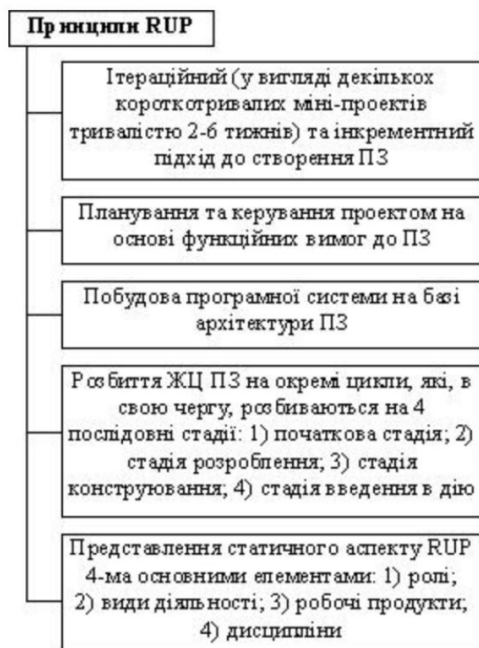


Рисунок 1.7 – Основні принципи технології RUP

Кожний цикл складається з чотирьох взаємопов'язаних стадій: початкова стадія, стадія розроблення, стадія конструювання, стадія впровадження. Завершення кожної стадії супроводжується досягненням визначеної контрольної точки (*milestone*), у якій оцінюються результати виконаних робіт, перевіряється досягнення поставлених цілей та приймаються рішення щодо переходу до наступної стадії проекту.

Початкова стадія (Inception) присвячена оцінюванню доцільності реалізації проекту та формуванню його загальної концепції. На цьому етапі здійснюється аналіз бізнес-потреб, визначаються межі майбутньої програмної системи, оцінюються очікувані витрати, економічний ефект і масштаб проекту. Крім того, формується початковий план реалізації та визначаються основні ризики. Результатами цієї стадії є загальна концепція програмної системи, початкова модель варіантів використання, глосарій основних термінів, бізнес-план, календарний план виконання робіт із розподілом на стадії та ітерації, а також один або декілька початкових прототипів.

Стадія розроблення (Elaboration) спрямована на деталізацію вимог до програмної системи, аналіз предметної області та проектування її архітектури. У процесі виконання цієї стадії визначається технологічна платформа, формується базова архітектура системи, уточнюються функціональні та нефункціональні вимоги, розробляється детальний план подальших робіт і проводиться усунення найбільш критичних ризиків. Основним результатом цього етапу є архітектурна модель програмної системи, яка слугує основою для подальшого програмування. Крім того, створюються уточнена модель варіантів використання, перелік вимог, працездатний архітектурний прототип, оновлений бізнес-план та детальний план виконання всіх ітерацій із визначенням

критеріїв їх оцінювання. Як правило, ця стадія займає близько 20% загальної тривалості проєкту.

Ознаками успішного завершення стадії розроблення є можливість достатньо точно оцінити трудомісткість реалізації всіх варіантів використання програмної системи, а також виявлення основних ризиків проєкту та визначення ефективних підходів до їх мінімізації.

Стадія конструювання (Construction) охоплює безпосереднє створення ПЗ відповідно до розробленої архітектури. На цьому етапі виконується програмування, інтеграція компонентів, тестування та підготовка ПЗ до експлуатації. Результатом є готовий до передачі користувачам програмний продукт, який повинен містити інтегроване ПЗ для цільових платформ, користувацьку документацію та опис реалізованої програмної системи.

Стадія впровадження (Transition) забезпечує передачу програмного продукту кінцевим користувачам та його введення в експлуатацію. До основних робіт цієї стадії належать проведення бета-тестування, паралельна експлуатація нової та існуючої систем, перенесення даних, оптимізація продуктивності, а також навчання користувачів і персоналу, відповідального за технічний супровід.

Статична структура методології RUP базується на чотирьох основних складових:

- 1) ролі – визначають обов'язки, відповідальність і функції окремих виконавців або команд;
- 2) види діяльності – окремі роботи, що виконуються конкретними учасниками процесу розроблення;
- 3) робочі продукти – результати виконання окремих видів діяльності (документи, моделі, програмний код, прототипи тощо);
- 4) дисципліни – логічно пов'язані процеси, що об'єднують сукупність робіт, необхідних для отримання певного результату.

Методологія RUP охоплює шість основних дисциплін: моделювання бізнес-процесів, керування вимогами, аналіз і проєктування, реалізацію, тестування та розгортання. Крім цього, вона включає три допоміжні дисципліни: керування конфігурацією та змінами, керування проєктом і створення та підтримку інфраструктури розроблення.

1.6 Гнучке розроблення програмного забезпечення

Гнучке розроблення ПЗ (Agile) – це загальний термін, що описує ітеративне та поступове надання якісних продуктів і цінності стейкхолдерам за допомогою командної співпраці, гнучкості та постійного планування. Це адаптивний підхід до розроблення ПЗ, який враховує невизначеність у мінливих середовищах і дозволяє командам розробників швидко реагувати на зміни.

Одним із ключових принципів Agile є визнання пріоритетності людей та їхньої взаємодії над процесами й інструментами. У центрі Agile-підходу перебуває ефективна співпраця стейкхолдерів, спрямована на спільний пошук оптимальних рішень у процесі розроблення ПЗ. Замовник, команда розробників та інші стейкхолдери підтримують постійну комунікацію, що забезпечує безперервний обмін знаннями, досвідом та ідеями, а також створює умови для колективного прийняття управлінських і технічних рішень. Такий підхід сприяє своєчасному виявленню проблем, зменшенню ризиків перевитрат часу та фінансових ресурсів, а також підвищує здатність команди ефективно працювати в умовах високої невизначеності й оперативно реагувати на зміни вимог. Разом із тим Agile не є універсальним рішенням і має певні обмеження. Надмірна кількість комунікацій без належної організації може призвести до зниження керованості проєктом і виникнення хаотичності в роботі команди. Тому ефективне застосування Agile передбачає дотримання низки умов:

формування невеликих за чисельністю команд, залучення компетентних і вмотивованих фахівців, організацію коротких ітерацій із чітко визначеними цілями, встановлення часових обмежень для виконання робіт та визначення зрозумілих критеріїв досягнення очікуваного результату.

Фундаментальним принципом Agile є пріоритетність створення працездатного програмного продукту над підготовкою надмірно деталізованої документації. Основна увага приділяється розробленню програмного забезпечення, яке вже на ранніх етапах може бути використане та оцінене замовником, тоді як документація формується в обсязі, достатньому для підтримки процесу розроблення й подальшої експлуатації програмної системи.

Ще одним основоположним принципом Agile є пріоритетність конструктивної співпраці із замовником над формальним дотриманням умов контракту. Agile-підхід передбачає постійну взаємодію між командою розробників і замовником протягом усього життєвого циклу проєкту, що дає змогу своєчасно уточнювати вимоги, оперативно реагувати на зміни та забезпечувати створення програмного продукту, який найбільш повно відповідає реальним потребам користувачів.

Реагування на зміни цінується більше, ніж дотримання плану. Традиційні процеси розробки часто дотримуються жорстких термінів, де зміни призводять до додаткових витрат та затримок. Agile-підхід включає зміни в процес планування через зворотний зв'язок від стейкхолдерів.

Історично розробка ПЗ виконувалася з використанням підходу SDLC, який є систематичним та процесоорієнтованим, переходячи від одного етапу до наступного. Як SDLC, так і Agile-розробка ПЗ включають планування документації, зворотний зв'язок зі стейкхолдерами та тестування системи. Однак, Agile-підхід

забезпечує набагато більшу гнучкість у врахуванні змін та відгуків кінцевих користувачів на всіх етапах процесу розробки. Підхід SDLC можна використовувати з проєктами будь-якого розміру, тоді як Agile-підхід краще підходить для проєктів меншого масштабу. При підході SDLC зазвичай не вноситься жодних змін після початкових етапів, і взаємодія з кінцевим користувачем (замовником) мінімальна. На противагу цьому, при Agile-підході з кінцевими користувачами регулярно консультуються.

Методологія Scrum базується на організації процесу розроблення у вигляді коротких, фіксованих за тривалістю ітерацій, які називаються спринтами (Sprint). У межах кожного спринту команда виконує визначений обсяг робіт, спрямованих на створення або вдосконалення програмного продукту.

Перед початком кожного спринту проводиться планування спринту (Sprint Planning), під час якого аналізується та пріоритизується список завдань продукту (Product Backlog), після чого формується беклог спринту (Sprint Backlog) – перелік завдань, що мають бути виконані протягом поточної ітерації. Саме цей перелік визначає обсяг робіт команди розробників на відповідний спринт.

Тривалість спринту є фіксованою і зазвичай становить від одного до чотирьох тижнів. Таким чином, увесь процес розроблення програмного продукту поділяється на послідовність рівних за тривалістю спринтів, кожен із яких завершується отриманням працездатного інкременту ПЗ.

Перший крок у процесі Agile-розробки включає планування та підготовку для залучення потрібних фахівців до роботи в команді (Team). Agile-команда складається з невеликої групи осіб і призначена для реалізації одного проєкту/продукту. Agile-команда також може включати ключових стейкхолдерів (рис. 1.8).

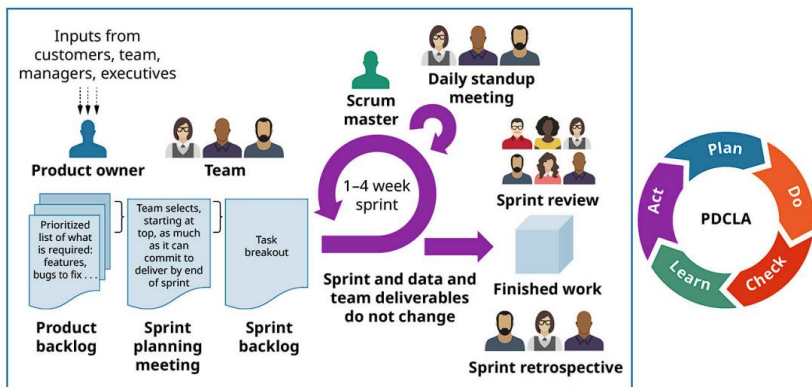


Рисунок 1.8 – Фази Agile-розробки

Agile-команда створює план виконання визначених вимог та отримання кінцевого продукту. Вона визначає функції, які становитимуть кінцевий продукт, та порядок, у якому ці функції будуть додані. Ці функції називаються беклогом продукту або дорожньою картою. Після того, як беклог продукту встановлено, починається робота з розробки ПЗ.

Регулярні зустрічі Agile-команди з власником продукту (Product owner) проводяться для оцінювання поточного стану розробки. Користувачькі історії (user stories) – це функціональні компоненти роботи, які визначаються спільно з власником продукту або стейкхолдерами. Очікується, що кожна з цих історій доповнить готовий кінцевий продукт.

Agile-командами керують менеджери проєктів (Scrum-master), використовуючи Agile-методологію, яка відрізняється від традиційно керованих проєктів. Менеджер Agile-проєктів веде команду через динамічні, швидкозмінні та нестабільні середовища з великою невизначеністю для досягнення цілей проєкту.

Навички, необхідні для ролі менеджера Agile-проєктів, включають комунікативні та організаторські здібності,

критичне мислення, здатність працювати в командному середовищі, а також глибоке розуміння процесів управління проєктами та Agile-процесів.

Важливим елементом менеджменту Agile-проєктів є планування спринтів – циклів виконання, протягом яких (від 1 до 6 тижнів) визначена робота виконується та перевіряється. Під час спринтів команда проводить щоденну зустріч, яку часто називають стендапом (standup), зазвичай від десяти до тридцяти хвилин, щоб обговорити досягнутий прогрес та виниклі труднощі. Крім того, команди розбивають великі завдання на підзадачі для кожного спринту.

Для кожного спринту Agile-команда визначає на основі навичок своїх учасників користувацькі історії, над якими вона працюватиме під час спринту, та оцінює час, необхідний для завершення спринту. Наприклад, команда може вирішити провести двотижневий спринт, прагнучи виконати два завдання з кодування, два завдання з тестування та два завдання з документування. Після цього користувацькі історії пріоритезуються.

Під час спринту члени команди активно працюють над дорученими їм завданнями. Коли під час спринтів виникає проблема, вона додається до беклогу та пріоритезується разом з роботою, яка вже зберігається в беклогу. Після завершення кожного спринту команда, як правило, має готовий до використання працездатний продукт, готовий для перегляду стейкхолдерами або власником продукту під час зустрічі з огляду спринту.

Команди можуть побачити обсяг роботи, що залишилася, за допомогою діаграми вигорання (burndown chart) або графічного представлення роботи, виконаної в спринті, та роботи, що залишилася. Після спринту команда визначає проблеми та можливості для наступного спринту. Цей процес триває до тих пір, поки беклог не буде

вичерпано, а функціональні компоненти програмного продукту не будуть завершені. Після цього команда проводить ретроспективну зустріч, на якій її учасники детально обговорюють спринти та області для покращення, які слід застосувати до майбутніх спринтів.

Визначаючи, як краще підійти до системного аналізу, слід враховувати переваги та недоліки підходів SDLC та Agile. Гнучкість, необхідна в сучасному бізнесі, може вимагати гібридного підходу до розробки ПЗ, який включає аспекти як підходу SDLC, так і Agile. Підхід SDLC може надати проєктам чіткість, детальну документацію, ретельне вивчення ризиків і підводних каменів у проєктуванні, а також пильну увагу до бюджету. Однак підхід SDLC не призначений для проєктів з високою невизначеністю на початкових етапах, а також не підходить для проєктів, у яких можуть виникнути неочікувані зміни. У цих випадках може бути корисним більш гнучкий Agile-підхід.

Застосування ітераційного підходу в поєднанні з постійною взаємодією із кінцевими користувачами дає змогу поступово уточнювати концепцію та архітектуру програмного продукту, активно залучати ключових стейкхолдерів до процесу розроблення й своєчасно виявляти та системно усувати проблеми, що виникають під час реалізації проєкту. Однією з головних переваг Agile-підходу є висока адаптивність до умов невизначеності. За зростання рівня невизначеності тривалість ітерацій, як правило, скорочується, що забезпечує швидший зворотний зв'язок і можливість оперативного внесення змін. В окремих випадках ітерація може тривати лише одну добу, як це практикується під час проведення хакатонів.

Кожна ітерація розпочинається з планування робіт і визначення поставлених цілей, а після її завершення проводяться оцінювання отриманих результатів, контроль виконання поставлених завдань, ретроспективний аналіз

процесу розроблення та планування наступної ітерації. Такий цикл безперервного вдосконалення забезпечує поступове підвищення якості програмного продукту та ефективності роботи команди.

Водночас Agile-підхід має і певні обмеження. Серед основних зауважень до нього відзначають відносно невеликий обсяг формалізованої документації, а також складність точного прогнозування термінів виконання робіт і загальної вартості проєкту. Це зумовлено високою гнучкістю методології, яка передбачає постійне уточнення вимог і швидке реагування на зміни в процесі розроблення.

Підсумовуючи, SDLC пропонує структуру, необхідну для дотримання термінів та бюджетів, тоді як Agile надає бізнесу гнучкість у прийнятті рішень у міру розвитку його бачення. Використовуючи гібридний підхід, компанії можуть пом'якшити слабкі сторони та використовувати сильні сторони кожного підходу для задоволення своїх потреб.

Основна ідея Lean-підходу полягає у максимально раціональному використанні ресурсів, зокрема часу, фінансів і людського потенціалу, шляхом пошуку найпростішого способу досягнення поставленої мети. Замість розроблення повнофункціонального продукту без підтвердження його ринкової затребуваності Lean-підхід пропонує спочатку перевірити бізнес-гіпотезу за допомогою мінімальних засобів. Наприклад, для оцінювання потенційного попиту може бути створена проста цільова вебсторінка (landing) із формою реєстрації або підписки, після чого за допомогою рекламної кампанії аналізується зацікавленість потенційних користувачів. Отримані результати дають змогу обґрунтовано прийняти рішення щодо доцільності подальшого розроблення програмного продукту.

Якщо підтверджено наявність ринкового попиту, наступним кроком є створення мінімально життєздатного продукту (Minimum Viable Product, MVP). Це найпростіша версія програмного продукту, що реалізує його ключову функціональність і водночас забезпечує достатню цінність для користувачів. Такий продукт використовується для перевірки прийнятих рішень, отримання зворотного зв'язку та визначення напрямів подальшого розвитку. За результатами експлуатації MVP команда розробників може ухвалити рішення щодо вдосконалення продукту або відмови від подальшої реалізації проєкту, мінімізувавши при цьому витрати ресурсів.

Lean-підхід орієнтований насамперед на перевірку потреб користувачів і підтвердження цінності запропонованого рішення для ринку. Його головна увага зосереджена на економічній доцільності та відповідності очікуванням споживачів. Методологія Lean передбачає безперервний пошук найпростішого технічного, організаційного або управлінського рішення, яке забезпечує досягнення поставленої мети без створення зайвої складності.

Разом із тим надмірне прагнення до спрощення може стати недоліком Lean-підходу. Якщо процес оптимізації призводить до вилучення функцій, що формують основну цінність продукту, ПЗ може втратити практичну корисність для користувачів. Тому принцип ощадливості повинен поєднуватися із збереженням функціональних характеристик, які визначають конкурентоспроможність програмного продукту. Слід також зазначити, що Lean не обмежується лише перевіркою попиту за допомогою цільових вебсторінок. Такий спосіб є лише одним із практичних прийомів, тоді як сама методологія охоплює широкий спектр підходів до управління розробленням та розвитку програмного продукту.

Канбан-підхід візуалізує процес виконання робіт. Процес розроблення ПЗ поділяється на окремі етапи, наприклад планування, програмування, тестування та розгортання. Усі завдання подаються у вигляді карток, які переміщуються між відповідними стовпцями Канбан-дошки відповідно до їхнього поточного стану. Нові завдання можуть додаватися в будь-який момент, а завершення роботи визначається не закінченням встановленого часового інтервалу, а переходом картки до статусу «Виконано».

Основною метою Dynamic Systems Development Method (DSDM) є своєчасне завершення проєкту в межах визначеного бюджету із можливістю адаптації вимог у процесі розроблення. DSDM застосовується не лише під час створення ПЗ, а й у реалізації проєктів інших предметних областей.

Rapid Application Development (RAD) – це методологія швидкого розроблення ПЗ, що широко використовує засоби візуального моделювання, прототипування та повторного використання компонентів. Для RAD характерними є невеликі команди розробників, короткі терміни реалізації проєктів (зазвичай до чотирьох місяців) і активна участь замовника протягом усього процесу розроблення. Методологія допускає уточнення та зміну вимог під час виконання проєкту, що забезпечує скорочення строків створення програмного продукту та зменшення витрат.

Extreme Programming (XP) є однією з гнучких методологій програмної інженерії, орієнтованою на ефективну роботу в умовах постійної зміни вимог до ПЗ. Вона ґрунтується на сукупності практик, спрямованих на підвищення якості програмного продукту та продуктивності команди розробників. До найважливіших із них належать:

- безперервне уточнення та коригування плану розроблення;
- використання простої та зрозумілої архітектури програмної системи;
- регулярне автоматизоване тестування;
- парне програмування, за якого два розробники спільно працюють над одним завданням;
- постійна інтеграція програмного коду та частий випуск невеликих функціональних релізів.

1.7 Європейський цифровий регуляторний простір: виклики для системного аналізу та програмної інженерії

ЄС активно розбудовує суверенну цифрову інфраструктуру, створюючи щільне регуляторне поле. Це вимагає від системних аналітиків обов'язкового врахування законодавчих вимог як першочергових нефункціональних обмежень при проєктуванні програмних систем. Дев'ять ключових європейських регламентів формують сьогодні нові правила розробки ПЗ.

General Data Protection Regulation (GDPR). Регламент GDPR вимагає системного впровадження концепцій Privacy by Design (приватність за дизайном) та Privacy by Default (приватність за замовчуванням). Системний аналітик зобов'язаний закласти в архітектуру бази даних принципи мінімізації даних та обмеження терміну зберігання. На практиці це означає проєктування підсистем автоматизованого очищення та анонімізації даних після завершення терміну їхнього використання. При розробці інтерфейсів мають бути передбачені механізми отримання згоди користувача (Consent Management), які не дозволяють за замовчуванням встановлювати прапорці згоди. Крім того, архітектура програмної системи повинна забезпечувати реалізацію прав суб'єктів даних, зокрема права на забуття та права на перенесення даних.

AI Act (Акт про штучний інтелект). Регламент (ЄС) 2024/1689 (EU AI Act) вводить суворий ризик-орієнтований підхід до систем штучного інтелекту (ШІ). Забороняються системи ШІ з неприйнятним ризиком (наприклад, соціальний скоринг). Для високоризикових систем ШІ встановлюються детальні вимоги до якості навчальних даних, прозорості алгоритмів, логування процесів та людського нагляду (Human-in-the-Loop). З метою забезпечення регуляторної відповідності системні аналітики повинні використовувати також міжнародний стандарт ISO/IEC 42001, який регламентує створення систем управління ШІ (Artificial Intelligence Management System, AIMS). Цей стандарт вимагає ведення ретельного обліку моделей ШІ, відстеження походження даних (Data Lineage) та оцінювання етичного впливу на кожному етапі життєвого циклу. Контролюючим органом є Європейський офіс ШІ (EU AI Office).

Cyber Resilience Act (CRA). Регламент CRA фокусується на безпеці продуктів із цифровими елементами, що постачаються на європейський ринок. Він вимагає від розробників програмного продукту забезпечення безпеки протягом усього життєвого циклу продукту (мінімум 5 років або протягом очікуваного терміну служби). Даний регламент вводить жорсткі вимоги щодо швидкого сповіщення про активно експлуатовані вразливості та серйозні кіберінциденти: розробники мають надіслати первинне попередження до ENISA (European Union Network and Information Security Agency) та національних CSIRT (Computer Security Incident Response Team) протягом 24 годин, детальний звіт – протягом 72 годин, а фінальний звіт – протягом 14 днів. Недотримання цих правил загрожує блокуванням продажів програмних продуктів на ринку ЄС та накладенням великих штрафів.

NIS2 Directive. Регулює організаційну кіберстійкість суб'єктів критичної інфраструктури. На відміну від регламенту CRA, який фокусується на кібербезпеці самих програмних продуктів, директива NIS2 висуває вимоги до кібербезпеки операційних процесів компаній, управління ризиками в ланцюгах постачання та безперервності бізнесу. Системні аналітики при проєктуванні корпоративних систем мають інтегрувати механізми виявлення кіберінцидентів, стійкі архітектури резервного копіювання та засоби швидкого відновлення після збоїв.

Data Act. Цей закон розширює права користувачів на доступ до даних, що генеруються підключеними пристроями Інтернету речей та супутніми сервісами. Він вимагає, щоб дані за замовчуванням проєктувалися як легкодоступні для користувачів. Системний аналітик має спроектувати архітектуру програмної системи таким чином, щоб забезпечити безкоштовний, безперешкодний та безперервний доступ до даних у реальному часі. Закон також встановлює механізми отримання даних державними органами в умовах надзвичайних ситуацій (B2G-взаємодія).

Data Governance Act (DGA). Спрямований на створення Common European Data Spaces (Єдиних європейських просторів даних) у стратегічних сферах (охорона здоров'я, енергетика, транспорт тощо). Він урегульовує діяльність посередників у обміні даними (Data Intermediaries) та створює умови для добровільного поширення даних на користь суспільства (Data Altruism). Для системного проєктування це означає необхідність розробки стандартизованих моделей метаданих, використання надійних технологій знеособлення (деідентифікації) даних та забезпечення сумісності інтерфейсів обміну даними на транскордонному рівні.

Digital Services Act (DSA). Регулює діяльність онлайн-посередників, маркетплейсів та соціальних мереж. Висуває

жорсткі вимоги до прозорості алгоритмів рекомендаційних систем, модерації контенту цифрових платформ і заборони маніпулятивних інтерфейсів (Dark Patterns). DSA забороняє таргетну рекламу на основі чутливих персональних даних або спрямовану на неповнолітніх. Всі онлайн-магазини в ЄС зобов'язані впровадити кнопку «Скасувати договір» (Cancel Contract), яка дозволяє розірвати угоду про покупку в один клік, що потребує відповідної побудови логіки оброблення замовлень на рівні backend-архітектури.

Digital Markets Act (DMA). Цей акт націлений на обмеження монопольного впливу великих технологічних платформ, визначених як гейткіпери (Gatekeepers), що мають капіталізацію понад €75 млрд або річний оборот в ЄС понад €7.5 млрд та обслуговують більше 45 млн активних користувачів щомісячно. DMA зобов'язує гейткіперів забезпечити безкоштовну та ефективну сумісність сторонніх програм із функціями своїх ОС та апаратного забезпечення. DMA вимагає від великих платформ надання кінцевим споживачам можливості безперервного перенесення даних (data portability) у реальному часі, незалежно від правової основи оброблення замовлень у межах регламенту GDPR. Для реалізації таких запитів системні аналітики сторонніх компаній мають розробляти інтеграційні шари, що взаємодіють з API гейткіперів. Водночас розробники зобов'язані здійснювати оцінку впливу на захист персональних даних (Data Protection Impact Assessment, DPIA) відповідно до вимог GDPR, що дає змогу своєчасно виявляти, оцінювати та мінімізувати ризики, пов'язані з обробкою персональних даних.

European Accessibility Act (EAA). Цей акт робить цифрову інклюзивність обов'язковою вимогою для вебсайтів і мобільних застосунків. Недотримання вимог ЕАА загрожує штрафами до €20 000 за кожне порушення та щоденними санкціями.

Контрольні запитання

1. Наведіть приклад робочого процесу.
2. Яка основна функція системного аналітика?
3. Дайте визначення процесу аналізу даних.
4. Опишіть концепцію системного аналізу.
5. Визначте методології та інструменти системного аналізу.
6. Поясніть процес SDLC.
7. Який інструмент ви б обрали для проведення системного аналізу і чому?
8. Наведіть класифікацію системних вимог.
9. Назвіть основні розділи звіту про системний аналіз.
10. Що включає розробка програмної системи?
11. Дайте визначення життєвого циклу програмного продукту.
12. Поясніть каскадну модель життєвого циклу управління проєктами та наведіть приклад, коли компанія повинна обрати саме її.
13. Поясніть основні принципи таких методологій управління проєктами, як SDLC, Agile, Lean.
14. На якому етапі проєкту використовується опис робіт?
15. Дайте визначення опису робіт та поясніть, як він використовується.
16. Що таке структура розподілу робіт і як вона допомагає менеджеру проєкту?
17. Дайте визначення стейкхолдеру.
18. Чому важливо залучати стейкхолдерів до оцінювання та управління ризиками?
19. Як можна залучити стейкхолдерів?
20. Розробіть короткий сценарій управління проєктом, де найкращим підходом до розробки є інкрементний.

РОЗДІЛ 2

ІДЕНТИФІКАЦІЯ ВИМОГ КОРИСТУВАЧІВ ТА ОСНОВИ ПРОЄКТУВАННЯ СИСТЕМ

2.1 Визначення бізнес-проблем

Бізнес-проблема – це будь-яка перешкода, яка спричиняє відхилення бізнесу від очікуваного результату. Вона може суттєво впливати на конкурентоспособність бізнесу. Щоб визначити бізнес-проблему, впровадити її рішення, а також оцінити його ефективність, потрібно виконати такі кроки.

1. Визначення проблеми. Для виявлення та всебічного аналізу проблеми компанія може залучати працівників і ключових стейкхолдерів. Глибше зрозуміти природу проблемної ситуації та здійснити її критичний аналіз допомагає використання системи послідовних запитань. Початковий етап аналізу доцільно розпочинати з таких запитань, як: «Чому це є проблемою?», «Чому виникла саме така ситуація?» або «Чому проблема проявилася саме зараз?». Подальше уточнення здійснюється шляхом постановки додаткових запитань, зокрема: «Які напрями діяльності компанії зазнають впливу цієї проблеми?» та «Які бізнес-ролі або категорії учасників бізнес-процесів вона зачіпає?». Такий підхід дає змогу комплексно оцінити масштаби проблеми, визначити її причини та наслідки, а також сформулювати обґрунтовані передумови для прийняття управлінських рішень.

2. Визначення можливих рішень. Використовуючи дані, зібрані під час визначення проблеми, а також висновки, отримані від учасників процесу, можна провести мозковий штурм щодо потенційних рішень, які можуть вирішити визначену бізнес-проблему. Важливо розглянути варіант нічого не робити, а також його наслідки з точки зору короткострокового та довгострокового планування. Можливі рішення повинні враховувати їх вартість, вплив,

час, ресурси та інші фактори, які визначають життєздатність можливих варіантів рішень.

3. Вибір та впровадження рішення. Потрібно розглянути запропоновані варіанти рішення, вибрати та впровадити той варіант рішення, який є найбільш доцільним та життєздатним.

4. Оцінювання рішення. Якщо рішення виявилось успішним, його зазвичай застосовують до подібних проблем. Для тих рішень, які не спрацювали, процес визначення проблеми може потребувати перезавантаження, а варіанти рішення – повторного оцінювання. Для досягнення бажаного результату може бути обраний інший варіант рішення.

2.2 Методи ідентифікації вимог користувачів

Вирішення бізнес-проблем вимагає розуміння того, що потребують кінцеві користувачі або власники продукту та що потрібно для підтримки їхніх бізнес-функцій. Вимоги користувача – це аспекти рішення, які визначаються стейкхолдерами та необхідні для підтримки конкретних потреб та очікувань бізнес-процесу або продукту. Вимоги користувача записуються у формі специфікацій і можуть бути класифіковані як функціональні або нефункціональні вимоги.

Функціональна вимога – це функція програмного застосунку, необхідна для виконання завдань відповідної бізнес-сфери. До них можуть належати бізнес-правила, транзакції, адміністративні функції, аудит, автентифікація, авторизація зовнішніх інтерфейсів та функції звітності.

Нефункціональна вимога – це атрибут системи, який покращує її функціональність. Нefункціональні вимоги часто розглядаються як елементи, що описують загальні властивості системи або те, як система повинна поводитися та працювати. До них можуть належати продуктивність,

масштабованість, безпека, гнучкість, експлуатаційні обмеження, зручність використання, сумісність, ремонтпридатність, доступність та потужність.

Вимоги зазвичай визначаються командою осіб, включаючи користувачів, аналітиків та ІТ-фахівців із досвідом роботи з системою, що проєктується або модифікується. Для виявлення вимог використовуються спостереження за користувачами, інтерв'ю, опитування, аналіз документації, мозковий штурм та аналіз інтерфейсу. Одним із способів зберігання цієї інформації є матриця відстеження вимог (requirements traceability matrix, RTM). Зазвичай це електронна таблиця. RTM використовується для запису кожної вимоги разом із додатковою інформацією: тип вимоги (функціональна/нефункціональна), опис, мета, бізнес-потреба, пріоритет, підрозділ та статус її розробки. Компанії використовують RTM як довідник для тестування, щоб переконатися, що кожна вимога була ретельно перевірена, протестована та підтверджена. Крім того, RTM допомагає у верифікації та валідації аналізу вимог.

Блок-схема (flowchart) – це діаграма, яка відображає послідовні зв'язки між даними, системами, програмами та процесами. Її часто використовують технічні та нетехнічні фахівці для документування, планування та передачі ідей у простому для розуміння візуальному форматі. Блок-схеми використовують спеціальні символи, схематичні стрілки та з'єднувачі для покращення візуалізації.

Карта процесу «Як є / Як має бути» використовується для детального моделювання поточного та цільового станів певного бізнес-процесу або функціональної складової ІС. Модель «Як є» (As-Is) відображає існуючий порядок виконання процесу, його структуру, взаємозв'язки та особливості функціонування на момент аналізу. Натомість модель «Як має бути» (To-Be) описує бажаний стан процесу

після впровадження змін, визначаючи необхідні удосконалення, заходи та рішення, що забезпечують досягнення поставлених цілей і підвищення ефективності функціонування ІС.

Діаграма прецедентів (use case) є візуальним представленням функцій ІС, що показує, як певні користувачі взаємодіють зі специфічними функціями ІС (рис. 2.1).



Рисунок 2.1 – Діаграма прецедентів ІС бібліотеки

Ці діаграми зазвичай містять такі компоненти: учасників (людей або будь-яку зовнішню сутність); блок, що представляє саму систему; стрілки напрямку, що показують зв'язки між учасниками та системами (вказують на потік даних до системи та з неї).

Контекстна діаграма — це діаграма високого рівня, яка візуалізує систему або її частини та суб'єкти середовища, з якими вона взаємодіє, за допомогою схематичних стрілок

для відображення потоку даних. Вони вказують на найвищий рівень взаємодії між системою та бізнес-процесами (рис. 2.2).

Ментальна карта — це зображення ідей у вільній формі з гілками, що відображають потік інформації та думок. Їх зазвичай використовують для мозкового штурму.



Рисунок 2.2 – Контекстна діаграма системи онлайн-продажу портативних зарядних пристроїв

Документ з бізнес-вимогами (business requirements document, BRD) – це офіційний посібник, який містить всі аспекти бізнес-потреб та деталі, пов’язані з новим проєктом або програмним рішенням, включаючи цілі, очікування та причини, чому компанії потрібне рішення.

Процес ідентифікації вимог не завжди призводить до ідеального набору інструкцій для вирішення бізнес-проблеми. Часто RTM може містити погані вимоги користувачів, що суттєво впливає на кінцевий продукт та створює труднощі для команди проєкту. Виникнуть пов’язані з цим витрати (ресурси, регуляторні штрафи, затримка виходу на ринок), а також недокументовані процеси, суперечливі вимоги, проблеми комунікації з кінцевими користувачами та помилкова зосередженість на візуальних аспектах дизайну, а не на функціональних елементах.

Для постійного вдосконалення продукту системні аналітики мають ітеративно вирішувати проблеми з поганими вимогами користувачів, виконуючи такі дії.

1. Проведення ретельного дослідження. Розширення знання про бізнес-проблему, розуміючи фактори та рушійні сили, сприяє врахуванню потреб бізнесу. Дослідження має зосереджуватися на контексті, в якому існує бізнес, та враховувати як внутрішні фактори (організаційна культура та бізнес-процеси), так і зовнішні фактори (ринкові умови, нормативні вимоги та конкуренти). Необхідно провести аналіз поширеності виявленої бізнес-проблеми та практик її розв'язання, надавши відповіді на такі ключові питання: чи виникає аналогічна проблема в інших компаніях; які підходи, методи або інструменти використовуються іншими компаніями для її вирішення; а також наскільки ефективними є застосовувані ними рішення. Дослідження може підтвердити або спростувати зібрані вимоги, або визначити нові вимоги для розгляду.

2. Повторний розгляд бізнес-проблеми. Після завершення дослідження необхідно використати зібрані знання для повторного розгляду бізнес-проблеми. Ситуацію можна розглянути з іншого боку, і це може дати додаткове розуміння.

3. Проведення сесії з огляду вимог. Потрібно ініціювати сесії для огляду вимог з усіма стейкхолдерами. Зворотній зв'язок з цих сесій може надати додаткову цінну інформацію. Обов'язково потрібно використати візуальні матеріали (схеми даних, блок-схеми, діаграми, звіти), оскільки вони передають інформацію з більшою чіткістю.

4. Отримання схвалення. Остаточний перегляд та схвалення від стейкхолдерів може бути корисними для вирішення неповних або відсутніх вимог, а також для перевірки зібраних вимог.

Все частіше компанії працюють глобально, а команди розробки розподілені по всьому світу. Це може ускладнити процеси збору вимог та розробки ПЗ і призвести до неефективності та перевитрати коштів. Часто компанії обирають глобальну розробку ПЗ (Global Software Development, GSD), яка дозволяє кваліфікованим працівникам у різних частинах світу розробляти програмні рішення для компаній. На відміну від традиційних команд розробки, де учасники працюють разом, а завдання та діяльність розподіляються для досягнення спільної мети, команди GSD є віртуальними та значною мірою покладаються на комунікаційні технології для розробки ПЗ. До проблем GSD відносяться нижча продуктивність та проблеми з комунікацією. Для забезпечення ефективності процесу GSD необхідна ефективна координація між віртуальними командами та керівниками проєктів.

Визначенням бізнес-проблем, а також управлінням процесом збору вимог займаються аналітики. Назва цієї посади має багато варіацій, включаючи бізнес-аналітиків і системних аналітиків. Ці ролі можна знайти в усіх ІТ-компаніях. Навички, необхідні для цих ролей, включають знання бізнес-аналізу, комунікації, збору вимог, аналітики, програмного забезпечення (Python, R) та інструментів, що використовуються для документування.

2.3 Загальні принципи та дорожня карта проєктування програмних систем

Після визначення бізнес-проблеми, проведення системного аналізу та ідентифікації вимог користувачів із метою вирішення бізнес-проблеми починається процес проєктування системи, що визначає архітектуру, інтерфейси та дані для системи, яка зможе задовольнити

задані функціональні та нефункціональні вимоги. До загальних принципів проєктування систем належать такі.

Простота. Спрощення дизайну системи є кращим, оскільки надмірне ускладнення вимагає більше роботи над проєктом. Потрібно мінімізувати складність, де це можливо.

Ясність. Дизайн системи має бути зрозумілим та простим у використанні. Слід враховувати потреби користувачів, які взаємодіятимуть з програмним продуктом.

Базовий функціонал. Основний функціонал та його взаємопов'язані частини є важливішими, а додаткові системні функції можуть зачекати.

Масштабованість. Для забезпечення готовності запропонованого рішення до організаційних змін необхідно оцінити його адаптивність і масштабованість, надавши відповіді на такі ключові питання: чи здатне рішення ефективно реагувати на зміни в діяльності компанії, зокрема на збільшення або зменшення кількості користувачів, продуктів, бізнес-процесів, послуг, обсягів даних, а також на трансформацію бізнес-середовища; якою є стратегія розвитку компанії на найближчі роки та яким чином її реалізація впливатиме на вимоги до масштабованості й подальшого розвитку програмної системи.

Надійність. Дизайн та системне рішення повинні бути надійними, якщо вони функціонують відповідно до розроблених специфікацій, без збоїв або з мінімальними збоями протягом зазначеного часу їх використання.

Безпека. Система вважається безпечною, коли лише авторизовані користувачі мають до неї доступ і коли вжито всіх заходів для контролю та захисту системи.

Проєктування програмних систем виконують розробники систем (system designer). Ця роль вимагає

технічних і аналітичних знань, досвіду роботи з ПЗ, що використовується для створення документації, а також володіння комунікативними, лідерськими навичками та навичками вирішення проблем.

Прототипування – це підхід до проектування, за якого розробники системи створюють робочу модель рішення. Прототипи зазвичай створюються за допомогою ітеративного процесу розробки або серії безперервних кроків планування, аналізу, впровадження та оцінювання, які призводять до проєкту, що має зростаючу функціональність і все більше відповідає вимогам користувача. Прототипи також дозволяють користувачам модифікувати або вносити проміжні зміни аж до отримання кінцевого продукту. Окрім гнучкості, яку пропонують прототипи, їхні інші переваги включають: раннє виявлення функціональних проблем; підвищення залученості користувачів, покращення співпраці команди; підвищення задоволеності кінцевим продуктом; більша економія часу та коштів, пов'язаних з допрацюванням проєкту.

Однак створення прототипу має свої недоліки. Створення прототипів може бути дорогим, а розробка – трудомісткою. Часто включені функції можуть відрізнятись від кінцевого продукту, і це може вводити в оману стейкхолдерів, оскільки не забезпечує повноцінного функціонуючого продукту. Розробка прототипу також може потребувати доопрацювання через зміну бізнес-вимог.

Керування вводом/виводом належить до процесу проектування програмних систем. Вихідні дані – це інформація, яку система надає користувачам, іншими словами, це дані, що утворюються в результаті оброблення вхідних даних відповідно до логіки роботи системи. Розробники систем створюють форми вводу та екрани

виводу, які враховують питання якості та зосереджені на взаємодії з користувачем.

Наведений нижче список завдань проєктування програмної системи надає дорожню карту для кожного етапу процесу проєктування, дозволяючи команді мати організований робочий процес і приймати обґрунтовані рішення на кожному кроці.

1. Визначити та задокументувати процеси проєктування. Потрібно створити чітку та лаконічну документацію, яка визначає рекомендації щодо поведінки системи, стандартів, атрибутів, доступності та будь-якої іншої інформації, щоб визначити очікування команди щодо кінцевого продукту. Необхідно визначити технології та елементи системи, а також задокументувати вимоги користувачів до функціональності системи.

2. Визначити характеристики дизайну. Потрібно визначити архітектурні компоненти системи та забезпечити можливість їхньої реалізації. Необхідно створити словники, наприклад, слів, моделей поведінки, зображень, фраз, для забезпечення узгодженості з елементами дизайну, а також узгодженості спільного досвіду команди у створенні єдиного користувацького інтерфейсу. Потрібно визначити та задокументувати характеристики дизайну для кожного визначеного компонента.

3. Оцінити альтернативні варіанти дизайну та остаточно визначити проєктні рішення. Потрібно оцінити альтернативні варіанти дизайну на основі подібних або нових розробок, які можливо впровадити як альтернативу. Слід обов'язково включити в процес оцінювання ті компоненти, які ризикують застаріти під час створення системи. Потрібно задокументувати обґрунтування всіх представлених альтернативних варіантів і узгоджене фінальне рішення, включивши до

нього основні апаратні та програмні компоненти, інтерфейси та підсистеми.

4. Керування проєктуванням. Потрібно розробити рішення, включивши основні апаратні та програмні компоненти, інтерфейси та підсистеми. При цьому необхідно забезпечити врахування стандартів доступності та інклюзивності.

5. Перевірка та впровадження проєкту системи. Потрібно переконаватися, що проєкт системи відповідає затвердженим вимогам користувачів, залучивши власників системи, користувачів, проєктувальників та інших стейкхолдерів до процесу перевірки. Потрібно забезпечити навчання всіх користувачів та персоналу служби підтримки системи для забезпечення її належного використання, підтримки та обслуговування.

6. Вимірювання успішності дизайну системи та внесення покращень. Потрібно зібрати відгуки користувачів та оцінити отримані дані за допомогою метрик, розроблених для вимірювання ефективності системи. Необхідно застосувати процеси постійного вдосконалення для реагування на відгуки користувачів, а також оновлення та зміни системи.

2.4 Дизайн, орієнтований на користувача

Розробка мобільних застосунків і платформ соціальних мереж зосереджена на створенні інтуїтивно зрозумілого та захопливого досвіду, ставлячи потреби, уподобання та поведінку користувача в центр процесу проєктування. Врахування потреб користувачів гарантує, що застосунки та платформи будуть не лише функціональними, але й забезпечать безперебійний та приємний досвід користувача на різних пристроях та в різних контекстах.

Бізнес дедалі більше усвідомлює необхідність врахування відгуків стейкхолдерів у процесі проєктування,

що відображає відхід від традиційних практик проєктування. Одним із способів задоволення цієї потреби є впровадження дизайну, орієнтованого на користувача (user-centered design, UCD), – ітеративного підходу до проєктування, який враховує потреби, поведінку та уподобання користувача. UCD прагне позитивно впливати на взаємодію з користувачем, завойовуючи його лояльність для подальшого використання продукту.

Користувач знаходиться в центрі процесу проєктування, а його досвід враховується в рішеннях, де бізнес прагне узгодити бізнес-практики та вимоги стейкхолдерів для створення дизайну, який відповідає потребам кінцевого користувача. Врахування відгуків користувачів не лише покращує результат дизайну, але й поглиблює стосунки з користувачем. До найкращих практик впровадження підходу UCD відносяться такі.

Визначення бізнес-мети. Щоб дійти до фінального рішення проблеми, бізнесу необхідно визначити свої організаційні цілі. Вони вказують напрямок процесу проєктування. Які стратегічні цілі бізнесу? Які заплановані результати досягнення цих цілей (у короткостроковій та довгостроковій перспективі)? Як дизайн системи відображатиме ці цілі та підвищуватиме соціально-економічну та фінансову цінність бізнесу? Як користувач системи вписується у стратегічні цілі бізнесу та як система відображатиме цей взаємозв'язок? Потрібно також враховувати ринкові сили, які допоможуть сформувати дизайн системи. А саме, компанія повинна визначити свій цільовий ринок, а також цільових користувачів свого продукту/рішення та своїх конкурентів.

Розуміння користувачів та узгодження бізнес-цілей. Клієнт є рушійною силою дизайну, оскільки кінцеві користувачі зрештою підтримуватимуть кінцеве рішення. Розуміння користувача є важливим та основоположним для

підходу UCD. Це розуміння може включати потреби, здібності та обмеження користувача, кожне з яких може вплинути на здатність користувача повноцінно взаємодіяти з дизайном ІС. Наприклад, щоб задовольнити потреби користувачів, вебсайт може містити змінний розмір шрифту, варіанти кольорів або більші кнопки. Передбачення завдань користувача (поточних та майбутніх) є ключовим для розуміння труднощів, з якими користувачі можуть зіткнутися під час використання рішення. Для цього:

– *Проведіть ретельне дослідження.* Ініціюйте інтерв'ю, фокус-групи, опитування та інші види досліджень клієнтів, щоб отримати їх відповіді. Запросіть на зустріч користувачів, які можуть надати уявлення про різні ролі чи функції системи. Розгляньте можливість оцінювання того, хто використовуватиме вебсайт, як користувач почувається під час використання дизайну, а також його потреби та обмеження. Зберіть та проаналізуйте дані за допомогою аналітичних інструментів. Дізнайтеся якомога більше про користувачів, щоб створити продукт, який відповідає їхнім потребам.

– *Оцініть потреби та вимоги бізнесу.* Залучіть стейкхолдерів для кращого розуміння цілей, обмежень, нормативних вимог та бюджетних міркувань вашої компанії. Врахуйте дані, отримані від користувачів, які проходили опитування. Визначте показники, які будуть використовуватися для вимірювання успішності дизайну.

Створення дизайнерського рішення та його оцінювання. Створіть комплексне рішення, яке враховує відгуки користувачів. Дослідіть інструменти вебдизайну, такі як ескізи (wireframe), користувацькі історії, макети та діаграми, використовуючи бізнес-вимоги та відгуки стейкхолдерів. Тестуйте дизайн системи, відтворюючи користувацький досвід. Цей процес відповідає ітеративній

природі UCD, а саме циклічній практиці створення, вдосконалення та покращення проєкту. Обов'язково врахуйте та перевірте всі бізнес-вимоги та вимоги користувачів. Оцініть дизайн системи, використовуючи визначені показники для вимірювання успішності дизайну.

Продовжуйте вдосконалювати проєкт. Продовжуйте залучати користувачів до процесу проєктування, вдосконалюючи дизайн системи за потреби та оцінюючи зручність його використання.

Незважаючи на свою різноманітність, мобільні застосунки зазвичай належать до однієї з таких категорій.

Застосунки соціальних мереж. Вони дозволяють обмінюватися інформацією на рівні спільноти через комунікацію та співпрацю. Ці застосунки сприяють об'єднанню членів спільноти за допомогою спільного контенту, розширюючи соціальні мережі Facebook, TikTok та WhatsApp.

Ігри та розваги. В них творці контенту, маркетологи та художники співпрацюють, щоб створити приємний інтерактивний досвід користувача, який генерує дохід.

Стиль життя. Застосунки для стилю життя підтримують аспекти, пов'язані зі стилем життя користувача. Наприклад, користувачі, які дивляться телебачення (ТБ), тепер можуть отримати доступ до ТБ-застосунку, щоб дивитися пропущені шоу в запису та отримувати ексклюзивний контент. Сюди також відносяться мобільні застосунки для носимих пристроїв. Організації охорони здоров'я широко впроваджують послуги через мобільні технології завдяки їх простоті використання, зручності та зниженню адміністративних витрат для організації.

Продуктивність. Застосунки для підвищення продуктивності дозволяють користувачам ефективно керувати завданнями.

Новини та інформація. Вони дозволяють користувачам залишатися в курсі поточних подій, представляючи новини та інформацію в унікальних форматах, які цікавлять користувачів.

Утиліта. Такі застосунки можуть бути попередньо встановлені на мобільному пристрої та зазвичай виконують одну швидку функцію (наприклад, календар).

Сьогодні багатьом користувачам потрібен доступ до вебсайтів, ПЗ і застосунків на різних платформах. Наприклад, співробітникам, які працюють дистанційно, потрібно мати доступ до робочого ПЗ через планшет, а не через настільний комп'ютер в офісі. Користувачі соціальних мереж часто мають доступ до своїх облікових записів через смартфон, але вони також можуть хотіти взаємодіяти з мобільним застосунком через носимий пристрій (наприклад, смарт-годинник).

2.5 Нормативне регулювання подолання цифрової нерівності в ЄС

Оброблення та управління даними, професійні обов'язки, сприяння цифровій інклюзивності є критично важливими етичними міркуваннями в галузі ІТ. Їх ключові аспекти включають:

– *Збір, зберігання, оброблення та обмін даними:*

- *Збір:* підприємства повинні бути прозорими щодо характеру та мети збору даних, забезпечуючи повне інформування користувачів;

- *Зберігання та оброблення:* необхідні надійні заходи безпеки для запобігання несанкціонованому доступу та потенційному зловживанню, захищаючи конфіденційну інформацію;

- *Обмін:* практика обміну даними повинна пріоритезувати конфіденційність користувачів та

відповідати відповідним нормам, забезпечуючи повагу до інформації.

– *Професійні обов'язки ІТ-фахівців:*

- Захист даних користувачів та забезпечення конфіденційності;

- Сприяння доступності та зменшення цифрової нерівності;

- Підтримка справедливості та інклюзивності в цифровій сфері.

– *Дотримання професійних етичних кодексів:*

- Дотримання етичних кодексів таких організацій, як Асоціація обчислювальної техніки (АСМ) та Інститут інженерів електротехніки та електроніки (ІЕЕЕ);

- Діяльність за такими ключовими принципами, як повага до конфіденційності, уникнення шкоди, а також внесок у суспільство та добробут людини.

– *Цифрова інклюзивність та доступність:*

- Забезпечення рівного доступу до технологій, незалежно від соціального походження, географічного розташування чи фізичних можливостей;

- Розробка технологій з урахуванням доступності, корисних для різноманітних потреб користувачів, покращуючи як охоплення, так і користувацький досвід.

Забезпечення рівного доступу для всіх до технологічних досягнень є важливим соціальним питанням. Цифрова інклюзивність, яка передбачає забезпечення доступності ІТ-рішень для всіх, має глибокий вплив на скорочення цифрового розриву. Цифровий розрив стосується розриву між окремими особами, громадами чи країнами, які мають доступ до сучасних інформаційно-комунікаційних технологій, і тими, які його не мають. Цей розрив може проявлятися в різних формах, таких як відмінності в доступі до Інтернету, цифровій грамотності та наявності доступних пристроїв та послуг. Технологічні

підприємства несуть відповідальність за сприяння інклюзивному цифровому середовищу. Розробляючи доступні технології, вони повинні забезпечити, щоб переваги цифрових інновацій досягли всіх верств суспільства, і щоб користувачі могли ефективно використовувати ці рішення.

Це вимагає розробки технологій з урахуванням різноманітних потреб користувачів: від створення вебсайтів, доступних для людей з вадами зору чи слуху, до створення ПЗ, в якому легко орієнтуватися людям з когнітивними або моторними проблемами. Зобов'язання щодо доступності є наріжним каменем етичного розвитку галузі ІТ. Проєктування з урахуванням доступності не лише розширює базу користувачів, але й покращує загальний користувацький досвід.

Усунення цифрової нерівності є одним із пріоритетних напрямів політики ЄС в рамках формування єдиного цифрового простору. Законодавство ЄС спрямоване на забезпечення рівного доступу громадян до цифрової інфраструктури, сучасних інформаційно-комунікаційних технологій, цифрових послуг та розвитку цифрових компетентностей. Комплексний підхід поєднує нормативно-правові акти, стратегічні програми та фінансові інструменти, що дозволяють зменшувати соціальні, територіальні та економічні прояви цифрової нерівності.

Основою сучасної цифрової політики ЄС є програма «Цифрове десятиліття до 2030 року» (Digital Decade Policy Programme 2030), яка встановлює кількісні цілі щодо розвитку цифрової інфраструктури та цифрових навичок населення. Зокрема, передбачено забезпечення всіх домогосподарств високошвидкісним гігабітним інтернетом, повне покриття населених територій мережами 5G, а також досягнення базового рівня цифрових компетентностей щонайменше у 80 % дорослого населення ЄС. Особлива

увага приділяється подоланню цифрового розриву між міськими та сільськими територіями, а також між різними соціальними групами населення.

Важливим нормативним актом є Європейський кодекс електронних комунікацій (European Electronic Communications Code, Directive (EU) 2018/1972), який закріплює принцип універсального доступу до електронних комунікаційних послуг. Документ зобов'язує держави-члени ЄС забезпечувати доступність широкосмугового Інтернету та голосового зв'язку незалежно від місця проживання громадян, сприяючи розвитку телекомунікаційної інфраструктури у сільських, віддалених та економічно менш розвинених регіонах ЄС.

Одним із ключових механізмів подолання цифрової нерівності є Регламент про програму «Цифрова Європа» (Regulation (EU) 2021/694), який передбачає фінансування розвитку високопродуктивних обчислень, штучного інтелекту, кібербезпеки, цифрових навичок та широкого впровадження цифрових технологій у державному секторі й бізнесі. Особлива увага приділяється підтримці малих і середніх підприємств, закладів освіти та органів публічного управління.

Важливе значення має також Європейський акт про доступність (European Accessibility Act, Directive (EU) 2019/882). Його дія поширюється не лише на державний, а й на приватний сектор і встановлює обов'язкові вимоги щодо доступності цифрових продуктів та послуг, включаючи електронну комерцію, банківські послуги, електронні книги, транспортні сервіси, електронні комунікації, термінали самообслуговування та мобільні застосунки. Реалізація вимог цього акта спрямована на усунення цифрової дискримінації та створення інклюзивного цифрового середовища для всіх категорій користувачів.

Зменшенню цифрової нерівності сприяє також Директива (ЄС) 2016/2102 щодо доступності вебсайтів і мобільних застосунків органів державного сектору, яка зобов'язує державні установи забезпечувати відповідність своїх цифрових ресурсів міжнародним стандартам вебдоступності (Web Content Accessibility Guidelines, WCAG). Це дозволяє громадянам, незалежно від їх фізичних можливостей, отримувати рівний доступ до державних електронних послуг.

Важливу роль відіграє Регламент про єдиний цифровий ринок (Digital Services Act, Regulation (EU) 2022/2065), який спрямований на створення безпечного та прозорого цифрового середовища. Документ встановлює вимоги щодо прозорості роботи онлайн-платформ, захисту користувачів, протидії незаконному контенту та забезпечення рівного доступу до цифрових сервісів.

Окремим напрямом є підвищення рівня цифрових компетентностей населення. У межах Європейського порядку денного навичок (European Skills Agenda) та Плану дій щодо цифрової освіти (Digital Education Action Plan 2021-2027) реалізуються програми розвитку цифрової грамотності, перепідготовки працівників, підготовки фахівців у сфері ІТ та підтримки цифрової трансформації освіти. Це сприяє скороченню цифрового розриву між різними віковими, професійними та соціальними групами населення.

Таким чином, законодавство ЄС формує багаторівневу систему подолання цифрової нерівності, яка поєднує правові, економічні та організаційні механізми. Основними напрямками цієї політики є забезпечення універсального доступу до цифрової інфраструктури, розвиток цифрових навичок населення, створення доступних цифрових сервісів, підтримка цифрової інклюзії та усунення бар'єрів для повноцінної участі громадян у цифровому суспільстві.

Комплексний характер європейського законодавства дозволяє розглядати його як одну з найбільш розвинених моделей правового регулювання цифрової трансформації та скорочення цифрової нерівності у світі.

Контрольні запитання

1. Поясніть чому важливо вирішувати бізнес-проблеми?
2. Дайте визначення та наведіть приклади функціональних вимог.
3. Дайте визначення та наведіть приклади нефункціональних вимог.
4. Що показує діаграма прецедентів?
5. Що показує контекстна діаграма?
6. Які проблеми пов'язані з процесом збору вимог? Як їх вирішувати?
7. Опишіть сутність процесу глобальної розробки програмного забезпечення.
8. Які риси має гарний проєкт системи?
9. Які переваги та недоліки створення прототипу?
10. Опишіть основні завдання проєктування системи.
11. Чому важливий дизайн, орієнтований на користувача?
12. Наведіть найкращі практики впровадження підходу до дизайну, орієнтованого на користувача.
13. Наведіть класифікацію собільних застосунків.
14. Опишіть ключові аспекти етичних міркувань щодо оброблення та управління даними.
15. Опишіть ключові аспекти етичних міркувань щодо професійних обов'язків ІТ-фахівців.
16. Опишіть ключові аспекти етичних міркувань щодо сприяння цифровій інклюзивності.
17. Назвіть цілі цифрової політики ЄС.

18. Що передбачає Європейський акт про доступність (European Accessibility Act)?

19. Які вимоги встановлює Регламент про єдиний цифровий ринок (Digital Services Act)?

РОЗДІЛ 3

ОСНОВИ ПРОЄКТУВАННЯ МОБІЛЬНИХ ПРОГРАМНИХ СИСТЕМ

3.1 Мобільна економіка та її вплив на бізнес

Мобільна економіка є глобальною екосистемою, що охоплює апаратне забезпечення, програмні платформи, телекомунікаційні мережі, а також широкий спектр сервісів та застосунків, доступних через мобільні пристрої. Фундаментальною її ознакою є не стільки технологія, скільки фундаментальна зміна у поведінці споживачів та парадигмах ведення бізнесу. Смартфон перестав бути просто засобом комунікації, перетворившись на первинну точку доступу до інформації, послуг, розваг та комерції.

Для сучасної компанії ігнорування цього каналу взаємодії з клієнтом є стратегічним ризиком, що може призвести до втрати ринкових позицій. Економічний вплив цього явища є колосальним. Внесок мобільних технологій та сервісів у світовий ВВП обчислюється трильйонами доларів США, і ця цифра демонструє стабільне зростання. Ключовим рушієм цього зростання є так звана економіка застосунків (App Economy), що самостійно генерує сотні мільярдів доларів доходу через продажі застосунків, внутрішні покупки та мобільну рекламу. Цей новий економічний простір створює робочі місця, стимулює інновації та відкриває можливості для бізнесу будь якого масштабу, від глобальних корпорацій до локальних стартапів.

Проникнення мобільних технологій у всі сфери життя стало каталізатором тектонічних зсувів у традиційних бізнес-моделях. Спостерігається чіткий перехід від підходу, орієнтованого на настільні комп'ютери, до стратегії, де мобільний досвід є пріоритетним. Більше того, для багатьох нових компаній мобільна платформа є єдиним каналом взаємодії з клієнтами.

Ця еволюція змінила фундаментальні очікування споживачів. Сучасний клієнт вимагає негайного доступу до послуг, високого рівня персоналізації та релевантності пропозицій, що враховують його поточний контекст, наприклад, місцезнаходження. Бізнес, що не здатний задовольнити ці вимоги, програє у конкурентній боротьбі.

Яскравим прикладом трансформації є ринок пасажирських перевезень. Такі компанії як Uber, Bolt та український Uklon повністю змінили ринок пасажирських перевезень. Вони замінили застарілу модель з диспетчерськими службами на ефективну цифрову платформу, що в реальному часі поєднує водіїв та пасажирів. З економічної точки зору це призвело до зниження бар'єрів входу на ринок для водіїв, оптимізації логістики, впровадження динамічного ціноутворення та, що найважливіше, генерації величезних обсягів даних для аналізу попиту та пропозиції.

Схожі процеси відбулися у фінансовому секторі. Такі проекти як Monobank продемонстрували, що банківські послуги можуть існувати без фізичних відділень (необанкінг). Мобільний застосунок став повноцінним банком у кишені клієнта. Це дозволило радикально скоротити операційні витрати, пов'язані з утриманням мережі філій, та переспрямувати зекономлені ресурси на покращення продукту та залучення клієнтів. Мобільний банкінг також забезпечує значно вищий рівень залученості клієнтів через миттєві сповіщення, гейміфікацію та персоналізовані фінансові поради, що було неможливим у класичній моделі.

Сфера роздрібної торгівлі також зазнала глибоких змін. Мобільні застосунки великих ритейлерів, наприклад, Rozetka або Amazon, – це значно більше, ніж просто мобільна версія вебсайту. Вони функціонують як потужні інструменти утримання клієнтів, інтегруючи програми

лояльності, персоналізовані рекомендації на основі історії переглядів та покупок, а також спрощуючи процес оплати завдяки інтеграції з платіжними системами Apple Pay та Google Pay.

Роль мобільних застосунків не обмежується взаємодією з кінцевим споживачем. Для цілісного розуміння їх стратегічного значення необхідно класифікувати їх за типом цільової аудиторії та бізнес-завдань, які вони вирішують. Існує три основних типи застосунків: B2C, B2B та B2E. Застосунки типу Business to Consumer, або B2C, є найбільш видимою категорією. Їх основні стратегічні цілі включають створення прямого каналу продажів, посилення маркетингових комунікацій, підвищення впізнаваності бренду, надання клієнтської підтримки та збір цінних даних про поведінку споживачів. У цьому сегменті мобільний застосунок виступає як ключовий актив для побудови довгострокових відносин з клієнтом.

Наступний тип, Business to Business, або B2B, фокусується на оптимізації взаємодії між компаніями. Такі застосунки не завжди є публічними та часто розробляються для вирішення специфічних галузевих завдань. Прикладами можуть слугувати мобільні CRM системи, як Salesforce, що надають менеджерам з продажу миттєвий доступ до бази клієнтів та історії угод безпосередньо «в полі». Іншим прикладом є логістичні платформи, що дозволяють відстежувати вантажі в реальному часі, чи системи для управління закупівлями. Економічна цінність B2B застосунків полягає у підвищенні операційної ефективності, скороченні витрат, мінімізації помилок, пов'язаних з людським фактором, та покращенні прозорості бізнес-процесів.

Третя категорія, Business to Employee, або B2E, спрямована на внутрішню аудиторію компанії, тобто на її співробітників. Мета таких застосунків – це підвищення

продуктивності праці та корпоративної залученості. Це можуть бути внутрішні корпоративні месенджери, цифрові платформи для дистанційного навчання та адаптації нових співробітників, системи для управління відпустками, подання звітів про витрати або доступу до розрахункових листів. З точки зору бізнесу, інвестиції в B2E рішення окупаються через скорочення адміністративного навантаження на HR (Human Resources) та фінансові відділи, прискорення внутрішніх комунікацій та процесів, а також створення більш комфортного цифрового робочого середовища.

Поняття цифрової трансформації є значно ширшим, ніж просте впровадження нових технологій. Це фундаментальна перебудова бізнес-процесів, корпоративної культури та моделей створення цінності для клієнта. У цьому складному процесі мобільні застосунки відіграють роль не просто інструмента, а ключового каталізатора. Вони є головним інтерфейсом, через який відбувається взаємодія компанії з її зовнішнім та внутрішнім середовищем у новій цифровій реальності.

Однією з найважливіших функцій мобільного застосунку в рамках цифрової трансформації є збір даних. Кожен застосунок, встановлений на смартфоні, є своєрідним сенсором, що збирає унікальну інформацію про користувачі в подобирання, патерни поведінки, географічне переміщення та соціальний контекст. Аналіз великих даних дозволяє бізнесу приймати більш обґрунтовані стратегічні рішення, точно сегментувати аудиторію для маркетингових кампаній, персоналізувати продукт та передбачати майбутній попит. Таким чином, мобільний застосунок перетворюється з простого каналу комунікації на джерело стратегічної інформації.

Крім того, мобільні платформи відкривають шлях до створення абсолютно нових бізнес-моделей та джерел

доходу, які були неможливими раніше. Наприклад, виробник побутової техніки може запропонувати через мобільний застосунок додаткові послуги з обслуговування або платні підписки на розширений функціонал для своєї розумної техніки. Страхова компанія може розробити мобільний застосунок, що аналізує стиль водіння клієнта та пропонувати індивідуальні тарифи на страхування, заохочуючи безпечну поведінку на дорозі. Такі інновації дозволяють компаніям диверсифікувати доходи та створювати додаткову цінність для споживача.

Мобільна економіка перебуває у стані безперервної еволюції, рушієм якої є поява нових технологічних парадигм. Для економіста вміння не просто реагувати на зміни, а передбачати їх та оцінювати їхній потенційний вплив на бізнес є ключовою компетенцією. Розглянемо три основні тренди, що визначають майбутнє мобільної індустрії та створюють нові ринкові можливості.

Першим і найважливішим трендом є інтеграція штучного інтелекту (ШІ) та машинного навчання безпосередньо в мобільні застосунки. Цінність ШІ для бізнесу полягає не у створенні футуристичних роботів, а у його здатності до гіперперсоналізації та предиктивної аналітики. Сучасні мобільні пристрої оснащені потужними процесорами, здатними виконувати складні обчислення локально, що дозволяє застосункам аналізувати поведінку користувача в реальному часі. Це відкриває шлях до створення продуктів, що адаптуються до потреб кожного окремого клієнта. Наприклад, застосунок для електронної комерції може в реальному часі перебудовувати свою головну сторінку, пропонуючи товари, які з високою ймовірністю зацікавлять саме цього користувача. Фінансовий застосунок може використовувати машинне навчання для миттєвого виявлення шахрайських транзакцій. Економічний ефект від такої інтеграції є

прямим та вимірюваним: підвищення коефіцієнту конверсії, зростання середнього чеку, підвищення утримання клієнтів та зниження фінансових втрат.

Другим важливим трендом є розвиток Інтернету речей. Концепція Інтернету речей передбачає, що мільярди повсякденних пристроїв, від автомобілів та побутової техніки до медичних сенсорів та промислового обладнання, будуть підключені до мережі Інтернет. У цьому новому світі роль смартфона трансформується. Він перестає бути просто пристроєм для комунікації та споживання контенту, а перетворюється на універсальний пульт управління та центральну панель моніторингу для екосистеми підключених пристроїв. Це створює абсолютно нові бізнес-моделі. Виробники обладнання зможуть заробляти не стільки на одноразовому продажі фізичного товару, скільки на наданні платних цифрових послуг через мобільний застосунок. Наприклад, власник розумного холодильника може отримувати через застосунок персоналізовані рецепти та автоматично замовляти продукти. Для бізнесу це означає перехід від моделі одноразових продажів до моделі прогнозованого доходу від підписок.

Третім перспективним напрямком є доповнена реальність. Технологія доповненої реальності дозволяє накладати цифрову інформацію та віртуальні об'єкти на зображення реального світу, що транслюється через камеру смартфона. Потенціал цієї технології для бізнесу є величезним, особливо у сфері роздрібної торгівлі. Застосунки, що використовують доповнену реальність, дозволяють вирішити одну з головних проблем онлайн-шопінгу: неможливість протестувати товар до покупки. Наприклад, застосунок меблевого магазину ІКЕА Place, дозволяє покупцям побачити, як обраний диван чи стіл буде виглядати безпосередньо в їхній кімнаті. Косметичні бренди пропонують застосунки, що дозволяють віртуально

нанести макіяж. З економічної точки зору, такий досвід значно підвищує впевненість покупця у правильності покупки, що призводить до зростання конверсії та, що не менш важливо, до зниження кількості повернень товару.

Підсумовуючи, важливо наголосити, що розробка мобільного застосунку сьогодні це не технічне, а стратегічне бізнес-рішення. Для майбутніх бізнес-лідерів критично важливо розуміти не стільки аспекти програмування, скільки економічне підґрунтя, яке стоїть за мобільним програмним продуктом: його роль у бізнес-моделі компанії, його здатність генерувати дохід та оптимізувати витрати, його потенціал як інструменту для збору даних та побудови довгострокових відносин з клієнтами. Успіх у сучасній мобільній економіці визначається не наявністю мобільного застосунку як такого, а наявністю чіткої стратегії його використання для досягнення ключових бізнес-цілей компанії.

3.2 Проєктування мобільних програмних систем

У процесі сучасної розробки важливо спроектувати систему таким чином, щоб вона відповідала вимогам користувачів на різних платформах з майже однаковою функціональністю. Зокрема, розробники вебсистем повинні враховувати в процесі розробки правильне відображення контенту вебсистеми на мобільному пристрої.

Ефективні програмні системи повинні забезпечувати цілісний і безперервний користувацький досвід незалежно від типу пристрою чи програмної платформи. Це передбачає збереження однакової функціональності, логіки роботи та зручності взаємодії із системою під час використання її на різних пристроях, зокрема смартфонах, планшетах і персональних комп'ютерах, а також на різних операційних платформах, таких як Android, iOS та Windows. У деяких випадках користувачі цінують, коли одні

програми інтегровані в інші програми. Наприклад, коли користувач змінює своє зображення профілю в одному застосунку, воно автоматично оновлюється в іншому застосунку.

Розробникам також слід враховувати вплив глобалізації на дизайн мобільних програмних систем, як з точки зору міжнародних регуляторних стандартів, так і потреб користувачів. Це включає вибір мови інтерфейсу та включення елементів, які не є регіональними або специфічними для однієї культури чи етнічної групи.

Таким чином, проєктування систем з часом ускладнюється зі зміною технологій та розвитком потреб і вимог користувачів. Під час розробки мобільної програмної системи або соціальної платформи слід враховувати такі рекомендації.

Мета та завдання. Почніть з визначення мети та завдань мобільної програмної системи або платформи. Проведіть дослідження та дізнайтеся про останні тенденції в дизайні, які можуть підвищити зручність використання вашого продукту.

Простота. Прості функціональні дизайни з приємною візуалізацією збільшують конверсію. Гарний дизайн може включати кольорову палітру, індикатор рядка стану для візуального відображення прогресу користувача та спрямовування його до наступного кроку за допомогою підказок.

Надійність. Дизайн має бути надійним, тобто користувачі повинні мати можливість цілодобово отримувати доступ до продукту. Щоб підтримувати цю надійність, програмний продукт повинен відповідати стандартам швидкості та узгодженості. Важливим аспектом, який слід врахувати, є архітектура мережі передачі даних. Крім того, зверніть увагу на політики мобільних програмних систем і соціальних платформ, щоб

захиститися від викривлення інформації (неточної або неправдивої інформації) та дезінформації (неправдивої інформації, наданої з метою введення в оману).

Зручна навігація. Дизайн мобільних програмних систем є успішним лише тоді, коли має зручну навігацію. Користувачі повинні мати можливість переміщатися між функціями вебсистеми за допомогою кінчиків пальців, голосових команд або інших простих способів.

Сумісність з платформами. Враховуйте різні платформи під час розробки, щоб створити безшовний користувацький досвід. Це формує довіру у користувачів, оскільки вони отримують доступ до застосунку в будь-якому середовищі.

Інтеграція соціальних мереж. Оскільки користувачі стають все більше схильними ділитися інформацією з іншими користувачами у своїх спільнотах, інтеграція соціальних мереж стає важливою функцією, яка піднімає дизайн на новий рівень. Під час проєктування інтеграції мобільної програмної системи із соціальними мережами доцільно проаналізувати низку ключових аспектів. Зокрема, необхідно визначити, наскільки ефективно мобільна програмна система взаємодіє з найпоширенішими платформами соціальних мереж; чи існує потреба користувачів у спільній взаємодії, наприклад у порівнянні досягнень або обміні результатами через соціальні сервіси; яким чином така інтеграція сприятиме розширенню аудиторії та підвищенню рівня використання мобільної програмної системи; а також які типи push-сповіщень і повідомлень із соціальних платформ є доцільними для забезпечення зручного та ефективного користувацького досвіду.

Основними етапами розробки мобільної програмної системи є такі.

Розробка стратегії. Вона допоможе визначити напрямок та бачення дизайну та дозволить передбачити майбутні тенденції галузі. На цьому етапі необхідно проаналізувати низку стратегічно важливих аспектів, надавши відповіді на такі ключові питання: які стратегічні пріоритети визначають розвиток компанії; яким чином мобільна програмна система сприятиме досягненню бізнес-цілей компанії; а також яку бізнес-проблему має вирішити запропоноване дизайнерське рішення. Створення мобільної програмної системи може підвищити задоволеність клієнтів і розширити клієнтську базу. Процес розроблення ПЗ повинен бути спрямований на створення позитивного досвіду для цільових користувачів завдяки функціональності мобільної програмної системи. Створіть детальний план, включаючи аналіз проблеми, загальні цілі та те, як дизайн мобільної програмної системи зрештою відповідатиме стратегічним цілям компанії, включаючи дані для підтримки плану.

Інтерфейс користувача та користувацький досвід. Ці елементи працюють разом у мобільній програмній системі та повинні бути частиною розробки її дизайну. Користувацький інтерфейс (User Interface, UI) – це точка взаємодії між користувачем та програмним продуктом. Проектування з урахуванням UI передбачає врахування елементів та функцій, з якими може зіткнутися користувач, а також того, як дизайн відповідає потребам користувача. Користувацький досвід (User Experience, UE або UX) – це підхід до дизайну, який інтегрує досвід користувача з компанією, її послугами та продуктами. UX розглядає процес проектування застосунку з усіх аспектів взаємодії клієнта з послугами компанії. Цей процес вимагає глибокого розуміння користувача, його потреб та його взаємодії з системою, що розробляється. Подібно до Agile-

підходу, UX-підхід ставить потреби користувача на перше місце в процесі розробки.

Розробка, тестування та розгортання мобільної програмної системи. Беручи до уваги визначену раніше стратегію та план інтеграції підходу UI/UX в дизайн мобільної програмної системи, розпочніть її розроблення. Використовуйте історії користувачів, розкадровки та інші візуальні елементи й інструменти, щоб окреслити кроки користувача під час навігації мобільною програмною системою. На цьому етапі необхідно оцінити якість дизайну мобільної програмної системи, проаналізувавши низку ключових аспектів. Зокрема, слід визначити, чи забезпечує дизайн цілісність і узгодженість інтерфейсу; чи є основні функції достатньо доступними для користувачів; чи характеризується текстовий контент належною читабельністю; чи є навігація інтуїтивно зрозумілою та зручною; чи підтримується коректне функціонування системи на різних програмних платформах; а також чи відповідає дизайн потребам і очікуванням користувачів, забезпечуючи позитивний користувацький досвід та ефективну взаємодію із системою. Перегляньте ваш дизайн, щоб переконатися, що він відповідає бізнес-вимогам і вимогам користувача. Протестуйте дизайн, щоб виявити помилки або збої, які можуть вплинути на його використання, функціональність та продуктивність (наприклад, час завантаження вебсторінки). Розгляньте можливість розширення тестування на фокус-групу (групу користувачів, зібраних для обговорення своїх думок щодо програмного продукту, оскільки їхнє розуміння може прояснити майбутнє використання продукту). Після перевірки мобільного дизайну розгорніть мобільну програмну систему та протестуйте її.

Підтримка мобільної програмної системи та моніторинг продуктивності. Продовжуйте моніторити

мобільну програмну систему, щоб оцінити зручність її використання. Відстежуйте відгуки користувачів та показники продуктивності, використовуючи отримані дані для оновлення майбутніх ітерацій дизайну.

3.3 Аналіз ринку та валідація бізнес-ідеї

В основі будь якого успішної мобільної програмної системи лежить вирішення конкретної проблеми або задоволення значущої потреби певної групи людей. Поширена помилка багатьох підприємців-початківців полягає у підході, коли команда розробляє технологічно цікавий програмний продукт, який, однак, не знаходить попиту на ринку. Економічні наслідки такого підходу є руйнівними, оскільки це призводить до марної витрати фінансових, часових та людських ресурсів. Тому процес створення мобільної програмної системи повинен починатися не з ідеї як такої, а з ідентифікації та глибокого розуміння бізнес-проблеми.

Джерела для генерації життєздатних бізнес-ідей є різноманітними. Часто вони виникають з особистого досвіду засновників компанії, які стикаються з певною незручністю або неефективністю у своєму повсякденному житті чи професійній діяльності та прагнуть її виправити. Іншим потужним джерелом є аналіз існуючих ринків з метою виявлення незадоволених потреб або недостатньо обслуговуваних клієнтських сегментів. Часто існуючі компанії пропонують універсальні рішення, ігноруючи специфічні запити вузьких ніш. Також перспективні бізнес-ідеї можуть народжуватися на перетині технологічних трендів, таких як ШІ, Інтернет речей або доповнена реальність, та соціальних змін, наприклад, зростання популярності дистанційної роботи або підвищення уваги до ментального здоров'я.

Первинне оцінювання бізнес-ідеї має ґрунтуватися на концепції відповідності проблеми та рішення (Problem–Solution Fit). На цьому етапі основна увага зосереджується не на технічній можливості реалізації продукту, а на визначенні того, чи здатне запропоноване рішення ефективно усунути проблему, яка є достатньо важливою для потенційних користувачів. Ключовим є питання не «Чи можемо ми створити цей продукт?», а «Чи вирішує запропонований продукт настільки значущу проблему, що користувачі будуть готові інвестувати в його використання свій час і кошти?».

Економічна значущість бізнес-проблеми визначається низкою чинників, серед яких її поширеність, інтенсивність прояву та готовність потенційних споживачів оплачувати рішення, що забезпечує її усунення. Саме оцінювання цих характеристик дає змогу зробити обґрунтований висновок щодо ринкових перспектив майбутнього продукту та доцільності його подальшого розроблення.

Після формулювання початкової гіпотези щодо бізнес-проблеми та її можливого вирішення наступним кроком є проведення ретельного аналізу ринку. Цей процес дозволяє об'єктивно оцінити потенціал бізнес-ідеї та уникнути виходу на ринок, що не існує або є перенасиченим. Одним з перших завдань є визначення розміру ринку за допомогою таких метрик як TAM, SAM та SOM. Total Addressable Market або TAM – це загальний обсяг ринку для певного продукту чи послуги, що відображає максимальний можливий дохід. Serviceable Available Market або SAM – це та частина TAM, яка є географічно та операційно доступною для компанії. Наприклад, для сервісу доставки їжі TAM може бути глобальним ринком, тоді як SAM обмежуватиметься містами, де компанія планує працювати. Нарешті, Serviceable Obtainable Market або SOM – це та частка SAM, яку компанія реально може захопити,

враховуючи конкуренцію, власні ресурси та маркетингову стратегію. Саме показник SOM є найбільш важливим для фінансового планування та залучення інвестицій.

Не менш важливим є глибокий аналіз конкурентного середовища. Конкуренти це не лише загроза, але й безцінне джерело інформації. Необхідно ідентифікувати як прямих конкурентів, що пропонують аналогічний продукт, так і непрямих, які вирішують ту саму проблему клієнта іншими засобами. Аналіз їхніх продуктів, цінових моделей, маркетингових стратегій та відгуків користувачів в магазинах застосунків дозволяє виявити їхні сильні та слабкі сторони. Скарги користувачів на функціонал чи сервіс конкурентів можуть вказати на вільну нішу або можливість для створення продукту з кращим користувацьким досвідом.

Фінальним елементом аналізу є чітке визначення цільової аудиторії. Спроба створити продукт для всіх зазвичай призводить до створення продукту, який по справжньому не потрібен нікому. Ефективний підхід полягає у створенні деталізованого портрета ідеального клієнта, або персони. Такий портрет повинен включати не лише демографічні характеристики, як вік, стать та дохід, але й психографічні, такі як стиль життя, цінності, мотивація та, що найголовніше, бізнес-проблеми, які спонукають клієнта шукати рішення.

Традиційні бізнес-плани є громіздкими документами, що погано адаптовані до динамічних умов сучасного ринку, особливо на ранніх стадіях розвитку проекту. Більш гнучким та ефективним інструментом для стартапів є Lean Canvas (рис. 3.1).

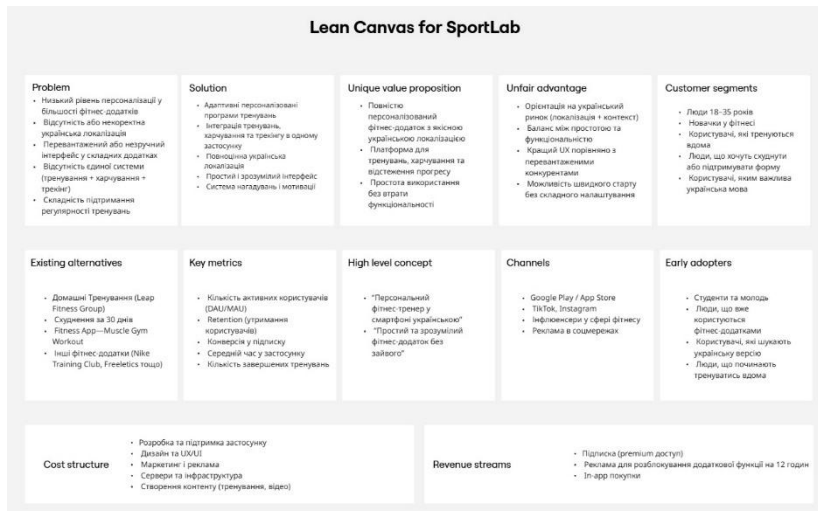


Рисунок 3.1 – Lean Canvas для мобільної програмної системи SportLab

Lean Canvas – це односторінковий шаблон, що дозволяє швидко описати, проаналізувати та презентувати бізнес-модель, фокусуючись на ключових гіпотезах, які потребують перевірки. Lean Canvas складається з дев'яти основних блоків.

Центральними елементами є блоки «Проблема» (Problem) та «Клієнтські сегменти» (Customer segments). Тут необхідно чітко сформулювати від однієї до трьох ключових проблем, з якими стикаються визначені раніше цільові сегменти вашого програмного продукту.

Навпроти них розташований блок «Унікальна ціннісна пропозиція» (Unique value proposition). Це концентроване, зрозуміле та переконливе повідомлення, що пояснює, чому ваш продукт є кращим вибором для клієнта і чим він відрізняється від конкурентів. У цьому блоці необхідно визначити та обґрунтувати унікальну ціннісну пропозицію

продукту або послуги, надавши відповідь на ключове питання: «Які конкурентні переваги зумовлюють вибір клієнтом саме вашого рішення?».

Блок «Рішення» (Solution) містить короткий опис ключових функцій мобільної програмної системи, кожна з яких повинна безпосередньо вирішувати одну з зазначених вище проблем.

Блок «Канали» (Channels) описує шляхи, якими унікальна ціннісна пропозиція буде донесена до клієнтських сегментів. Це можуть бути магазин застосунків Google Play, соціальні мережі, контекстна реклама чи контент-маркетинг.

Два нижні блоки, «Структура витрат» (Cost structure) та «Потоки доходів» (Revenue streams), описують фінансову складову бізнес-моделі. Перший блок перелічує основні операційні та капітальні витрати, необхідні для функціонування бізнесу, такі як розробка, маркетинг, заробітна плата та серверна інфраструктура. Другий блок описує, яким чином компанія буде генерувати дохід: через платні завантаження, підписки, внутрішні покупки, рекламу чи інші моделі монетизації.

Блок «Ключові метрики» (Key metrics) визначає набір показників, за допомогою яких буде вимірюватися успішність бізнесу. Це повинні бути не так звані метрики марнославства (наприклад, загальна кількість завантажень), а дієві показники, що відображають реальну залученість користувачів, наприклад, коефіцієнт утримання, вартість залучення клієнта та його пожиттєва цінність.

Останній блок, «Нечесна перевага» (Unfair advantage), є найскладнішим для заповнення. Це унікальний актив, який неможливо легко скопіювати або придбати конкурентам. Це може бути ексклюзивний доступ до даних, сильний бренд, ефект мережі, унікальна технологія або ключові партнерства.

Таким чином, процес переходу від бізнес-ідеї до життєздатної бізнес-моделі є систематичною та ітераційною роботою. Він вимагає глибокого аналізу ринку, чіткого визначення проблеми та аудиторії, а також структурування всіх ключових бізнес-гіпотез за допомогою такого інструменту, як Lean Canvas. Важливо розуміти, що Lean Canvas – це не статичний документ, а динамічний інструмент. Кожен його блок є гіпотезою, яку необхідно перевірити через експерименти та зворотний зв'язок від реальних користувачів. Такий підхід, заснований на даних, дозволяє мінімізувати ризики та суттєво підвищити шанси на створення комерційно успішного програмного продукту.

3.4 Основи UI/UX дизайну для бізнес-застосунків

У контексті розробки мобільних програмних систем поняття дизайну часто помилково зводять лише до його візуальної складової. Однак для бізнесу дизайн є стратегічною дисципліною, що спрямована на вирішення бізнес-проблем користувача задля досягнення конкретних комерційних цілей. Це інвестиція, рентабельність якої можна і необхідно вимірювати.

Щоб зрозуміти економічну цінність дизайну, вкрай важливо розмежовувати два його фундаментальні компоненти: досвід користувача, або User Experience (UX), та інтерфейс користувача, або User Interface (UI). Ці поняття є тісно пов'язаними, проте не тотожними. UX – це сукупність вражень, емоцій та відчуттів, які отримує людина під час взаємодії з мобільним застосунком. UX-дизайн фокусується на логіці, структурі та ефективності програмного продукту. Його головне завдання – зробити продукт корисним, зручним та цінним для користувача. Якщо уявити мобільну програмну систему як фізичний супермаркет, то UX-дизайн – це його архітектурне планування: наскільки логічно розташовані відділи, чи

легко знайти потрібний товар, чи є перешкоди на шляху покупця, наскільки швидким та зрозумілим є процес оплати на касі. Мета UX-архітектора полягає у проектуванні оптимального шляху клієнта, що мінімізує його зусилля та максимізує ймовірність досягнення цілі, наприклад, здійснення покупки. З точки зору бізнесу, якісний UX-дизайн забезпечує корисність продукту, тобто його здатність вирішувати реальну проблему, та його юзабіліті, тобто легкість у використанні. Продукт, що потребує довгого вивчення, має високий ризик бути видаленим користувачем одразу після першого знайомства.

Натомість UI – це конкретні візуальні та інтерактивні елементи, з якими користувач безпосередньо контактує. Це графічне оформлення екранів, кнопки, іконки, шрифти, кольорова палітра та анімації. Повертаючись до аналогії з супермаркетом, UI – це його внутрішнє оздоблення: дизайн вивісок, цінників, брендування продукції, зовнішній вигляд касового апарату та навіть форма візка для покупок.

Завдання UI-дизайнера полягає у тому, щоб зробити функціональну структуру, спроектовану UX-дизайнером, естетично привабливою, зрозумілою та зручною для взаємодії. Таким чином, UI є частиною UX, але не навпаки. Прекрасний інтерфейс не зможе врятувати продукт, який є нелогічним, заплутаним та не вирішує жодної проблеми користувача. І навпаки, навіть геніальна ідея з бездоганною логікою може провалитися на ринку, якщо її візуальна реалізація буде відштовхувати користувачів. Для комерційного успіху необхідна синергія обох компонентів.

Інвестиції в якісний UI/UX дизайн мають прямий та вимірюваний вплив на ключові показники ефективності (Key Performance Indicators, KPI) будь якого мобільного бізнесу. Одним з найважливіших KPI є утримання клієнтів, або *customer retention*. Залучення нового клієнта, за даними численних досліджень, коштує в рази дорожче, ніж

утримання існуючого. Позитивний досвід взаємодії з мобільною програмною системою є одним з головних факторів, що спонукає користувачів повертатися знову і знову. Інтуїтивно зрозумілий, стабільний та приємний у використанні продукт формує звичку та емоційну прихильність, що є основою довгострокової лояльності. Яскравим прикладом на українському ринку є успіх банківських застосунків Monobank та Privat24, для яких саме виняткова зручність та простота використання стали ключовими конкурентними перевагами та драйверами утримання мільйонів клієнтів.

Ще одним критично важливим показником є коефіцієнт конверсії. Конверсія – це виконання користувачем цільової дії, заради якої створювався застосунок: покупка товару, оформлення підписки, реєстрація на подію тощо. Якісний дизайн безпосередньо впливає на цей показник, усуваючи будь які перешкоди та тертя на шляху користувача до цілі. Кожен зайвий крок, кожне незрозуміле поле у формі реєстрації, кожна неочевидна кнопка – це потенційна точка втрати клієнта. Проєктування простого та зрозумілого процесу, наприклад, оформлення замовлення, може підвищити конверсію на десятки відсотків, що прямо трансформується у збільшення доходу.

Окрім впливу на доходи, продуманий дизайн також сприяє зниженню операційних витрат. Мобільна програмна система, логіку роботи якої користувачі розуміють без додаткових інструкцій, генерує значно менше звернень до служби підтримки. Це дозволяє компанії утримувати менший штат служби підтримки користувачів продукту та економити значні кошти. Дизайн у цьому випадку виконує функцію превентивної підтримки клієнтів.

Нарешті, у сучасному перенасиченому ринку мобільних програмних систем дизайн стає потужним інструментом диференціації та формування бренду. Коли

функціонал та ціни конкурентів є приблизно однаковими, користувачі обирають той продукт, взаємодія з яким приносить більше задоволення. Послідовний та професійний візуальний стиль формує у споживача відчуття надійності та довіри до компанії.

Ігнорування фундаментальних принципів дизайну неминуче призводить до негативних економічних наслідків. Однією з найдорожчих помилок є розробка продукту на основі припущень власників бізнесу, а не на основі досліджень реальних потреб та поведінки цільової аудиторії. Створення функціоналу, який нікому не потрібен, – це прямий шлях до провалу продукту та втрати всіх інвестованих у розробку коштів.

Іншою розповсюдженою проблемою є інформаційне перевантаження інтерфейсу, відоме як *feature creep*. У прагненні додати якомога більше можливостей, розробники часто створюють складні та заплутані екрани, що викликають у користувача відчуття розгубленості та когнітивного перенавантаження. У результаті людина не може виконати навіть базові завдання та з високою ймовірністю назавжди залишає застосунок. Економічний наслідок – це втрата потенційного клієнта, на залучення якого вже були витрачені маркетингові кошти.

Непослідовність у дизайні також є серйозною проблемою. Коли в різних частинах мобільної програмної системи використовуються різні стилі кнопок, шрифти, іконки та логіка взаємодії, це змушує користувача кожного разу заново вчитися працювати з інтерфейсом. Це не лише дратує, але й підриває довіру до програмного продукту, створюючи враження непрофесійності та хаотичності. Бізнес ризикує втратити довіру, яка є основою будь-яких комерційних відносин.

Окремо варто відзначити важливість технічної продуктивності як частини загального досвіду користувача.

Повільне завантаження екранів, затримки при виконанні дій, збої та помилки в роботі мобільної програмної системи є абсолютно неприйнятними для сучасного споживача. Згідно з дослідженнями Google, навіть кілька секунд затримки при завантаженні мобільної сторінки драматично збільшують показник відмов. Кожна секунда очікування – це потенційно втрачений дохід.

Отже, дизайн мобільної програмної системи є невід’ємною частиною бізнес-стратегії. Це потужний інструмент, що безпосередньо впливає на прибутковість, конкурентоспроможність та довгострокову стійкість компанії в умовах цифрової економіки. Розуміння його основних принципів та економічного обґрунтування є необхідною компетенцією для сучасного економіста.

3.5 Процес створення прототипу: від схеми до інтерактивного макету

Прототипування в контексті розробки мобільних програмних систем є процесом створення спрощених моделей майбутнього продукту. Це не просто технічний етап, а потужний інструмент стратегічного управління ризиками та оптимізації інвестицій. Основна економічна цінність прототипування полягає в його здатності виявляти та виправляти концептуальні, логічні та дизайнерські помилки на ранніх стадіях, коли вартість внесення змін є мінімальною.

Уявімо процес створення застосунку як будівництво споруди. Початок розробки програмного коду без попереднього прототипування є аналогічним до початку кладки цегли без затвердженого архітектурного плану. Будь яка зміна в проєкті на етапі будівництва, наприклад, переміщення несучої стіни, є надзвичайно дорогою та трудомісткою. У світі ПЗ внесення суттєвих змін у вже написаний код може коштувати в десятки, а іноді й у сотні

разів дорожче, ніж виправлення тієї самої помилки на рівні схематичного макету.

Прототипування дозволяє бізнесу провести валідацію ідеї з мінімальними фінансовими вкладеннями. Замість того, щоб інвестувати величезні кошти у розробку повноцінного продукту, який може виявитися непотрібним ринку, компанія може створити інтерактивний прототип із значно меншим бюджетом. Цей прототип можна продемонструвати потенційним інвесторам для залучення фінансування, показати фокус-групам для отримання зворотного зв'язку або використовувати для проведення перших користувацьких тестів. Таким чином, прототип виступає як інструмент для перевірки ключових бізнес-гіпотез, що лежать в основі продукту, до моменту здійснення значних капіталовкладень. Крім того, наявність детального прототипу суттєво покращує комунікацію всередині команди та з зовнішніми підрядниками. Він слугує єдиним та однозначним джерелом істини для менеджерів, дизайнерів та розробників, усуваючи двозначність та неправильне тлумачення вимог. Це безпосередньо впливає на точність оцінки термінів та бюджету розробки, знижуючи ризик непередбачуваних витрат та зриву дедлайнів.

Першим етапом у процесі візуалізації є створення низькорівневих схем, або вайрфреймів (*wireframe*) (рис. 3.2).

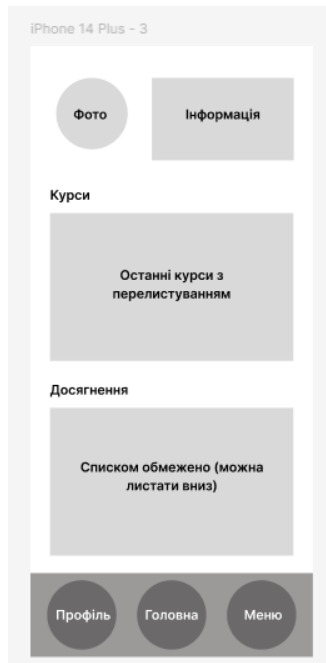


Рисунок 3.2 – Вайрфрейм екрану профілю користувача

Вайрфрейм – це схематичне, монохромне зображення екрану застосунку, яке фокусується виключно на структурі, ієрархії інформації та розташуванні ключових функціональних елементів, таких як кнопки, поля вводу та навігаційні панелі. На цьому етапі свідомо ігноруються всі візуальні аспекти: кольори, шрифти, зображення та іконки. Головна мета вайрфрейму полягає у тому, щоб визначити скелет майбутнього інтерфейсу та спроектувати логіку користувацьких сценаріїв.

З точки зору бізнесу, головна перевага вайрфреймів – це їхня швидкість та низька вартість створення. Їх можна малювати від руки на папері або створювати за допомогою простих онлайн-інструментів. Ця швидкість дозволяє

команді генерувати та обговорювати велику кількість варіантів компонування екранів за короткий проміжок часу.

Фокусування виключно на структурі дозволяє уникнути суб'єктивних дискусій про візуальний стиль та зосередитися на фундаментальних питаннях юзабіліті. Наприклад, чи зрозуміло користувачеві, яка дія є головною на цьому екрані? Чи логічно розташовані елементи навігації? Чи не перевантажений екран зайвою інформацією? Відповіді на ці питання на етапі створення вайрфреймів дозволяють виявити та виправити ключові недоліки у користувацьких потоках (*user flows*), ще до того, як буде написаний перший рядок коду чи створено перший піксель фінального дизайну.

Вайрфрейми є ефективним інструментом для узгодження бачення продукту між усіма стейкхолдерами. Вони дозволяють візуалізувати функціональні вимоги, що зазвичай описуються у текстових документах, роблячи їх більш зрозумілими та предметними для обговорення. Це допомагає переконатися, що і замовник, і команда розробки мають однакове уявлення про те, як буде працювати майбутня мобільна програмна система.

Після того, як структура та логіка мобільної програмної системи визначені та затверджені на рівні вайрфреймів, наступним кроком є створення високодеталізованого прототипу, який також називають інтерактивним макетом або клікабельним прототипом (рис. 3.3).

Цей тип прототипу є повнокольоровим, піксельно точним відображенням фінального продукту. Він включає в себе всі елементи UI-дизайну: фірмові кольори, типографіку, іконки, зображення та анімації. Більше того, такий прототип є інтерактивним.



Рисунок 3.3 – Клікабельний прототип

За допомогою спеціалізованих інструментів, таких як Figma, екрани пов'язуються між собою, що дозволяє імітувати реальну роботу мобільної програмної системи. Користувач може натискати на кнопки, переходити між екранами, відкривати меню та взаємодіяти з елементами інтерфейсу так, ніби він тримає в руках готовий продукт.

Головна бізнес-цінність клікабельного прототипу полягає в можливості провести реалістичне тестування користувацького досвіду до початку дороговартісної розробки. Компанія може надати прототип представникам своєї цільової аудиторії та спостерігати за тим, як вони виконують ключові завдання. Таке тестування дозволяє виявити проблеми, які були неочевидними на етапі статичних схем. Наприклад, чи помічають користувачі важливу кнопку? Чи розуміють вони значення певної іконки? Де у них виникають труднощі або вагання?

Отримання відповідей на ці питання дозволяє внести необхідні корективи в дизайн та логіку продукту, що значно підвищує шанси на його ринковий успіх.

Клікабельний прототип є також незамінним інструментом для презентації проєкту. На відміну від статичних зображень чи текстових описів, він дозволяє інвесторам, керівництву або потенційним клієнтам протестувати майбутній продукт. Це робить презентацію значно більш переконливою та ефективною. Для команди розробників високодеталізований прототип слугує вичерпним технічним завданням. Він усуває будь яку невизначеність щодо того, як повинен виглядати та поводитися кожен елемент інтерфейсу. Це мінімізує кількість питань до дизайнерів та менеджерів, прискорює процес розробки та гарантує, що фінальний продукт буде максимально відповідати початковому задуму.

Таким чином, процес прототипування є логічною послідовністю етапів, кожен з яких має свою унікальну мету та економічну доцільність. Перехід від низькорівневих схем до високодеталізованих інтерактивних макетів дозволяє компанії ітераційно та з мінімальними ризиками пройти шлях від абстрактної ідеї до чіткого та перевіреного бачення майбутньої мобільної програмної системи, створюючи міцний фундамент для подальшої розробки та комерційного успіху.

3.6 Основи нативного розроблення для Android

Вибір технологічної платформи для розроблення мобільної програмної системи є одним з ключових стратегічних рішень, яке приймає компанія. Це рішення має довгострокові фінансові та операційні наслідки, впливаючи на вартість розробки, швидкість виходу на ринок, якість кінцевого продукту та можливості його подальшого розвитку. Тому розуміння особливостей основних

мобільних екосистем є необхідною компетенцією для сучасного економіста.

Екосистема Android є домінуючою на світовому ринку мобільних операційних систем (ОС). Згідно з аналітичними даними, частка Android на глобальному ринку смартфонів стабільно перевищує 70%, а в деяких країнах, що розвиваються, вона сягає ще вищих показників. Для бізнесу ці цифри означають доступ до величезної потенційної аудиторії. Стратегія бізнесу, орієнтована на платформу Android, дозволяє охопити максимально широкі сегменти споживачів, включаючи ті, для яких пристрої компанії Apple є фінансово недоступними.

Фундаментальною характеристикою екосистеми Android є її відкритість. ОС Android базується на ядрі Linux та має відкритий вихідний код, що дозволяє будь-якому виробнику створювати власні пристрої на її основі. З одного боку, це є значною перевагою, оскільки стимулює конкуренцію та призводить до появи на ринку широкого асортименту пристроїв у всіх цінових категоріях – від бюджетних до флагманських, що дає бізнесу гнучкість у виборі цільової аудиторії.

З іншого боку, ця відкритість породжує явище, відоме як фрагментація. На ринку одночасно існують сотні моделей пристроїв від десятків виробників, з різними розмірами екранів, технічними характеристиками та, що найважливіше, різними версіями ОС Android. Для бізнесу фрагментація є суттєвим економічним викликом. Вона означає, що процес розробки та тестування мобільної програмної системи ускладнюється та здорожчується, оскільки необхідно забезпечити її коректну роботу на максимально можливому парку пристроїв. Компанія змушена приймати стратегічне рішення щодо того, які версії ОС Android та які типи пристроїв підтримувати,

балансуючи між бажанням охопити більшу аудиторію та необхідністю контролювати витрати.

Центральним елементом екосистеми Android з точки зору розповсюдження мобільних застосунків є магазин Google Play. Він функціонує як глобальний цифровий ринок, що надає компаніям доступ до мільярдів користувачів. Процес публікації мобільних застосунків у Google Play є відносно швидким та менш суворим у порівнянні з App Store від Apple. Це дає бізнесу важливу перевагу, а саме, швидкість виходу на ринок. Можливість оперативно публікувати оновлення та виправлення дозволяє швидше реагувати на потреби ринку та відгуки користувачів.

Нативне розроблення – це створення мобільної програмної системи спеціально для однієї конкретної ОС з використанням офіційних інструментів та мов програмування, які надає власник платформи. Для ОС Android це означає використання мови програмування Java або Kotlin та середовища розробки Android Studio. Такий підхід є найдорожчим, але він пропонує низку унікальних переваг, які можуть бути критично важливими для досягнення бізнес цілей.

Головною перевагою нативного розроблення є максимальна продуктивність та якість користувацького досвіду. Нативний код виконується безпосередньо ОС без будь яких проміжних шарів, що забезпечує найвищу швидкість роботи, плавність анімацій та миттєвий відгук інтерфейсу. З економічної точки зору, висока продуктивність безпосередньо впливає на задоволеність та утримання користувачів. Повільна мобільна програмна система, що працює із затримками, викликає роздратування та з високою ймовірністю буде видалена. Інвестиції у нативне розроблення – це інвестиції у преміальну якість продукту та, як наслідок, у лояльність клієнтів.

Другою важливою перевагою є повний та негайний доступ до всіх апаратних та програмних можливостей пристрою. Нативні мобільні програмні системи можуть без обмежень використовувати найновіші функції камери смартфона, модуль NFC для безконтактних платежів, біометричний сенсор, Bluetooth та інші специфічні компоненти. Це дозволяє бізнесу створювати інноваційні та унікальні функції, які неможливо або вкрай складно реалізувати за допомогою інших технологій. Для компаній, чия бізнес-модель базується на унікальному функціоналі, що тісно пов'язаний з можливостями смартфона, нативний підхід є єдиним можливим вибором для збереження конкурентної переваги.

Крім того, нативні мобільні програмні системи зазвичай є більш надійними та безпечними. Пряма інтеграція з ОС та використання офіційних інструментів розробки мінімізують ризики виникнення помилок та вразливостей. Для бізнесу у сферах, що вимагають високого рівня довіри, таких як фінансові технології, охорона здоров'я чи корпоративні сервіси, безпека є не просто перевагою, а фундаментальною вимогою. Нативна розробка дозволяє забезпечити найвищий рівень захисту даних та стабільності роботи мобільної програмної системи.

Нарешті, нативні мобільні програмні системи ідеально відповідають візуальним та функціональним стандартам своєї платформи. Для Android – це принципи Material Design, розроблені Google. Коли мобільна програмна система виглядає та поводить ся так, як очікує користувач, це значно знижує поріг входження та робить взаємодію з нею інтуїтивно зрозумілою. Це прискорює процес адаптації користувачів та підвищує загальну зручність використання програмного продукту. Попри всі свої переваги, нативний підхід має один суттєвий недолік, який є вирішальним для

багатьох компаній, – це висока вартість та тривалий час розробки.

Якщо бізнес планує бути присутнім як на платформі Android, так і на платформі iOS, вибір нативного шляху означає необхідність фінансування двох окремих, паралельних проєктів. Це вимагає утримання двох команд розробників з різними компетенціями, створення та підтримки двох незалежних кодових баз, а також проведення подвійного циклу тестування для кожної нової функції чи оновлення. Фактично, це подвоює бюджет на розробку та значно збільшує загальний час виходу продукту на обидва ринки. Для стартапів з обмеженим фінансуванням або для проєктів, де швидкість є ключовим фактором успіху, такі витрати можуть бути неприйнятними.

Додаткові витрати пов'язані з довгостроковою підтримкою продукту. Загальна вартість володіння (Total Cost of Ownership, TCO) для двох нативних мобільних програмних систем є значно вищою. Кожне оновлення ОС, кожне виправлення помилки, кожна нова функція потребують подвійних зусиль та ресурсів. Це створює постійне операційне навантаження на компанію. Ринок праці також відіграє свою роль. Пошук та утримання висококваліфікованих нативних розробників під конкретну платформу може бути складним та дорогим завданням.

Отже, вибір нативної розробки для Android є складним стратегічним компромісом. Цей підхід є виправданим, коли бізнес-пріоритети включають безкомпромісну якість, максимальну продуктивність та унікальний функціонал, що тісно інтегрований з апаратними можливостями мобільних пристроїв. Проте компанія повинна бути готовою до значних початкових інвестицій та високих довгострокових витрат на підтримку продукту. Роль менеджменту полягає в тому, щоб тверезо оцінити ці економічні фактори та обрати

шлях, що найкращим чином відповідає фінансовим можливостям та стратегічним цілям бізнесу.

Для ефективного управління процесом створення мобільної програмної системи менеджеру продукту не потрібно володіти навичками програмування. Однак, розуміння базових архітектурних компонентів, з яких складається мобільна програмна система, є критично важливим. Таке розуміння дозволяє чіткіше формулювати вимоги для команди розробників, адекватніше оцінювати складність та вартість реалізації нових функцій, а також приймати більш обґрунтовані стратегічні рішення щодо розвитку продукту.

Уявімо, що мобільна програмна система – це велика офісна будівля. Кожен елемент цієї будівлі має своє призначення та взаємодіє з іншими, створюючи єдину функціональну систему. Розглянемо три фундаментальні будівельні блоки будь-якого Android застосунку: Activity, Layouts та Intent, розглядаючи їх як приміщення, їхні креслення та внутрішні комунікації нашої уявної будівлі.

Ключовим компонентом, з яким безпосередньо взаємодіє користувач, є Activity. У нашій аналогії, Activity – це окреме приміщення або функціональна зона всередині офісної будівлі. Кожне приміщення має своє чітке призначення та надає інструменти для виконання конкретного завдання. Наприклад, екран входу та реєстрації – це приймальня на першому поверсі (одна Activity). Головний екран зі стрічкою новин – це великий робочий простір або шоурум (друга Activity). Екран профілю користувача – це персональний кабінет (третя Activity). З точки зору бізнесу, кожна Activity репрезентує один логічний крок у користувацькому сценарії. Коли менеджер продукту (*product manager*) планує новий функціонал, він мислить саме в категоріях таких екранів та завдань, які на них вирішуються.

Важливою концепцією, пов'язаною з Activity, є життєвий цикл. Це набір станів, через які проходить екран від моменту його появи до моменту зникнення. Уявіть, що відбувається з офісним приміщенням: його відкривають (користувач заходить на екран), у ньому починається активна робота (користувач взаємодіє з елементами), яка може бути призупинена (користувачеві надходить телефонний дзвінок, що перекриває мобільний застосунок), і нарешті приміщення закривають (користувач переходить на інший екран або виходить із мобільного застосунку).

Розуміння цього циклу має пряме економічне значення. Наприклад, якщо користувач знаходиться на екрані оформлення замовлення (біля каси) і його процес переривається дзвінком, бізнес повинен прийняти рішення: чи повинні дані, введені у кошик, зберігатися? Якщо так, то як довго? Неправильне управління життєвим циклом Activity може призвести до втрати даних, негативного користувацького досвіду та, як наслідок, до втрати продажів. Таким чином, Activity – це не просто картинка на екрані, а самостійна функціональна одиниця з власною бізнес-логікою.

Якщо Activity – це функціональне призначення приміщення, то Layout – це його детальний архітектурний план або креслення. Layout визначає, як саме виглядає екран та де на ньому розташовані всі візуальні елементи. Він описує, що кнопка «Купити» знаходиться внизу екрана, зображення товару займає центральну частину екрана, а його опис розміщений праворуч. В Android-розробці ці креслення зазвичай створюються за допомогою мови розмітки XML, що є аналогом інструкції для будівельників.

Ключова бізнес-цінність такого підходу полягає в принципі розділення відповідальності. Логіка роботи екрану (що саме він показує), яка реалізується в Activity, є відокремленою від його зовнішнього вигляду (як саме він

виглядає), що визначається у файлі Layout. Це розділення надає бізнесу надзвичайну гнучкість та є економічно вигідним. Уявіть, що компанія вирішила провести повний ребрендинг: змінити кольори, шрифти, стилі кнопок у своєму мобільному застосунку. Завдяки такому архітектурному рішенню, для цього не потрібно переписувати всю бізнес-логіку програмного продукту. Достатньо лише оновити креслення, тобто внести зміни у файли Layout. Це значно швидше, дешевше та безпечніше, ніж втручання у фундаментальний код мобільної програмної системи. Такий підхід дозволяє компанії швидко адаптуватися до нових ринкових трендів або проводити А/В-тестування різних варіантів дизайну для підвищення конверсії, не залучаючи до цього значних ресурсів на програмування. Отже, Layout – це інструмент, що забезпечує не лише візуальну привабливість, але й економічну ефективність у довгостроковій підтримці та розвитку мобільної програмної системи.

Офісна будівля не може функціонувати як набір ізольованих приміщень. Для її повноцінної роботи необхідні коридори, ліфти, вказівники та система внутрішнього зв'язку, що дозволяють працівникам переміщуватися між приміщеннями та обмінюватися інформацією. В екосистемі Android цю функцію виконує компонент під назвою Intent – повідомлення або запит, який один компонент мобільної програмної системи (наприклад, одна Activity) надсилає ОС Android, щоб активувати інший компонент.

Розрізняють два основні типи Intent-запитів. Явні (Explicit Intent) застосовуються для організації навігації між компонентами одного мобільного застосунку. У цьому випадку застосунок чітко визначає цільовий компонент, який необхідно відкрити. Наприклад, після успішної автентифікації користувача може бути виконано перехід від

екрана входу (Login Activity) до екрана особистого профілю (Profile Activity). Отже, явні Intent є базовим механізмом реалізації внутрішньої навігації мобільної програмної системи, оскільки забезпечують однозначне визначення компонента, до якого має бути передано керування.

Неявні (Implicit Intent) є більш універсальним механізмом взаємодії компонентів мобільної ОС та забезпечують ширші можливості інтеграції мобільного застосунку із зовнішніми сервісами. На відміну від явних Intent, вони містять лише опис необхідної дії без зазначення конкретного компонента, який повинен її виконати.

Наприклад, якщо користувач натискає кнопку «Поділитися», застосунок формує Intent із запитом на передачу текстового вмісту. Android аналізує цей запит, визначає всі встановлені на пристрої застосунки, що підтримують відповідний тип дії (поштові клієнти, месенджери, соціальні мережі тощо), після чого надає користувачеві можливість самостійно обрати необхідний сервіс для виконання операції.

Використання неявних Intent має суттєві переваги як із технічної, так і з економічної точки зору. Зокрема, розробникам не потрібно створювати власні механізми надсилання електронних листів, повідомлень або публікацій у соціальних мережах. Натомість мобільний застосунок використовує вже наявні можливості ОС та сторонніх застосунків, що значно скорочує витрати часу, фінансових ресурсів і трудомісткість розроблення. Крім того, такий підхід забезпечує користувачам можливість виконувати необхідні дії за допомогою звичних для них сервісів, що позитивно впливає на зручність використання та загальний користувацький досвід.

Підсумовуючи, базову структуру Android-застосунку можна уявити як систему, де Activity є функціональними модулями, Layout визначають їхній зовнішній вигляд, а

Intent забезпечують їхню злагоджену взаємодію. Розуміння цієї тріади дозволяє менеджеру продукту вести більш предметний діалог з технічною командою та приймати рішення, що ґрунтуються не лише на бізнес-потребах, але й на базових архітектурних реаліях технологічної платформи.

3.7 Основи крос-платформного розроблення

Основна економічна проблема нативного розроблення полягає у необхідності підтримувати дві абсолютно незалежні кодові бази та дві окремі команди для платформ Android та iOS. Такий підхід створює значне фінансове навантаження, особливо для стартапів. Саме як відповідь на цей виклик на ринку з'явилися та набули величезної популярності технології кросплатформного розроблення – підходу, що дозволяє створювати мобільні програмні системи для кількох ОС, використовуючи єдину кодову базу. Найбільш очевидною та переконливою перевагою цього підходу є його економічна ефективність. Замість того, щоб наймати дві команди спеціалізованих розробників, компанія може сформувати одну універсальну команду. Замість того, щоб писати, тестувати та підтримувати два окремі проекти, зусилля концентруються на одному. За різними оцінками, це дозволяє скоротити прямі витрати на розробку іноді навіть на п'ятдесят відсотків у порівнянні з нативним підходом.

Окрім прямої економії коштів, не менш важливою перевагою є значне прискорення виходу продукту на ринок. Оскільки нова функція розробляється одночасно для обох платформ, час від ідеї до її реалізації в руках кінцевих споживачів суттєво скорочується. У висококонкурентних галузях, де бути першим означає захопити лівову частку ринку, цей фактор може стати вирішальним для успіху всього бізнесу. Швидкість стосується не лише першого запуску, але й подальших ітерацій. Можливість оперативно

випускати оновлення, виправляти помилки та реагувати на відгуки користувачів одночасно на Android та iOS надає компанії значну конкурентну гнучкість. З точки зору менеджменту, управління одним проєктом замість двох також є простішим, що знижує адміністративні та комунікаційні накладні витрати.

Одним із найпопулярніших фреймворків для кросплатформного розроблення є React Native, створений та підтримуваний компанією Meta. Щоб зрозуміти його бізнес-цінність, необхідно розглянути його ключовий архітектурний принцип. На відміну від деяких інших технологій, що створюють вебсторінку всередині мобільного застосунку, React Native використовує інший, значно ефективніший підхід. Розробники пишуть логіку мобільного застосунку, використовуючи мову програмування JavaScript. А далі спеціальний механізм, який називається міст (bridge), транслює ці команди у справжні, нативні компоненти інтерфейсу для кожної платформи. Коли розробник пише команду, щоб створити кнопку, міст дає вказівку Android створити стандартну кнопку для Android, а iOS – створити стандартну кнопку для iOS. В результаті користувач взаємодіє зі звичними для нього нативними елементами, що забезпечує значно кращий досвід, ніж у випадку з вебтехнологіями. З точки зору бізнесу, це означає, що основна частина бізнес-логіки продукту, наприклад, правила розрахунку вартості замовлення, алгоритми валідації даних, логіка авторизації, пишеться лише один раз. Це і є джерелом тієї самої економії ресурсів, яке дозволяє команді максимально переносити свою роботу між платформами.

Однак наявність такого мосту є водночас і джерелом головного компромісу React Native. Хоча цей механізм є дуже швидким, процес перекладу команд з JavaScript у нативні виклики все ж потребує певного часу та ресурсів

процесора. Для більшості стандартних мобільних застосунків, таких як програми для електронної комерції, соціальні мережі, новинні портали чи корпоративні застосунки, ця різниця у продуктивності є непомітною для кінцевого користувача. Проте, для мобільних застосунків, що вимагають інтенсивної графіки, складних анімацій, оброблення відео в реальному часі або інших ресурсомістких операцій, цей додатковий шар абстракції може стати вузьким місцем та призвести до погіршення продуктивності.

Вибір React Native, як і будь-якого технологічного рішення, є стратегічним компромісом. Компанія отримує значну вигоду у вартості та швидкості, але натомість приймає певні ризики та обмеження. Перший та найважливіший компроміс, як вже було зазначено, – це продуктивність. Для 90% мобільних програмних систем її буде більш ніж достатньо, але бізнес повинен чітко усвідомлювати, чи не потрапляє його продукт у ті 10%, де нативна швидкість є критично важливою. Наприклад, запуск ресурсомісткої гри або професійного відеоредактора, створеного на React Native, є ризикованим рішенням, яке може призвести до негативних відгуків користувачів та провалу на ринку.

Другий ризик пов'язаний із залежністю від сторонньої платформи. React Native є продуктом з відкритим кодом, який підтримується компанією Meta та великою спільнотою розробників. Це робить його досить надійним. Проте, бізнес стає залежним від дорожньої карти та пріоритетів Meta. Якщо компанія вирішить уповільнити розвиток фреймворку React Native або кардинально змінити його архітектуру, це може створити серйозні проблеми для всіх, хто побудував свій бізнес на цій технології.

Третій важливий аспект – це доступ до нових функцій ОС. Коли Google або Apple анонсують нову унікальну

можливість у своїх ОС, нативні розробники отримують доступ до неї в перший же день. Команді React Native та спільноті потрібен час, щоб створити підтримку цієї нової функції та інтегрувати її у фреймворк. Ця затримка може тривати від кількох тижнів до кількох місяців. Для компаній, які прагнуть бути технологічними лідерами та першими пропонувати своїм клієнтам найсучасніші інновації, така затримка може бути неприйнятною з точки зору конкурентної боротьби.

Підсумовуючи, React Native є потужним стратегічним інструментом для бізнесу. Він дозволяє значно скоротити витрати та прискорити вихід на ринок, що робить його ідеальним вибором для створення мінімально життєздатних продуктів (MVP) для більшості B2C та B2E застосунків, а також для компаній з обмеженим бюджетом. Світові гіганти, такі як Shopify, Tesla, Instagram, використовують React Native у своїх продуктах, що підтверджує його життєздатність. Однак, прийняття рішення про його використання вимагає від менеджменту ретельного аналізу специфіки власного програмного продукту та чіткого розуміння компромісів, на які доведеться піти.

3.8 Взаємодія з сервером

Більшість мобільних програмних систем, якими ми користуємося щодня, не є самодостатніми програмами. Їхня справжня цінність полягає у здатності відображати актуальну, динамічну інформацію: стрічку новин, каталог товарів, баланс банківського рахунку, повідомлення від друзів. Ця інформація не зберігається безпосередньо в пам'яті смартфона, оскільки це зробило б мобільний застосунок надзвичайно великим за розміром та миттєво застарілим. Натомість, сучасні цифрові продукти побудовані за моделлю, відомою як клієнт-серверна

архітектура. Її розуміння є фундаментальним для усвідомлення того, як функціонує цифровий бізнес.

У клієнт-серверній архітектурі існує дві основні сторони. Клієнт – це мобільний застосунок, встановлений на пристрої користувача. Його головні завдання – це відображення інформації у зручному та привабливому вигляді, а також збір та передача даних, які вводить користувач. Сервер – це потужний комп'ютер, що знаходиться у віддаленому центрі оброблення даних. Він виконує функції централізованого сховища даних і є ключовим компонентом програмної системи, відповідальним за реалізацію основної бізнес-логіки. Саме на сервері здійснюються оброблення запитів клієнтів, виконання бізнес-процесів, керування даними та координація взаємодії між окремими компонентами програмної системи.

Для кращого розуміння, проведемо аналогію з ресторанним бізнесом. Клієнт – це відвідувач, що сидить за столиком та має в руках меню (інтерфейс застосунку). Сервер – це кухня ресторану, де зберігаються всі інгредієнти (дані) та відбувається приготування страв (оброблення даних).

Відвідувач ніколи не заходить на кухню безпосередньо. Існує чіткий розподіл функцій, що є економічно обґрунтованим з кількох причин. По перше, він забезпечує централізацію та консистентність даних. Усі користувачі мобільної програмної системи підключаються до одного сервера, що гарантує, що вони бачать однакову та актуальну інформацію, наприклад, однакову ціну на товар. По друге, це питання безпеки. Вся конфіденційна інформація та ключова бізнес-логіка зберігаються на захищеному сервері, а не на пристрої користувача, який може бути загублений, вкрадений або зламаний. Це критично важливий аспект управління ризиками. По третє, така архітектура забезпечує

масштабованість. Якщо кількість користувачів зростає, бізнесу достатньо збільшити потужність сервера (розширити кухню), не вимагаючи від кожного користувача оновлювати свій застосунок.

Клієнт та сервер повинні якимось чином спілкуватися між собою. Цей процес комунікації регулюється спеціальним набором правил та угод, який називається Application Programming Interface, або API. Повертаючись до нашої ресторанної аналогії, API – це офіціант. Він є посередником, що діє за чітко визначеними правилами. Відвідувач не може просто крикнути на кухню, щоб йому принесли чогось поїсти. Він повинен зробити замовлення з меню (API-запит), використовуючи назви страв з меню. Офіціант (API) приймає це замовлення та передає його на кухню (сервер) у зрозумілому для кухарів форматі. Кухня готує страву (обробляє API-запит та отримує дані з бази даних) і віддає її офіціанту, який, у свою чергу, приносить її відвідувачу у стандартному вигляді на тарілці (повертає API відповідь у структурованому форматі, наприклад, JSON).

API є формальним контрактом, що описує, які саме запити може робити клієнт, і які відповіді він отримає від сервера. Для бізнесу API є не просто технічним інструментом, а потужним стратегічним активом. Сучасна цифрова економіка значною мірою використовує API. Компанії можуть надавати доступ до своїх даних та функціоналу іншим бізнесам через публічні API. Наприклад, сервіс продажу авіаквитків може використовувати API різних авіакомпаній, щоб збирати інформацію про рейси, та API платіжної системи, щоб проводити оплату. Це дозволяє створювати складні та цінні для споживача програмні продукти шляхом комбінування сервісів від різних постачальників.

Крім того, наявність чітко визначеного API прискорює внутрішні процеси розробки. Команда, що працює над мобільним застосунком (клієнтом), та команда, що відповідає за серверну частину, можуть працювати паралельно та незалежно одна від одної. Їм достатньо на самому початку узгодити меню, тобто контракт API. Це значно скорочує загальний час розробки продукту.

Важливою особливістю взаємодії між клієнтом та сервером є її асинхронна природа. Коли мобільний застосунок надсилає запит на сервер, він не чекає відповіді. Процес отримання даних відбувається у фоновому режимі. В нашій аналогії, зробивши замовлення офіціанту, відвідувач не замирає, поки той не повернеться з кухні. Він може продовжувати розмову, пити воду або вивчати меню далі. Мобільний застосунок повинен поводитися так само. Якщо цього не зробити, інтерфейс буде блокуватися при кожному завантаженні даних, що створить жахливий користувацький досвід. З точки зору бізнесу, правильна обробка цього періоду очікування є критично важливою для утримання користувачів. Коли дані завантажуються, професійно розроблений мобільний застосунок повинен показувати користувачеві індикатор завантаження: анімований спінер або так звані скелетні екрани, які схематично показують структуру контенту, що скоро з'явиться. Це інформує користувача про те, що система працює, і що потрібно трохи зачекати. Мобільний застосунок, який просто зупиняється без будь яких пояснень, сприймається як неякісний та ненадійний. Користувач, особливо новий, може вирішити, що програма зламалася, і видалити її, не дочекавшись відповіді від сервера. Таким чином, інвестиції у якісну реалізацію асинхронної взаємодії є інвестиціями у задоволеність та довіру клієнтів.

Життєвою силою будь якої мобільної програмної системи є дані, які отримуються з віддаленого сервера за допомогою чітко регламентованих запитів через API. Розуміння клієнт-серверної архітектури та асинхронних принципів обміну інформацією дозволяє менеджеру усвідомити, де знаходиться справжня цінність та складність продукту.

Контрольні запитання

1. Що таке мобільна економіка?
2. Охарактеризуйте програмні системи типу B2C.
3. Охарактеризуйте програмні системи типу B2B.
4. Охарактеризуйте програмні системи типу B2E.
5. Що означає безшовний користувацький досвід між пристроями та платформами?
6. Назвіть основні етапи розроблення мобільної програмної системи.
7. Що ви вважаєте важливішим: UI чи UX? Поясніть, як би ви розставили пріоритети у дизайні вебсистеми.
8. Поясніть, як глобалізація впливає на дизайн мобільних програмних систем.
9. Дайте визначення метрикам TAM, SAM та SOM.
10. Опишіть основні блоки шаблону Lean Canvas.
11. Як дизайн впливає на показник утримання клієнтів?
12. Як дизайн впливає на конверсію?
13. Опишіть сутність проблеми інформаційного перевантаження інтерфейсу мобільної програмної системи.
14. У чому полягає основна економічна цінність прототипування?
15. Яка головна перевага вайрфрейму з точки зору бізнесу?
16. У чому полягає головна бізнес-цінність клікабельного прототипу?

РОЗДІЛ 4

ПІДВИЩЕННЯ КОНКУРЕНТОСПРОМОЖНОСТІ МОБІЛЬНИХ ПРОГРАМНИХ СИСТЕМ

4.1 Моделі монетизації мобільних програмних систем

Вибір способу, у який мобільна програмна система буде генерувати дохід, є одним із найважливіших стратегічних рішень, що приймається на ранніх етапах планування програмного продукту. Модель монетизації це не просто механізм отримання прибутку. Вона фундаментально визначає характер відносин між бізнесом та його користувачами, впливає на стратегію залучення та утримання клієнтів, а також формує сприйняття бренду. Неправильно обрана модель монетизації може звести нанівець успіх навіть технологічно досконалого продукту, тоді як вдала стратегія монетизації здатна перетворити простий застосунок на високоприбутковий бізнес. Рішення про те, як саме продукт буде заробляти гроші, повинно бути інтегрованою частиною його бізнес-моделі, відображеною ще на етапі її валідації у Lean Canvas. Воно має органічно впливати з унікальної ціннісної пропозиції продукту та відповідати очікуванням цільової аудиторії.

Існує кілька основних моделей монетизації, кожна з яких має свої переваги, недоліки та найкраще підходить для певних типів бізнесу. Модель платних завантажень, або Premium, є найбільш прямолінійною. Користувач сплачує фіксовану ціну один раз, щоб завантажити застосунок з магазину Google Play чи App Store. З економічної точки зору, ця модель є найпростішою для прогнозування доходу: дохід дорівнює ціні застосунку, помноженій на кількість завантажень. Кожен новий користувач миттєво приносить прибуток. Більше того, вимога попередньої оплати діє як фільтр, залучаючи більш вмотивовану та платоспроможну аудиторію. Однак у сучасній мобільній економіці, де

користувачі звикли до безкоштовних альтернатив, ця модель має суттєвий недолік: вона створює високий бар'єр для входу. Необхідність платити за продукт, який ще неможливо випробувати, відлякує переважну більшість потенційних користувачів. Це значно обмежує охоплення аудиторії та віральний потенціал продукту. Конкурувати з безкоштовними аналогами стає вкрай важко. Стратегічно, ця модель є виправданою лише для нішевих продуктів, що пропонують унікальну та значну цінність, не мають прямих безкоштовних конкурентів та орієнтовані на професійну аудиторію. Прикладами можуть бути спеціалізовані наукові калькулятори, професійні інструменти для обробки фото чи відео, або високоякісні ігри без реклами та внутрішніх покупок.

Модель внутрішніх покупок (In-App Purchases, IAP) передбачає, що мобільна програмна система розповсюджується безкоштовно, що дозволяє усунути початковий бар'єр для завантаження та залучити широку аудиторію. Дохід генерується за рахунок продажу різноманітних цифрових товарів або послуг безпосередньо всередині мобільної програмної системи. Ця модель домінує, зокрема, в індустрії мобільних ігор. Існує два основних типи внутрішніх покупок. Перший – це витратні матеріали, або *consumables*. Це товари, які користувач може купувати та використовувати багаторазово, наприклад, ігрова валюта, підказки, додаткові життя чи енергія. Ця модель стимулює повторні покупки від найбільш залучених користувачів. Другий тип – це невитратні товари, або *non-consumables*. Це покупки, що здійснюються один раз та надають користувачеві постійний доступ до певного контенту чи функціоналу. Прикладами є розблокування повної версії гри, видалення реклами, придбання додаткових наборів фільтрів для фоторедактора. Головна перевага цієї моделі полягає у можливості побудувати

величезну базу користувачів та монетизувати лише невеликий відсоток найбільш лояльних з них. Однак, вона несе в собі ризик створення негативного досвіду, якщо неплатні користувачі відчують себе обмеженими. Пошук балансу між безкоштовним та платним функціоналом є ключовим викликом для бізнесу.

Модель підписок (Subscriptions) за останні роки набула величезної популярності. Вона передбачає, що користувач сплачує регулярний платіж, зазвичай щомісяця або щорічно, за безперервний доступ до контенту або сервісу. З точки зору фінансового менеджменту, це найбільш приваблива модель, оскільки вона створює стабільний та прогнозований потік доходу, відомий як Monthly Recurring Revenue, або MRR. Прогнозованість доходу дозволяє бізнесу більш ефективно планувати свої витрати, інвестиції та стратегію зростання. Інвестори також високо цінують підписні бізнес-моделі через їхню стабільність. Ця модель змінює фокус бізнесу з одноразових продажів на побудову довгострокових відносин з клієнтами. Щоб утримувати користувачів та запобігати їхньому відтоку, компанія повинна постійно надавати цінність: регулярно оновлювати контент, додавати нові функції та покращувати сервіс. Це стимулює інновації та підвищує якість продукту. Модель підписок ідеально підходить для сервісів, що надають постійний доступ до контенту, як от Netflix чи Spotify, для новинних видань, для хмарних сховищ, а також для застосунків у сфері фітнесу та ментального здоров'я, таких як Calm або Headspace.

Монетизація за рахунок реклами дозволяє зробити мобільну програмну систему абсолютно безкоштовною для кінцевого користувача. Дохід генерується шляхом продажу рекламних площ всередині мобільної програмної системи іншим компаніям. Існують різні формати реклами – від невеликих банерів до повноекранних відео. Ефективність

цієї моделі прямо залежить від розміру аудиторії – чим більше у мобільної програмної системи щоденних активних користувачів, тим вищим є дохід від реклами. Головною перевагою є можливість максимального охоплення ринку. Однак, надмірна або нав'язлива реклама може сильно погіршити користувацький досвід та викликати роздратування.

Модель Freemium є гібридною та однією з найпопулярніших на сьогодні. Вона поєднує в собі безкоштовну та платну версії продукту. Користувачам пропонується безкоштовна версія мобільної програмної системи з базовим набором функцій, яка може містити рекламу або мати певні обмеження. Мета безкоштовної версії полягає у тому, щоб виступити потужним маркетинговим інструментом, залучити якомога більше користувачів та дозволити їм оцінити цінність продукту. Найбільш активним та лояльним користувачам згодом пропонується перейти на платну, преміальну версію, яка не містить реклами та пропонує розширений функціонал. Класичними прикладами є Spotify або Duolingo. Ключовим стратегічним завданням для бізнесу, що використовує модель Freemium, є пошук ідеального балансу. Безкоштовна версія має бути достатньо корисною, щоб залучати та утримувати масову аудиторію, але водночас преміум версія повинна пропонувати настільки вагомі переваги, щоб стимулювати значну частину користувачів до здійснення оплати.

Отже, вибір моделі монетизації це комплексне рішення, що вимагає глибокого аналізу продукту, ринку та психології споживачів. Часто компанії використовують комбінацію кількох моделей. Головний принцип успішної стратегії полягає в тому, щоб спосіб отримання доходу не суперечив, а навпаки, підсилював основну цінність, яку продукт надає своєму користувачеві.

4.2 Ключові показники ефективності та мобільна аналітика

Відомий постулат менеджменту стверджує, що неможливо керувати тим, що неможливо виміряти. У контексті мобільного бізнесу, де конкуренція є надзвичайно високою, а поведінка споживачів стрімко змінюється, прийняття рішень на основі інтуїції чи припущень є прямою дорогою до фінансових втрат. Мобільна аналітика є дисципліною, що перетворює управління продуктом з мистецтва на науку. Вона являє собою процес збору, оброблення та інтерпретації даних про те, як користувачі взаємодіють із застосунком. Цей процес дозволяє бізнесу отримувати об'єктивну картину реальності, діагностувати проблеми, знаходити точки росту та, зрештою, приймати обґрунтовані рішення, що максимізують рентабельність інвестицій. Аналітика не є одноразовою дією, а безперервним циклом, що супроводжує продукт на всіх етапах його існування.

Ще до початку розробки аналітичні дані використовуються для дослідження ринку та валідації бізнес-ідей. На етапі запуску продукту основний фокус аналітики зміщується на вимірювання ефективності маркетингових каналів. Вона допомагає відповісти на ключове питання: звідки приходять найбільш цінні користувачі? Аналізуючи дані, компанія може зрозуміти, яка рекламна кампанія, соціальна мережа чи публікація у ЗМІ приводить не просто більше завантажень, а більше активних та платоспроможних клієнтів. Це дозволяє оптимізувати маркетинговий бюджет, відмовляючись від неефективних каналів та інвестуючи в ті, що забезпечують найкращу віддачу. Після того, як продукт набрав початкову базу користувачів, фокус аналітики переходить на їхню поведінку всередині мобільної програмної системи. На цьому етапі бізнес прагне зрозуміти, наскільки продукт є

цікавим та корисним. Якими функціями користуються найчастіше, а які залишаються непоміченими? На якому етапі користувачі стикаються з труднощами та залишають застосунок? Відповіді на ці питання є безцінними для визначення пріоритетів у подальшій розробці. Нарешті, на етапі зрілості продукту аналітика допомагає оптимізувати його монетизацію. Вона дозволяє сегментувати аудиторію та зрозуміти, які групи користувачів найбільш схильні до здійснення покупок, як зміни в ціновій політиці впливають на дохід та як можна збільшити позитивну цінність кожного клієнта.

Існує величезна кількість показників, які можна відстежувати. Однак, важливо фокусуватися на тих ключових показниках ефективності, або КРІ, що найкраще відображають стан бізнесу. Одними з базових метрик є DAU та MAU, що означають кількість унікальних активних користувачів за день та за місяць відповідно. MAU демонструє загальний розмір аудиторії продукту, тоді як співвідношення DAU до MAU показує, наскільки регулярно люди повертаються до мобільного застосунку. Для месенджера високий показник цього співвідношення є нормою, тоді як для мобільного застосунку з пошуку авіаквитків він очікувано буде низьким. Проте, найважливішим показником здоров'я продукту є коефіцієнт утримання, або Retention Rate. Ця метрика показує, який відсоток користувачів повертається до мобільного застосунку через певний проміжок часу після першого встановлення, наприклад, на перший, сьомий та тридцятий день. Високий коефіцієнт утримання свідчить про те, що продукт дійсно вирішує проблему користувача та створює для нього цінність. Низький показник утримання є тривожним сигналом. Бізнес, що страждає від низького утримання, можна порівняти з дірявим відром: скільки б нових користувачів не залучалося за допомогою

маркетингу, вони швидко зникають. Жоден бізнес не може бути прибутковим у довгостроковій перспективі, якщо він не здатний утримувати своїх клієнтів.

Глибину залученості допомагають виміряти такі метрики, як тривалість сесії та частота сесій. Вони показують, скільки часу користувачі проводять у мобільному застосунку під час одного візиту та як часто вони його відкривають. Ці дані дозволяють зрозуміти патерни використання продукту.

Основною метою будь-якого комерційного програмного продукту є забезпечення прибуткової діяльності. Тому оцінювання ефективності мобільної програмної системи не повинно обмежуватися лише аналізом поведінкових характеристик користувачів, а має також включати дослідження ключових фінансових показників. Одним із найважливіших інструментів такого аналізу є економіка юніта (Unit Economics), яка дає змогу оцінити економічну доцільність бізнес-моделі на рівні окремого клієнта. Базовими показниками економіки юніта є вартість залучення клієнта (Customer Acquisition Cost, CAC) та пожиттєва цінність клієнта (Lifetime Value, LTV).

Customer Acquisition Cost (CAC) характеризує середні витрати компанії на залучення одного нового клієнта. Значення цього показника визначається як відношення сукупних витрат на маркетингову діяльність і продажі за певний період до кількості нових клієнтів, залучених протягом цього самого періоду. Таким чином, CAC відображає, скільки фінансових ресурсів необхідно витратити для залучення одного користувача.

Lifetime Value (LTV) відображає прогнозований сукупний дохід, який компанія очікує отримати від одного клієнта протягом усього періоду його взаємодії з програмним продуктом. Для оцінювання цього показника враховуються середній дохід, отриманий від одного

користувача, а також тривалість його активного використання продукту. Отже, LTV характеризує економічну цінність клієнта для бізнесу в довгостроковій перспективі.

Порівняння значень LTV і САС є ключовим критерієм оцінювання ефективності бізнес-моделі. Для забезпечення фінансової стійкості компанії величина пожиттєвої цінності клієнта повинна суттєво перевищувати витрати на його залучення. Якщо ж витрати на залучення користувачів перевищують дохід, який вони генерують протягом усього життєвого циклу взаємодії з продуктом, така бізнес-модель є економічно неефективною та не забезпечує сталого розвитку. У зв'язку з цим одним із головних завдань управління мобільним бізнесом є одночасне зниження показника САС шляхом підвищення ефективності маркетингових заходів і збільшення показника LTV завдяки вдосконаленню функціональних можливостей програмного продукту, підвищенню його цінності для користувачів та оптимізації механізмів монетизації.

Для збору та аналізу всіх цих даних компаніям не потрібно створювати власні складні системи. Існує розвинений ринок готових аналітичних інструментів. Стандартним рішенням для багатьох є Google Analytics for Firebase, безкоштовний та потужний інструмент від Google, що дозволяє відстежувати ключові події, аналізувати поведінку користувачів та сегментувати аудиторію. Для більш глибокого аналізу продукту існують спеціалізовані платформи, такі як Amplitude або Mixpanel. Вони дозволяють будувати складні воронки, проводити когортний аналіз та детально досліджувати користувацькі шляхи. Завдання менеджменту полягає не у технічному налаштуванні цих інструментів, а у правильній постановці бізнес-питань, відповіді на які можна знайти за допомогою зібраних даних. Таким чином, мобільна аналітика є

компасом, що вказує напрямок розвитку продукту. Вона дозволяє перейти від суб'єктивних думок до об'єктивних фактів, оптимізувати витрати, підвищувати доходи та будувати бізнес, що базується на глибокому розумінні своїх клієнтів.

4.3 Стратегія виходу на ринок та оптимізація сторінки мобільного застосунку в магазинах застосунків

Створення якісної мобільної програмної системи є необхідною, але не достатньою умовою для комерційного успіху. Без продуманої стратегії виходу на ринок, або *go to market strategy*, навіть найкращий продукт ризикує залишитися непоміченим серед мільйонів конкурентів. Процес запуску це не фінішна пряма для команди розробки, а стартовий постріл для команди маркетингу та розвитку бізнесу. Він вимагає ретельної підготовки, що починається задовго до того, як мобільний застосунок стає доступним для загального завантаження.

Одним з ключових етапів підготовки є бета-тестування. З точки зору бізнесу, це аналог тестового відкриття ресторану чи магазину, коли послуги надаються обмеженій, лояльній аудиторії до офіційного грандіозного відкриття. Мета бета-тестування полягає у виявленні та виправленні критичних помилок, оцінці стабільності роботи продукту в реальних умовах та отриманню першого зворотного зв'язку від справжніх користувачів. Запуск продукту з серйозними помилками може призвести до хвилі негативних відгуків у магазині застосунків, які вкрай важко виправити і які можуть назавжди зіпсувати репутацію продукту. Таким чином, бета-тестування є критично важливим інструментом управління ризиками.

Паралельно з технічною підготовкою відбувається створення маркетингових матеріалів, які будуть представляти продукт у магазині застосунків. Це включає

фінальний дизайн іконки, підготовку інформативних та привабливих скріншотів, а також створення короткого промоційного відео. Ці візуальні активи відіграють вирішальну роль у першому враженні, яке продукт справить на потенційного користувача. Процес публікації мобільного застосунку на платформі Android відбувається через спеціальний сервіс, що називається Google Play Console. Це онлайн кабінет для розробника або власника бізнесу, де відбувається завантаження фінальної версії мобільного застосунку, заповнення всієї необхідної інформації, налаштування моделі монетизації, вибір країн для розповсюдження та управління сторінкою продукту. Google стягує невелику одноразову реєстраційну плату, що робить поріг входу на найбільший у світі ринок мобільних застосунків дуже низьким. Після завантаження мобільний застосунок проходить процес перевірки, який здебільшого є автоматизованим та займає від кількох годин до кількох днів. Це забезпечує бізнесу відносну швидкість та гнучкість у публікації оновлень.

Після публікації мобільного застосунку одним із ключових завдань бізнесу стає забезпечення його видимості та залучення цільової аудиторії. У зв'язку з цим постає основне питання: «Яким чином потенційні користувачі дізнаються про існування мобільного застосунку та його функціональні можливості?».

Залучення користувачів до мобільного застосунку здійснюється переважно за допомогою двох основних каналів. Перший передбачає використання платного просування, коли компанія інвестує кошти в рекламні кампанії та оплачує покази оголошень або переходу користувачів за ними. Другий канал ґрунтується на органічному залученні, за якого потенційні користувачі самостійно знаходять мобільний застосунок без застосування платної реклами.

Одним із ключових інструментів збільшення органічного трафіку є оптимізація сторінки мобільного застосунку в магазинах застосунків (App Store Optimization, ASO). За своєю сутністю ASO є аналогом пошукової оптимізації вебресурсів (Search Engine Optimization, SEO), оскільки обидва підходи спрямовані на підвищення видимості цифрового продукту в результатах пошуку за релевантними ключовими запитами. Оскільки значна частина користувачів знаходить нові мобільні застосунки саме через пошукові механізми магазинів застосунків, зокрема Google Play, високі позиції у результатах пошуку безпосередньо впливають на рівень видимості продукту та кількість потенційних користувачів, які можуть ознайомитися із застосунком і встановити його.

Економічна цінність ASO є колосальною. Кожен користувач, що прийшов через органічний пошук, є для бізнесу умовно безкоштовним. Таким чином, ефективна стратегія ASO дозволяє значно знизити вартість залучення клієнта. Це напряду впливає на прибутковість бізнесу, оскільки розширює позитивну різницю між позитивною цінністю клієнта та вартістю його залучення. Інвестиції в ASO – це довгострокові інвестиції у створення стабільного та економічно ефективного каналу зростання.

Стратегія ASO складається з двох основних компонентів: текстової та візуальної оптимізації. Текстова оптимізація спрямована на те, щоб допомогти алгоритмам магазину застосунків правильно зрозуміти, про що ваш мобільний застосунок, та показувати його за відповідними пошуковими запитами. Найважливішим елементом тут є назва мобільного застосунку. Вона повинна містити не лише назву бренду, але й одне або два ключових слова, що найкраще описують його функціонал. Наприклад, назва «Bolt: Швидкі та доступні поїздки» є більш інформативною та ефективною, ніж використання лише назви Bolt, оскільки

вона не лише чітко пояснює потенційним користувачам призначення мобільного застосунку, а й надає пошуковим алгоритмам додаткову інформацію для його коректної індексації та ранжування за релевантними пошуковими запитам.

Наступним елементом є короткий та повний опис. Короткий опис – це ваш головний рекламний текст, який користувач бачить першим. Повний опис дозволяє більш детально розповісти про переваги продукту та природно інтегрувати в текст додаткові ключові слова. Процес вибору цих слів є окремим аналітичним завданням, що вимагає дослідження того, які саме терміни використовує ваша цільова аудиторія для пошуку схожих рішень.

Візуальна оптимізація спрямована на те, щоб переконати користувача, який вже знайшов вашу сторінку, натиснути кнопку «Встановити». Вона безпосередньо впливає на коефіцієнт конверсії. Центральним елементом тут є іконка. Це обличчя вашого продукту. Іконка має бути простою, впізнаваною, естетично привабливою та вирізнятися на тлі конкурентів. Погана іконка може звести нанівець усі зусилля з текстової оптимізації. Скріншоти та промо відео відіграють роль вітрини та демонстрації продукту. Їхнє завдання не просто показати екрани застосунку, а розповісти візуальну історію про те, яку проблему користувача вирішує продукт та які переваги він отримає. Якісні, професійно оформлені візуальні матеріали створюють враження надійності та підвищують довіру, що є вирішальним фактором у прийнятті рішення про завантаження.

Таким чином, стратегія виходу на ринок – це комплексний процес, що поєднує технічну готовність, ретельне планування та маркетингову оптимізацію. ASO є його невід’ємною частиною, яка дозволяє закласти фундамент для довгострокового органічного зростання. Це

безперервна робота, що вимагає аналізу даних, експериментів та постійного вдосконалення, але її результат – це стабільний потік нових користувачів та стійка бізнес-модель у висококонкурентному світі мобільних застосунків.

4.4 Підтримка програмного продукту після запуску

Запуск мобільного застосунку в магазині Google Play є початком його публічного життя, що неминуче пов'язане не лише з комерційними, але й з юридичними зобов'язаннями. Ігнорування цих аспектів може призвести до значних фінансових штрафів, блокування застосунку в магазині та, що найважливіше, до непоправної шкоди для репутації бізнесу. В сучасній цифровій економіці, де дані є одним з найцінніших активів, ключова юридична відповідальність лежить у сфері захисту персональних даних користувачів. Фундаментальним документом, що регулює ці відносини, є Політика конфіденційності. Це не просто формальний текст, а публічний договір між компанією та її клієнтами. У цьому документі бізнес зобов'язаний чітко, повно та у доступній формі пояснити, які саме дані користувачів він збирає, з якою метою, як він їх обробляє та зберігає, а також чи передає він їх третім сторонам.

Наявність прозорості та зрозумілої Політики конфіденційності є не лише вимогою платформ Google та Apple, але й ключовим елементом побудови довіри. Користувачі все більше обізнані у питаннях приватності, і будь яка нечіткість або спроба приховати реальні обсяги збору даних може викликати негативну реакцію та відтік аудиторії. Особливої ваги це питання набуває в контексті міжнародного законодавства, зокрема Загального регламенту про захист даних, або GDPR. Важливо розуміти, що дія GDPR є екстратериторіальною. Це означає, що його вимоги є обов'язковими для будь якої компанії у світі,

включно з українськими, якщо її програмна система обробляє персональні дані громадян ЄС. Враховуючи глобальний характер мобільних ринків, практично будь-яка мобільна програмна система з амбіціями міжнародного зростання підпадає під дію цього регламенту.

З економічної та правової точок зору недотримання вимог GDPR становить суттєвий ризик для діяльності компанії. За порушення положень Регламенту можуть застосовуватися фінансові санкції у розмірі до 20 млн євро або 4% річного світового обороту підприємства, залежно від того, яка із сум є більшою. Такі штрафи можуть мати критичний вплив на фінансову стійкість бізнесу та його подальший розвиток. Регламент GDPR зобов'язує компанії впроваджувати низку обов'язкових процедур щодо оброблення персональних даних, зокрема забезпечувати отримання однозначної згоди користувачів на їх оброблення, надавати можливість доступу до власних персональних даних, а також реалізовувати право на їх видалення, відоме як право на забуття. У зв'язку з цим вимоги щодо захисту персональних даних доцільно враховувати ще на етапі проєктування програмної системи, оскільки їх інтеграція після завершення розроблення ПЗ може потребувати суттєвого перегляду архітектури ПЗ, додаткових фінансових витрат і збільшення термінів реалізації проєкту.

Успішний запуск є лише початком довгого шляху. Щоб продукт залишався актуальним, конкурентоспроможним та прибутковим, він повинен постійно розвиватися. Цей процес розвитку базується на безперервному циклі, що складається з трьох етапів: отримання зворотного зв'язку, його аналіз та планування оновлень. Існує кілька каналів для отримання зворотного зв'язку. Перший і найбільш публічний – це відгуки та оцінки в магазині Google Play. Ці відгуки є надзвичайно важливими, оскільки вони

впливають не лише на рейтинг мобільного застосунку та його видимість у пошуку, але й на рішення потенційних користувачів. Бізнес повинен систематично моніторити ці відгуки та, що важливо, реагувати на них. Публічна відповідь на негативний відгук з пропозицією допомоги демонструє, що компанія дбає про своїх клієнтів, і може перетворити незадоволеного користувача на лояльного. Іншими каналами є прямі звернення до служби підтримки через електронну пошту чи чат, а також кількісні дані з аналітичних систем.

Комплексне використання якісних і кількісних даних забезпечує всебічне розуміння поведінки користувачів та створює надійну інформаційну основу для прийняття обґрунтованих управлінських рішень. Якісний зворотний зв'язок дає змогу з'ясувати причини поведінки користувачів, відповідаючи на питання «Чому вони діють саме так?», тоді як кількісні показники дозволяють визначити «Які саме дії виконують користувачі?». Поєднання цих двох підходів формує цілісне уявлення про взаємодію користувачів із програмним продуктом та сприяє ефективному вдосконаленню його функціональних можливостей. На основі аналізу цієї інформації формується дорожня карта продукту, або *product roadmap*. Вона визначає пріоритети для наступних оновлень. Найвищим пріоритетом завжди є виправлення критичних помилок, які перешкоджають нормальній роботі застосунку. Далі йдуть вдосконалення існуючого функціоналу та розробка нових функцій, яких найбільше потребують користувачі. Такий підхід, що базується на даних, гарантує, що обмежені ресурси команди розробників інвестуються у ті зміни, які принесуть найбільшу цінність для клієнтів і, як наслідок, для бізнесу.

Служба підтримки клієнтів це не центр витрат, а важливий інструмент утримання та побудови лояльності.

Кожне звернення користувача, який зіткнувся з проблемою, є одночасно і загрозою, і можливістю. Якщо проблему не вирішити або зробити це неякісно, компанія з високою ймовірністю назавжди втратить цього клієнта. Більше того, він може поділитися своїм негативним досвідом з іншими, завдаючи шкоди репутації бренду. З іншого боку, швидке та ефективне вирішення проблеми може перетворити розчарованого користувача на одного з найпалкіших прихильників бренду. З економічної точки зору, утримання існуючого клієнта є значно дешевшим, ніж залучення нового. Тому інвестиції в якісну підтримку є прямою інвестицією в підвищення показника LTV. Окрім функції утримання, служба підтримки є безцінним джерелом якісного зворотного зв'язку. Саме сюди надходить найбільш детальна інформація про проблеми, з якими стикаються користувачі, про їхні побажання та ідеї щодо покращення продукту. Організація процесу систематичного збору, класифікації та передачі цієї інформації до продуктової команди дозволяє бізнесу тримати руку на пульсі потреб своєї аудиторії. Для ефективного управління зверненнями компанії зазвичай використовують спеціалізовані програмні системи, такі як Zendesk або Intercom. Вони дозволяють автоматизувати процеси, відстежувати історію спілкування з кожним клієнтом та аналізувати загальні тренди у зверненнях.

Підсумовуючи, період після запуску вимагає від бізнесу не меншої уваги та ресурсів, ніж етап розроблення. Систематична робота зі зворотним зв'язком та регулярні оновлення забезпечують довгострокову конкурентоспроможність програмного продукту. А професійна служба підтримки перетворює проблеми на можливості, зміцнюючи лояльність клієнтів, яка є фундаментом будь якого сталого бізнесу.

Контрольні запитання

1. У яких випадках доцільно використовувати модель платних завантажень?

2. Опишіть переваги та недоліки моделі внутрішніх покупок.

3. У яких випадках доцільно використовувати модель підписок?

4. Опишіть переваги та недоліки монетизації за рахунок реклами.

5. Для чого розраховують метрики DAU та MAU?

6. Що показує коефіцієнт утримання?

7. Що показують тривалість сесії та частота сесії?

8. Що показує вартість залучення одного клієнта?

9. Що показує позитивна цінність клієнта?

10. У чому полягає мета бета-тестування програмної системи?

11. Як пошукова оптимізація може допомогти компанії виділитися серед інших?

12. Охарактеризуйте основні компоненти стратегії ASO.

13. Який фундаментальний документ регулює відносини між компанією та її клієнтами.

14. Які вимоги регламенту GDPR необхідно враховувати ще на етапі проєктування програмної системи?

15. Охарактеризуйте канали для отримання зворотнього зв'язку.

16. Для чого формується дорожня карта продукту?

17. Яку роль в утриманні клієнтів відіграє служба підтримки клієнтів?

РОЗДІЛ 5

АНАЛІТИКА ДАНИХ

5.1 Процес аналізу даних

У сучасному світі бізнес значною мірою покладається на аналітику даних. Компанії зараз більше ніж будь-коли усвідомлюють неймовірний потенціал цього зростаючого ресурсу та використовують його для отримання корисної інформації, прийняття обґрунтованих рішень та збільшення доходів. Мета аналітики даних полягає в отриманні значущої інформації з величезних обсягів необроблених даних. У основі аналітики даних лежать фундаментальні навички, необхідні для виявлення закономірностей та отримання аналітичних висновків.

Наприклад, компанія хоче знати, наскільки добре її продукт працює на певному ринку. Заглиблюючись у дані, аналітик даних може виявити такі тенденції, як вищі продажі протягом певних сезонів, уподобання щодо певних варіацій продукту або кореляції між маркетинговими кампаніями та стрибками продажів. Ці закономірності надають цінну інформацію про поведінку споживачів та динаміку ринку, дозволяючи компанії оптимізувати свою маркетингову стратегію, ефективніше адаптувати продукти до потреб клієнтів та, зрештою, покращити загальну ефективність на цільових ринках.

У сфері бізнес-операцій застосування аналітики даних забезпечує основу для інформування процесу бізнес-аналізу та надає компаніям можливість приймати обґрунтовані рішення на основі глибоких інтерпретацій ринкових тенденцій та поведінки споживачів.

Процес аналізу даних включає вивчення наборів даних для формування висновків, зазвичай з використанням статистичних методів. Аналітика даних – це інструмент, який дозволяє компаніям отримувати конкурентну перевагу

шляхом аналізу історичних даних, прогнозування тенденцій та оптимізації бізнес-процесів.

Великі дані дозволяють компаніям отримати повне розуміння свого цільового ринку та клієнтської бази. Компанії збирають інформацію про такі характеристики, як демографічні дані клієнтів, уподобання, історія покупок та поведінка в Інтернеті. Використовуючи цю інформацію, аналітики даних можуть виявляти закономірності та тенденції. Потім керівники маркетингової команди можуть адаптувати продукти, послуги та маркетингові кампанії до конкретних потреб своїх клієнтів, підвищуючи їхню задоволеність та лояльність.

Збір та використання великих даних стають дедалі важливішими в сучасному бізнес-середовищі, проте використання реального потенціалу великих даних пов'язане зі значними труднощами через їх величезний обсяг, швидкість виробництва та різноманітність. Проблеми створює також якість великих даних, оскільки неструктуровані та зашумлені дані можуть перешкоджати їх точному аналізу та інтерпретації. Проблеми з надійністю виникають з кількох причин: неточні дані, надлишкові записи та людські помилки під час введення даних.

Надлишкові записи – це кілька записів, помилково розміщених в одному наборі даних. Існують різні методи реагування на надлишкові записи. Найочевидніший – це просто їх видалити. Аналітики даних використовують такі інструменти, як код на Python та функції електронних таблиць, щоб відфільтрувати пошкоджені дані на заданому рівні та отримати точніший набір даних. Іншим методом вирішення проблеми надлишковості є призначення іншого значення спостереженню, яке значно відрізняється від решти спостережень, що потенційно вказує на аномалію або помилку, – наприклад, середнє значення вибірки.

Компанії, що ведуть бізнес з державами-членами ЄС, повинні дотримуватися регламенту GDPR, який суворо забороняє купівлю та продаж персональних даних, водночас надаючи особам право бути забутими. GDPR базується на фундаментальних принципах, спрямованих на захист персональних даних фізичних осіб у ЄС.

Аналіз даних – це процес систематичного використання статистичних та логічних методів для перегляду, сортування та узагальнення даних з метою отримання корисної інформації. Це вимагає спеціальних навичок та експертизи. Аналітики даних повинні володіти статистичними знаннями, навичками програмування та предметної експертизи, щоб розбиратися в складнощах аналізу великих даних. Крім того, важливо вміти інтерпретувати результати та ефективно доносити їх до широкої аудиторії. Також слід пам'ятати, що правила ЄС забороняють збір та зберігання значущих для забезпечення конфіденційності даних або розшифрування зашифрованих даних.

Завдяки сучасним інструментам вебаналітики компанії у режимі реального часу аналізують ринкові тенденції та діяльність конкурентів. Збираючи та аналізуючи дані з різних джерел, включаючи соціальні мережі, галузеві звіти, відгуки клієнтів та онлайн-форуми, компанії отримують інформацію про динаміку ринку, нові тенденції та стратегії конкурентів. Особи, що приймають рішення, можуть використовувати цю інформацію для виявлення можливостей, передбачення змін на ринку та проактивної адаптації своїх бізнес-стратегій для підтримки конкурентної переваги.

Аналітики даних використовують різноманітні методи для ідентифікації та отримання даних з різних джерел, наприклад, вебскрейпінг, збір даних датчиків, моніторинг

соціальних мереж, торговельні майданчики даних та API, а також внутрішній збір даних.

Вебскрейпінгом називається автоматизоване отримання даних з онлайн-джерел за допомогою ПЗ, що імітує поведінку людини в Інтернеті. Методи вебскрейпінгу передбачають використання автоматизованих інструментів, які можуть збирати інформацію з вебсторінок, включаючи відгуки клієнтів, дані соціальних мереж, новинні статті або загальнодоступні набори даних.

З поширенням пристроїв Інтернету речей аналітики даних можуть використовувати збір даних з датчиків, призначених для реагування на такі фізичні умови, як температура, вологість, тиск, місцезнаходження або рух. Ці датчики генерують дані в режимі реального часу, надаючи цінну інформацію для таких галузей економіки, як виробництво, охорона здоров'я та логістика.

Моніторинг соціальних мереж включає моніторинг та збір даних із платформ соціальних мереж, щоб отримати уявлення про настрої, поведінку та тенденції клієнтів. Аналізуючи розмови, коментарі, лайки та поширення в соціальних мережах, аналітики даних можуть виявляти нові теми, уподобання споживачів та потенційні проблеми з брендом.

Деякі компанії надають торговельні майданчики даних або API, за допомогою якого програмні застосунки спілкуються та взаємодіють один з одним з метою обміну даними. Торговельний майданчик даних – це онлайн-платформа, де постачальники даних та споживачі можуть купувати, продавати або обмінюватися наборами даних та пов'язаними послугами. Ці онлайн-платформи пропонують фінансові дані, дані про погоду, демографічні дані та галузеві набори даних. Вони сприяють доступу до даних у різних форматах, часто інтегруючи інструменти для аналітики та візуалізації.

Компанії часто мають внутрішні джерела даних, включаючи записи про транзакції, бази даних клієнтів, дані про продажі або операційні журнали. Аналітики даних можуть використовувати ці джерела для збору відповідних даних для аналізу та отримання уявлення про власні бізнес-операції. Компанії, які використовують можливості великих даних для прийняття стратегічних рішень, отримують конкурентну перевагу. Під час збору великих даних аналітики даних повинні дотримуватися правил конфіденційності даних та отримувати відповідні дозволи або згоду, коли це необхідно.

Процес аналізу даних складається з кількох етапів, кожен з яких впливає на наступний та надає йому інформацію (рис. 5.1).

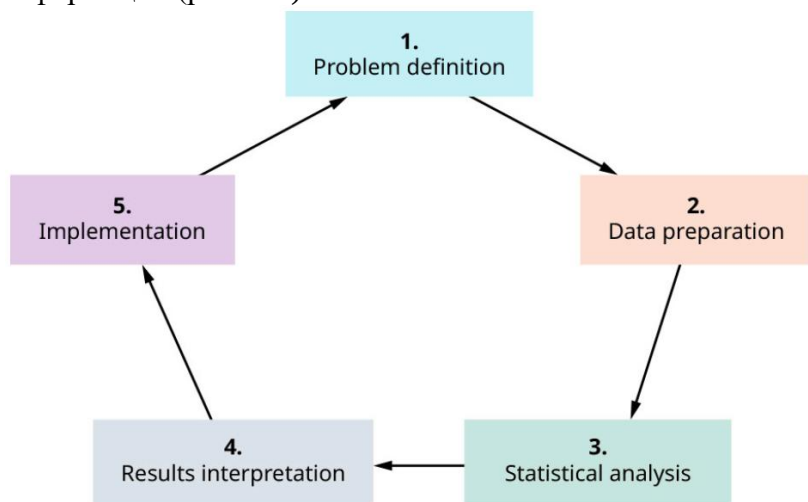


Рисунок 5.1 – Процес аналізу даних

Процес аналізу даних дозволяє компаніям отримувати цінну інформацію з даних. Першим етапом у цьому процесі є визначення проблеми (Problem definition). Компанія повинна сформулювати проблему, яка потребує вирішення за допомогою аналітики даних. Чітко визначивши

проблему, керівники компанії можуть зосередити свої зусилля на аналізі даних.

Далі йде збір, очищення та перетворення даних для аналізу (Data preparation). Цей етап включає збір даних з різних джерел, інтеграцію окремих наборів даних та забезпечення якості даних. Для забезпечення придатності даних для аналізу використовують методи очищення та нормалізації даних. Нормалізація даних передбачає коригування даних таким чином, щоб вони мали стандартний масштаб. Це полегшує порівняння різних типів значень та гарантує, що одна ознака не домінує над іншими через свій масштаб.

Існують такі підходи до збору даних: використання вбудованих бібліотек, використання зовнішніх наборів даних та ручне введення даних. Кожен підхід має свої переваги. Вбудовані бібліотеки, наприклад, Python або R, заощаджують час, але можуть бути неповними. Зовнішні дані зручні, але з великими наборами даних складно працювати, особливо якщо є кілька джерел даних. Недоліки ручного введення даних включають час, необхідний для введення набору даних, та виникнення помилок через людський фактор.

Після підготовки даних аналітики застосовують статистичний аналіз (Statistical analysis) для виявлення закономірностей та взаємозв'язків. Цей етап включає використання статистичних методів та алгоритмів машинного навчання. Залежно від характеру даних, аналітики даних можуть використовувати описову статистику, регресійний аналіз, кластеризацію, класифікацію або прогнозне моделювання. Аналітики даних використовують ці інструменти для проведення прогнозного моделювання. Описова статистика допомагає отримати моментальний знімок даних та забезпечити відправну точку для їх аналізу.

Наступний етап – інтерпретація результатів (Results interpretation), під час якого отримані результати перетворюються на практичну інформацію. Аналітики даних оцінюють результати в контексті проблеми, що розглядається, інтерпретують статистичні результати та роблять висновки. Вони часто створюють візуалізації, діаграми або звіти на основі даних, щоб ефективно доносити свою думку до стейкхолдерів.

Впровадження (Implementation) передбачає застосування отриманих висновків та рекомендацій на практиці – прийняття стратегічних рішень, удосконалення процесів та операційні зміни на основі отриманих результатів. Впровадження також може вимагати співпраці між підрозділами компанії або інтеграції аналітичних моделей у існуючі системи чи робочі процеси.

5.2 Правила конфіденційності даних у ЄС

Правила та стандарти конфіденційності даних стають дедалі важливішими у сучасному світі, керованому даними. З огляду на те, що щодня збираються та обробляються величезні обсяги персональних даних, ці правила забезпечують захист персональної інформації та надають стандартизований підхід для бізнесу до управління конфіденційністю даних. У всьому світі країни розробляють правила для задоволення цієї постійно зростаючої потреби. Загальний регламент про захист даних (GDPR) – це комплексний закон про захист даних у ЄС, який встановлює принципи збору та обробки персональної інформації фізичних осіб у межах ЄС. Компанії, які працюють із громадянами ЄС, повинні дотримуватися його вимог незалежно від свого місцезнаходження.

GDPR встановлює сім основоположних принципів оброблення персональних даних, яких повинні

дотримуватися всі компанії, що здійснюють їх збирання, зберігання та використання.

1. Законність, справедливість і прозорість. Оброблення персональних даних повинно здійснюватися на законних підставах, бути добросовісним і зрозумілим для суб'єктів даних, які мають бути належним чином поінформовані про мету та порядок їх використання.

2. Обмеження мети. Персональні дані можуть збиратися виключно для конкретних, чітко визначених і законних цілей та не повинні використовуватися способом, несумісним із такими цілями.

3. Мінімізація даних. Обсяг персональних даних має бути адекватним, релевантним і обмежуватися лише тією інформацією, яка є необхідною для досягнення визначеної мети оброблення.

4. Точність. Персональні дані повинні бути достовірними, актуальними та за потреби своєчасно оновлюватися або виправлятися.

5. Обмеження строків зберігання. Зберігання персональних даних допускається лише протягом періоду, необхідного для досягнення цілей їх оброблення, після чого вони підлягають видаленню або анонімізації.

6. Цілісність і конфіденційність. Компанія повинна забезпечити належний рівень захисту персональних даних від несанкціонованого доступу, втрати, знищення, зміни чи інших неправомірних дій шляхом застосування відповідних технічних і організаційних заходів.

7. Підзвітність. Контролер персональних даних несе відповідальність за дотримання вимог GDPR і повинен бути здатним документально підтвердити відповідність процесів оброблення встановленим принципам, забезпечуючи взаємодію з регуляторними органами та відкритість щодо виконання вимог законодавства у сфері захисту персональних даних.

Оброблення персональних даних дозволяється лише за наявності однієї із законних підстав:

- добровільна згода суб'єкта даних;
- виконання договору;
- виконання юридичного обов'язку;
- захист життєво важливих інтересів особи;
- виконання завдання в суспільних інтересах або здійснення офіційних повноважень;
- законний інтерес контролера, якщо він не порушує права суб'єкта даних.

GDPR запровадив суттєві зміни, зокрема суворіші вимоги до згоди, тобто явного дозволу, наданого особою на збір, оброблення та використання її персональної інформації. Окрім визначення принципів оброблення персональних даних, GDPR суттєво розширює перелік прав суб'єктів даних, надаючи громадянам більше можливостей контролювати використання власної персональної інформації. До основних прав належать:

1. Право на отримання інформації про порядок, мету та підстави оброблення персональних даних.

2. Право на доступ до власних персональних даних і відомостей щодо їх оброблення.

3. Право на виправлення неточних або неповних персональних даних.

4. Право на видалення персональних даних (право на забуття), яке передбачає можливість вимагати їх вилучення за наявності визначених законодавством підстав.

5. Право на обмеження оброблення, що дає змогу тимчасово призупинити використання персональних даних у встановлених випадках.

6. Право на перенесення даних, яке забезпечує можливість отримати персональні дані у структурованому форматі та передати їх іншому контролеру.

7. Право на заперечення проти оброблення персональних даних у випадках, передбачених GDPR.

8. Право не бути об'єктом виключно автоматизованого прийняття рішень, зокрема рішень, що ґрунтуються на профілюванні та можуть мати юридичні або інші суттєві наслідки для суб'єкта даних.

Компанії, діяльність яких пов'язана з обробленням персональних даних, зобов'язані забезпечити виконання комплексу організаційних і технічних вимог, передбачених GDPR. До основних обов'язків належать:

1. Ведення обліку операцій з персональними даними, що забезпечує прозорість процесів їх оброблення та можливість підтвердження відповідності вимогам законодавства.

2. Запровадження технічних і організаційних заходів захисту, спрямованих на забезпечення конфіденційності, цілісності та доступності персональних даних.

3. Застосування принципів Privacy by Design і Privacy by Default, відповідно до яких вимоги щодо захисту персональних даних інтегруються в архітектуру програмної системи ще на етапі її проєктування, а налаштування за замовчуванням забезпечують максимальний рівень захисту інформації.

4. Укладання договорів між контролером і обробником персональних даних, які визначають права, обов'язки та відповідальність сторін під час здійснення оброблення даних.

5. Проведення оцінки впливу на захист персональних даних (Data Protection Impact Assessment, DPIA) у випадках, коли заплановані операції можуть створювати високий ризик для прав і свобод суб'єктів персональних даних.

6. Призначення уповноваженої особи із захисту даних (Data Protection Officer, DPO) у випадках, передбачених законодавством, для здійснення контролю за дотриманням

вимог GDPR, консультування організації та взаємодії з наглядовими органами.

У разі порушення безпеки персональних даних, що призвело до їх витоку, несанкціонованого розкриття або іншої компрометації, GDPR покладає на контролера персональних даних низку обов'язків щодо реагування на інцидент. Зокрема, контролер зобов'язаний повідомити компетентний наглядовий орган не пізніше ніж через 72 години після виявлення порушення. Якщо інцидент може спричинити високий ризик для прав і свобод фізичних осіб, контролер також повинен без невиправданої затримки поінформувати про це суб'єктів персональних даних.

Недотримання вимог GDPR може призвести до застосування значних адміністративних санкцій. За порушення меншої тяжкості GDPR передбачає штраф у розмірі до 10 млн євро або 2% світового річного обороту компанії за попередній фінансовий рік (залежно від того, яка сума є більшою). За найбільш серйозні порушення, зокрема пов'язані з недотриманням основних принципів оброблення персональних даних або прав суб'єктів даних, розмір штрафу може сягати 20 млн євро або 4% річного обороту компанії, залежно від того, яка величина є більшою.

GDPR вимагає від компаній документувати, як вони обробляють дані. Тому розуміння того, які дані збираються, з якою метою, а також як вони обробляються та зберігаються, є надзвичайно важливим. Крім того, GDPR зобов'язує компанії вести детальний облік своєї діяльності з оброблення даних, від збору даних до зберігання та остаточного видалення. Таке ведення обліку є не лише вимогою дотримання регламенту, воно також виконує стратегічну функцію, сприяючи культурі підзвітності та прозорості. Недотримання цього детального рівня

документування може призвести до значних юридичних наслідків згідно з GDPR.

Згідно з практикою Європейського Суду, поняття підприємства охоплює будь-якого суб'єкта господарювання, який здійснює економічну діяльність, незалежно від правового статусу суб'єкта господарювання чи способу його фінансування. Таким чином, підприємство може складатися не лише з однієї окремої компанії у сенсі юридичної особи, а й з кількох фізичних осіб або корпоративних утворень. Таким чином, цілу групу компаній можна розглядати як одне підприємство, а її загальний світовий річний оборот можна використовувати для розрахунку штрафу за порушення GDPR однією з її компаній. Порушення в компанії можна виявити завдяки проактивним інспекційним заходам, що проводяться органами захисту даних, незадоволеними працівниками або клієнтами, які скаржаться до органів влади, самозверненням компанії або через журналістські розслідування.

Галузеві стандарти також відіграють важливу роль у формуванні того, як бізнес захищає персональні дані. Наприклад, Міжнародна організація зі стандартизації (ISO) запровадила міжнародний стандарт ISO/IEC 27701:2025 для систем управління інформаційною безпекою, що дозволяють компаніям комплексно та структуровано керувати безпекою, гарантуючи, що всі потенційні вразливості усунені, а системи стійкі до потенційних атак.

ISO/IEC 27701:2025 надає рекомендації щодо управління інформацією про конфіденційність, по суті перетворюючи принципи конфіденційності таких нормативних актів, як GDPR, на дієві засоби контролю. ISO/IEC 27701:2025 включає не лише технічні заходи, але й адміністративні, такі як визначення ролей та обов'язків, ведення обліку діяльності з оброблення та забезпечення належних процедур реагування на порушення безпеки

даних. Впроваджуючи стандарт ISO/IEC 27701:2025, компанії демонструють свою відданість конфіденційності, зберігають довіру стейкхолдерів, а також отримують конкурентну перевагу.

Компаніям потрібно бути пильними та адаптивними не лише для того, щоб уникнути штрафів, але й для того, щоб заслужити та зберегти довіру та лояльність своїх клієнтів. Це особливо актуально в епоху, коли конфіденційність даних все частіше розглядається як конкурентна перевага. Довіра до того, як компанії обробляють персональні дані, може суттєво вплинути на репутацію їхнього бренду, стосунки з клієнтами та, зрештою, на їхній прибуток. Ландшафт конфіденційності даних стає дедалі складнішим, і бути в курсі правил та стандартів конфіденційності даних має вирішальне значення для бізнесу.

Оцінювання прогалин та ризиків – це аудит, який допомагає визначити, де політика та практика компанії відповідають світовим стандартам або відхиляються від них. Його метою є встановлення поточної ситуації. Після завершення аудиту компанія може використати його результати для створення нових політик та протоколів, які приведуть компанію у відповідність до GDPR. Потрібно створити дорожню карту для досягнення відповідності GDPR, яка включає визначення того, що потрібно зробити, та детальний опис того, як це зробити. У таблиці 5.1 наведено приклади нових політик, які організація може запровадити після аудиту відповідності GDPR.

Будь-які нові політики компанії повинні пройти юридичну перевірку, щоб переконатися в їхній відповідності GDPR. Вони також повинні бути перевірені різними підрозділами компанії на предмет доцільності:

- *Юридична перевірка на відповідність GDPR.* Проект політики повинні переглянути юристи, що спеціалізуються на дотриманні вимог захисту даних.

Таблиця 5.1 – Політики узгодження з GDPR

Політика	Опис
Зберігання даних з обмеженим часом	Розробити політику, згідно з якою дані користувачів видаляються після певного періоду, якщо не отримано поновленої згоди, відповідно до принципів GDPR щодо мінімізації даних та обмеження зберігання
Інструкції щодо оброблення даних третіх сторін	Встановити суворіші правила для сторонніх розробників, включаючи ретельні процеси перевірки та контрольні списки відповідності, щоб забезпечити відповідальне поводження з даними користувачів відповідно до принципу підзвітності GDPR
Суворіші заходи безпеки даних	Впровадити методи шифрування та двофакторну автентифікацію як налаштування за замовчуванням для покращення захисту даних користувачів відповідно до принципу цілісності та конфіденційності GDPR
Автоматизовані сповіщення про оброблення даних	Розробити правила, які вимагатимуть чіткого сповіщення користувачів про автоматизоване прийняття рішення на основі даних користувачів, а також можливості відмовитися від нього, як того вимагає GDPR

– *Огляд підрозділами на предмет доцільності.* Після того, як юридична перевірка підтвердить, що проєкт політики відповідає GDPR, наступним кроком є перевірка політики підрозділами, на які вона вплине, наприклад, маркетинг, продажі та обслуговування клієнтів, щоб забезпечити операційну доцільність.

– *Схвалення керівництва.* Остаточним арбітром у процесі перевірки політики, як правило, є керівництво компанії. Його згода є критично важливою не лише для схвалення, але й для ефективного впровадження політики.

Працівників компанії необхідно інформувати про будь-які нові політики, щоб забезпечити дотримання вимог GDPR. Для забезпечення відповідності GDPR вирішальне значення має навчання працівників компанії новим політикам.

Впровадження нових політик для узгодження практик оброблення даних з вимогами GDPR перетворює результати оцінювання на практичні дії. Наприклад, якщо виявлено прогалини в управлінні згодою користувачів, відповідну політику слід оновити, щоб забезпечити чітку згоду та належну фіксацію згоди користувачів. Аналогічно, якщо виявлено діяльність з високим рівнем ризику, компанія має запровадити такі заходи, як шифрування або суворіший контроль доступу.

Ефективне впровадження часто вимагає координації між кількома підрозділами, забезпечуючи узгодженість та інтеграцію змін в усій компанії. У таблиці 5.2 наведено приклади стратегій впровадження нових політик оброблення даних, які може використовувати компанія.

Постійний моніторинг є необхідним для ефективного впровадження змін в поточну діяльність компанії. Він включає регулярне проведення автоматизованих та ручних перевірок відповідності компанії вимогам GDPR.

Таблиця 5.2 – Приклади стратегій впровадження нових політик оброблення даних

Стратегія впровадження	Опис
Оновлення ПЗ для всієї системи	Впровадження оновлень ПЗ для всієї системи для застосування нових політик щодо оброблення даних, які включають налаштування конфіденційності. Може включати зміни на рівні інтерфейсу користувача, такі як додавання прапорців для отримання згоди та сповіщень про використання даних користувачів
Реінжиніринг бекенд-процесів	Заміна ПЗ серверних систем, включаючи алгоритми зберігання та пошуку даних для відповідності принципу мінімізації даних GDPR. Може включати модифікацію алгоритмів для анонімізації даних користувачів для усунення порушень конфіденційності
Підвищення кваліфікації працівників	Навчальні сесії для працівників для забезпечення обізнаності та належного впровадження нових політик

Графік планових перевірок забезпечує структуровані інтервали для перевірки, що, в свою чергу, з часом мінімізує ризик повторного виявлення невідповідностей GDPR у практиці компанії.

Важливим етичним питанням у галузі ІТ є прозорість. У контексті управління прозорість означає відкритість, комунікацію та підзвітність, де дії та рішення є чіткими та зрозумілими для стейкхолдерів. Користувачі мають право знати, як їхні дані збираються, зберігаються, використовуються та поширюються. Компанії повинні бути прозорими у своїх практиках оброблення даних, забезпечуючи чіткі та зрозумілі політики конфіденційності та механізми отримання згоди. Прозорі практики не лише відповідають нормативним вимогам, але й сприяють довірі між клієнтами та компаніями, що має вирішальне значення для довгострокової взаємодії з клієнтами.

Роль технологічних компаній та ІТ-фахівців у відповідальності за соціальні та етичні наслідки технологій ще ніколи не була такою важливою. Обидві сторони є ключовими стейкхолдерами у формуванні норм та цінностей цифрової сфери. Компанії повинні наповнювати свої бізнес-стратегії етичними міркуваннями – від захисту даних користувачів до забезпечення цифрової інклюзивності. ІТ-фахівці повинні дотримуватися професійних етичних кодексів, пропонуючи рішення, що поважають права користувачів та суспільні цінності. Саме завдяки їхнім спільним зусиллям технології можуть по-справжньому виконувати своє призначення як інструмент для покращення добробуту суспільства.

5.3 Основи бізнес-аналітики

У сучасному управлінні бізнесом компанії постійно використовують інформацію для отримання цінної аналітики та керування процесами прийняття рішень. Один із способів, як компанії можуть це зробити, – це використання бізнес-аналітики (Business Intelligence, BI), яка допомагає виявити нові тенденції та вподобання споживачів, спрямовуючи зусилля з розробки програмних

продуктів. Аналіз відгуків та поведінки клієнтів допомагає компаніям визначити бажані функції та включити їх у свої програмні продукти.

Бізнес-аналітика допомагає компаніям випереджати зміни ринку, виявляючи нові тенденції та можливості. Персоналізовані рекомендації на основі поведінки споживачів збільшують продажі та задоволеність клієнтів. Визначення регіональних моделей попиту та уподобань щодо запасів допомагають компаніям зменшити надлишок товарів та уникнути дефіциту.

Успіх чи невдача бізнесу може залежати від здатності ставити правильні запитання та, таким чином, приймати обґрунтовані рішення в умовах обмежених ресурсів. Процес бізнес-аналітики включає збір, аналіз та інтерпретацію даних для прийняття бізнес-рішень та покращення організаційної ефективності. Бізнес-аналітика надає набір інструментів, які дозволяють особам, що приймають рішення, працювати розумніше. За допомогою бізнес-аналітики компанії можуть виявляти тенденції, закономірності та кореляції у своїх даних, що дозволяє їм зрозуміти поведінку клієнтів, динаміку ринку та операційну ефективність.

У сучасному бізнес-ландшафті прийняття рішень має ґрунтуватися на точній та актуальній на даний момент інформації. У цьому процесі життєво важливу роль відіграє бізнес-аналітика. Консолідуючи дані з різних джерел та перетворюючи їх на змістовні висновки, бізнес-аналітика надає особам, які приймають рішення, всебічне розуміння поточного стану компанії та її перспектив. За допомогою бізнес-аналітики особи, які приймають рішення, оцінюють різні сценарії та виконують прогнозне моделювання. Це дозволяє їм приймати обґрунтовані рішення з меншою невизначеністю.

Важливою перевагою інструментів бізнес-аналітики є можливість створювати візуалізацію або графічне представлення даних для полегшення розуміння, аналізу та передачі аналітичних висновків. Інформаційна панель (*dashboard*) сприяє ефективнішому донесенню складних ідей до аудиторії з мінімальним досвідом у технічних галузях.

Бізнес-лідери часто аналізують стан компанії, і KPI, що є важливою частиною цього аналізу. KPI – це вимірювана величина, яка демонструє, наскільки ефективно компанія досягає своїх ключових бізнес-цілей. З цим пов'язана концепція бенчмаркінгу, яка полягає в порівнянні ефективності компанії зі стандартами галузі та показниками конкуруючих компаній для визначення областей для покращення ефективності та оптимізації. Інструменти бізнес-аналітики дозволяють визначати KPI та контрольні рівні, а також оцінювати ефективність проєкту в режимі реального часу.

Завдяки ретельному аналізу даних клієнтів, керівники бізнесу краще розуміють їх очікування, звички та вподобання. Ці дані використовуються для підвищення залученості та лояльності клієнтів, покращення ефективності маркетингових кампаній і визначення додаткових каналів збуту.

Контрольні запитання

1. Які ключові права регламент GDPR надає фізичним особам і які їхні наслідки для бізнесу?
2. Які ключові рушійні сили швидкого створення даних у сучасну цифрову епоху, і як це впливає на конфіденційність та безпеку даних?
3. Чому для бізнесу важливо розуміти різні правила та практики конфіденційності в різних регіонах світу?
4. Охарактеризуйте регламент GDPR.

5. Охарактеризуйте галузевий стандарт ISO/IEC 27001.
6. Як регламент GDPR впливає на соціальну відповідальність компаній та ІТ-фахівців, що розробляють вебтехнології та технології Інтернету речей?
7. Яких ключових принципів регламенту GDPR повинні дотримуватися компанії?
8. Опишіть потенційні ризики, з якими компанія може зіткнутися через прогалини в її політиці безпеки даних та конфіденційності. Як компанія може зменшити ці ризики?
9. Які проблеми пов'язані зі збором та використанням великих даних, і як компанії можуть вирішити ці проблеми, щоб розкрити потенціал великих даних для прийняття стратегічних рішень?
10. Яким чином компанії можуть використовувати великі дані для отримання конкурентної переваги та покращення ефективності своєї діяльності?
11. Опишіть ключові етапи процесу бізнес-аналітики, виділивши основні проблеми, з якими стикаються компанії під час впровадження методів бізнес-аналітики.
12. Для чого використовуються інформаційні панелі?
13. У чому полягає концепція бенчмаркінгу?
14. Опишіть інструменти, що використовуються у вебаналітиці для збору даних.
15. Наведіть приклад з власного досвіду про те, як бізнес-аналітику можна застосувати для прийняття обґрунтованих рішень.
16. Як повинні поводитися технологічні компанії та ІТ-фахівці, щоб технології були інструментом для покращення добробуту суспільства?
17. Що означає прозорість у галузі ІТ у контексті управління?
18. Наведіть приклади стратегій впровадження нових політик оброблення даних у компанії.
19. Наведіть приклади політик узгодження з GDPR.

ТЕСТОВІ ЗАВДАННЯ

1. Який недолік системного аналізу?

☐ затримує раннє виявлення потенційних загроз для процесів;

☐ збільшує витрати ресурсів та загальні витрати;

☐ перешкоджає узгодженню стратегічних цілей;

☐ знижує якість, ефективність, продуктивність та зручність використання системи.

2. Який інструмент використовується в системному аналізі для документування деталей проєкту?

☐ моделювання;

☐ схема потоку даних;

☐ псевдокод;

☐ словник даних.

3. Яке твердження найкраще характеризує процес розробки програмного забезпечення за методологією Agile?

☐ це лінійний підхід до розробки програмного забезпечення, який послідовно переходить від одного етапу до наступного;

☐ зміни в середовищі не враховуються;

☐ фінальний продукт постачається в одному пакеті в кінці процесу;

☐ він цінує людей та взаємодію більше, ніж процеси та інструменти.

4. Невеликий стартап розробляє мобільний застосунок і має обмежені ресурси. Команда розробників хоче швидко випустити базову версію застосунку, зібрати відгуки користувачів і внести ітеративні покращення на основі цих відгуків. Команда розробників також планує внести зміни в деякі функції застосунку після того, як дізнається більше про потреби користувачів. Враховуючи ці умови, який з підходів найкраще підходить для команди розробників?

○ використовувати каскадний підхід (waterfall), щоб забезпечити повну розробку та тестування всіх функцій до першого релізу, запобігаючи будь-яким суттєвим змінам продукту після запуску;

○ дотримуватись гнучкого процесу розробки, щоб зосередитися на створенні мінімально життєздатного продукту (MVP), збирати відгуки користувачів та повторювати роботу над застосунком після кожного нового релізу;

○ використовувати гібридний підхід, поєднуючи елементи Agile та waterfall, щоб зберегти гнучкість у розробці, дотримуючись при цьому суворого послідовного графіка;

○ використовувати модель швидкої розробки застосунків (Rapid Application Development, RAD), щоб якомога швидше завершити розробку функцій застосунку з мінімальною участю користувача в процесі розробки.

5. Що б свідчило про необхідність оновлення вимог користувача?

- документовані процеси;
- раннє постачання продукту;
- відповідність функціональним вимогам користувача;
- збільшення вартості.

6. Який приклад демонструє важливість чіткого визначення бізнес-проблем та вимог користувачів на початку процесу розробки системи?

○ зосередження на вимогах користувачів може запобігти непорозумінням під час процесу розробки, але не вплине на вартість проекту чи терміни;

○ неповні або неправильні вимоги користувачів можуть призвести до значних затримок та збільшення витрат, але їх вирішення на ранніх етапах допомагає уникнути дорогих змін у майбутньому;

○ аналіз потреб користувачів лише на завершальних етапах розробки гарантує, що система відповідає очікуванням стейкхолдерів без марнування ресурсів;

○ покладання виключно на візуальні інструменти гарантуватиме повне розуміння потреб користувачів та усуне ризики затримки проєкту.

7. Коли слід розпочати процес проєктування системи?

○ до визначення бізнес-проблеми, до завершення системного аналізу, під час процесу збору вимог та до створення схем проєктування;

○ після визначення бізнес-проблеми, до завершення системного аналізу, під час процесу збору вимог та після створення схем проєктування;

○ після визначення бізнес-проблеми, після завершення системного аналізу, після завершення процесу збору вимог та після створення схем проєктування;

○ після визначення бізнес-проблеми, перед завершенням системного аналізу, під час процесу збору вимог та перед створенням схем проєктування.

8. Чим відрізняються логічні проєкти від фізичних проєктів?

○ логічні проєкти включають конкретне обладнання, яке використовуватиметься в системі;

○ фізичні проєкти включають бізнес-процеси для візуального представлення потоку даних та зв'язків між ними;

○ логічні проєкти забезпечують загальне представлення системи;

○ фізичні проєкти використовуються для побудови логічного проєкту.

9. Яке твердження найкраще ілюструє важливість процесу проєктування системи для забезпечення успішного результату?

○ проєктування системи зосереджується, головним чином, на виборі правильного обладнання для задоволення вимог до продуктивності, забезпечуючи функціонування системи належним чином;

○ процес проєктування системи створює детальні архітектурні та інформаційні специфікації, балансуючи логічні та фізичні компоненти, а також виявляє потенційні проблеми на ранній стадії за допомогою прототипів та ітеративного проєктування;

○ проєктування системи стосується лише розробки інтерфейсу користувача, зосереджуючись, головним чином, на естетиці та зручності використання.

○ процес проєктування системи в основному стосується створення необхідної документації та не потребує врахування відгуків користувачів чи майбутньої масштабованості.

10. Яка ключова мета використання орієнтованого на користувача підходу до проєктування?

○ надання найдешевшого рішення користувачеві в найкоротші терміни;

○ інтеграція відгуків клієнтів один раз під час процесу розробки;

○ розробка детальної документації та звітів про функціональність системи;

○ використання інформації від користувачів протягом усього процесу проєктування для забезпечення найкращого рішення, що відповідає потребам користувачів.

11. Чому важливо проводити дослідження перед розробкою мобільного застосунку?

○ дослідження допомагає створити застосунок з найменшими витратами часу та коштів;

○ дослідження підказує, як розробити продукт;

○ дослідження запобігає дороговартісним помилкам та значним змінам на подальших етапах процесу розробки;

○ дослідження надає нові відомості про потреби користувачів.

12. Що слід враховувати при розробці дизайну мобільних застосунків?

- модність дизайну;
- заздалегідь визначений час відгуку;
- зручна навігація пальцями;
- складний, але функціональний дизайн.

14. Яке твердження є правильним щодо UI та UE?

- UI та UE працюють незалежно один від одного;
- між UI та UE існує мінімальна різниця;
- UE враховує взаємодію користувача з дизайном;
- UI розглядає процес дизайну, охоплюючи досвід користувача з усіма аспектами діяльності компанії.

15. Яке твердження найкраще описує роль ітеративного дизайну та відгуків користувачів у підході до дизайну, орієнтованого на користувача (User-Centered Design, USD), для створення продукту?

○ ітеративний дизайн гарантує, що продукт буде ідеальним з самого початку, уникаючи необхідності збору відгуків користувачів після першого запуску;

○ відгуки користувачів в UCD актуальні лише на початковій фазі, а ітерація зосереджена виключно на технічній продуктивності;

○ UCD використовує постійний відгук користувачів для керівництва ітеративними вдосконаленнями, створюючи продукт, який відповідає потребам користувачів та бізнес-цілям;

○ ітеративний дизайн в UCD зосереджений переважно на естетичних покращеннях, знижуючи пріоритетність вимог до функціональності та доступності.

16. У контексті кібербезпеки, що в першу чергу означає цифрова довіра?

☐ стандарти шифрування, що використовуються в цифрових комунікаціях

☐ довіра стейкхолдерів до здатності компанії захищати дані та системи;

☐ цифрові сертифікати, що використовуються для автентифікації вебсайту;

☐ надійність кваліфікованих електронних підписів.

17. Який суттєвий виклик кібербезпеці створює зростання хмарних обчислень?

☐ спрощена ІТ-інфраструктура;

☐ зменшення потреб у зберіганні даних;

☐ унікальні ризики, такі як витoki даних, несанкціонований доступ та порушення цілісності спільних ресурсів;

☐ зменшення потреби в мережевій безпеці.

18. Що таке Privacy by Design?

☐ принцип, який виступає за цілеспрямоване вбудовування конфіденційності в дизайн ІТ-систем і бізнес-практик;

☐ принцип, що зосереджений на технічних аспектах впровадження засобів контролю захисту даних, таких як шифрування та анонімізація;

☐ принцип, що зосереджений лише на зберіганні та управлінні даними сторонніми постачальниками;

☐ принцип, що реагує на захист даних і враховується лише після того, як стався витік даних.

19. Загальний регламент про захист даних (GDPR) поширюється на _____.

☐ лише на компанії, що знаходяться в ЄС;

☐ лише на компанії, що знаходяться за межами ЄС;

☐ на всі компанії, які здійснюють оброблення персональних даних громадян ЄС, незалежно від країни їх реєстрації чи фактичного місцезнаходження;

○ лише на організації, що мають фізичну присутність в ЄС.

20. Стандарт Міжнародної організації зі стандартизації (ISO) 27701:2025 є розширенням стандарту ISO 27001. Чим керує стандарт ISO 27701:2025?

○ як керувати конфіденційною інформацією;

○ як керувати відносинами з третіми сторонами у сфері зберігання даних;

○ як створювати нові закони про конфіденційність даних;

○ як інтегрувати конфіденційність в ІТ-системи та бізнес-практики.

21. Як називається практика впровадження засобів контролю конфіденційності в проєктування та розробку програмних систем та бізнес-практик?

○ конфіденційність за дизайном;

○ інженерія конфіденційності

○ Інженерія безпеки

○ мережа конфіденційності.

22. Який міжнародний стандарт забезпечує основу для системи управління інформаційною безпекою?

○ ISO/IEC 27001;

○ ISO 9001;

○ ISO 31000.

23. Розробляючи вебтехнології, компанії та ІТ-фахівці несуть соціальну відповідальність за _____.

○ максимізацію прибутку;

○ забезпечення залучення користувачів;

○ захист персональної інформації та конфіденційності користувачів;

○ мінімізацію витрат на розробку

24. Розробляючи вебтехнології, компанії та ІТ-фахівці несуть соціальну відповідальність за _____.

25. Які компанії, найімовірніше, потребуватимуть, щоб їх центри обробки даних були сертифіковані за стандартом ISO/IEC 27001?

- ☐ великі компанії;
- ☐ малий онлайн-бізнес;
- ☐ індивідуальні клієнти.

26. Щодо дотримання регламенту GDPR, який принцип вимагає збирати лише ті дані користувачів, які конче необхідні для бізнес-цілей?

- ☐ максимізація даних;
- ☐ прозорість;
- ☐ мінімізація даних;
- ☐ розширення даних.

27. Яка головна мета проведення аудиту в контексті безпеки даних та конфіденційності?

- ☐ узгодити практику компанії зі стандартами безпеки даних і конфіденційності;
- ☐ змінити організаційну структуру компанії;
- ☐ оцінити ефективність роботи співробітників.

28. Що означає термін «прозорість» у регламенті GDPR?

- ☐ чіткість у тому, як використовуються та управляються дані користувачів;
- ☐ фізичне планування офісу компанії;
- ☐ державні транзакції.

29. Що з переліченого можна вважати значним ризиком, пов'язаним із прогалинами в політиці безпеки даних?

- ☐ зростання плинності кадрів;
- ☐ втрата довіри користувачів та потенційні штрафи з боку регуляторних органів;
- ☐ зниження цін на акції.

30. Яке точне визначення аналітики даних?

- ☐ процес збору та зберігання великих обсягів даних;

☐ практика дослідження, очищення та перетворення даних для виявлення корисних ідей;

☐ використання статистичних методів для прогнозування майбутніх ринкових тенденцій;

☐ інтеграція структурованих та неструктурованих даних у централізовану базу даних.

31. Що таке великі дані?

☐ сукупність даних, яка є занадто великою для оброблення традиційними методами;

☐ набір даних з високою точністю;

☐ набір даних, що містить лише структуровані дані;

☐ процес аналізу даних для виявлення прихованих закономірностей та тенденцій.

32. Яка проблема пов'язана зі збором та використанням великих даних?

☐ брак доступних джерел даних;

☐ низька швидкість оброблення;

☐ недостатня ємність сховища;

☐ обсяг, швидкість та різноманітність даних.

33. Який завершальний етап процесу бізнес-аналітики перед початком наступного циклу?

☐ інтерпретація результатів;

☐ статистичний аналіз;

☐ впровадження;

☐ підготовка даних.

34. Які переваги впровадження бізнес-аналітики в компанії?

☐ оптимізоване прийняття рішень, підвищення ефективності та покращення безпеки даних;

☐ підвищення задоволеності, зменшення плинності кадрів та покращення управління ланцюгом поставок;

☐ покращена візуалізація даних, покращене управління даними та збільшення ємності зберігання даних;

☐ збільшення частки ринку, підвищення впізнаваності бренду та збільшення доходу від продажів.

35. Яка основна мета візуалізації даних у контексті бізнес-аналітики?

☐ представити дані в естетично привабливому вигляді;

☐ узагальнити складні дані та виділити закономірності та тенденції;

☐ забезпечити безпеку даних та захистити конфіденційну інформацію;

☐ зберігати та впорядковувати великі обсяги даних для подальшого аналізу.

36. Яка основна мета бізнес-аналітики?

☐ збирати та зберігати великі обсяги даних офлайн;

☐ аналізувати дані та генерувати висновки для прийняття рішень;

☐ виявляти та вирішувати проблеми якості даних;

☐ класифікувати дані.

37. Який крок у процесі прийняття рішень на основі даних включає використання таких методів, як класифікація, регресія, кластеризація та асоціативний аналіз, для виявлення закономірностей та тенденцій у зібраних даних?

☐ визначення проблеми;

☐ збір даних;

☐ інтерпретація аналітики;

☐ аналіз даних.

38. Який варіант відповіді найкраще описує роль вебаналітики в компаніях?

☐ виявлення можливостей для покращення;

☐ підвищення офлайн-продуктивності;

☐ зниження взаємодії з користувачем;

☐ прийняття суб'єктивних рішень.

39. Що компанії можуть вимірювати за допомогою вебаналітики?

- емоційні реакції користувачів;
- вплив маркетингових кампаній;
- колірна схема вебсайту компанії.

40. Яка основна мета А/В-тестування в оптимізації вебсайту?

- оптимізувати контент вебсайту за допомогою релевантних ключових слів;
- проаналізувати поведінку користувачів та джерела вебтрафіку;
- порівняти та визначити, який варіант вебсторінки працює краще;
- отримати високоякісні зворотні посилання для вебсайту.

41. Що представляє собою проєкт?

- починання, спрямоване на створення продукту або послуги з чітко визначеними датами початку та завершення;
- застосування знань, навичок, інструментів та технологій до проєктної діяльності;
- портфель проєктів, що управляються як група для досягнення стратегічних цілей компанії;
- починання, що використовується для підвищення цінності в наданні продуктів та послуг.

42. Яке твердження найкраще розрізняє стейкхолдерів та команди в сфері управління проєктами?

- зацікавлені сторони є частиною команди;
- команда є частиною сфери діяльності стейкхолдера;
- команда зацікавлена в результатах проєкту;
- стейкхолдери вимірюють рівень якості проєкту.

43. Якими є перші три кроки у розробці плану проєкту?

- визначити обсяг, визначити бюджет та визначити ресурси;
- визначити обсяг, визначити бюджет та оцінити ризики;

- оцінити ризики, визначити обсяг та визначити ресурси;

- визначити бюджет, визначити ресурси та оцінити ризики.

44. Визначте п'ять основних етапів розробки та управління проектом.

- ініціювання проекту, графік проекту, ризики виконання та моніторингу проекту, а також закриття проекту;

- статут проекту, планування проекту, реалізація та моніторинг проекту, а також закриття проекту;

- ініціювання проекту, планування проекту, ризики розробки та моніторингу проекту, а також закриття проекту;

- ініціювання проекту, планування проекту, виконання проекту, моніторинг та контроль.

45. Які кроки входять до виконавчої частини проекту?

- координація ресурсів, управління ризиками, вирішення проблем та забезпечення контролю якості;

- розподіл усіх ресурсів, моніторинг ризиків, дотримання графіка та встановлення заходів контролю якості;

- управління ресурсами, моніторинг ризиків, вирішення питань графіка та вимірювання контролю якості;

- моніторинг ресурсів, управління ризиками, виправлення проблем та гарантування контролю якості.

46. Яке твердження щодо наукових досліджень та інновацій у сфері технологій не є правильним?

- вони сприяли значному зростанню ринку аналізу даних та прогнозної аналітики;

- глобальне зростання технологій спостерігається в кількох країнах у різних секторах бізнесу;

- дослідницька та інноваційна діяльність сприяє розвитку нових технологій та процесів;

○ дослідницька та інноваційна діяльність фінансується переважно урядами країн.

47. Яка мета наукового дослідження стосовно технологій?

○ наукове дослідження є передумовою інновацій;

○ наукове дослідження є вторинним компонентом у процесі проєктування;

○ наукове дослідження надає повне уявлення про те, як технологія має функціонувати;

○ наукове дослідження перешкоджає прийняттю інновацій споживачами.

48. Яке з наведених тверджень є правильним щодо процесу інновацій?

○ процес інновацій проходить через систематичні кроки: концепція, впровадження, маркетинг та комунікація;

○ аналіз та запис інформації є першим кроком у визначенні та виборі ресурсів;

○ процес наукового дослідження та пов'язані з ним навички є основою дослідницького процесу;

○ інформацію необхідно записувати та негайно передавати.

49. Чому ефективно управління ризиками важливе для організацій, що займаються глобальним обміном даними?

○ дозволяє уникнути затримок із термінами виконання проєктів;

○ дозволяє забезпечити використання даних для маркетингових цілей;

○ дозволяє мінімізувати потенційні загрози для даних і систем;

○ дозволяє спростити процес обміну даними.

50. Яка ключова перевага міжфункціональних команд?

○ підвищена конкуренція між підрозділами;

○ покращена комунікація та співпраця;

○ обмежений обмін інформацією;

☐ повільніші процеси прийняття рішень.

51. Яка практика передбачає постійне тестування систем штучного інтелекту з використанням різноманітних наборів даних для виявлення неточних упереджень?

- ☐ шифрування;
- ☐ прозорість;
- ☐ тестування на упередженість;
- ☐ підзвітність.

52. Яке поняття стосується здатності описувати логіку та рішення моделі штучного інтелекту зрозумілим способом?

- ☐ обмеження;
- ☐ управління;
- ☐ пояснювальність;
- ☐ анонімізація.

53. Хто має бути відповідальним за прийняття важливих рішень на основі штучного інтелекту?

- ☐ система штучного інтелекту;
- ☐ державні регулятори;
- ☐ керівники компаній;
- ☐ люди-наглядачі.

54. Який термін описує відкритість та прозорість функціонування системи штучного інтелекту?

- ☐ пояснимість;
- ☐ прозорість;
- ☐ справедливість;
- ☐ нагляд.

55. Яка практика допомагає забезпечити, щоб системи штучного інтелекту доповнювали людей, а не замінювали їхню свободу дій?

- ☐ автоматизоване прийняття рішень;
- ☐ підзвітність;
- ☐ технологічне безробіття;
- ☐ людський контроль.

ПРАКТИЧНА РОБОТА 1

Тема: Розроблення та специфікація вимог до програмного забезпечення.

Мета роботи: оволодіння навичками аналізу, розроблення та специфікації вимог до програмного забезпечення.

Теоретичні відомості

Вимоги (requirements) до програмного забезпечення – це можливості або умови, яким повинна відповідати система або проєкт. Вимоги можуть виражатися у вигляді текстових тверджень і графічних моделей. Розробка вимог може включати виявлення вимог (збір, розуміння, розгляд та з'ясування потреб стейкхолдерів), аналіз (перевірка цілісності і закінченості), специфікація (документування вимог) та верифікація (перевірка правильності).

Згідно класифікації FURPS+ вимоги діляться на такі категорії, які можна згрупувати в три групи:

1. **Функціональні вимоги (Functionality)** – це те, що хоче користувач – властивості, можливості, безпека.
2. **Нефункціональні вимоги**, які визначають якісні характеристики програмної системи:

- **зручність використання (Usability)** – забезпечення комфортної взаємодії користувача із програмною системою, зрозумілого інтерфейсу, привабливого дизайну та наявності необхідної супровідної документації.

- **надійність (Reliability)** – здатність програмної системи стабільно функціонувати, мінімізувати кількість відмов, забезпечувати відновлення після збоїв, гарантувати передбачуваність роботи та точність виконання операцій.

- **продуктивність (Performance)** – ефективність функціонування програмної системи, що характеризується швидкістю виконання операцій, часом відгуку, пропускнуою здатністю, а також раціональним використанням оперативної пам'яті та інших обчислювальних ресурсів.

- **підтримуваність (Supportability)** – здатність програмної системи до тестування, супроводу, модифікації, масштабування, конфігурування, адаптації та моніторингу в процесі експлуатації.

3. **Обмеження (Constraints)**, які необхідно враховувати під час розроблення програмної системи:

- **обмеження проєктування (Design Constraints)** – вимоги, пов'язані з використанням зовнішніх пристроїв, баз даних, програмних платформ та інших елементів технологічного середовища.

- **обмеження реалізації (Implementation Constraints)** – необхідність дотримання визначених стандартів, мов і технологій програмування, архітектурних підходів, методологій розроблення (наприклад, **Test-Driven Development (TDD)**) та інших технічних вимог.

- **обмеження інтерфейсів (Interface Constraints)** – вимоги щодо сумісності та взаємодії програмної системи з іншими програмними продуктами, сервісами або інформаційними системами.

- **фізичні обмеження (Physical Constraints)** – вимоги, зумовлені характеристиками апаратного забезпечення, програмних платформ, кінцевих користувацьких пристроїв, засобів відображення інформації (моніторів, сенсорних екранів тощо) та іншого технічного обладнання.

Основні методи виявлення вимог до програмної системи включають:

- **комунікаційні методи** – проведення інтерв'ю, опитувань та анкетування користувачів, замовників і інших стейкхолдерів з метою з'ясування їхніх потреб та очікувань;

- **колективні методи генерації ідей** – використання мозкового штурму, робочих зустрічей, семінарів і воркшопів для спільного формування та уточнення вимог;

- **методи спостереження** – дослідження реальних бізнес-процесів шляхом безпосереднього спостереження за

діяльністю працівників, зокрема із застосуванням методики «фотографії робочого дня»;

- **аналіз нормативної документації** – вивчення законодавчих актів, стандартів, внутрішніх регламентів та інших документів, що визначають вимоги до програмної системи;

- **аналіз моделей діяльності** – дослідження існуючих моделей бізнес-процесів, інформаційних потоків та організаційної структури компанії;

- **аналіз конкурентних рішень** – вивчення функціональних можливостей, переваг і недоліків аналогічних програмних продуктів, представлених на ринку.

- **аналіз експлуатаційної статистики** – використання даних щодо функціонування попередніх версій програмної системи, включаючи статистику використання, журнали подій, показники продуктивності та відгуки користувачів.

Вимоги схильні до проблем двозначності, неповноти, і неузгодженості. Їх усунення на етапі визначення вимог коштує на кілька порядків менше, ніж усунення цих же проблем на пізніх стадіях розробки. Аналіз вимог спрямований на вирішення даних проблем. Існує технічний компроміс між занадто невизначеними вимогами та вимогами настільки деталізованими, що вони:

- 1) вимагають багато часу для розробки, іноді навіть ризикують застаріти до кінця розробки;
- 2) обмежують можливі способи реалізації;
- 3) є занадто дорогими.

У процесі аналізу та специфікації вимог результати документуються у вигляді артефактів, кожен із яких відображає окремий аспект майбутньої програмної системи:

- **модель прецедентів (Use Case Model)** – містить сукупність типових сценаріїв взаємодії користувачів із

програмною системою та визначає її функціональні (поведінкові) вимоги;

- **додаткова специфікація (Supplementary Specification)** – призначена для опису нефункціональних вимог, зокрема вимог до продуктивності, надійності, безпеки, звітності, документування, супроводу, ліцензування та інших якісних характеристик програмної системи;

- **словник термінів (Glossary)** – забезпечує єдине трактування ключових понять предметної області шляхом наведення визначень основних термінів, що використовуються в документації проекту;

- **документ бачення (Vision)** – формулює стратегічні цілі проекту, його основні ідеї та високорівневі вимоги, які визначають концепцію майбутньої програмної системи. Цей документ є стислим описом проекту та використовується для швидкого ознайомлення зацікавлених сторін із його призначенням і напрямками розвитку;

- **бізнес-правила (Business Rules)** – містять нормативні вимоги, корпоративні політики та регламенти, що впливають на функціонування програмної системи. До них можуть належати внутрішні правила компанії, вимоги до бухгалтерського обліку, податкове законодавство, галузеві стандарти та інші нормативно-правові акти. Бізнес-правила можуть бути включені до додаткової специфікації або оформлені як окремий артефакт.

Технічне завдання – це результат виявлення вимог до програмної системи. Зазвичай створюється на основі затверджених шаблонів. Шаблон визначає структуру, зміст і стиль документа. Окрім формулювання вимог технічне завдання включає також проєктні питання. На початку документа викладають бізнес-контекст проєкту, учасників

проєкту та основні обмеження. В кінці подають план-графік робіт, бюджет, ризики, документацію та ін.

Приклад змісту технічного завдання:

1 Вступ.

1.1 Мета і масштаб проєкту.

1.2 Діловий контекст.

1.3 Учасники проєкту.

1.4 Ідеї рішень.

1.5 Огляд документу.

2 Системні сервіси.

2.1 Границі системи.

2.2 Функціональні вимоги.

2.3 Вимоги до даних.

3 Системні обмеження.

3.1 Обмеження стосовно інтерфейсу.

3.2 Обмеження стосовно продуктивності.

3.3 Обмеження стосовно безпеки.

3.4 Експлуатаційні обмеження.

3.5 Політичні та юридичні обмеження.

3.6 Інші обмеження.

4 Проєктні питання.

4.1 Невирішені проблеми.

4.2 Попередній календарний план.

4.3 Попередній кошторис.

Додатки

Глосарій

Бізнес-документи і форми

Посилання

Порядок виконання роботи

1. Ознайомитися з теоретичним матеріалом з теми.

2. Сформувані команди чисельністю 3–5 осіб та вибрати тему навчального міні-проєкту (по одному проєкту на команду).

3. Проаналізувати потенційну програмну систему з точки зору замовника та розробити документ бачення проєкту (Vision).

4. У ході роботи над вимогами сформувати словник термінів (глосарій).

5. Розробити модель випадків використання програмної системи (Use Case).

6. Розробити додаткову специфікацію.

ПРАКТИЧНА РОБОТА 2

Тема: Аналіз ринку та конкурентів для мобільного застосунку

Мета роботи: оволодіння навичками проведення дослідження ринку з метою реалізації бізнес-ідеї мобільного застосунку, ідентифікації основних конкурентів, аналізу їхніх сильних і слабких сторін для формування унікальної ціннісної пропозиції.

Порядок виконання роботи

1. Оберіть бізнес-ідею для вашого майбутнього мобільного застосунку (наприклад, доставка здорової їжі, застосунок для вивчення мов, трекер звичок).

2. За допомогою пошукових систем та магазинів мобільних застосунків (Google Play, App Store) знайдіть 3-5 прямих та непрямих конкурентів.

3. Проаналізуйте сторінки конкурентів у магазинах мобільних застосунків. Зверніть увагу на кількість завантажень, рейтинг, відгуки користувачів (особливо негативні), візуальне оформлення та перелік функцій.

4. Створіть у Google Sheets порівняльну таблицю, що містить інформацію про назву конкурента, його ключові функції, модель монетизації, сильні сторони, слабкі сторони (на основі відгуків).

5. На основі результатів аналізу недоліків конкурентних рішень і виявлених незадоволених потреб цільового ринку

необхідно сформулювати унікальну ціннісну пропозицію (Unique Value Proposition, UVP) мобільного застосунку. Її доцільно подати за таким шаблоном: «Мобільний застосунок [назва] допомагає [цільова аудиторія] вирішити [бізнес-проблему або потребу] завдяки [ключовій функціональній можливості або перевазі], що вигідно відрізняє його від [конкурентних рішень]».

Вимоги до звіту

Надайте текстовий документ, що містить: опис вашої бізнес-ідеї, заповнену порівняльну таблицю конкурентів, сформульовану унікальну ціннісну пропозицію.

ПРАКТИЧНА РОБОТА 3

Тема: Вибір та економічне обґрунтування моделі монетизації.

Мета роботи: оволодіння навичками обґрунтування моделі монетизації для власного проєкту, а також розрахунку його основних економічних показників.

Порядок виконання роботи

1. Для вашого проєкту проаналізуйте та оберіть одну або комбінацію моделей монетизації (Freemium, Subscription, In-App Purchases, Ads).

2. У текстовому документі обґрунтуйте ваш вибір: чому ця модель найкраще підходить для вашої аудиторії та функціоналу.

3. Створіть нову таблицю в Google Sheets для фінансового моделювання.

4. Сформулюйте та запишіть ключові припущення:
- прогнозована кількість нових користувачів на місяць;
 - конверсія в платного користувача (%);
 - вартість підписки або середня вартість покупки;
 - орієнтовні щомісячні витрати (сервер, маркетинг).

5. На основі припущень розрахуйте прогнозований щомісячний дохід, витрати та чистий прибуток на 12 місяців уперед.

6. Визначте орієнтовний місяць, коли проєкт досягне точки беззбитковості.

Вимоги до звіту

Надайте текстовий документ з обґрунтуванням моделі монетизації та посилання на електронну таблицю з розрахунками.

ПРАКТИЧНА РОБОТА 4

Тема: Розробка бізнес-моделі мобільного застосунку за методологією Lean Canvas.

Мета роботи: оволодіння навичками структурування бізнес-ідеї, валідації її ключових гіпотез та візуалізації бізнес-моделі за допомогою інструменту Lean Canvas.

Порядок виконання роботи

Заповніть блоки шаблону Lean Canvas, спираючись на бізнес-ідею, сформульовану у попередній роботі:

- **Problem.** Опишіть 1-3 головні проблеми, які існують у ваших потенційних клієнтів.

- **Customer Segments.** Визначте вашу цільову аудиторію.

- **Unique Value Proposition.** Перенесіть та уточніть унікальну ціннісну пропозицію з попередньої роботи.

- **Solution.** Опишіть ключові функції вашого застосунку, що вирішують проблеми.

- **Channels.** Перелічіть канали, через які ви будете залучати користувачів (ASO, соцмережі, контент-маркетинг).

- **Revenue Streams.** Опишіть, як застосунок буде заробляти гроші (підписка, реклама, комісія).

- **Cost Structure.** Перелічіть основні статті витрат (розробка, маркетинг, сервери).

– **Key Metrics.** Визначте ключові показники, за якими будете вимірювати успіх (кількість завантажень, Retention Rate).

– **Unfair Advantage.** Подумайте, що у вас є такого, що важко скопіювати конкурентам.

Вимоги до звіту

Надайте посилання на вашу дошку в Miro/FigmaJam з правами на перегляд або експортуйте канву у форматі PDF/PNG.

ПРАКТИЧНА РОБОТА 5

Тема: Аналіз доцільності бізнес-ідеї мобільного застосунку.

Мета роботи: Оволодіння методами проведення аналізу доцільності бізнес-ідеї.

Порядок виконання роботи

Аналіз доцільності бізнес-ідеї проводять до виділення значних фінансових ресурсів на її реалізацію, перед написанням бізнес-плану. Заповніть такі розділи звіту про аналіз доцільності бізнес-ідеї, сформульованої у попередній роботі.

Розділ 1. Ринкова доцільність.

У цьому розділі основна увага зосереджується на аналізі ринкової ситуації та її динаміки, зокрема галузевих тенденцій, ринкових сегментів, рівня конкуренції, особливостей цільових сегментів, каналів розподілу, а також маркетингової та цінової політики, пов'язаних із запропонованою бізнес-ідеєю.

Необхідно визначити, наскільки результати галузевого аналізу, актуальні тенденції розвитку ринку та потреби й уподобання потенційних клієнтів підтверджують доцільність запропонованої бізнес-ідеї. Окрему увагу слід приділити обґрунтуванню напрямів і перспектив її подальшого розвитку.

У межах цього аналізу потрібно охарактеризувати цільовий сегмент ринку та визначити особливості реалізації бізнес-стратегії залежно від обраної моделі взаємодії з клієнтами: B2C (Business-to-Consumer), B2B (Business-to-Business) або B2G (Business-to-Government). Також необхідно сформувати маркетингову та комунікаційну стратегії, визначити підходи до ціноутворення і вибрати оптимальні канали поширення програмного продукту.

Важливою складовою є аналіз конкурентного середовища. Необхідно ідентифікувати основних конкурентів у відповідній галузі та на цільовому ринку, оцінити їхні сильні й слабкі сторони, особливості пропонованих продуктів та визначити конкурентні переваги власної бізнес-ідеї.

Розділ 2. Технічна доцільність.

У цьому розділі необхідно обґрунтувати технічну здійсненність запропонованої бізнес-ідеї та продемонструвати наявність у команди розробників відповідних технічних знань, професійних компетентностей, практичних навичок і ресурсів, необхідних для створення цінності, розроблення та надання запланованого програмного продукту.

У межах цього аналізу слід визначити, чи володіє команда розробників необхідними технічними знаннями, навичками та компетентностями для самостійного створення запропонованого програмного продукту. Також потрібно оцінити доступність на ринку праці висококваліфікованих фахівців, яких можна залучити до команди розробників для забезпечення реалізації проєкту.

Окремо необхідно представити операційний план розроблення та надання програмного продукту, у якому визначаються основні етапи створення продукту, необхідні ресурси, відповідальні виконавці, технології та інструменти розроблення, а також порядок його тестування,

впровадження, супроводу й подальшого надання користувачам.

Розділ 3. Доцільність реалізації бізнес-ідеї.

У цьому розділі необхідно розкрити механізм створення, забезпечення та отримання цінності запропонованою бізнес-моделлю, зокрема визначити, яким чином бізнес формує цінність для цільового ринку, забезпечує її доставку споживачам та генерує економічну вигоду, у тому числі дохід від реалізації програмного продукту.

Необхідно представити розроблену бізнес-модель за методологією Business Model Canvas (BMC) та оцінити, наскільки її складові підтверджують спроможність бізнесу створювати ціннісні пропозиції у формі програмних продуктів, послуг або рішень, забезпечувати їх ефективно надання цільовим клієнтам та отримувати відповідну економічну цінність на ринку.

Окрему увагу слід приділити деталізації бізнес-моделі, необхідній для формування обґрунтованих фінансових прогнозів. Потрібно визначити основні джерела та механізми формування доходів, оцінити їхню стабільність і перспективність, а також з'ясувати, чи передбачає бізнес-модель використання декількох потоків доходу, що може сприяти диверсифікації фінансових надходжень та підвищенню стійкості бізнесу.

Розділ 4. Доцільність структури управління.

У цьому розділі необхідно охарактеризувати організаційне забезпечення та кадрові аспекти реалізації бізнес-ідеї, а також визначити підходи до управління інтелектуальною власністю, розробленням програмного продукту та операційною діяльністю.

Насамперед слід представити організаційну структуру бізнесу, яка відображає розподіл управлінських повноважень, функцій і відповідальності між учасниками

команди. Необхідно також визначити поточні та перспективні кадрові потреби компанії, встановити необхідну кількість працівників і сформувавши перелік професійних компетентностей, навичок та кваліфікацій, необхідних для ефективного виконання відповідних функцій.

Окрему увагу потрібно приділити плану управління інтелектуальною власністю. Цей план має охоплювати питання захисту та використання торгової марки, торгового найменування, патентів, авторських прав, комерційної таємниці та інших об'єктів інтелектуальної власності, що мають значення для діяльності бізнесу.

Важливою складовою розділу є визначення плану розвитку програмного продукту – від створення мінімально життєздатного продукту (Minimum Viable Product, MVP) або прототипу до його вдосконалення, масштабування та повної комерціалізації. Необхідно визначити основні етапи такого розвитку, критерії переходу між ними та очікувані результати.

Для планування та контролю операційної діяльності доцільно розробити діаграму Ганта, яка відображатиме перелік основних завдань, послідовність їх виконання, строки реалізації, взаємозв'язки між роботами та відповідальних виконавців. Такий інструмент дає змогу систематизувати бізнес-операції, контролювати дотримання встановлених термінів і своєчасно виявляти відхилення від запланованого графіка.

Розділ 4. Фінансова доцільність.

У цьому розділі необхідно оцінити фінансову доцільність і довгострокову економічну стійкість запропонованої бізнес-ідеї, визначивши співвідношення між необхідними інвестиціями, очікуваними витратами та прогнозованими економічними результатами. Особливу увагу слід приділити оцінюванню прибутковості бізнесу,

його здатності генерувати позитивний грошовий потік, досягати беззбитковості та забезпечувати безперервність діяльності в довгостроковій перспективі.

У межах фінансового аналізу необхідно визначити, чи існує на цільовому ринку розрив між попитом і пропозицією (Demand–Supply Gap, DSG) щодо запропонованого програмного продукту, а також оцінити основні показники його економічної ефективності. Зокрема, потрібно проаналізувати, чи є річний грошовий потік (Cash Flow Analysis, CFA) позитивним, а ціна реалізації одиниці продукту (Selling Price per Unit, SPU) – вищою за собівартість одиниці продукту (Cost per Unit, CPU). Необхідно також визначити точку беззбитковості в натуральному вираженні (Break-Even Units, BEU) та оцінити можливість її досягнення протягом року.

Важливим показником є маржа внеску (Contribution Margin, CM), яка характеризує частину доходу, що залишається після покриття змінних витрат і може бути спрямована на покриття постійних витрат та формування прибутку. Для оцінювання інвестиційної привабливості бізнес-ідеї необхідно розрахувати рентабельність інвестицій (Return on Investment, ROI) та термін окупності (Payback Period, PBP), визначивши їх відповідність встановленим критеріям економічної ефективності.

Завершальним етапом фінансового аналізу має стати формування прогнозних значень ключових фінансово-економічних показників, зокрема SPU, CPU, CM, BEU, ROI та PBP. Отримані результати дають змогу оцінити прибутковість, інвестиційну привабливість і перспективи довгострокової фінансової стійкості запропонованої бізнес-моделі.

Розділ 5. Регуляторна доцільність.

У цьому розділі необхідно здійснити аналіз правових, нормативних та зовнішніх факторів, які можуть впливати на

реалізацію та розвиток запропонованої бізнес-ідеї. Особливу увагу слід приділити визначенню нормативно-правових вимог, яких має дотримуватися бізнес, а також оцінюванню сприятливості зовнішнього середовища для його функціонування та подальшого розвитку. Для комплексної оцінки доцільно застосувати PESTEL-аналіз, що дає змогу дослідити політичні, економічні, соціальні, технологічні, екологічні та правові чинники, здатні створювати як можливості, так і потенційні загрози для бізнесу. У межах цього аналізу необхідно визначити, які державні нормативно-правові акти, стандарти та регуляторні вимоги поширюються на запропоновану бізнес-ідею та які зобов'язання вони створюють для компанії.

Окремо слід оцінити вплив політичного та економічного середовища країни на перспективи реалізації бізнес-ідеї, зокрема стабільність економіки, державну політику, рівень підтримки підприємництва та інвестиційний клімат. Не менш важливим є дослідження соціальних та екологічних факторів, які можуть впливати на попит, поведінку споживачів і умови функціонування бізнесу. Також необхідно проаналізувати технологічне середовище, визначивши доступність сучасних технологій, цифрової інфраструктури, фахівців та інших технологічних ресурсів, необхідних для реалізації бізнес-ідеї.

Завершальним елементом є оцінювання правового середовища, зокрема рівня його сприятливості для започаткування та ведення бізнесу, наявності правових механізмів захисту підприємницької діяльності та відповідності нормативного регулювання потребам запропонованої бізнес-моделі.

Розділ 6. Доцільність стратегії виходу.

У цьому розділі необхідно визначити стратегічні підходи до подальшого розвитку, масштабування, передачі

права власності або припинення діяльності компанії. Основна увага приділяється формуванню довгострокової стратегії підприємця, яка має передбачати можливі варіанти виходу з бізнесу, зміну структури власності, залучення партнерів та інвесторів, масштабування діяльності, а також механізми мінімізації ризиків у разі виникнення операційних труднощів. Такий підхід дає змогу забезпечити стійкість бізнесу та створити умови для отримання економічної вигоди у випадку його успішного розвитку.

У межах цього розділу необхідно визначити можливість переходу засновника (генерального директора) від оперативного управління до іншої форми участі у власності бізнесу як одного з можливих сценаріїв виходу. Також слід розглянути стратегії внутрішнього та зовнішнього зростання, які можуть використовуватися для масштабування бізнесу або як механізм поступового виходу власника з операційної діяльності.

Важливим питанням є визначення частки власності, яку засновник планує зберегти у разі залучення партнерів, інвесторів або інших учасників бізнесу. Необхідно також оцінити можливість часткового продажу корпоративних прав у майбутньому, зокрема шляхом приватного розміщення акцій або залучення до числа співвласників друзів, членів родини чи інших інвесторів.

Окрему увагу слід приділити розробленню стратегії припинення діяльності та ліквідації бізнесу у випадку його операційної неефективності, фінансової неспроможності або банкрутства. Така стратегія має передбачати механізми мінімізації можливих фінансових і репутаційних втрат.

Необхідною складовою довгострокового планування є також план наступництва, спрямований на забезпечення безперервності функціонування бізнесу у випадку тривалої непрацездатності, хвороби, досягнення власником похилого віку або його смерті. Такий план має визначати

порядок передачі управлінських повноважень, прав власності та відповідальності за подальше функціонування компанії.

Вимоги до звіту

Надайте текстовий документ, що містить звіт про аналіз доцільності вашої бізнес-ідеї.

ПРАКТИЧНА РОБОТА 6

Тема: Створення макетів (wireframes) ключових екранів мобільного застосунку.

Мета роботи: оволодіння навичками створення низькодеталізованих макетів (вайрфреймів) для визначення структури, ієрархії контенту та логіки навігації майбутнього мобільного застосунку.

Порядок виконання роботи

1. Зареєструйтесь у Figma та створіть новий дизайн-файл.

2. Визначте 3-4 ключові екрани для вашого проєкту (наприклад: екран реєстрації/входу, головний екран, екран профілю, екран налаштувань).

3. Створіть фрейми (Artboards) для кожного екрану, вибравши розмір екрана мобільного пристрою.

4. Використовуючи базові фігури (прямокутники, кола) та текстові блоки, відтворіть структуру кожного екрану. На цьому етапі кольори, шрифти та зображення неважливі. Мета – показати, де будуть розташовані кнопки, поля вводу, зображення, заголовки та основний текст.

5. Згрупуйте елементи для кращої організації проєкту.

Вимоги до звіту

Надайте посилання на ваш проєкт у Figma з правами на перегляд.

ПРАКТИЧНА РОБОТА 7

Тема: Розробка інтерактивного клікабельного прототипу в Figma.

Мета роботи: Оволодіння навичками створення інтерактивного прототипу для тестування основних користувацьких сценаріїв та демонстрації логіки роботи мобільного застосунку.

Порядок виконання роботи

1. Відкрийте файл із вайрфреймами з попередньої роботи.

2. Перейдіть у режим прототипування (вкладка Prototype на правій панелі).

3. Виберіть інтерактивний елемент (наприклад, кнопку «Увійти»). Навколо нього з'явиться вузол у вигляді кола.

4. Натисніть на вузол і, утримуючи кнопку миші, перетягніть стрілку на той екран, на який має відбутися перехід.

5. Налаштуйте параметри взаємодії у вікні Interaction details: тригер (On click), дія (Navigate to), анімація (Smart animate або Instant).

6. Зв'яжіть усі ключові екрани, щоб створити логічний ланцюжок дій (наприклад, від екрану входу до головного екрану, а звідти – до профілю).

7. Натисніть кнопку «Present» (іконка Play у верхньому правому куті), щоб запустити та протестувати ваш інтерактивний прототип.

Вимоги до звіту

Надайте посилання на готовий інтерактивний прототип у Figma.

ПРАКТИЧНА РОБОТА 8

Тема: Практикум з ASO: підготовка текстових та графічних матеріалів для Google Play.

Мета роботи: оволодіння навичками оптимізації сторінки застосунку (ASO), підготовки набору текстових і візуальних матеріалів для публікації в Google Play.

Порядок виконання роботи

1. Проведіть дослідження ключових слів, релевантних для вашого застосунку.
2. У Google Docs підготуйте текстові матеріали:
 - назва (до 30 символів).
 - короткий опис (до 80 символів): привабливий заклик до дії.
 - повний опис (до 4000 символів): детально опишіть функціонал, переваги та унікальну цінову пропозицію, використовуючи ключові слова.
3. У Figma (або Canva) розробіть візуальні матеріали:
 - іконка (512x512 px): створіть просту, впізнавану та привабливу іконку.
 - скріншоти: створіть 3-5 скріншотів, використовуючи ваш прототип. На кожному скріншоті додайте короткий заголовок, що пояснює функцію мобільного застосунку.

Вимоги до звіту

Надайте текстовий документ, що містить усі текстові матеріали, а також експортовані у форматі PNG іконку та скріншоти.

ПРАКТИЧНА РОБОТА 9

Тема: Знайомство з Android Studio: створення та запуск першого проєкту.

Мета роботи: оволодіння навичками налаштування середовища розробки Android Studio, створення та запуску проєкта на віртуальному пристрої.

Порядок виконання роботи

1. Завантажте та встановіть Android Studio з офіційного сайту. Дотримуйтесь інструкцій майстра установки.

2. Після першого запуску відкрийте Virtual Device Manager (AVD Manager) та створіть новий віртуальний пристрій.

3. Поверніться на стартовий екран та створіть новий проєкт.

4. Оберіть шаблон Empty Views Activity.

5. Вкажіть назву проєкту (наприклад, MyFirstApp), мову (Java або Kotlin) та мінімальний SDK. Натисніть «Finish».

6. Дочекайтесь, поки проєкт буде зібрано. Ознайомтеся зі структурою проєкту (папки java, res, файл AndroidManifest.xml).

7. Запустіть проєкт, вибравши створений емулятор у списку пристроїв. Переконайтеся, що на екрані емулятора з'явився мобільний застосунок із написом «Hello World!».

Вимоги до звіту

Надайте текстовий документ, що містить такі скріншоти:

- 1) головне вікно Android Studio;
- 2) вікно AVD Manager зі створеним емулятором;
- 3) запущений застосунок «Hello World!» на емуляторі.

ПРАКТИЧНА РОБОТА 10

Тема: Верстка простого UI в Android Studio.

Мета роботи: оволодіння навичками створення статичного користувацького інтерфейсу для екрана Android-застосунку за допомогою мови розмітки XML та візуального редактора.

Порядок виконання роботи

1. Відкрийте проєкт, створений у попередній роботі.
2. Перейдіть до файлу розмітки `res/layout/activity_main.xml`.
3. Використовуючи режим Split (код та візуальний редактор одночасно), видаліть стандартний компонент `TextView`.
4. Відтворіть один із ваших вайрфреймів (наприклад, екран входу). Для цього з панелі `Palette` перетягніть на екран:
 - два елементи `EditText` (для email та пароля);
 - один елемент `Button` (для входу).
5. Налаштуйте атрибути для кожного елемента в XML:
 - `android:id` – унікальний ідентифікатор;
 - `android:layout_width` та `android:layout_height` – ширина та висота;
 - `android:hint` – підказка для полів вводу;
 - `android:text` – текст для кнопки.
6. Розташуйте елементи на екрані, використовуючи `ConstraintLayout` (прив'яжіть їх один до одного та до меж екрану).
7. Запустіть застосунок на емуляторі та переконайтеся, що інтерфейс виглядає так, як ви планували.

Вимоги до звіту

Надайте файл `activity_main.xml` та скріншот отриманого екрану мобільного застосунку на емуляторі.

ПРАКТИЧНА РОБОТА 11

Тема: Налаштування середовища для крос-платформного розроблення на React Native та створення стартового проєкту.

Мета роботи: оволодіння навичками налаштування програмного середовища для крос-платформної розробки на React Native (Expo) та запуску стартового проєкта.

Порядок виконання роботи

1. Встановіть Node.js з офіційного сайту.
2. Відкрийте термінал і встановіть Expo CLI глобально командою: `npm install -g expo-cli`.
3. Запустіть емулятор Android зі створеним віртуальним пристроєм.
4. У терміналі створіть новий проєкт командою: `expo init MyReactNativeApp`. Оберіть шаблон `blank`.
5. Перейдіть у теку проєкту: `cd MyReactNativeApp`.
6. Відкрийте теку проєкту у Visual Studio Code.
7. Запустіть проєкт на емуляторі командою: `npm run android` або `expo start` і в браузері натисніть «Run on Android device/emulator».
8. Відкрийте файл `App.js`, змініть текст у компоненті `<Text>` та збережіть файл. Переконайтеся, що зміни автоматично відобразилися в емуляторі.

Вимоги до звіту

Надайте скріншоти, що демонструють процес створення та запуску проєкту в терміналі, а також фінальний результат на емуляторі з вашим зміненим текстом.

ПРАКТИЧНА РОБОТА 12

Тема: Створення списку та отримання даних з публічного API в React Native.

Мета роботи: оволодіння навичками отримання даних з віддаленого сервера (API) та відображення їх у вигляді списку в застосунку React Native.

Порядок виконання роботи

1. Використовуйте проєкт з попереднього завдання.
2. Оберіть безкоштовне публічне API, що повертає масив даних (<https://jsonplaceholder.typicode.com/posts>).
3. У файлі App.js імпортуйте хуки та компоненти: `useState`, `useEffect`, `FlatList`, `View`, `Text`, `StyleSheet`.
4. Створіть стан для зберігання даних: `const [data, setData] = useState([])`.
5. Використайте хук `useEffect` для виконання запиту до API при першому завантаженні компонента. Всередині `useEffect` викличте функцію `fetch('https://jsonplaceholder.typicode.com/posts')`, обробіть відповідь та збережіть результат за допомогою `setData()`.
6. Замість стандартного `View` та `Text` у рендері використовуйте компонент `FlatList`.
7. Налаштуйте `FlatList`, передавши йому необхідні дані:
 - `data={data}` – масив даних зі стану;
 - `renderItem={({item}) =>`
`<View><Text>{item.title}</Text></View>}` – функція, що описує, як відображати кожен елемент списку;
 - `keyExtractor={item => item.id.toString()} – функція для генерації унікального ключа.`
8. Додайте стилі для списку та його елементів.
9. Запустіть застосунок і переконайтеся, що на екрані відображається список заголовків, отриманих з API.

Вимоги до звіту

Надайте повний код файлу App.js та скріншот застосунку з відображенням списком.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Bass L., Clements P., Kazman R. Software Architecture in Practice. Boston: Addison-Wesley, 2021.
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York: McGraw-Hill Education, 2019.
3. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1719343340075&uri=CELEX%3A32016R0679>.
4. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). URL: <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>.
5. Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act). URL: <https://eur-lex.europa.eu/eli/reg/2024/2847/oj>.
6. Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union (NIS 2 Directive). URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1755292862480&uri=CELEX%3A32022L2555>.
7. Regulation (EU) 2023/2854 of the European Parliament and of the Council of 13 December 2023 on harmonised rules on fair access to and use of data (Data Act). URL: <https://eur-lex.europa.eu/eli/reg/2023/2854>.
8. Regulation (EU) 2022/868 of the European Parliament and of the Council of 30 May 2022 on European data governance

(Data Governance Act). URL: <https://eur-lex.europa.eu/eli/reg/2022/868/oj>.

9. Regulation (EU) 2022/2065 of the European Parliament and of the Council of 19 October 2022 on a Single Market For Digital Services (Digital Services Act). URL: <https://eur-lex.europa.eu/eli/reg/2022/2065/oj/eng>.

10. Regulation (EU) 2022/1925 of the European Parliament and of the Council of 14 September 2022 on contestable and fair markets in the digital sector (Digital Markets Act). URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1712177161976&uri=CELEX%3A32022R1925>.

11. Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services (European Accessibility Act). URL: <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32019L0882>.

12. Decision (EU) 2022/2481 of the European Parliament and of the Council of 14 December 2022 establishing the Digital Decade Policy Programme 2030. URL: <https://eur-lex.europa.eu/eli/dec/2022/2481/oj/eng>.

13. Directive (EU) 2018/1972 of the European Parliament and of the Council of 11 December 2018 establishing the European Electronic Communications Code. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1772157124798&uri=CELEX%3A32018L1972>.

14. Regulation (EU) 2021/694 of the European Parliament and of the Council of 29 April 2021 establishing the Digital Europe Programme. URL: <https://eur-lex.europa.eu/eli/reg/2021/694/oj/eng>.

15. Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the

websites and mobile applications of public sector bodies. URL: <https://eur-lex.europa.eu/eli/dir/2016/2102/oj>.

16. European Skills Agenda. URL: https://employment-social-affairs.ec.europa.eu/policies-and-activities/skills-and-qualifications/european-skills-agenda_en.

17. Digital Education Action Plan 2021-2027. URL: https://education.ec.europa.eu/sites/default/files/document-library-docs/deap-communication-sept2020_en.pdf.

18. Андрущак І.Є., Ліщина Н.М., Сіваковська О.М., Ящук А.А. Інженерія програмного забезпечення: магістерський курс: навч. посіб. / за заг. ред. І.Є. Андрущака. Луцьк: ЛНТУ, 2024. 188 с.

19. Антонюк А.І. Інженерія програмного забезпечення: конспект лекцій. Київ: КПІ ім. Ігоря Сікорського, 2022.

20. Бородкіна, І.Л. Бородкін Г.Л. Інженерія програмного забезпечення. Київ: ЦУЛ, 2019. 204 с.

21. Бревус В. М., Бревус Г. Б. Основи програмної інженерії (SE201): конспект лекцій для здобувачів освітньої програми «Інженерія програмного забезпечення». Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2025. 127 с. URL: <https://elartu.tntu.edu.ua/handle/lib/50332>.

22. Бугасєва І.Г., Новікова Н.О., Панченко Т.Д. Системний аналіз. Навчальний посібник для спеціальності 122 Комп'ютерні науки. ОНМУ. 2023. 106 с.

23. Гетьман І.А., Міхєєнко Д.Ю., Держевецька М. А. Системний аналіз: конспект лекцій. Запоріжжя: ТОВ «Технічний університет «Метінвест політехніка», 2025. 140 с. URL: <https://dspace.mipolytech.education/handle/mip/2970>.

24. Грицюк П.М., Джоші О.І., Гладка О.М. Основи теорії систем і управління: навч. посіб. Рівне: НУВГП, 2021. 272 с. URL:

<https://ep3.nuwm.edu.ua/20653/1/Основи%20теорії%20систем%20і%20управління.pdf>.

25. Дегтярьова Л.М., Гроза П.М., Сомов С.В. Технології розробки програмного забезпечення: навч. посіб. Полтава: ПолтНТУ, 2017. 212 с. URL:

<https://reposit.nupp.edu.ua/item/4720>.

26. Зінов'єва О.Г., Малкіна В.М. Системний аналіз: конспект лекцій. Запоріжжя, 2023. 150 с.

27. Катренко А.В. Системний аналіз: підручник. Львів: «Новий Світ-2000», 2026. 396 с.

28. Кучеров Д.П., Артамонов Є.Б. Інженерія програмного забезпечення: навч. посіб. Київ: НАУ, 2017. 386 с.

29. Левус Є., Мельник Н. Вступ до інженерії програмного забезпечення. Навчальний посібник. Видавництво Львівської політехніки. 2018. 248 с.

30. Мазурок Т.Л., Яновський А.О. Системний аналіз: навчальний посібник. Одеса: ПНПУ, 2022. 250 с. URL: <http://dspace.pdpu.edu.ua/bitstream/123456789/15826/1/Mazurok%20T.%20L.pdf>.

31. Панкратова Н.Д. Системний аналіз: теорія та застосування: підручник. Київ: Вид-во «Наукова думка» НАН України, 2019. 352 с.

32. Прокопенко Т.О. Теорія систем і системний аналіз: навч. посіб. Черкаси: ЧДТУ, 2019. 139 с. URL: https://er.chdtu.edu.ua/bitstream/ChSTU/986/1/TCICA_НП_Прокопенко.pdf.

33. Соколов С.В. Теорія систем і системний аналіз: конспект лекцій. Суми: Сумський державний університет, 2020. 171 с.

34. Стативка Ю.І. Проектування та розроблення програмного забезпечення: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2023. 208 с.

35. Теорія систем і системний аналіз: навч. посіб. / О. А. Балтовський та ін. Одеса: РВВ ОДУВС, 2021. 156 с. URL: <https://dspace.oduvs.edu.ua/items/fe031d77-2202-432b-8b0b-533bc62ae6>.

36. Тихоход В.О., Гурін А.Л., Беспала О.М. Технології розробки програмного забезпечення : курс лекцій. Київ: КПІ ім. Ігоря Сікорського, 2024. 238 с. URL: <https://ela.kpi.ua/items/01a7b076-7727-4fa1-aa1d-dff2fadf98a6>.

37. Тонконогий В.М., Вайсман В.О., Бовнегра Л.В., Кіркопуло К.Г. Системний аналіз: навч. посіб. Одеса : Нац. ун-т «Одеська політехніка», 2022. 84 с. URL: <http://dspace.opu.ua/jspui/handle/123456789/12878>.

38. Трофименко О.Г., Манаков С.Ю., Ларін Д.Г. Основи програмної інженерії : навч. посіб. Одеса : Фенікс, 2022. 196 с. URL: <https://dspace.onua.edu.ua/handle/11300/22773>.

39. Федоров М. В. Системний аналіз: конспект лекцій. Харків: ХНУМГ ім. О.М. Бекетова, 2018. 62 с. URL: <https://core.ac.uk/download/pdf/158567194.pdf>.

40. Цибульник С.О., Барандич К.С. Технології розроблення програмного забезпечення. Частина 1. Підручник. Електронне мережне навчальне видання. Київ. КПІ ім. Ігоря Сікорського. 2022.

41. Швець С.В., Швець У.С. Основи системного аналізу: навчальний посібник. Суми: Сумський державний університет, 2017. 126 с. URL: <https://essuir.sumdu.edu.ua/server/api/core/bitstreams/3626f244-8958-4943-92c5-a0b43b80bc88/content>.