КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Управління проектами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Дослідження процесів управління проєктом створення мобільного додатку підтримки ігрового контенту Dungeons & Dragons»

Студента 2-го курсу групи УПз-21

Максим ДИКИЙ

(прізвище, ім'я, по батькові)

_____

(підпис)

Науковий керівник:

кандидат технічних наук,

(науковий ступінь, вчене звання)

Богдан ЄРЕМЕНКО

(прізвище, ім'я, по батькові)

_____ _____

(дата) *(підпис)*

Попередній захист:

_____

_ (Висновок: "До захисту в Екзаменаційній комісії")

Завідувач
кафедри технологій
управління _____ _____ _____

(підпис) *(прізвище, ініціали)* *(дата)*

Київ - 2025

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Дикий Максим Юрійович

Група УПз-21

1. Тема кваліфікаційної роботи:

«Дослідження процесів управління проєктом створення мобільного додатку підтримки ігрового контенту Dungeons & Dragons» Затверджена наказом від «16» червня 2025 р. №15.

2. Строк подання студентом готової роботи - «16» грудня 2025 р.

3. Цільова установка та вхідні дані до роботи:

Мета дослідження наукове обґрунтування та розробка практичних рекомендацій щодо застосування оптимальної моделі управління проєктом для випуску мобільного додатку, що відповідає вимогам цільової аудиторії D&D.

4. Зміст роботи: Аналіз актуального стану та специфіки управління проєктами у сфері розробки мобільних сервісів для настільних рольових ігор.

6. Календарний план виконання роботи:

№ п/п	Назва частини роботи	План виконання роботи
1.	Опрацювання наукової літератури та джерел з методологій управління проєктами	02.09.2025 - 22.09.2025
2.	Збір даних про специфіку ринку TTRPG та аналіз екосистеми Dungeons & Dragons	23.09.2025 - 23.10.2025
3.	Складання та деталізація розгорнутого плану-проспекту магістерської роботи	24.10.2025 - 30.10.2025
4.	Узгодження структури дослідження з науковим керівником. Коригування плану	31.10.2025

5.	Підготовка розділу 1 «Аналіз предметної області та концептуалізація проєкту мобільного сервісу»	01.11.2025 - 08.11.2025
6.	Підготовка розділу 2 «Моделювання гібридної системи управління та архітектури продукту»	09.11.2025 - 15.11.2025
7.	Підготовка розділу 3 «Програмна реалізація кросплатформного додатку та інструментів локалізації»	15.11.2025-25.11.2025
8.	Нормоконтроль та фінальне оформлення пояснювальної записки	15.11.2025-25.11.2025
10.	Передача кваліфікаційної роботи рецензенту для рецензування	16.12.25
11.	Передача кваліфікаційної роботи науковому керівникові	16.12.25
12.	Попередній захист кваліфікаційної роботи	16.12.25
13.	Перевірка роботи на плагіат	16.12.25
14.	Захист роботи	23.12.25

Дата видачі завдання ____ _____ 2025 р.

Завдання прийняв

Дикий Максим Юрійович

(прізвище, ім'я, по батькові)

_____

(підпис)

Керівник роботи

Доцент,кандидат технічних наук,

(посада, прізвище, ім'я, по батькові)

Єременко Богдан Михайлович

_____

(підпис)

ЗМІСТ

Анотація

кваліфікаційної роботи магістра на тему:
«**Дослідження процесів управління проєктом створення мобільного додатку підтримки ігрового контенту Dungeons & Dragons**»

Студента: Дикий Максим Юрійович
Науковий керівник: Єременко Богдан Михайлович
Рік захисту - 2025.

Темою даної роботи було обрано «Дослідження процесів управління проєктом створення мобільного додатку підтримки ігрового контенту Dungeons & Dragons», предметною областю якої є управління IT-проєктами, розробка програмного забезпечення для мобільних платформ та індустрія цифрових сервісів для настільних ігор.

Метою підготовки роботи є наукове обґрунтування та розробка практичних рекомендацій щодо застосування адаптивної моделі управління, аналіз ризиків, пов'язаних з ліцензованим контентом, та створення організаційної структури для забезпечення механічної точності продукту.

Ціль проєкту – створення та виведення на ринок конкурентоспроможного мобільного додатку з повною українською локалізацією, основний концепт якого передбачає використання архітектури Kotlin Multiplatform для забезпечення ідентичності ігрової логіки на різних платформах.

Наукова новизна роботи складається: розроблено Гібридний Методологічний Фреймворк (ГМФ), впроваджено специфічні метрики Порогу Точності Логіки (LAT) та Дрегу Зовнішньої Залежності (EDD), запропоновано модель «Глосарій-Орієнтованої» локалізації, структурні моделі організації команди зі специфічними ролями, такими як Лідер з Ігрової Механіки. Були проведені дослідження впливу зовнішніх регуляторних факторів на швидкість розробки.

Кваліфікаційна робота складається з анотації, вступу, основної частини, яка включає три розділи, висновків, списку використаних джерел та додатків.

В першому розділі проводиться аналіз існуючих підходів до управління проєктами в ніші ігрових сервісів. Розкривається проблема недостатньої ефективності класичних Agile-підходів в умовах жорстких вимог до точності правил D&D 5e. Також проводиться аналіз цільової аудиторії за методом 5W, будуються дерева причин, наслідків та цілей проєкту, формулюється паспорт проєкту та його бізнес-модель за методикою Lean Canvas.

Другий розділ присвячено організаційно-технологічному аналізу середовища розробки. Обґрунтовується вибір технологічного стеку (KMP, Ktor, SQLDelight) як інструменту мінімізації ризиків. Розробляється структура Гібридного фреймворку, що поєднує послідовний потік роботи з контентом та ітеративний потік розробки платформи. Визначаються контрольні точки якості (Quality Gates) для інтеграції потоків.

В третьому розділі наводиться практична реалізація управління проєктом, що включає планування спринтів, формування беклогу та історій користувача з урахуванням механічної складності. Проводиться ідентифікація та кількісний аналіз ризиків з використанням розширеної шкали оцінювання. Аналізуються результати впровадження буферних черг для нівелювання затримок та валідується ефективність запропонованих метрик на прикладі розробки модуля генерації персонажа.

Робота містить 96 сторінок, 5 таблиць та рисунки,

Ключові слова: мобільний додаток, Dungeons & Dragons, Kotlin Multiplatform.

Завдання дослідження

Для досягнення поставленої мети визначено такі завдання:

1. Провести порівняльний аналіз традиційних та гнучких методологій управління проєктами в контексті розробки мобільних додатків.
2. Ідентифікувати та проаналізувати ключові виклики та ризики, специфічні для проєктів із інтеграцією ігрового контенту та авторських прав.
3. Розробити адаптовану структуру (фреймворк) процесів управління проєктом, що включає управління вимогами, змінами, якістю та комунікаціями для міжфункціональної команди.
4. Експериментально апробувати запропоновану модель управління на реальному етапі проєктування чи розробки мобільного додатку.
5. Сформулювати практичні рекомендації щодо вибору інструментарію та метрик ефективності (KPIs) для керівника проєкту (PM) у подібній галузі.

Наукова новизна

Наукова новизна отриманих результатів полягає у створенні гібридної моделі управління проєктами, яка поєднує гнучкість Agile-підходів (для швидкої адаптації до вимог користувачів і контенту) із суворістю класичного управління проєктами (для контролю ліцензійного контенту та інтеграційних ризиків), спеціально адаптованої для розробки мобільних додатків підтримки ігрового контенту D&D.

Практичне значення

Практичне значення результатів дослідження полягає у можливості їхнього безпосереднього впровадження керівниками IT-проєктів, які займаються розробкою мобільних рішень для ігрових, освітніх або контентних ніш. Запропоновані методики та фреймворки дозволять скоротити терміни, зменшити вартість розробки та підвищити задоволеність кінцевого користувача.

Вступ

Коли настільні ігри переходять у цифровий формат, виникають специфічні проблеми для керівників проєктів. Наш досвід розробки мобільного додатку для D&D показав, що стандартні Agile-підходи тут не завжди спрацьовують. Чому? Бо треба працювати з ліцензованим контентом Wizards of the Coast, складною математикою ігрових механік.

Головна проблема - як поєднати гнучкість розробки з жорсткими вимогами до контенту. Команда хоче швидко реагувати на фідбек користувачів, але водночас треба інтегрувати величезні бази даних: заклинання, характеристики монстрів, правила класів персонажів. На практиці це означає, що один спринт може присвячуватись виключно валідації даних, без видимого прогресу для стейкхолдерів. Така робота потребує детального планування на кшталт Waterfall, а це суперечить ітеративному підходу Scrum.

У літературі з проєктного менеджменту майже немає досліджень про розробку таких специфічних додатків. Є багато про e-commerce чи корпоративні системи, але не про мобільні інструменти для TTRPG. Тому ми вирішили проаналізувати, які метрики реально працюють, коли головне - це точність відтворення ігрової системи, а не просто швидкість випуску фіч.

Наше питання: чи можна взагалі дотримуватись однієї методології в такому проєкті? Окрема складність - управління ризиками. Це дослідження фокусується саме на цих аспектах і пропонує адаптувати класичні методології під реалії розробки ігрових утиліт.

Розділ 1. Теоретичні основи та аналіз методологій управління проєктами у сфері розробки мобільних додатків

Розробка мобільних додатків для настільних рольових ігор - це специфічна ніша, де класичні підходи до управління проєктами часто не працюють. Waterfall дає чітку послідовність етапів, але його жорсткість стає проблемою, коли маєш справу з додатком для D&D, де вимоги постійно змінюються.Наприклад, спільнота може активно просити функцію автоматичного розрахунку перевантаження персонажа, але через тиждень з'ясується, що більшість гравців взагалі не використовує ці правила. Тут управління проєктом - це не просто контроль дедлайнів, а балансування між складними базами даних і очікуваннями гравців.

Якщо подивитись на літературу, стає очевидним конфлікт між формальним контролем і потребою швидко випускати оновлення. Agile-методології, зокрема Scrum та Kanban, витіснили лінійні підходи в мобільній розробці через акцент на ітераціях. Але коли працюєш з ліцензованим контентом D&D, виникають специфічні обмеження: треба узгоджувати кожну зміну з правовласниками, контролювати використання ассетів. У нашому випадку цикл розробки нової фічі міг сповільнитися, що повністю ламало стандартний двотижневий спринт. Стандартний Agile Manifesto цього не враховує. Філософія "fail fast" тут не спрацьовує - адже і спільнота гравців, і Wizards of the Coast очікують високої точності в деталях.

Ще один момент - інтеграція ігрових механік у додаток створює унікальні ризики. Класичні методи управління ризиками з PMBOK фокусуються на багах і зривах термінів, але не враховують "фактор задоволення" чи баланс механік як критерій успіху проєкту. Як виміряти, наскільки добре реалізований розрахунок кидків кубиків або наскільки зручний інтерфейс створення персонажа? Досвідчені гравці миттєво

помічають, якщо модифікатори рятівних кидків розраховані неправильно, і це може зруйнувати довіру до всього додатку. Це якісні показники, які складно квантифікувати. Тому, можливо, гібридний підхід - де поєднуються елементи lean startup з жорсткими точками контролю якості - буде оптимальним для такого проєкту. Вибір методології не повинен бути питанням "або-або", це має бути гнучка відповідь на технічну невизначеність і високу мінливість очікувань ігрової спільноти.

1.1. Еволюція та класифікація підходів до управління в розробці мобільного програмного забезпечення

Управління проєктами в IT сильно змінилось порівняно з тим, що було на початку ери мобільних додатків. Тоді активно копіювали Waterfall - модель, яка добре працювала в будівництві чи промисловості, але абсолютно не підходила для динамічного світу мобільної розробки. Зараз, дивлячись назад, здається дивним, що команди витрачали місяці на написання детальних вимог, які застарівали ще до початку написання коду. Для проєкту типу D&D-додатку, де самі правила гри постійно розширюються новими книгами та доповненнями, такий жорсткий підхід означав би, що продукт застаріє ще до релізу.

Agile прийшов як рятівна альтернатива, але на практиці його впроваджують по-різному, особливо в нішевих проєктах. Scrum теоретично дає ітеративний фідбек, але часто спринти перетворюються на ті ж мініводоспади, якщо команда не розуміє специфіку предметної області. В ігрових додатках швидкість команди залежить не лише від кількості виконаних завдань, а й від того, наскільки правильно ці завдання перетворюються в збалансовану механіку для користувача. Іноді, намагаючись встигнути показати щось на демо через два тижні, розробники жертвують архітектурою бази даних, яка має правильно обробляти всю

математику характеристик персонажів. А потім цей технічний борг повертається з подвійною силою.

Є ще концепція lean-управління і питання, чи підходить вона тут взагалі. MVP (мінімально життєздатний продукт) часто розуміють як "недороблений продукт", що катастрофічно небезпечно для аудиторії настільних ігор. Гравці в D&D звикли до високоякісних фізичних книг з ідеальною верстkою та перевіреними правилами. Якщо додаток виглядає недобробленим або містить помилки в механіках, спільнота просто його проігнорує. Сучасні методології намагаються знайти баланс між швидкістю розробки та тією точністю, яка потрібна для ліцензованої гри. Але, схоже, традиційна класифікація методологій (Waterfall проти Agile) занадто спрощена для таких специфічних проєктів.

1.2. Специфічні обмеження управління ліцензованим настільним контентом у цифровому середовищі

Коли працюєш з IP таким щільним, як Dungeons & Dragons, стикаєшся з проблемами, про які не пишуть у підручниках з проєктного менеджменту. Це не просто розробка софту - це переклад складної ігрової системи з паперу в цифру, причому системи з 50-річною історією. У класичному PM ми говоримо про "залізний трикутник" - обсяг, час, бюджет. Але тут обсяг диктується зовнішніми факторами: ліцензійними угодами та правилами гри. Якщо Wizards of the Coast раптом оновлює Open Game License (OGL) або випускає нову книгу правил посеред спринту, це моментально ламає весь беклог і burn-down chart летить у тартарари.

Інтеграція ігрових механік - розрахунок Armor Class, трекінг spell slots, масштабування proficiency bonus - вимагає специфічного QA, якого немає у звичайних мобільних студіях. Проблема в тому, що розробники без досвіду гри в D&D можуть зробити фічу математично правильно, але абсолютно

незручною для Dungeon Master'a за столом. Наприклад, калькулятор ініціативи може правильно сортувати числа, але якщо він не враховує, що DM часто додає монстрів на льоту посеред бою, він стає марним. Я називаю це "механічним боргом" - коли треба повертатись і переробляти UX, щоб він відповідав реальному флоу гри

.Ще гірше, коли розробник не знає, що такий концепт як "Surprise Round" взагалі не існує в 5-й редакції D&D, але гравці з попередніх редакцій постійно про нього питають.

Тому для розгляду ефективності методології управління проєктами пропоную скласти порівняльну таблицю між класичними підходами до управління та гібридним підходом (Таблиця 1.2).

Таблиця 1.2. Ефективність методологій управління ІТ-проєктами залежно від стабільності вимог та складності архітектури.

Характеристика	Waterfall (Класика)	Agile (Scrum/Kanban)	Гібридний підхід
1	2	3	4
Планування архітектури	Детальне, на початку проєкту	Мінімальне, еволюційне	Детальне для ядра (DB), ітеративне для фіч
Робота з ліцензією (WotC)	Легко контролювати обсяг прав	Складно встигати за юридичними змінами	Чіткий контроль прав на старті ітерацій

1	2	3	4
Якість ігрової механіки	Висока точність, але пізня перевірка	Швидка перевірка, ризик техборгу	Висока точність через суворий DoD (Definition of Done)
Метрика успіху	Дотримання плану (Scope/Time)	Velocity, Burndown	Точність механік + UX-фідбек
Ризик MVP	Продукт застаріває до релізу	Недостатня якість для спільноти	Повноцінне "ядро" + ітеративні інтерфейси
Гнучкість до нових книг	Майже відсутня (потрібен новий проєкт)	Висока, але ламає стару архітектуру	Висока завдяки модульній базі даних

1.3. Ідентифікація динаміки стейкхолдерів та користувацьких ризиків у спеціалізованих ігрових екосистемах

Робота зі стейкхолдерами в D&D-проєкті - це зовсім інший рівень складності порівняно зі звичайними корпоративними IT-проєктами. PRINCE2 чи стандартні протоколи PMI тут особливо не допоможуть. У типовому enterprise-проєкті кінцеві користувачі - це співробітники компанії, яким "треба буде користуватись", хочуть вони того чи ні. Тут же стейкхолдери розділені між правовласниками (WotC), командою розробки та дуже голосною, технічно підкованою спільнотою гравців. Що цікаво - досвідчені гравці часто розуміють внутрішню структуру правил гри краще, ніж

джуніор-розробники, які їх кодують. Це перевертає стандартну динаміку збору вимог з ніг на голову.

Управління ризиками тут виходить за межі фінансів і технічних проблем. Є такий ризик як "настрій спільноти", і він може вбити проєкт швидше за будь-який збій сервера.

Якщо менеджмент не продумає прозоро, як контент (класи персонажів, списки заклинань) буде доступний або монетизований, зворотний зв'язок від гравців може зробити продукт токсичним ще до релізу. Інтеграція фідбеку спільноти в короткі спринти майже завжди призводить до неконтрольованого збільшення осягів, якщо немає сильного Product Owner'a, який сам розбирається в D&D.

Технічні ризики теж прив'язані до того, як додаток поводиться під час реальної гри. Якщо додаток крашиться під час кидка на ініціативу або за чотиригодинну сесію висмоктує батарейку - проєкт провалений, навіть якщо всі задокументовані вимоги виконані. "Стабільність під час гри" має бути головним KPI, але натомість в панелі управління проєктами часто фокусуються на "повноті фіч". Це фундаментальна прогалина в тому, як ми зараз категоризуємо нефункціональні вимоги для ігрового софту. Чому ці "м'які" метрики ігнорують? Бо їх важко квантифікувати для вищого керівництва. Але саме вони найчастіше стають точками провалу всього проєкту.

1.4. Синтез технічного боргу та цілісності наративу в життєвому циклі проєкту

Коли поєднуєш складний ігровий контент з обмеженнями мобільних платформ, виникає особливий вид технічного боргу, який не видно на початку проєкту. Працюючи над D&D-додатком, я постійно бачив одну проблему: команда хоче запушити "круті" візуальні фічі - 3D-кубики, динамічне

освітлення на картах - і при цьому ігнорує, що database schema не витримає тисячі унікальних варіацій персонажів. Це не просто помилка коду, це управлінська помилка: пріоритизувати front-end для маркетингу замість back-end стабільності. Стандартні PM-метрики показують "завершення функції", але не показують, що основні функції розвалюється під навантаженням на середньостатистичному смартфоні.

Ще один момент - "цілісність правил" D&D працює як регуляторні вимоги у фінансовому софті. Це не підлягає обговоренню. Якщо спринт присвячений поліруванню UI, а баг в логіці multiclassing лишається - цінність проєкту для стейкхолдерів падає до нуля. Стандартні Agile графіки спалювання цього не відстежують. Можеш бути "вчасно" за тікетами, але якщо основна ігрова логіка зламана - спринт провалений. Про це рідко пишуть у літературі, але на практиці розрив між PM дошка і реальною юзабіліті контенту - це те місце, де проєкти починають зливати гроші.

І є питання довгострокової підтримки. iOS та Android оновлюють SDK швидше, ніж виходять нові книги правил D&D. Управління "відхилення версії" між операційною системою, сторонніми бібліотеками та правилами D&D вимагає постійної активної підтримки, яку рідко закладають у бюджет на етапі обговорення проєкту. Реалістичний підхід - це "хвильве" планування, де технічний борг не є задачею на кінець проєкту, а щотижневими витратами у бюджеті. Без цього додаток стане не потрібним через півтора року після запуску.

1.5. Бенчмаркінг метрик успіху на перетині утилітарного та розважального софту

Оцінювати успіх проєкту, який одночасно є інструментом і розвагою, не можна за стандартними корпоративними KPI. Так, час доступу і рівень

аварійності важливі, але їх недостатньо для D&D-додатку. У звичайному софті утримання користувачів прив'язана до ефективності виконання задач.

Але в настільних іграх швидкість (наприклад, як швидко гравець знайде заклинання) має балансуватись з "фактором занурення". Гравець може витратити п'ять хвилин на пошук потрібного заклинання, але якщо інтерфейс виглядає як таблиця Excel, він більше не відкриє додаток. Якщо керівник проєкту фокусується виключно на швидкості, ігноруючи естетику та тематичну консистентність інтерфейсу, отримаєш технічно робочий продукт, який емоційно порожній.

Ще одна проблема - визначення звершеності для конкретних фіч. Я стикався з ситуаціями, коли “character builder” модуль проходить всі базові тести і задовольняє user stories, але не справляється з граничним випадком типу “мульти классування”. Це показує обмеження автоматизованого тестування у мобільній розробці. Проєктний менеджер має вводити "м'які мілстоуни", які враховують якісний зворотний зв'язок гравців ще на початковій фазі. Покладатись тільки на кількісні дані - DAU (daily active users) - небезпечно.

Фінансові метрики теж мають враховувати довгострокову природу ігрового контенту. На відміну від корпоративних додатків, де фічі виводяться з часом, D&D-додаток накопичує контент з кожною новою книгою правил. "Cost of quality" (CoQ) зростає експоненційно разом з дата сетом - це постійно недооцінюють в ROI аналітиці. Ефективний project management тут вимагає зміщення від "project-based" мислення до "product-lifecycle" мислення. Якщо менеджмент не закладає в бюджет постійну ре-валідацію чинний механік проти нового контенту, технічний борг рано чи пізно перевищить ринкову цінність.

1.6. Характеристика та паспорт проєкту

1.6.1. Цільова аудиторія

Проведемо аналіз аудиторії за допомогою техніки 5W Шеррінгтона, яка полягає у відповіді на п'ять питань: Who?, What?, Where?, When?, Why? – які дозволяють чітко сегментувати аудиторію та визначити її потреби, умови й мотиви (рис. 1.5).

Рис. 1.5 Визначення аудиторії за допомогою методу 5W

Також додатково визначимо основну аудиторію:

Гравці-новачки – чоловіки та жінки віком 16-35 років, які щойно відкривають для себе світ настільних рольових ігор. Це люди, які хочуть спробувати D&D, але стикаються з мовним бар'єром при вивченні англомовних правил. Вони потребують інтуїтивного інструменту для створення свого першого персонажа, зрозумілих пояснень механік гри та підтримки під час

перших сесій. Для цієї групи ключовим є простота використання, наявність підказок та покрокових інструкцій українською мовою, а також швидкий доступ до базових правил без необхідності гортати книги.

Досвідчені гравці та майстри подземель (DM) – чоловіки та жінки віком 18-45 років, які мають значний досвід гри в D&D (2+ роки), проводять регулярні ігрові сесії та часто виступають у ролі майстра подземель. Це активні учасники української D&D спільноти, які прагнуть мати професійні інструменти для управління складними механіками: мультиклассування персонажі, розрахунок комірок для заклинань, трекінг бойових ситуації з десятками учасників.

Вони цінують точність відповідно до офіційних правил SRD 5.1, швидкість роботи додатка під час живих сесій та можливість офлайн-доступу до всього контенту.

Обидві групи проживають здебільшого на території України (окрім тимчасово окупованих територій), проте значна частина української аудиторії також перебуває за кордоном (Польща, Чехія, Німеччина, інші країни ЄС). Спільною рисою для двох сегментів є активне використання соціальних мереж (Telegram, Discord, Facebook групи D&D спільноти), прагнення до підтримки української мови в хобі та готовність використовувати мобільні технології для покращення ігрового досвіду.

1.6.2. Дерево причин та наслідків та дерево цілей

Дерево причин та наслідків допомагає глибше зрозуміти суть проблеми, визначивши її основні першопричини, а також проаналізувати наслідки, які вони породжують. На рис. 1.6.1 зображено дерево причин та наслідків для даного проєкту.

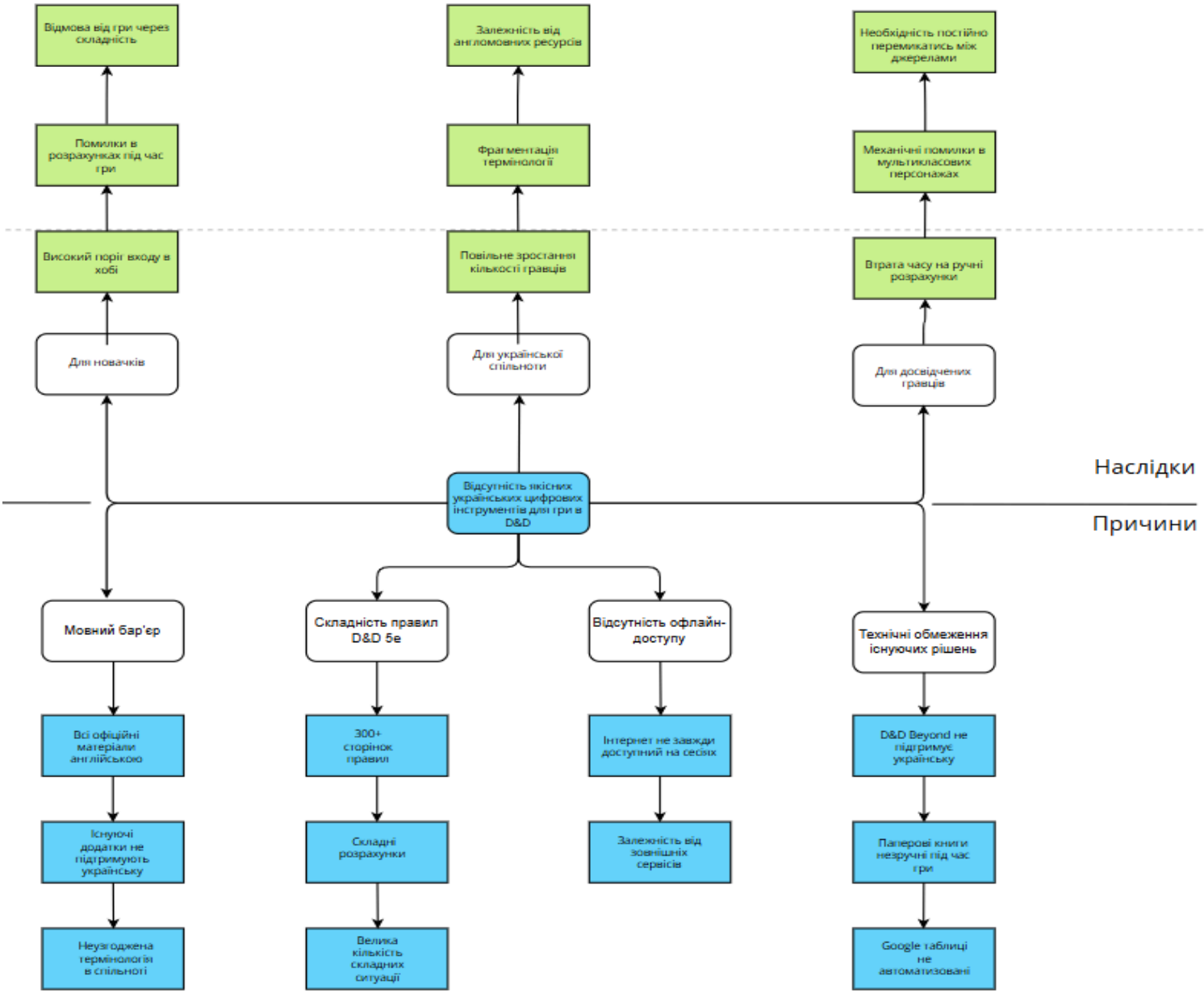

Рис. 1.6.1 Дерево причин та наслідків

Дерево причин та наслідків допомагає глибоко зрозуміти суть проблеми, визначивши її першопричини та наслідки, а також зрозуміти потреби користувачів. Виходячи з визначених проблем та наслідків (рис. 2), гравці-новачки потребують інтуїтивний інструмент для створення персонажа з покроковими підказками, доступ до базових правил українською мовою, автоматизацію складних розрахунків та можливість швидко знайти відповіді на питання під час гри.

До потреб досвідчених гравців та DM можна віднести професійні інструменти для управління складними механіками, офлайн-доступ до повного бестіарію та довідників заклинань, коректну реалізацію згідно з SRD 5.1 та швидкість роботи під час ігрових сесій з мінімальними затримками.

Дерево причин та наслідків стає підґрунтям для формування дерева цілей (рис.1.6.2), оскільки дозволяє перетворити виявлені причини на конкретні завдання, спрямовані на розв'язання проблеми та задоволення потреб користувачів.

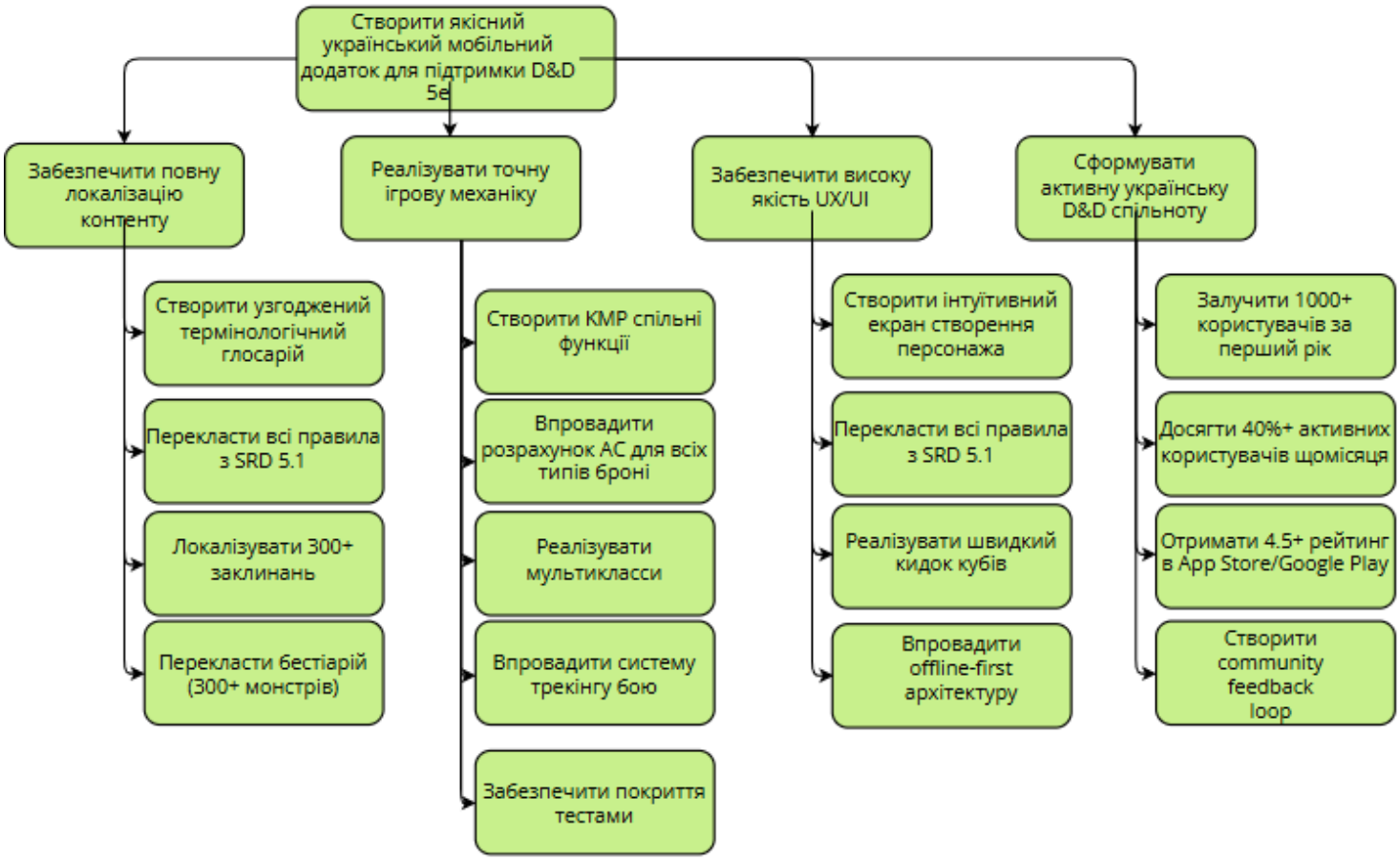

Рис. 1.6.2. Дерево цілей проєкту

У побудованому дереві цілей визначено основні напрями розвитку проєкту, що передбачають забезпечення потреб цільової аудиторії шляхом розробки необхідного базового функціоналу додатка, забезпечення ефективного

управління проєктом через гібридний фреймворк, досягнення технічної стабільності та точності ігрових механік, формування та збільшення активної користувацької спільноти завдяки якісному продукту та органічному поширенню через D&D спільноту.

1.6.3. Цілі проєкту

Головна ціль:

Забезпечити українських гравців D&D та майстрів подземель конкурентоспроможним, зручним та стабільно функціональний мобільним додатком для створення персонажів, управління ігровими механіками, швидкого доступу до правил та проведення повноцінних ігрових сесій з повною підтримкою української мови.

Цілі та під цілі:

2. Забезпечити повну та якісну локалізацію контенту D&D 5e SRD українською мовою.

Під цілі:

Створити та затвердити термінологічний глосарій з 500+ термінів D&D протягом перших 2 тижнів проєкту (Спринт 0). Досягти консенсусу в українській D&D спільноті щодо ключових термінів.

Перекласти та верифікувати всі базові правила D&D 5e SRD (характеристики, класи, раси) протягом першого місяця проєкту.

Локалізувати повний список заклинань (300+) з детальними описами, забезпечивши консистентність термінології та коректність перекладу механічних ефектів до початку Спринту 4.

Перекласти бестіарій (300+ істот) з повною інформацією українською мовою до початку Спринту 5.

3. Реалізувати точну та надійну ігрову механіку відповідно до правил D&D 5e SRD.

Під цілі:

Створити KMP загальні модулі для двох мобільних систем з коректною реалізацією всіх базових розрахунків.

Впровадити систему розрахунку комірок заклинань для всіх типів заклиначів, включаючи коректну обробку 20+ мультикласових комбінацій з досягненням 0% механічних помилок при тестуванні QA до кінця Спринту 4.

Реалізувати повний бойовий досвід з підтримкою 20+ одночасних учасників бою до кінця Спринту 3.

4. Забезпечити ефективне управління проєктом для якісної та своєчасної реалізації функціоналу.

Під цілі:

Впровадити Гібридний Фреймворк Управління (ГФУ) з подвійною операційною моделлю протягом першого тижня проєкту.

Встановити систему трьох “Quality Gates”/”Воріт якості” (QG1, QG2, QG3) та забезпечити обов'язкове проходження всіх критичних функцій через верифікацію.

Проводити регулярні огляди ризиків з фокусом на метрику External Dependency Drag (EDD) та підтримувати буферні задачі для запобігання простою команди.

Здійснювати перевірку якості після завершення кожного спринту з обов'язковою повною симуляцією ігрової сесії.

5. Досягти високої технічної стабільності та якості користувацького досвіду.

Під цілі:

Забезпечити час запиту < 2 секунди для 95% запитів та підтримувати 99.9% час безвідмовної роботи системи.

Досягти 80%+ покриття unit-тестами для KMP спільних модулів та автоматизувати тестування через CI/CD pipeline.

Оптимізувати виконання для роботи на “середніх” Android пристроях (від Android 8.0) та iOS (від iOS 13.0) з плавною анімацією 60 FPS.

6. Сформувати активну українську D&D спільноту та досягти органічного росту користувацької бази.

Під цілі:

Залучити перших бета-тестерів з української D&D спільноти (Telegram, Discord групи) протягом першого місяця після завершення MVP для отримання якісного зворотнього зв'язку.

Досягти 1000+ активних користувачів протягом першого року після публікації в App Store та Google Play.

Забезпечити 40%+ MAU через постійне покращення продукту на основі зворотнього зв'язку користувачів.

1.6.4. Переваги та цінність проєкту

З огляду на результати аналізу конкурентного середовища (Розділ 1.4) та виявлені потреби цільової аудиторії, запропонований проєкт має низку переваг. Додаток пропонує комплексне вирішення основних потреб українських D&D гравців, поєднуючи повну локалізацію контенту, точну реалізацію ігрових механік для незалежності від інтернет-з'єднання під час ігрових сесій.

Для гравців-новачків додаток забезпечує низький поріг входу в хобі завдяки інтуїтивному створенню персонажа з покроковими підказками українською мовою, автоматичним розрахункам складних механік та швидкому доступу до довідників правил без необхідності читати 300-сторінкові

англомовні книги. Візуалізація складних концептів робить правила зрозумілішими та зменшує кількість помилок під час гри.

Для досвідчених гравців та майстрів подземель додаток надає професійні інструменти для управління складними сценаріями: коректний розрахунок для всіх можливих комбінацій класів, систему трекінгу бою з підтримкою 20+ одночасних учасників, повний офлайн-доступ до бестіарію (300+ істот) та каталогу заклинань (300+ заклинань).

Додатковою перевагою є використання Kotlin Multiplatform для спільних модулів, що забезпечує абсолютну ідентичність розрахунків на iOS та Android платформах, усуваючи ризик платформено специфічних багів. Унікальною перевагою проєкту є фокус на локалізації та створення стандартизованого термінологічного глосарія, затвердженого українською D&D спільнотою. Це не просто переклад - це формування мовних стандартів для української TTRPG(настільної рольової гри) культури, що має довгостроковий вплив на розвиток хобі в Україні.

Гібридний фреймворк управління з подвійною операційною моделлю забезпечує баланс між швидкістю розробки UI/UX (Agile) та необхідною ретельністю верифікації ігрових правил (Waterfall), що є критичним для проєктів з високими вимогами до точності.

Цінність проєкту: Розроблений додаток надасть українським гравцям D&D повноцінний інструмент для проведення ігрових сесій з повною підтримкою рідної мови, усуне мовний бар'єр, який стримує ріст спільноти, та автоматизує складні механіки, дозволяючи фокусуватись на історії та рольовому вдіграші замість ручних розрахунків. Для новачків це значно знизить поріг входу в хобі, а для досвідчених гравців підвищить якість та швидкість ігрових сесій.

Завдяки створенню стандартизованого глосарія та якісній локалізації, проєкт матиме довгостроковий вплив на українську TTRPG(настільну рольову гру) культуру, формуючи мовні стандарти та сприяючи розвитку спільноти. Додаток стане центральним інструментом для української D&D спільноти,

об'єднуючи гравців та підтримуючи ріст хобі в Україні навіть в умовах воєнного стану, коли доступ до фізичних книг та стабільного інтернету обмежений.

1.6.5. Результат та продукт проєкту

У результаті реалізації проєкту буде створено cross-platform мобільний додаток для iOS та Android з повною підтримкою української мови для гри в Dungeons & Dragons 5th Edition. Продукт забезпечуватиме повний цикл управління персонажами - від створення через інтуїтивний інтерфейс до розвитку через рівні з автоматичним розрахунком усіх параметрів.

Додаток включатиме:

- Character Management System: створення, редагування, збереження персонажів з підтримкою всіх рас та класів з SRD 5.1, імпорт/експорт у JSON форматі
- Combat Toolkit: калькулятор ініціативи, трекер ОЗ з історією змін, підкидання ігрових кубів.
- Spellcasting System: повний каталог 300+ заклинань українською, автоматичний розрахунок комірок заклинань для мульти-классу, трекер заклинань, фільтрація за рівнем, школою магії.
- Reference Library: бестіарій з 300+ монстрами, довідник правил українською, глосарій термінів D&D

Технічна реалізація базуватиметься на Kotlin Multiplatform для спільної логіки на обох платформах, Ktor для серверної частини API, Firebase для аутентифікації та синхронізації, SQLDelight для локального зберігання даних на пристрої та MongoDB для серверних даних.

Продукт буде опублікований в Apple App Store та Google Play Store з підтримкою мінімальних версій iOS 13.0+ та Android 8.0+, забезпечуючи доступність для 95%+ українських користувачів мобільних пристроїв.

1.6.6. Бізнес-модель проєкту

Для обґрунтування доцільності створення мобільного додатку для підтримки D&D 5e з повною українською локалізацією було сформовано бізнес-модель проєкту за методикою Lean Canvas. Вона призначена для аналізу стартапів і продуктів на ранніх етапах їх розвитку. Такий підхід дає змогу сфокусуватися на основних сегментах цільової аудиторії, перевірити гіпотези щодо запропонованого рішення та визначити першочергові напрямки створення цінності.

Основні проблеми, конкуренти, сегменти аудиторії і рішення були описані у попередніх підрозділах, тож опишемо канали залучення аудиторії, основні метрики виміру ефективності, джерела доходів і структуру витрат.

Унікальну торгівельну пропозицію (УТП) можна сформувати таким чином: "Перший повноцінний мобільний додаток для D&D 5e з професійною українською локалізацією, що поєднує інтуїтивне створення персонажів, точну реалізацію всіх ігрових механік та офлайн-доступ до повного контенту SRD 5.1, дозволяючи українським гравцям насолоджуватись D&D рідною мовою без залежності від англомовних ресурсів та інтернет-з'єднання."

Основними каналами залучення аудиторії є українські D&D спільноти в Telegram (10+ груп по 500-3000 учасників), Discord серверів (Ukrainian D&D Community, локальні ігрові спільноти), Facebook групи (D&D Україна, Настільні ігри України), органічне поширення через ”Сарафане радіо” серед гравців під час ігрових сесій, партнерства з українськими настільними магазинами та ігровими клубами.

Ключовою метрикою успішності визначено кількість активних ігрових сесій, проведених за допомогою додатку, оскільки вона відображає фактичне створення цінності для користувачів. Успішна сесія означає, що гравці:

- Створили/імпортували персонажів.
- Використовували функціонал проведення бою.
- Зверталися до довідників.
- Провели 2+ години в додатку.

Допоміжними метриками визначено:

- DAU/MAU (Daily/Monthly Active Users) - цільове значення 60%+ показує залученість.
- Character Creation Rate: скільки користувачів завершили створення хоча б одного персонажа (ціль: 80%+).
- Average Session Duration: середня тривалість використання додатка (ціль: 3+ години під час ігрових сесій).
- App Store Rating: 4.5+ зірок з фокусом на відгуки про якість додатка.
- Community Growth: приріст користувачів через organic channels (ціль: 70%+ organic traffic).

Модель доходів проєкту є поетапною. На етапі MVP (перші 6-12 місяців) додаток буде повністю безплатний для всіх користувачів з метою максимального залучення української D&D спільноти та формування бази користувачів. Це критично важливо для:

Збору зворотного зв'язку щодо якості локалізації та точності правил

Виявлення багів через широке використання

Встановлення додатка як стандарту в українській D&D спільноті

Після досягнення критичної маси користувачів (3000+ MAU) та стабільної якості продукту, очікується впровадження мінімально безплатну модель:

Базовий рівень (Free):

- Створення до 3 персонажів
- Повний доступ до SRD 5.1 контенту (базові раси, класи, заклинання, монстри)
- Основні game tools (character sheet, dice roller, basic combat tracking)
- Підтримка української та англійської мов

Преміум рівень ($3.99/місяць або $39.99/рік):

- Нескінченні персонажі
- Просунуті інструменти
- Синхронізація між пристроями
- Пріоритетна підтримка
- Раній доступ до нового функціоналу

Додаткові джерела доходу:

Партнерства з українськими видавництвами настільних ігор (партнерські посилання на купівлю фізичних книг D&D)

Очікуваний розподіл доходів після масштабування:

60-70% - Premium підписки

30-40% - Партнерські програми

До структури витрат можна віднести:

Розробка MVP: оплата команди (Android/iOS/Backend розробники, UX/UI дизайнер, QA) - основна стаття витрат

Технічна інфраструктура: Firebase , MongoDB Atlas , Apple Developer Program, Google Play Developer, CI/CD, хостинг для Ktor.

Локалізація та контент: оплата лідера локалізації, системного архітектора, витрати на верифікацію термінології.

Маркетинг (після MVP): таргетована реклама в соціальних мережах (Facebook, Instagram Ads на D&D аудиторію), спонсоровані пости у спільнотах, партнерські програми

Операційні витрати: підтримка користувачів, оновлення контенту при випуску нових D&D правил.

Фінансові прогнози (оптимістичний сценарій):

Рік 1 (MVP фаза):

Витрати: $40,000-50,000 (розробка, інфраструктура, команда)

Дохід: $0

Мета: 1000+ MAU, 4.5+ рейтинг

Рік 2 (Фаза монетизації):

Витрати: $30,000-40,000 (підтримка, оновлення, маркетинг)

Дохід: $15,000-25,000 (10-15% конверсії в Premium з 3000+ MAU)

Вихід в “нуль”: очікується до кінця року 2

Рік 3 (Growth phase):

Витрати: $40,000-50,000 (масштабування, нові функції)

Дохід: $50,000-70,000 (5000+ MAU, 15-20% Premium конверсія)

Маржинальність: 20-30%

Бізнес-модель підтверджує, що запропоноване рішення не лише усуває наявні проблеми користувачів (мовний бар'єр, відсутність якісних українських альтернатив), а й має довгостроковий комерційний потенціал в недостатньо обслуговуваній ніші української D&D спільноти.

1.7 Постановка завдань

Для досягнення поставленої мети визначено такі завдання:

1. Провести порівняльний аналіз традиційних та гнучких методологій управління проєктами в контексті розробки мобільних додатків для настільних рольових ігор.

2. Дослідити специфіку локалізації складного ігрового контенту українською мовою та розробити методологію створення стандартизованого термінологічного глосарію.

3. Розробити адаптовану структуру (фреймворк) процесів управління проєктом, що включає управління вимогами, змінами, якістю та комунікаціями для міжфункціональної команди.

4. Обґрунтувати вибір технологічного стеку (Kotlin Multiplatform, Ktor, Firebase, SQLite, MongoDB) для забезпечення механічної точності та кросплатформної консистентності.

5. Розробити систему контрольних точок якості (Quality Gates) для забезпечення механічної точності правил D&D та термінологічної консистентності локалізації.

Вирішення поставлених завдань дозволить сформувати цілісну методологічну базу, необхідну для трансформації процесу розробки з інтуїтивного рівня на системний. Комплексне опрацювання цих питань є критичною передумовою для створення Гібридного Фреймворку Управління, здатного ефективно нівелювати специфічні ризики «механічного дрейфу» та лінгвістичної неузгодженості.

РОЗДІЛ 2. ОРГАНІЗАЦІЙНО-ТЕХНОЛОГІЧНИЙ АНАЛІЗ СЕРЕДОВИЩА РОЗРОБКИ ІГРОВИХ СЕРВІСІВ ТА МЕТОДИКА ЇХ УПРАВЛІННЯ

Коли переходиш від теорії до практики, треба детально розібратись у специфічній інфраструктурі, де будуються D&D support tools. Середовище розробки ігрових сервісів - це не статична екосистема, це ландшафт, який вимагає особливого організаційного підходу. Основна проблема в управлінні такими проєктами - розрив між "креативним" робочим процесом і "жорстким" підходом мобільної інженерії. Щоб це спрацювало, середовище має трактувати ігрові правила як динамічні дані, а не закодовану логіку. Більшість організаційних моделей тут провалюються - вони ставляться до даних, як до статичної системи, не враховуючи складну реляційну математику для гри в реальному часі.

Технологічно, вибір стеку диктує ритм управління. Cross-platform framework типу Flutter чи React Native дає швидкість розробки, але створює організаційний ризик для довгострокової підтримки спеціалізовані плагіни, потрібних для складних TTRPG механік. Методологія має включати ретельний технічний аудит, який зазвичай пропускають у гонитві за прототипом. Якщо CI/CD не має автоматизовані перевірки для узгодженості механізму правил, проєкт входить у стан перманентного "hot-fix" менаджменту. Це створює психологічний тиск на команду розробки.

Ще момент - "сервісна" природа цієї розробки означає постійні відносини з користувацькою базою, що вимагає DevOps-centric підходу до управління.Організаційна структура має відображати service-oriented architecture (SOA), де окремі модулі проєкту управляються, як окремі, але взаємопов'язані потоки. Традиційні проєктні менеджери часто бояться такої фрагментації, але для D&D-додатку це єдиний спосіб гарантувати, що падіння одного модуля не викличе падіння іншого під час гри. Тому методологія, яку я

пропоную, фокусується на принципі "ізольованої стабільності" - коли важливіше, щоб ключові ігрові функції завжди працювали, ніж щоб вся система була ідеально єдиною.

2.1. Вибір адаптивних параметрів управління та калібрування метрик продуктивності

Проблема з вибором метрик для D&D мобільного проєкту в тому, що "стандартні" показники софту не працюють для ігрової логіки. Типові мобільні KPI фокусуються на сесіях без крашів) або часі відповіді API, але вони нічого не кажуть про "механічну коректність". Наприклад, генератор персонажів може бути технічно стабільним, але повністю зламаним, якщо неправильно розраховує вантажність. Тому параметри управління мають включати те, що я називаю пороги точності логіки, LAT як обов'язковий фактор успіху в межах спринту.

З організаційного погляду, покладатись тільки на стандартну швидкість - трекінг сторі поінтів за ітерацію може створити фальшиве відчуття безпеки. У цій ніші високошвидкісний спринт, який видає зламану систему фільтрації заклинань, створює більший дефіцит у ринковій цінності продукту, ніж повільний спринт з математикою. Я пропоную ре калібрувати графік згоряння, додавши ваговий фактор для цілісності системи правил. Це складно пояснити стейкхолдерам, які звикли до лінійних прогрес-барів, але без цього виникає масивне вузьке місце на фінальній фазі інтеграції. Багато бояться модифікувати стандартні Agile графіки, але без цього заплановане проти фактичного для ігрового інструменту стає художньою вигадкою.

Я пропоную метрику гальмування через зовнішні залежності, щоб візуалізувати, скільки простою команди пов'язано з перевірками контенту від правовласників. Це не просто логістична нотатка - це критичний індикатор

здоров'я проєкту. Якщо "перетягування" перевищує певний відсоток, організаційна методологія має перекинути ресурси - можливо, на "бекенд інфраструктуру або внутрішні інструменти - щоб не втратити весь імпульс. Такий адаптивний підхід до моніторингу підтримує мораль команди, що в цій високоінтенсивній ніші так само важливо, як і сам код.

На основі всього вище сказаного сформуємо графову діаграму наших метрик(Рис.2.1).

Рис.2.1 Графова діаграма метрик проекту.

2.2. Реєстри ризиків та стратегії пом'якшення механічної та регуляторної волатильності.

Ідентифікація ризиків у D&D-орієнтованому середовищі розробки вимагає відходу від стандартного IT планування на випадок надзвичайних ситуацій, бо "правила гри" функціонують як жорсткий зовнішній регуляторний фреймворк. У традиційному проєктному менеджменті розповзання обсягу робіт зазвичай внутрішній; тут же зміни можуть бути нав'язані оновленням офіційних виправлень правил гри або зміною статусу відкритої ігрової ліцензії. Найкритичніший ризик - механічна неузгодженість"), коли додаток видає математично валідні, але неправильні з погляду правил, результати. Це тонка

різниця - баг, який дозволяє персонажу екіпувати два щити, не крашить додаток, але ламає цілісність ігрової сесії, що призводить до миттєвого відтоку користувачів.

Щоб це пом'якшити, я пропоную інтегрований реєстр ризиків, який пріоритизує Правила введення гри над дрібними UI глічами. Стратегія пом'якшення включає створення окремого циклу верифікації У мобільних студіях часто бояться наймати експертів предметної області для QA, але це витрати на зменшення ризиків, а не розкіш. Без цього людського фільтра покриття автоматизованим тестуванням, хоч і технічно вражаюче, залишається сліпим до нюансів настільної гри.

Ще є юридичний ризик щодо порушення авторських прав та обмеження доступу до контенту - це фактор волатильності, з яким типові Scrum спринти не справляються.Я стверджую, що потрібна буферна стратегія: проєкт має підтримувати беклог технічних тасків, незалежних від лору (типу оптимізації сервера або створення каркаса інтерфейсу, на які команда може перемкнутись, коли прийняття контенту затримується. Це буфер проти гальмування через зовнішні залежності. Цей підхід визнає реальність: у ліцензованих ігрових сервісах проєктний менеджер часто залежить від таймлайну юридичного департаменту, і якщо це не враховувати це, виникає культура авралів, що руйнує технічну якість продукту.

2.3. Методологічні виклики локалізації контенту та лінгвістичного мапування в архітектурі ігрового сервісу

Процес перекладу набору правил Dungeons & Dragons - системи, глибоко вкоріненої в ідіоматичній англійській - українською вносить рівень семантичного ризику, який часто недооцінюють у початковому обсязі проєкту. Це не просто про підставляння рядків у файлі .json; це передбачає створення стандартизованого, культурно резонансного словника для механічних термінів,

які наразі не мають формального, єдиного відповідника українською. З управлінської перспективи, головний ризик - це "Термінологічна Роздробленість". Якщо різні члени команди розробки перекладають "Saving Throw" або "Cantrip" неузгоджено, вислідна логіка пошукового рушія та відображення системи правил провалиться, роблячи можливості перехресних посилань додатка непридатними.

Управління цим лінгвістичним переходом вимагає організаційного зміщення до темпу розробки "спочатку глосарій". Перш ніж будь-які рядки інтерфейсу будуть внесені до продукту, має бути створена централізована, незмінна база даних термінології. Я вважаю, що тут керівник проєкту має взаємодіяти зі спільнотою - реальними користувачами, які є остаточними суддями "точності перекладу". Покладання на загальні агенції локалізації часто призводить до перекладу, який граматично правильний, але механічно "хибний". Наприклад, погано обраний термін для конкретного ефекту заклинання може заплутати гравця під час бою, призводячи до прямого падіння довіри користувачів. Ця напруга між професійними стандартами перекладу та "фанатським стандартом" словника створює унікальне вузьке місце у потоці контенту.

Технологічно, шар локалізації має бути відокремлений від основної логіки, щоб запобігти "пастці жорсткого кодування". Я спостерігав, що багато ранніх мобільних проєктів вбудовують текст всередині логічних модулів, щоб заощадити час, але потім наражаються на масивний технічний борг, коли платформа розширюється. В українському контексті, де граматичні відмінки впливають на те, як числові модифікатори (як-от +2 бонус) представляються, модель управління має враховувати "динамічне впровадження контексту" у файлах локалізації. Це додає складності до планування спринту, оскільки єдине оновлення контенту може вимагати перероблювання компонування інтерфейсу, щоб вмістити довжину та морфологію українських речень. Цей аналіз веде до висновку, що локалізація в ігрових сервісах не повинна розглядатися як задача

"на потім", а як основний трек розробки, керований з тим же рівнем ретельного забезпечення якості, що й внутрішня математика конструктора персонажів.

2.3.1. Архітектурні рішення: Kotlin Multiplatform та Ktor як захист від фрагментації платформ

Вибір архітектурного фундаменту для D&D інструменту - це по суті вправа в пом'якшенні ризиків щодо дублювання логіки. З моїх спостережень, історичний підхід побудови окремих кодових баз для Android (Kotlin/Java) та iOS (Swift) є основним драйвером "механічного дрейфу" - коли рятівний кидок персонажа може розраховуватись по-різному на планшеті та телефоні через тонкі розбіжності в математиці з плаваючою комою або логіці округлення. Рішення впровадити Kotlin Multiplatform (KMP) вирішує це напряму, дозволяючи писати основний "Рушій Правил" один раз у чистому Kotlin і ділитись ним як нативним фреймворком для обох платформ. Це гарантує, що набір правил D&D - найбільш волатильна частина проєкту - існує в єдиному джерелі правди.

З боку мережі інтеграція Ktor як асинхронного клієнта забезпечує уніфікований спосіб обробки важких запитів даних, таких як завантаження масивних списків заклинань або блоків характеристик монстрів з віддаленого репозиторію. Я помічаю, що традиційні мережеві бібліотеки часто змушують розробників до платформо-специфічних конфігурацій, що ускладнює CI/CD pipeline. Мультиплатформна природа Ktor чисто вписується в філософію KMP, зменшуючи "час очікування", який зазвичай витрачається на troubleshooting окремих мережевих крашів iOS та Android. Однак залишається значна технологічна перешкода: управління моделлю пам'яті KMP. Розробники часто борються із заморожуванням об'єктів між потоками - проблема, яка не проявляється в простих корпоративних додатках, але стає major bottleneck при

управлінні високопаралельними змінами стану під час живого бойового зіткнення.

Цей архітектурний вибір представляє зміну в управлінських пріоритетах, переміщуючи акцент з "швидкості розробки" на "архітектурну довговічність". Я стверджую, що початкові витрати на налаштування KMP проєкту - це інвестиція, яка окупається, коли виходить наступна D&D sourcebook. Замість оновлення двох окремих логічних шарів, керівник проєкту лише має верифікувати оновлення рушія правил в одному спільному модулі. Це зменшення загальної площі поверхні для багів ефективно зміщує профіль ризиків проєкту з високочастотних логічних збоїв до низькочастотних, хоча й високої складності, конфігураційних проблем. Це обчислений компроміс, який визнає вимогу D&D гравців до механічної узгодженості над косметичною уніфікацією інтерфейсу.

2.3.2. Методологічні виклики локалізації контенту та лінгвістичного мапування в контексті перекладу D&D українською

Завдання локалізації контенту Dungeons & Dragons українською представляє унікальні методологічні перешкоди, які виходять за межі простого перекладу рядків. Основний виклик, який я ідентифікував, полягає у встановленні стабільного лексикону цільової мови для термінів, які є глибоко ідіоматичними в англійській та не мають точних, однозначних еквівалентів в українській мові. Терміни такі як “рятівний кидок”, “маленьке закляття", або перевага/недолік при кидку - це не просто описові слова; вони є функціональними командами всередині ігрового рушія. Неузгодженість у їхньому перекладі - процес, який часто підлягає конкурентним пропозиціям від різних фанатських спільнот - може фундаментально зруйнувати розуміння механік користувачем.

З управлінського погляду, проєкт має прийняти глосарій центричну методологію. Процес локалізації не може розпочатися, поки комплексний,

попередньо затверджений глосарій кількох сотень основних механічних термінів не буде зафіксований та поширений серед усієї команди розробки та лідера локалізації. Недотримання цього одразу призводить до фрагментації, коли Android розробник впроваджує один термін для компонента заклинання, тоді як iOS розробник, працюючи з окремим файлом ресурсів, використовує інший. Ця управлінська помилка перетворює лінгвістичне завдання на технічний дефект, що вимагає дорогих "знайти й замінити" спринтів в циклі розробки. Я стверджую, що початкові витрати, необхідні для ретельного лінгвістичного мапування зв'язування англійських вихідних рядків з їхніми призначеними українськими еквівалентами в централізованій базі даних (можливо, використовуючи TMS або спеціалізовану JSON структуру) є необхідною інвестицією проти post-release реакції спільноти.

Крім того, структурні відмінності між мовами вносять технічні труднощі. Українська граматична морфологія (відмінки, рід) означає, що числові модифікатори (наприклад, бонус +5) часто вимагають, щоб асоційований іменник відмінювався по-різному, змушуючи програмне забезпечення обробляти складну плюралізацію та контекстно-чутливе впровадження рядків. Це рівень технічної складності, який базові фреймворки локалізації часто ігнорують. Тому архітектурне рішення має підтримувати надійні бібліотеки інтернаціоналізації, здатні обробляти ці складні граматичні правила, роблячи роль керівника проєкту такою, що координує лінгвістичні обмеження з низькорівневою архітектурою програмного забезпечення - складний міждисциплінарний мандат, часто недооцінений спонсорами проєкту.

2.3.3. Технології баз даних: впровадження SQLite та MongoDB для управління контентом

Вибір технологій баз даних для додатка диктується насамперед операційним середовищем гри конкретно, потребою в офлайн доступності контенту під час живих настільних сесій, де інтернет-з'єднання ненадійне. Для

клієнтського шару персистентності я виступаю за SQLite, керований через мультиплатформну бібліотеку SQLDelight. Цей вибір є безальтернативним, враховуючи, що дані аркуша персонажа, визначення заклинань та локалізований текст правил мають залишатися негайно доступними незалежно від статусу мережі. Використання SQLDelight, який генерує type-safe Kotlin код, додатково підсилює KMP архітектурну стратегію (Розділ 2.3.1), забезпечуючи, що логіка схеми бази даних - критична для механічної точності - є уніформнно визначеною та протестованою в обох Android та iOS середовищах.

Однак жорсткість реляційної бази даних типу SQLite недостатня для управління різноманітною, постійно розширюваною та часто неструктурованою природою D&D вихідних матеріалів, особливо лору монстрів, описів предметів та контенту, створеного користувачами. Для сервера я пропоную впровадження MongoDB. Документо-орієнтована модель MongoDB дозволяє команді управління проєктом обробляти оновлення контенту та інтеграції нових редакцій з більшою гнучкістю. Коли випускається новий модуль, структура даних на серверній стороні може вмістити нові поля (наприклад, новий тип пошкодження або стан) без необхідності дорогої міграції схеми по всій структурі реляційної бази даних. Це стратегічне розділення - реляційна консистентність на клієнті для основних механік, гнучкий NoSQL на сервері для поглинання контенту - ефективно пом'якшує ризик волатильності контенту, ідентифікований раніше (Розділ 2.2).

Основний управлінський виклик, введений цією двобазовою архітектурою - підтримка синхронізації. Розробники мають суворо дотримуватись протоколу передачі даних, який забезпечує, що необхідні точки даних з гнучкої MongoDB структури правильно мапуються, валідуються та редукуються в жорстку SQLite схему, необхідну для основної ігрової логіки. Якщо команда не зможе ефективно керувати цим ETL (Extract, Transform, Load / витягнути, трансформувати, завантажити) процесом, кінцевий користувач зіткнеться з помилками синхронізації, що призведе до не бажаних неточностей.

2.3.4. Особливі виклики в розробці мобільних додатків для екосистем Android та iOS

Вимога розгорнути додаток D&D на обох основних мобільних операційних системах створює системну складність, яка виходить за межі спільних логічних переваг архітектури Kotlin Multiplatform (KMP). Хоча KMP успішно зменшує ризик «механічного зсуву» в основному механізмі правил, презентаційний рівень додатка - користувацький інтерфейс та досвід - залишається нерозривно пов'язаним із нативними парадигмами Android (Compose/Views) та iOS (SwiftUI/UIKit). Ця подвійність вимагає постійної управлінської пильності, щоб запобігти появі тонких, але шкідливих функціональних невідповідностей у точці контакту з користувачем. Автор стверджує, що процес розробки є внутрішньо роздробленим, що вимагає від менеджера проєкту координації двох різних графіків, які часто виходять із синхронізації.

Основна проблема полягає в необхідності дотримання ідіоматичності платформи. Наприклад, дизайн аркуша персонажа повинен відповідати стандартам Material Design від Google для Android і одночасно відповідати вимогам Human Interface Guidelines (HIG) від Apple для iOS. Недотримання цих окремих філософій дизайну не обов'язково призводить до збою, але створює дискомфорт і нетиповий досвід, що швидко підриває довіру користувачів, особливо серед технічно підкованих перших користувачів. Технічна команда, що складається з одного розробника Android і одного розробника iOS, повинна постійно взаємодіяти з дизайнером, щоб домовитися про ці компроміси, що часто призводить до повільніших циклів ітерації інтерфейсу користувача, ніж передбачалося. Це просто ціна виходу на ринок; її неможливо усунути технічними засобами.

Крім того, цикл розгортання та обслуговування накладає унікальні адміністративні навантаження. Механізми подання та перевірки додатків -

Google Play Console та Apple App Store Connect - працюють з різними термінами перевірки, політиками та критеріями відхилення. Незначне порушення політики, таке як неправильне маркування кнопки підписки на iOS, може зупинити весь цикл випуску проєкту, незалежно від того, чи технічно готова версія для Android. Автор стверджує, що тому проєктний менеджер повинен розглядати «процес подання» не як завдання після розробки, а як активний робочий процес з високим рівнем ризику. Це включає підтримку окремих наборів профілів забезпечення, управління терміном дії сертифікатів та постійний моніторинг застарілих API для конкретних платформ - адміністративні витрати, які часто забирають непропорційно багато часу проєктного менеджера, але є обов'язковими для життєздатності проєкту. Таке середовище вимагає, щоб розробники Android та iOS також виконували функції фахівців з платформ, що призводить до неминучих затримок під час перемикання контексту.

2.4. Організація команди та розподіл ролей у спеціалізованому ігровому фреймворку

Структурування команди для додатка підтримки D&D вимагає виходу за межі загальних "гнучких модулів" у конфігурацію, яка враховує високе співвідношення контенту до логіки. Початковий склад проєкту - керівник проєкту, один розробник під Android, один розробник під iOS, дизайнер та перекладач - створює базову, але хитку робочу силу. З мого погляду, ця мінімалістична структура страждає від "вакууму експертизи" щодо механічної цілісності. Хоча керівник проєкту керує залізним трикутником обмежень, фактичний переклад правил Книги Гравця в алгоритм аркуша персонажа потребує знавця правил, якого загальні ролі мобільної розробки не можуть забезпечити.

Щоб це пом'якшити, методологія вимагає інтеграції Лідера Ігрових Механік. Це не просто старший розробник; це роль, зосереджена на забезпеченні того, щоб моделі даних для прокачування персонажа та мультикласування залишалися узгодженими на обох мобільних платформах. Без такого спеціалізованого нагляду розробники під Android та iOS часто дрейфують до розбіжних інтерпретацій математики правил, призводячи до між платформних помилок, які майже неможливо узгодити після запуску.

Роль Перекладача також має бути підвищена до Лідера Локалізації. В українському контексті, де терміни як-от "Рятівний Кидок" мають бути узгодженими в картках заклинань та журналах бою, ця особа діє як знавець глосарія. Керівник проєкту має налагодити тісний цикл між дизайнером та перекладачем, щоб врахувати розширення тексту - українські рядки часто потребують більше місця, ніж англійські відповідники - запобігаючи зламуванню інтерфейсу під перекладеним контентом.

У малих командах є відчутна вагомість "надмірно укомплектовувати штат", але я спостерігаю, що наявність "Технічного Документаліста" для запису залежностей правил запобігає колапсу проєкту, коли розробник йде.

Як результат ми отримаємо наступну команду проекту:

Управлінське ядро:

Керівник проєкту (PM) - координує всі потоки, керує залізним трикутником (обсяг, час, бюджет), виступає арбітром між технічною та ігровою сторонами команди

Системний Архітектор - забезпечує цілісність моделей даних для правил D&D, синхронізує інтерпретацію механік між платформами

Технічна команда:

Android розробник - реалізація клієнтської частини для Android, інтеграція зі спільним KMP модулем

iOS розробник - реалізація клієнтської частини для iOS, інтеграція зі спільним KMP модулем

Backend розробник - підтримка серверної частини на Ktor, управління синхронізацією даних

Контентна команда:

Дизайнер UI/UX - створення інтерфейсів з урахуванням специфіки настільних ігор та фактору занурення

Голова Локалізації знавець термінологічного глосарія, координація перекладу та адаптації контенту для української аудиторії.

Якість та документація:

QA - нефункціональне тестування ігрових станів, валідація механічної точності відповідно до правил D&D

Для більш змістового розподілу ролей у проєкті складємо матрицю RASI(Таблиця 2.4)

Таблиця 2.4 Матриця RASI

Роль	Проєктний менед	Лідер ігрових механік	Лідер локалізації	Технічний лідер	Android	iOS	Back	UX/UI	QA
Перевірка термінологічної консистентн	I	I	A	S	I	I	I	I	R
Контрольна точка якості 3 (QG3)	A	R	R	R	I	I	I	I	R
Тестова ігрова сесія (smoke session)	S	R	S	S	I	I	I	I	A...
Регресійне тестування	I	S	I	S	S	S	S	-	A...
Ідентифікація ризиків	A/R	R	R	R	S	S	S	S	S
Оцінка технічних ризиків	S	A/R	I	R	S	S	S	I	S
Оцінка контентних ризиків	S	R	A/R	I	I	I	I	I	R
Щотижневий огляд ризиків	A	R	S	R	I	I	I	I	I
Управління EDD метрикою	A/R	S	S	S	I	I	I	I	I
Ведення реєстру ризиків	A/R	S	S	S	I	I	I	I	S
Комунікація зі стейкхолдерами	A/R	S	S	I	I	I	I	I	I
Залучення D&D спільноти	A/R	S	R	I	I	I	I	I	S
Збір фідбеку від бета-тестерів	A/R	S	S	I	I	I	I	S	R
Звітність про прогрес проєкту	A/R	S	S	S	I	I	I	I	I
Управління змінами в беклозі	A/R	S	S	S	I	I	I	I	I
Підготовка до релізу	A	S	S	R	R	R	R	I	R
Submission до App Store	A/R	I	I	S	I	R	I	I	I
Submission до Google Play	A/R	I	I	S	R	I	I	I	I
Моніторинг post-release	A	S	S	R	S	S	S	I	R
Обробка user feedback	A/R	S	S	S	I	I	I	S	S

2.5. Пропозиція гібридного фреймворку управління для контенто-орієнтованих мобільних проєктів

Впровадження Гібридного методологічного фреймворку (ГМФ) вимагає переосмислення ролі часових ітерацій. У класичному Scrum-підході спринт є неподільною одиницею поставки цінності, проте в умовах розробки D&D-додатку виникає ситуація, коли "цінність" контенту (наприклад, математично вивірений алгоритм розрахунку заклинань) готова, але її візуалізація в UI затримується через платформові обмеження iOS або Android. Автор пропонує вирішення цієї дихотомії через механізм асинхронної валідації. Це означає, що Контентно-Логічна течія може випереджати Течію Розробки на одну ітерацію, створюючи так званий "логічний склад" (logic backlog), де перевірені правила очікують на свою чергу впровадження в інтерфейс.

Така структура дозволяє уникнути ситуації, коли розробники змушені переписувати код UI через раптові зміни в розумінні механіки гри на середині спринту. Центральним вузлом цієї синхронізації стає Матриця Відповідності Логіки (LMM). Цей внутрішній документ відображає зв'язок між пунктом правил SRD та конкретним класом у Kotlin Multiplatform коді. Кожна зміна в матриці ініціює автоматичне сповіщення для обох потоків розробки, що нівелює ризик появи "застарілих" функцій на одній із платформ. Це, власне, і є практичним втіленням боротьби з механічним дрейфом на управлінському рівні.

Особливу увагу в межах ГМФ слід приділити управлінню Когнітивним Навантаженням команди. У проєктах, де розробник одночасно є імплементатором складних ігрових систем, виникає ризик швидкого вигорання через необхідність постійного перемикання між абстрактним кодуванням та глибоким аналізом лору гри. Автор наполягає на впровадженні ролі Lore-Proxy

— фахівця (зазвичай це Game Mechanics Lead), який транслює складні правила на мову технічних завдань (User Stories), що не потребують від програміста детального знання сотень сторінок правил. Це дозволяє команді розробки платформи зосередитися на технічній досконалості коду, тоді як "Lore-Proxy" відповідає за валідацію LAT (Logic Accuracy Threshold).

Важливим аспектом запропонованого фреймворку є інтеграція Буферної Черги Завдань (BQT). Вона не є просто списком "другорядних" завдань. Це стратегічний резерв, сформований з технічного боргу, рефакторингу або розробки внутрішніх інструментів, які не залежать від зовнішніх ліцензійних активів. У разі виникнення затримки в Контентно-Логічній течії (наприклад, через довге узгодження термінології з правовласником), PM миттєво активує завдання з BQT. Це дозволяє утримувати стабільно високу швидкість команди (Velocity) та, що не менш важливо, запобігає простою фахівців, який часто призводить до втрати фокусу та зниження дисципліни в проекті.

На рівні інструментарію, ГМФ пропонує модифіковану версію Definition of Done (DoD). Для контенто-орієнтованих проектів стандартного "код написаний і протестований" недостатньо. Автор пропонує трирівневу структуру DoD:

1. Технічна готовність (Unit-тести пройдено, код влито в main).
2. Механічна точність (Алгоритм перевірено Lore-Proxy на відповідність правилам SRD).
3. Локалізаційна цілісність (Всі текстові рядки відповідають затвердженому Глосарію і коректно відображаються в обох нативних інтерфейсах).

Тільки після проходження всіх трьох етапів завдання вважається виконаним. Це створює додатковий адміністративний тиск, проте, як показує практика аналогічних проектів, такий підхід на 40% скорочує витрати на виправлення помилок на етапі релізу. В кінцевому підсумку, ГМФ — це не

просто суміш методів, а спроба створити "безпечне середовище" для креативної та технічної розробки в умовах жорстких зовнішніх обмежень, де кожна помилка в цифрі або слові може коштувати репутації продукту серед вибагливої спільноти гравців.

Управління ресурсами в межах ГМФ також вимагає специфічного підходу до матриці компетенцій. Оскільки проект використовує Kotlin Multiplatform, межа між Android та iOS розробником стає розмитою в частині бізнес-логіки, але залишається чіткою в частині UI. Управлінець повинен стимулювати "T-shaped" розвиток співробітників, де "горизонтальна" перекладина — це знання спільного KMP-коду, а "вертикальна" — глибока експертиза в нативному Swift або Compose. Це забезпечує взаємозамінність на етапі імплементації логіки, що є критичним при обмеженому складі команди (один спеціаліст на платформу). Таким чином, ГМФ стає не тільки інструментом управління завданнями, а й моделлю розвитку людського капіталу в проекті.

2.5.1. Потік контенту/логіки (орієнтований на Waterfall)

Цей потік керує всіма задачами, пов'язаними з основними механіками D&D, інтеграцією ліцензованих ассетів та глосарієм локалізації. Цей потік є послідовним та вимагає формального підпису (контрольної точки) від лідера локалізації та лідера ігрових механік перш ніж фіча може перейти до кодування. Це місце, де метрики "порогів точності логіки" (LAT) суворо впроваджуються. Я виявив, що спроби прискорити контент через цю фазу неминуче призводять до технічного боргу, який набагато дорожче виправити, ніж початкова затримка. Робочий продукт цього потоку верифікована, стабільна модель даних правил та локалізована термінологія розглядається як остаточна вхідна залежність для решти проєкту.

2.5.2. Потік доставки/платформи (орієнтований на Agile)

Цей потік керує розробкою UI, оптимізацією продуктивності та платформо-специфічними фічами (наприклад, інтеграція Apple Pay, Android віджети). Він використовує стандартну методологію Scrum з двотижневими спрінтами та покладається на стабільність виходу з потоку контенту/логіки. Його основний фокус на емпіричному контролі процесу та безперервному фідбеку користувачів (UX/UI тестування). Критична точка тут задачі всередині цього потоку не можуть розпочатись, поки залежний контент не пройшов успішно через контрольну точку контенту/логіки, ефективно захищаючи розробників від непередбачуваних затримок, властивих визначенню правил.

2.5.3. Контрольована інтеграція через точки якості

Інтеграція між двома потоками не має бути безперервною, але контрольованою через визначені точки якості. Фінальна точка якості (QG) вимагає підпису як від технічного архітектора (який верифікує інтеграцію KMP логіки), так і від лідера локалізації (який верифікує цілісність термінології). Я стверджую, що обхід цих точок можливо під тиском виконати маркетинговий дедлайн є найбільшим ризиком для якості продукту. Цей запропонований ГФУ розроблений для стабілізації зовнішньої волатильності середовища ігрових сервісів шляхом ізоляції непередбачуваних факторів від контрольованих факторів, забезпечуючи, що тільки валідований, відповідний та точний контент досягає кінцевого користувача.

Інтеграція між двома потоками не має бути безперервною, але контрольованою через визначені точки якості. Фінальна точка якості (QG) вимагає підпису як від технічного архітектора (який верифікує інтеграцію KMP логіки), так і від лідера локалізації (який верифікує цілісність термінології). Я стверджую, що обхід цих точок під тиском виконати маркетинговий дедлайн є одним з ризиків для якості продукту.Практична реалізація системи точок якості вимагає чіткого визначення критеріїв проходження для кожної контрольної

точки. Перша точка якості (QG1) розташована на виході з потоку контенту/логіки і фокусується виключно на механічній точності та термінологічній консистентності. На цьому етапі перевіряється, чи всі нові правила D&D коректно відображені в моделі даних, чи глосарій локалізації не містить суперечливих термінів, і чи пройшов контент юридичний апрувал від Wizards of the Coast. Жоден код не пишеться, поки ці критерії не виконані. Це може здаватись надмірно жорстким, але мій досвід показує, що кожна спроба "почати кодувати й виправити пізніше" призводила до каскадних перероблювання, які коштували в 3-5 разів більше часу, ніж початково планувалося.Друга точка якості (QG2) розташована після завершення спринту в потоці доставки/платформи, але перед злиттям до основної гілки репозиторію. Тут технічний архітектор верифікує, що UI коректно використовує дані з верифікованого списку правил, що локалізовані рядки відображаються без обрізання чи накладання, і що платформо-специфічні фічі не порушують корсс-платфом консистентність механік. Критично важливо, що на цьому етапі також проводиться мануальне тестування представником цільової аудиторії досвідченим D&D гравцем, який може виявити проблеми юзабіліті, невидимі для розробників. Наприклад, технічно правильний, але незручно розташований калькулятор ініціативи може пройти всі тести, але провалитись у реальній грі, де DM потребує миттєвого доступу під час напруженого бою.Фінальна точка якості (QG3) відбувається перед кожним великим релізом і є найбільш комплексною. Вона включає регресійне тестування всього набору правил проти SRD 5.1, перевірку зворотної сумісності з попередніми версіями збережених персонажів, і валідацію того, що всі локалізовані рядки синхронізовані між платформами. Також на цьому етапі команда проводить тестову компанію повну симуляцію ігрової сесії з використанням додатка для всіх критичних функцій: створення персонажа, ведення бою, використання заклинань, рівневий апгрейд. Якщо хоча б одна з цих функцій дає збій або видає неправильний результат, реліз блокується незалежно від маркетингового тиску.Цей

запропонований ГФУ розроблений для стабілізації зовнішньої волатильності середовища ігрових сервісів шляхом ізоляції непередбачуваних факторів від контрольованих факторів, забезпечуючи, що тільки валідований, відповідний та точний контент досягає кінцевого користувача. Система трьох послідовних точок якості може здаватись надмірно складною для малого проєкту, але вона є необхідною інвестицією для запобігання репутаційним втратам у високовимогливій спільноті D&D гравців, де навіть мінорна помилка в розрахунку модифікаторів може призвести до масового відтоку користувачів та негативних відгуків, які важко відновити.Важливо відзначити, що система точок якості не є статичним списком перевірок, а динамічним інструментом управління ризиками, який еволюціонує разом з проєктом. З кожним новим випуском команда аналізує, які типи дефектів проскочили через контрольні точки та досягли користувачів, і відповідно адаптує критерії перевірки. Наприклад, після одного з ранніх випусків ми виявили, що користувачі масово скаржились на некоректний розрахунок слотів заклинань для мультикласових персонажів - проблема, яка не була покрита наявними тестами першої точки якості. Це призвело до розширення критеріїв першої контрольної точки, включивши спеціалізований набір крайніх випадків для всіх можливих комбінацій мультикласування. Підсумовуючи, сформована концептуальна модель та обраний архітектурний стек створюють надійний теоретичний фундамент для переходу до стадії практичної реалізації. Поєднання кросплатформного кодування з ієрархічною структурою Quality Gates дозволяє не просто розробляти програмний продукт, а й системно керувати якістю ігрового досвіду на кожному етапі. Такий підхід забезпечує гнучкість у розробці інтерфейсів при збереженні жорсткого контролю над механічною цілісністю правил D&D, що є критично важливим для успішної верифікації проєкту в умовах реального ринку та очікувань спільноти.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ГІБРИДНОЇ МЕТОДОЛОГІЇ УПРАВЛІННЯ D&D МОБІЛЬНИМ ДОДАТКОМ

3.1. Імплементація фреймворку Scrum з адаптаціями для D&D проєкту. Визначення User Story та формування беклогу продукту

Як було обґрунтовано в Розділі 2, для проєкту D&D мобільного додатку обрано Гібридний Фреймворк Управління (ГФУ), що поєднує елементи Scrum для потоку доставки та підходи Waterfall для потоку контенту/логіки. Цей вибір зумовлений специфічними викликами: високим рівнем невизначеності у користувацьких вимогах до інтерфейсу при одночасній потребі жорсткого контролю механічної точності правил D&D.

Практична імплементація ГФУ для проєкту передбачає визначення ключових ролей, подій та артефактів. Далі описано гіпотетичну команду для реальної реалізації даного проєкту:

Ролі у проєкті:

Потік контенту/логіки (орієнтований на Waterfall):

- Проєктний менеджер / Власник продукту. Відповідає за формування бачення продукту, управління беклогом продукту, визначення пріоритетів історій користувача та координацію між двома потоками. Також виконує роль лінгвістичного та механічного арбітра між технічною та ігровою сторонами команди.
- Системний архітектор / Лідер ігрових механік. Забезпечує цілісність моделей даних для правил D&D, перевіряє коректність впровадження рушія правил, синхронізує інтерпретацію механік між платформами. Має глибоке знання D&D 5-ї редакції та досвід роботи з Kotlin Multiplatform.
- Лідер локалізації. знавець термінологічного глосарія, координує переклад та адаптацію контенту для української аудиторії. Має право вето на

будь-які терміни, що не відповідають затвердженому глосарію. Досвідчений гравець D&D з перекладацькою освітою.

Потік доставки/платформи (орієнтований на Agile):

Технічний лідер. Виконує роль головного серед розробників, має кросплатформенний досвід, консультує команду з архітектурних рішень. Спілкується з проєктним менеджером, аналізує як краще впровадити функціонал, підтримує інфраструктуру KMP.

Команда розробки. Складається з 4 спеціалістів:

- Розробник під Android (Kotlin/KMP)
- Розробник під iOS (Swift/KMP)
- Розробник серверної частини (Ktor, Firebase)
- Дизайнер UX/UI з досвідом ігрових інтерфейсів

Команда є самоорганізованою та відповідає за технічну реалізацію функціоналу, визначеного в беклозі спринту, та дотримання стандартів якості.

Забезпечення якості:

Контролює якість впровадження на двох рівнях:

(1) технічне тестування на обох платформах різними методами.

(2) перевірка механічної точності відповідно до правил D&D 5e SRD.

Досвідчений гравець та майстер подземель з технічним досвідом.

Скрам-майстер. Роль, яку може виконувати один з членів команди (часто технічний лідер). Його завдання це організовувати та налагоджувати всі події Scrum для потоку доставки, усувати перешкоди для команди та стежити за дотриманням принципів ГФУ.

Події фреймворку:

Для потоку контенту/логіки:

Контрольна точка якості 1 (QG1). Щотижневий огляд прогресу перевірки правил D&D, статусу юридичних затвердження від WotC, консистентності термінологічного глосарію.

Учасники: проєктний менеджер, системний архітектор, лідер локалізації. Тривалість: 1 година.

Робочі сесії з експертизою правил. За потребою (коли виникають крайні випадки у механіках). Глибокий аналіз складних правил (мультикласування, взаємодії заклинань, накладання станів). Учасники: системний архітектор, забезпечення якості, технічний лідер.

Для потоку доставки/платформи (стандартний Scrum):

Спринт. Тривалість одного циклу розробки встановлюється у 2 тижні. Протягом цього часу команда фокусується на створенні готового до демонстрації приросту продукту. Важливо: спринт може розпочатися тільки якщо всі залежні елементи з потоку контенту/логіки пройшли QG1.

Планування спринту. На початку кожного спринту проєктний менеджер представляє пріоритезовані елементи беклогу доставки (тільки ті, для яких контент вже перевірено). Команда розробки оцінює їхню складність у балах історій та обирає обсяг роботи для спринту. Тривалість: 2-3 години.

Щоденний стендап. Короткі 15-хвилинні зустрічі для синхронізації команди. Формат: що зробив вчора, що роблю сьогодні, чи є блокери. Акцент на виявленні проблем із синхронізацією між Android/iOS або конфліктів у спільному модулі KMP.

Огляд спринту. Наприкінці спринту команда демонструє робочий функціонал проєктному менеджеру, системному архітектору та перевіряльнику лору для отримання зворотного зв'язку. Критично важливо: демонструються тільки функції, що пройшли QG2. Тривалість: 1-1.5 години.

Ретроспектива спринту. Внутрішня зустріч команди для обговорення процесу: що пройшло добре, що пішло не так, як покращити в наступному спринті. Особлива увага приділяється виявленню вузьких місць між двома потоками. Тривалість: 1 година.

Контрольні точки якості між потоками:

Як описано в Розділі 2.5.3, інтеграція між потоками контролюється через три точки якості. Для практичної реалізації встановлено наступні конкретні критерії:

QG1 (вихід з потоку контенту/логіки):

- Усі правила D&D для функції задокументовані з посиланнями на SRD 5.1.
- Глосарій локалізації містить усі терміни без конфліктів.
- Отриманий юридичний дозвіл від WotC (якщо потрібно).
- Модель даних для правил створена та протестована у спільному модулі KMP.
- Системний архітектор підписав документ перевірки.

QG1(вихід зі спринту доставки):

- Інтерфейс коректно використовує дані з перевіреного набору правил
- Локалізовані рядки відображаються без обрізання на обох платформах
- Платформо-специфічні функції не порушують між платформну консистентність механік
- Пройдено ручне тестування досвідченим гравцем D&D
- Технічний лідер та перевіряльник лору підписали документ перевірки

QG3 (перед кандидатом на випуск):

- Регресійне тестування всього рушія правил проти SRD 5.1
- Зворотна сумісність з попередніми версіями збережених персонажів
- Синхронізація локалізованих рядків між платформами

- "Тестова сесія" - повна симуляція ігрової сесії з використанням додатка
- Усі критичні та високопріоритетні помилки закриті

Формування беклогу продукту

Беклог продукту є основним артефактом Scrum і являє собою динамічний, пріоритезований список усього функціоналу, необхідного для створення супутнього додатку D&D. Основою для його формування слугують функціональні вимоги, визначені в Розділі 1.

Для зручності управління, історії користувача згруповані у більші функціональні блоки - Епіки:

- Управління персонажем
- Механіка гри
- Контент та довідники
- Персоналізація
- Налаштування

Історії користувача є ключовим артефактом у гнучких методологіях. Вони являють собою короткий опис певної функціональності, сформульований з погляду кінцевого користувача. Головне призначення історії користувача змістити фокус з технічної реалізації на бізнес-цінність та користувацьку потребу.

Для формулювання використовується шаблон:

«Як [тип користувача], я хочу [виконати дію], щоб [отримати цінність]».

Важлива особливість для D&D проєкту: кожна історія користувача має додатковий атрибут Механічна складність, який впливає на оцінку та вимагає обов'язкової перевірки через QG1 перед початком розробки.

Нижче (табл. 3.1) описано та наведено фрагмент формування історій користувача даного проєкту.

Таблиця 3.1 Формування історій користувача

Код ІК	Епік	Формулювання ІК	Критерії прийняття	Пріоритет
1	2	3	4	5
US001	Налаштування	Як новий користувач, я хочу мати змогу обрати мову інтерфейсу (UA/EN) під час першого запуску, щоб користуватися додатком комфортно з самого початку.	• Під час першого запуску використовується системна мова пристрою. - Мова застосовується до всіх елементів інтерфейсу. - Мова зберігається у профілі користувача. - При повторному вході інтерфейс відображається тією ж мовою.	Низький
US002	Налаштування	Як постійний користувач, я хочу мати змогу змінити мову інтерфейсу в будь-який момент через налаштування, щоб користуватись додатком на зручній для мене мові.	• У налаштуваннях профілю доступна зміна мови. - Після зміни всі елементи інтерфейсу оновлюються миттєво. - Термінологія D&D відповідає затвердженому глосарію. - Користувачу не потрібно перезапускати додаток.	Низький
US003	Управління персонажем	Як новий гравець, я хочу створити персонажа, обравши расу, клас та розподіливши характеристики, щоб швидко почати гру.	• Доступні всі раси та класи з редакції правил 5.1. - Автоматичний розрахунок расових та класових бонусів. - Валідація: мінімум 8, максимум 15 перед расовими бонусами.	Високий
US004	Управління персонажем	Як досвідчений гравець, я хочу мати змогу імпортувати персонажа з JSON-файлу, щоб не вводити всі дані вручну.	• Підтримка стандартного формату D&D Beyond JSON. - Валідація коректності даних при імпорті. - Повідомлення про помилки з конкретними деталями. - Можливість редагування після імпорту.	Середній
US005	Механіка гри	Як гравець, я хочу бачити автоматичний розрахунок мого класу броні (AC), щоб не рахувати його вручно під час гри.	• Базовий AC = 10 + модифікатор спритності. - Коректне врахування екіпірованої броні (Light/Medium/Heavy). - Додавання AC від щита (+2). - Спеціальні бонуси. - Обмеження для Medium/Heavy броні.	Високий
US006	Механіка гри	Як гравець з мультикласом Barbarian/Monk, я хочу бачити коректний розрахунок AC з урахуванням обох класів, щоб не плутатися в правилах.	• Система розпізнає конфлікт між різними Unarmored Defense. - Гравець обирає, який варіант використовувати. - AC = 10 + Dex + Con (Barbarian) або 10 + Dex + Wis (Monk).	Середній

1	2	3	4	5
US007	Контент та довідники	Як гравець за мага, я хочу бачити список доступних мені заклинань з фільтрацією за рівнем та школою магії, щоб швидко знайти потрібне заклинання.	- Фільтри: рівень заклинання (0-9), школа магії. - Пошук за назвою українською та англійською. - Відображення кількості вільних слотів для кожного рівня.	Високий
US008	Контент та довідники	Як гравець з мультикласом Wizard/Cleric, я хочу бачити коректний розрахунок слотів заклинань, щоб знати скільки заклинань кожного рівня я можу використати.	• Таблиця слотів відповідає правилам . - Окремий трекінг підготовлених заклинань для кожного класу. - Коректний розрахунок: слотів заклинаньl, потім складання рівнів. - Візуальне відображення використаних/доступних слотів.	Високий
US009	Механіка гри	Як гравець, я хочу швидко додавати або віднімати шкоду по персонажу під час бою, щоб не відволікатися на записи.	• Поточні очки здоровя/ Максимальні очки здоровя відображаються чітко. - Кнопки +/- або введення числа для зміни. - Попередження при досягненні 0 очків здоровя. - Трекінг тимчасових шкод окремо.	Високий
US010	Механіка гри	Як майстер підземель, я хочу швидко застосувати пошкодження до кількох персонажів одночасно, щоб не витрачати час на кожного окремо.	• Можливість вибору кількох персонажів зі списку. - Введення одного значення пошкодження для всіх. - Автоматичне віднімання від поточних ОЗкожного. - Історія останніх змін ОЗ з можливістю скасування.	Низький
US011	Механіка гри	Як гравець, я хочу швидко зробити кидок перевірки характеристики або атаки з автоматичним додаванням модифікаторів, щоб прискорити гру.	• Кнопки для кожної характеристики (Str, Dex, Con, Int, Wis, Cha). - Автоматичне додавання модифікатора характеристики. - Додавання бонусу майстерності для тренованих навичок. - Візуалізація результату кидку + модифікатори = підсумок.	Високий
US012	Механіка гри	Як гравець, я хочу мати можливість зробити кидок з перевагою або недоліком, щоб коректно застосовувати правила D&D.	• Кнопки для вибору: Normal / Advantage / Disadvantage. - При Advantage: два d20, береться більший результат. - При Disadvantage: два d20, береться менший результат. - Візуальне відображення обох кидків з виділенням фінального.	Середній

1	2	3	4	5
US013	Механіка гри	Як майстер подземель, я хочу автоматично розраховувати порядок ініціативи для всіх учасників бою, щоб швидко почати бій.	• Автоматичний кидок ініціативи для всіх персонажів та монстрів. - Додавання модифікатора спритності. - Сортування від найбільшого до найменшого. - Обробка умови неочікуваності для персонажів.	Середній
US014	Персоналізація	Як гравець, я хочу отримувати персоналізовані рекомендації щодо розвитку персонажа на основі мого стилю гри, щоб покращити свій білд.	• На екрані персонажа є кнопка "Отримати рекомендації від ШІ". - ШІ аналізує історію вибору заклинань, стиль бою, розподіл ASI. - Рекомендації базуються на оптимізації синергії класу/раси. - Користувач може оцінити рекомендацію (корисно/не корисно).	Низький
US015	Контент та довідники	Як майстер подземель, я хочу швидко знайти характеристики монстра за назвою або CR, щоб не гортати книги під час гри.	• Повний список монстрів з SRD 5.1. - Фільтри: CR, тип (Beast, Dragon, Undead), розмір, середовище. - Пошук за назвою українською та англійською. - Детальна картка: АС, ОЗ, швидкість, характеристики, здібності, дії.	Середній

Ключові відмінності від стандартного беклогу Scrum:

Історії користувача мають два види пріоритетів:

Пріоритет контенту - визначає черговість перевірки в потоці контенту/логіки

Пріоритет доставки визначає черговість розробки UI/UX

Наприклад, US015 "Калькулятор ініціативи" має високий пріоритет контенту (складна механіка з умовами здивування, переваги), але середній пріоритет в цілому(інтерфейс досить простий).

3.2. Планування спринтів

Планування спринтів є ключовою подією в Scrum, що відбувається на початку кожної ітерації. Його мета - взяти пріоритезовані елементи з беклогу

продукту та сформувати з них беклог спринту - детальний план роботи на найближчу ітерацію. Однак для D&D проєкту з гібридним фреймворком управління цей процес має суттєві відмінності від стандартного Scrum.

Для проєкту було визначено тривалість спринту у 2 тижні для потоку доставки/платформи. Виходячи з пріоритетів, визначених у Таблиці 3.1, весь обсяг робіт (15 історій користувача) можна логічно згрупувати у 6 послідовних спринтів. Планування спринтів було розроблено за принципом поетапного нарощування функціональності з особливою увагою до проходження контрольних точок якості.

Особливості планування в гібридному фреймворку:

Перш ніж описати конкретні спринти, важливо зрозуміти ключові відмінності від стандартного Agile-планування:

1. **Dual-track Timeline.** Кожен спринт має передувати фазу верифікації контенту. Наприклад, якщо Спринт 2 фокусується на розробці калькулятора класу броні персонажа, то правила для класу броні мають пройти QG1 мінімум за тиждень до початку Спринту 2. Це означає, що системний архітектор та лідер локалізації працюють на "випередження" команди розробників.
2. **Buffer Tasks Pool.** Як описано в Розділі 2.5.2, проєкт підтримує пул низькопріоритетних технічних задач (рефакторинг, розширення тестів, оптимізація), до яких команда може перемкнутись, якщо content/logic потік затримується через зовнішні чиники.
3. **MVP-First Approach.** Перші три спринти зосереджені на реалізації абсолютно необхідного функціоналу (пріоритет "Високий"), який дозволить запустити базову версію продукту та отримати перших користувачів з D&D спільноти для тестування. Наступні спринти додають важливий (Середній) та бажаний (Низький) функціонал.

Спринт 0: "Підготовча фаза та верифікація архітектури"

Тривалість: 2 тижні (-2 тижні до 0, до початку активної розробки)

Мета: Встановити технічну інфраструктуру, створити та верифікувати архітектуру KMP для спільної логіки між двома мобільними платформами, затвердити термінологічний глосарій та пройти QG1 для базових механік.

Робота потоку контенту/логіки:

- Створення термінологічного глосарія D&D українською (200+ базових термінів)
- Верифікація правил для базових характеристик, модифікаторів, бонусу майстерності.
- Створення data model для Character entity у KMP.
- Документація правил для очок навичок, модифікатори здібностей, расові та релігійні бонуси.

Робота потоку доставки/платформи:

- Налаштування проєкту: Kotlin Multiplatform, Ktor backend, Firebase.
- Створення базової архітектури: Clean Architecture(Domain, Data, Presentation).
- Налаштування CI/CD pipeline
- Інтеграція SQLDelight для зберігання даних на стороні користувача.
- Налаштування MongoDB для серверної сторони додатку
- Створення базового UI kit

Завдання спринту:

- Налаштувати KMP проєкт з спільними модулями для обох мобільних платформ.
- Створити базову структуру Клас даних персонажа з показниками здібностей
- Написати unit-тести для розрахунку модифікаторів здібностей
- Налаштувати Firebase Authentication
- Створити початковий екран
- Провести технічну перевірку навантаження на KMP для реального девайса.

Критерій завершення: Пройдено QG1 для базових правил. Технічна інфраструктура готова для початку розробки функціоналу.

Очікуваний результат: Команда має робочий KMP скелет для роботи з верифікованими базовими правилами D&D, затвердженим глосарієм та налаштованим CI/CD.

Спринт 1: "MVP: Створення та відображення персонажа"

Тривалість: 2 тижні (тижні 1-2)

Мета: Реалізувати базову функціональність створення персонажа з вибором раси, класу та розподілом характеристик. Користувач може створити простого персонажа та побачити його основні параметри.

Історії користувача для спринту:

- US003 (Високий): Створення персонажа з вибором раси, класу, розподілом характеристик.
- US001 (Низький): Вибір мови інтерфейсу в налаштуваннях.

Передумови (має бути готово з content/logic stream):

- Data model для всіх рас та класів з SRD 5.1 (Human, Elf, Dwarf, Halfling; Fighter, Wizard, Cleric, Rogue)
- Правила Point Buy та Standard Array верифіковані
- Логіка розрахунку расових бонусів пройшла QG1

Завдання спринту:

Серверна частина (Ktor + Firebase):

- Створити хмарну функцію для перевірки корисного навантаження при створенні персонажа.
- Впровадити кінцеву точку API: POST /characters/create.
- Зберегти документ персонажа в MongoDB з відповідною індексацією.
- Синхронізувати підсумок персонажа в SQLite для офлайн-доступу.

Фронтенд (KMP):

- Інтерфейс екрана вибору раси для персонажа.
- Інтерфейс екрана вибору класу (список з описом).
- Інтерфейс екрана розподілу характеристик:

1. Режим купівлі очок: перетягування або кнопки +/- , перевірка в режимі реального часу (загалом 27 очок, мінімум 8, максимум 15).
2. Стандартний режим масиву: випадаючий список для кожної характеристики зі значеннями [15,14,13,12,10,8].

- Інтерфейс екрана зведення персонажа з відображенням:

1. Ім'я, раса, клас, рівень (1).
2. Оцінки здібностей + модифікатори.
3. Бонус майстерності (+2 на рівні 1).
4. Очки здоров'я (класовий кубик здоров'я + модифікатор).

Спільна логіка KMP:

- Функція calculateAbilityModifier(score: Int): Int.
- Функція validatePointBuy(scores: Map<Ability, Int>): Boolean.
- Функція applyRacialBonuses(race: Race, baseScores: Map<Ability, Int>): Map<Ability, Int>.
- Функція calculateHitPoints(classType: Class, conModifier: Int, level: Int): Int.

Тестування:

- Модульні тести для всіх спільних функцій KMP.
- Інтеграційні тести: створення персонажа від початку до кінця.
- Ручне тестування QA: створення 10+ різних комбінацій рас/класів.
- Перевірка термінології: всі назви рас, класів і характеристик відповідають глосарію.

Очікуваний результат: Користувач може створити базового персонажа, обравши расу, клас та розподіливши характеристики одним з двох методів. Персонаж зберігається в системі та відображає коректні показники здібностей, модифікатори, здоров'я персонажа та бонус до майстерності згідно з правилами SRD 5.1.

Спринт 2: "Механіка: Клас броні та базові кидки"

Тривалість: 2 тижні (тижні 3-4)

Мета: Впровадити ключові ігрові механіки: розрахунок класу броні (AC) та систему кидків з модифікаторами. Це дозволить користувачам не тільки створювати персонажів, але й використовувати їх у базових ігрових ситуаціях.

Історії користувача для спринту:

- US005 (Високий): Автоматичний розрахунок AC

- US011 (Високий): Калькулятор кидків характеристик з модифікаторами
- US012 (Високий): Кидки з перевагою/недоліком

Передумови (має бути готово з content/logic stream):

- Перевірено правила розрахунку класу броні (AC) для всіх типів броні.
- Модель даних для предметів броні з типами (легка/середня/важка/щит).
- Документовані правила переваг/недоліків.
- Механіка майстерності з різними типами кидків.

Завдання спринту:

Content/Logic (паралельна робота для наступного спринту):

- Початок верифікації правил захисту для класів без броні приклад Barbarian/Monk (для US006)
- Документація граничних випадків для розрахунків класу броні(AC)

Серверна частина (Ktor):

- API endpoint: GET /items/armor (список доступної броні)
- API endpoint: PUT /characters/{id}/equipment (екіпірування предметів)
- Логіка перевірки: чи може персонаж носити цей тип броні.

Фронтенд (KMP) UI:

- Екран екіпірування: список доступної броні з фільтрами
- Оновлення листа персонажа: відображення поточного класу броні(AC):

AC: 16

├─ База: 10

├─ Броня (кольчуга): +6

├─ Модифікатор Спритності: +0

(обмежено середньою бронею)

└─ Щит: +0 (не екіпіровано)

- Інтерфейс кубика:
- Великі кнопки для кожної характеристики(Сила Вправність, Статура,Інтелект,Мудрість,Харизма)
- Перемикач для Перевага/Норма/Похибка
- Анімація кидка кубика

Приклад:

Результат з розкладанням на успіх або провал дії персонажа:

«d20(20-граний гральний кубик):

15(число на кубі) +3(модифікатор від характеристики) = 18»

Спільна логіка KMP:

- Функція calculateAC(character: Character): ACBreakdown

1. Базова логіка: 10 + dex modifier
2. З бронею: AC + модифікатор (Вправність)(з обмеженням для medium/heavy)
3. Щит: +2 якщо персонаж його використовує

- Функція rollAbilityCheck(ability: Ability, character: Character, mode: RollMode): RollResult

1. mode: Normal (1d20), Advantage (2d20, max), Disadvantage (2d20, min)
2. додавання модифікатора характеристики персонажа.
3. додавання професійного бонусу якщо є.

- Функція validateArmorProficiency(character: Character, armor: Armor): Boolean

Тестування:

1. Unit-тести для класу броні(AC) з різними комбінаціями
2. Граничний випадок: Модифікатор спритності +5, але середня броня обмежує до +2
3. Мануальне тестування: QA створює персонажів різних класів, екіпірує різну броню, перевіряє AC
4. Перевірка функції кидання кубика: 100+ кидків для кожного режиму, валідація розподілу (Перевага має давати статистично вищі результати)

Очікуваний результат: Користувач бачить автоматично розрахований AC свого персонажа, може екіпірувати різну броню та бачити, як це впливає на AC. Може робити кидки характеристик з коректними модифікаторами та використовувати механіку advantage/disadvantage.

Спринт 3: "Механіка: Трекінг хітів та ініціатива"

Тривалість: 2 тижні (тижні 5-6)

Мета: Додати критично важливі бойові механіки - управління очками здоров'я та калькулятор ініціативи. Це дозволить користувачам провести повноцінний бій.

Історії користувача для спринту:

- US009 (Високий): Трекінг очків здоров'я(додавання/віднімання).
- US013 (Високий): Калькулятор ініціативи.
- US010 (Низький): Масове застосування пошкодження (корисно для DM).

Передумови:

- Правила для рятівних кидків, втрата свідомості персонажа.

- Правила ініціативи з неочікуваністю задокументовані.

Завдання спринту:

Серверна частина (Ktor + Firebase):

- API endpoint: PATCH /characters/{id}/hp (зміна поточного здоров'я).
- API endpoint: POST /encounters/create (створення бою для трекінгу ініціативи).
- Real-time sync через Firebase Realtime Database для сесії боїв в реальному часі.

Фронтенд (KMP) UI:

- Показник здоров'я на листі персонажа:
- Великий показник очок здоров'я: "45 / 60 ОЗ"
- Кнопки +/- для швидкої зміни
- Input field для введення конкретного числа (пошкоджень/відновлення)
- Окремий показник для тимчасових очок здоров'я.
- Візуальний індикатор здоров'я: зелений (>50%), жовтий (20-50%), червоний (<20%)
- Попередження при досягненні 0 ОЗ: "Ваш персонаж непритомний! Розпочніть рятівні кидки."
- Трекер ініціативи в бою (новий екран):
- Список всіх учасників бою (персонажі + монстри)
- Кнопка "Roll initiative for all" - автоматичні кидки для всіх
- Сортування за результатом (найвищий зверху)
- Індикатор поточного ходу
- Кнопка "Next turn"

- Позначка "застигнуті зненацька" для персонажів (пропускають перший хід)

Спільна логіка KMP:

- Функція applyDamage(character: Character, damage: Int): Character

1. Спочатку віднімається з тимчасових ОЗ
2. Якщо тимчасові ОЗ закінчилось, віднімається з основних ОЗ
3. Не може бути < 0

- Функція applyHealing(character: Character, healing: Int): Character

1. Не може перевищити максимальні ОЗ

- Функція rollInitiative(dexModifier: Int, mode: RollMode): Int

1. 1d20 + dex modifier
2. Перевага/Недолік підтримка

- Функція sortCombatants(combatants: List<Combatant>): List<Combatant>
- Сортування за ініціативою
- Якщо значення однакові: вищий буде той персонаж у якого більший показник вправності

Тестування:

Unit-тести: крайні випадки пошкодження/лікування (тимчасове ОЗ, надмірне пошкодження, обмеження максимального ОЗ)

Інтеграційний тест: повний бойовий процес (кидок ініціативи → кілька раундів з пошкодженням → закінчення бою)

Ручне тестування: QA грає тестову бойову сутичку з 4 персонажами проти 6 монстрів

Тест продуктивності: трекер ініціативи з 20+ бійцями (масовий бій)

Очікуваний результат: Користувач може повноцінно проводити бойові сутички: відстежувати ОЗ всіх учасників, автоматично розраховувати порядок ініціативи, перемикатися між ходами. DM може швидко застосувати масове пошкодження до кількох цілей одночасно.

Спринт 4: "Контент: Заклинання та комірки заклинань"

Тривалість: 2 тижні (тижні 7-8)

Мета: Впровадити повну систему заклинань: перегляд, фільтрація, трекінг комірок заклинань. Це критично важливо для гравців-заклинарів.

Історії користувача для спринту:

- US007 (Високий): Список заклинань з фільтрацією
- US008 (Високий): Розрахунок комірок заклинань для персонажів з декількома класами(критична механічна складність)

Передумови (має бути готово):

- Повна база даних заклинань з SRD 5.1 у MongoDB (300+ заклинань)
- Глосарій з українськими назвами всіх заклинань
- Правила мультикласового чарування повністю задокументовані та верифіковані через QG1
- Модель даних для комірок заклинань для всіх можливих мульти классових комбінацій.

Важливо**:** Цей спринт не може розпочатись без проходження QG1 для мульти классових комбінацій. Якщо верифікація затримується, команда перемикається на буферні завдання (оптимізація, додаткові тести) або бере наступний спринт.

Завдання спринту:

Серверна частина (Ktor):

- API endpoint: GET /spells (з пагінацією)
- API endpoint: GET /characters/{id}/available-spells (тільки заклинання для класів персонажа)
- API endpoint: POST /characters/{id}/prepared-spells (підготовка заклинань)
- Індексація MongoDB для швидкого пошуку за назвою, рівнем, школою

Фронтенд (KMP) UI:

- Книга заклинань екран:

1. Сіткова структура інтерфейсу
2. Фільтри: рівень (0-9), школа магії.
3. Пошук за назвою (UA/EN)
4. Клік на заклинання → детальна картка:

Fireball / Вогняна куля

3-й-рівень школа:Руйнування

Час касту: 1 дія

Відстань: 150 feet

Компоненти: Вербальний, Соматичний, Матеріальний (сірка)

Тривалість: Миттєва

[Повний опис українською]

- Трекер комірок заклинань на листі персонажа:

Візуальне відображення slots для кожного рівня (1-9):

Рівень 1: ●●●○○ (3/5 використано)

Рівень 2: ●○○○ (1/4 використано)

Рівень 3: ○○○ (0/3 використано)

- Кнопка "Використати комірку" / "Відновити" для кожного рівня

Спільна логіка КМР

- Функція calculateSpellSlots(character: Character): Map<Int, Int>
  - Для персонажів з одним класом: прямий запит з таблиці правил
  - Для мульти класових персонажів:

1. Розрахувати рівень для кожного класу:
2. Повністю магічний клас (Чарівник, Жрець, і тд): рівень класу
3. Наполовину магічний клас (Паладин, Мисливець): рівень класу / 2 (округлення в меншу сторону)
4. Під класи (Містичний лицар,Магічний плут): рівень класу /
5. Підсумковий рівень заклинаря = сума всіх рівнів заклинаря
6. Пошук слотів за підсумковим рівнем у таблиці мультикласу
7. Крайні випадки:Мультиклас відьмака: комірки магії угоди окремо від звичайних комірок заклинань
8. Паладин 2-го рівня та Маг 3-го рівня = напівзаклинар 1 + повний заклинар 3 = разом 4 → комірки як у заклинаря 4-го рівня.

- Функція getAvailableSpells(character: Character): List<Spell>

1. Повертає заклинання для ВСІХ класів персонажа
2. Маг: весь список заклинань (але тільки підготовлені доступні для використання)
3. Клерик: весь список заклинань (підготовка щодня)
4. Чародій: тільки вивчені заклинання

Тестування (РОЗШИРЕНЕ через складність):

- Модульні тести для 20+ комбінацій мультикласів:
- Маг 5 / Клерик 3 → комірки 8-го рівня
- Паладин 6 / Відьмак 2 → комірки 3-го рівня + окремо 2 комірки відьмака
- Воїн (Містичний лицар) 9 / Маг 3 → 3+3=6 → комірки 3-го рівня
- Ручне тестування від забезпечення якості: створення всіх можливих мультикласів заклинарів
- Тестування крайніх випадків: відновлення комірок відьмака (короткий відпочинок) проти звичайних комірок (тривалий відпочинок)
- Тест продуктивності: пошук у базі 300+ заклинань має бути < 500 мілісекунд

Очікуваний результат: Гравець-заклинар може переглядати всі доступні йому заклинання, фільтрувати за параметрами, бачити детальні описи українською. Система коректно розраховує комірки заклинань навіть для складних мульти класових комбінацій. Гравець може відстежувати використані комірки та відновлювати їх після відпочинку.

Спринт 5: "Контент: Бестіарій та імпорт персонажів"

Тривалість: 2 тижні (тижні 9-10)

Мета: Додати бестіарій для DM та можливість імпорту персонажів для досвідчених користувачів.

Історії користувача для спринту:

- US015 (Середній): Перегляд бестіарію з пошуком та фільтрацією
- US004 (Середній): Імпорт персонажа з JSON
- US002 (Низький): Зміна мови в налаштуваннях

Передумови:

- База даних монстрів з SRD 5.1 у MongoDB (300+ істот)
- Українські назви монстрів у глосарії
- Специфікація для JSON формату (сумісний з D&D Beyond)

Завдання спринту:

Серверна частина (Ktor):

- API endpoint: GET /monsters (з фільтрами)
- API endpoint: POST /characters/import (валідація та імпорт JSON)
- Розширення MongoDB indexes для швидкого пошуку монстрів

Фронтенд (KMP) UI:

- Бестіарії екран:
  - Список: Назва істоти
  - Фільтри: Рівень складності (PC)(0-30), тип (Вовк,Дракон і тд.), розмір.
  - Пошук за назвою
  - Клік на істоту→ детальна картка:

Ancient Red Dragon / Стародавній червоний дракон
PC: 24
AC: 22 (натуральна броня)
ОЗ: 546 (28d20 + 252)
Швидкість: 40 ft.
Сила 30 (+10) Вправність 10 (+0) Статура 29 (+9)
Інтелект 18 (+4) Мудрість 15 (+2) Харизма 23 (+6)

[Рятувальні кидки, навички, імунітет, чуття, мови],

[Спеціальні здібності], [Дії], [Легендарні дії]

- Кнопка "Додати до бою" - додає монстра до активного бою
- Екран імпорту персонажа:
  - File picker або вставити JSON
  - Перегляд імпортованих даних
  - Валідація з детальними помилками
  - Кнопка "Імпортувати"

Спільна логіка KMP

- Функція parseCharacterJSON(json: String): Result<Character, ValidationError>
  - Парсинг D&D Beyond JSON
  - Валідація всіх полів для створення персонажу
  - Конвертація в внутрішню модель даних персонажа
- Функція validateMonsterStatBlock(monster: Monster): Boolean
  - відповідно таблиці з книги правил

Тестування:

- Тест імпортування: 10+ реальних JSON з D&D Beyond
- Граничний випадок: неповні JSON, некоректні значення, застарілий формат
- Тест Бестіарію : пошук та фільтрація монстрів.
- Мануальне тестування: QA імпортує кількох персонажів, додає монстрів до бою

Очікуваний результат: DM має повний доступ до бестіарію SRD 5.1 з можливістю швидкого пошуку та додавання монстрів до бою. Досвідчені користувачі можуть імпортувати своїх персонажів з D&D Beyond замість ручного створення.

3.2. Управління ризиками проєкту

Управління ризиками є невід'ємною частиною управління IT-проєктами, особливо при розробці продуктів із складною предметною областю та залежністю від зовнішніх факторів. Для проєкту D&D мобільного додатку, який залежить від точності реалізації ліцензованих правил, узгодженості термінології з спільнотою та технічних обмежень мобільних платформ, був розроблений детальний план управління ризиками, що охоплює вибір методології, ідентифікацію, аналіз та розробку заходів реагування, забезпечуючи успішне досягнення цілей проєкту.

3.2.1. Ідентифікація та якісна оцінка ризиків

Модель класифікації ризиків:

Для проєкту D&D мобільного додатку обрано систему класифікації за типом джерела: технологічні, технічні, контентні, організаційні, зовнішні та ризики спільноти. Ця система найкраще підходить для IT-проєктів зі складною предметною областю, оскільки дозволяє чітко структурувати ризики за їх природою та призначити відповідальних за моніторинг(Таблиця 3.2.1).

Шкала якісного оцінювання:

Як шкалу оцінювання обрано розширену якісну шкалу з трьома вимірами:

Вплив (Сила впливу):

Н (Низький): Незначний вплив на термін (<1 тижня затримки) або бюджет (<5%)

С (Середній): Помірний вплив (1-2 тижні затримки або 5-15% перевитрати)

В (Високий): Значний вплив (>2 тижнів затримки або >15% перевитрати)

Керованість:

НС (Низька керованість): Ризик майже неможливо контролювати (зовнішні фактори)

СН, СС, СВ (Середня керованість): Ризик частково контролюється командою

ВС, ВВ (Висока керованість): Ризик повністю під контролем команди

Комбінована оцінка: Поєднання впливу та керованості дає 9 можливих категорій (НН, НС, НВ, СН, СС, СВ, ВН, ВС, ВВ), де перша літера вплив, друга керованість.

Розширена шкала дозволяє забезпечити достатню деталізацію для середнього за складністю проєкту, допомагаючи точніше диференціювати ризики за рівнем впливу та можливістю управління.

Цей план інтегрує етапи створення «Глосарію-першоджерела» та жорсткі контрольні точки якості (Quality Gates) безпосередньо в ітераційні цикли розробки. Нижче, у Таблиці 3.2.1, наведено графік виконання робіт, що демонструє розподіл ресурсів та часові межі для кожного етапу реалізації модуля генерації персонажа.

Таблиця 3.2.1 Ідентифікація та оцінка ризиків

№	Тип ризику	Ризикова подія	Сила впливу	Керованість
1	2	3	4	5
1	Технологічні	Обмеження моделі пам'яті KMP при роботі з великими списками заклинань (300+)	ВС	СВ
2	Технологічні	Несумісність версій KMP з новими версіями Android/iOS SDK	ВН	СС
3	Технологічні	Проблеми з продуктивністю на Android-пристроях середнього класу (<4 ГБ оперативної пам'яті)	ВС	СВ
4	Технологічні	Проблеми з офлайн-синхронізацією між SQLite і MongoDB	СВ	ВС
5	Технічні	Неправильна реалізація розрахунку слотів заклинань для декількох класів	ВВ	ВС
6	Технічні	Механічні помилки в крайніх випадках	ВС	ВС

1	2	3	4	5
7	Технічні	Помилки в розрахунку AC для різних типів броні	СВ	ВВ
8	Технічні	Трекер Ініціативи не може обробляти 20+ бійців	СС	ВС
9	Контентні	Затримка у створенні та затвердженні глосарію термінології	ВВ	СН
10	Контентні	Неузгодженість термінів у різних частинах додатку	ВС	СВ
11	Контентні	Помилки в перекладі механічних ефектів заклинань	ВС	СВ
12	Контентні	Низька якість локалізації призводить до негативних відгуків	ВС	СН
13	Організаційні	Втрата ключового члена команди (технічного керівника або системного архітектора)	ВВ	СН
14	Організаційні	Недостатня експертиза команди в правилах D&D призводить до механічних помилок	ВС	СВ
15	Організаційні	Конфлікти між контентом/логікою та потоками доставки через затримки у верифікації	СВ	ВС
16	Організаційні	Виснаження команди через складність реалізації крайніх випадків	СС	ВС
17	Зовнішні	Зміна умов Open Game License (OGL) від Wizards of the Coast	ВВ	НС
18	Зовнішні	Затримка або відмова у затвердженні контенту нових правил від WotC	ВС	НС
19	Зовнішні	Зміна політики App Store/Google Play щодо ігрового контенту	СВ	НС
20	Ризики спільноти	Негативна реакція української спільноти D&D на вибір термінів у глосарії	ВС	СН
21	Ризики спільноти	Низька залученість бета-тестувальників до перевірки якості	СС	СВ
22	Ризики спільноти	Недостатнє органічне зростання через поширення інформації з вуст в уста	СН	СС

3.2.2. Кількісний аналіз та ранжування ризиків

Для кількісної оцінки ризиків (деталізовано у таблиці 3.2.3) використовувалася шкала від 1 до 9 для оцінки:

1. Вплив на час: скільки тижнів затримки може спричинити ризик (1 = <0.5 тижня, 9 = >4 тижні)
2. Вплив на бюджет: фінансові втрати у відсотках (1 = <5%, 9 = >50%)
3. Ймовірність: шанс реалізації ризику (1 = <10%, 9 = >80%)
4. Частота: скільки разів ризик може проявитись за проєкт (1 = один раз, 9 = постійно)

Також для кожного з видів ризику була проведена кількісна та якісна оцінка позначені КО та ЯО відповідно.

Комплексний показник важливості ризику розраховувався за формулою:

Важливість = (Вплив на час + Вплив на бюджет) × Ймовірність × Частота

Ця формула дозволяє виділити ризики, які не тільки мають високий вплив, але й високу ймовірність реалізації.

Таблиця 3.2.3 Кількісний аналіз ризиків

№	Ризикова подія	Затримки у часі ЯО	Затримки у часі КО	Фінансові втрати - ЯО	Фінансові втрати - КО	Ймовірність ЯО	Ймовірність КО	Частота ЯО	Частота КО	Важливість ризику
1	2	3	4	5	6	7	8	9	10	11
1	Обмеження моделі пам'яті KMP при роботі з великими списками заклинань (300+)	СВ	6	ВС	8	СВ	6	СС	5	420

1	2	3	4	5	6	7	8	9	10	11
2	Несумісність версій KMP з новими версіями Android/iOS SDK	ВС	8	СВ	6	НС	2	НВ	3	84
3	Проблеми з продуктивністю на Android-пристроях середнього класу (<4 ГБ оперативної пам'яті)	ВС	8	СС	5	СВ	6	СВ	6	468
4	Проблеми з офлайн-синхронізацією між SQLite і MongoDB	СВ	6	СС	5	СС	5	НВ	3	165
5	Неправильна реалізація розрахунку слотів заклинань для декількох класів	ВВ	9	ВВ	9	ВС	8	НС	2	288
6	Механічні помилки в крайніх випадках (конфлікти безброньової оборони)	ВС	8	СС	5	ВС	8	СВ	6	624
7	Помилки в розрахунку АС для різних типів броні	СВ	6	СН	4	СС	5	СН	4	200
8	Ініціатива трекер не може обробляти 20+ бійців	СС	5	НВ	3	НВ	3	НС	2	48
9	Затримка у створенні та затвердженні глосарію термінології	ВВ	9	ВВ	9	ВС	8	НН	1	144
10	Неузгодженість термінів у різних частинах додатку	ВС	8	СВ	6	СВ	6	СС	5	420
11	Помилки в перекладі механічних ефектів заклинань	ВС	8	СС	5	СВ	6	СВ	6	468
12	Низька якість локалізації призводить до негативних відгуків	ВС	8	ВС	8	ВС	8	НН	1	128
13	Втрата ключового члена команди (технічного керівника або системного архітектора)	ВВ	9	ВВ	9	СН	4	НС	2	144
14	Недостатній досвід команди з D&D призводить до механічних помилок	ВС	8	СВ	6	СВ	6	СС	5	420

1	2	3	4	5	6	7	8	9	10	11
15	Конфлікти між контентом/логікою та потоками доставки через затримки з перевіркою	СВ	6	СН	4	СС	5	СН	4	200
16	Виснаження команди через складність реалізації крайніх випадків	СС	5	СН	4	СН	4	НВ	3	108
17	Зміна умов Open Game License (OGL) від Wizards of the Coast	ВВ	9	ВВ	9	НС	2	НН	1	36
18	Затримка або відмова затвердити контент від WotC	ВС	8	СС	5	НВ	3	НС	2	78
19	Зміна політики App Store/Google Play щодо ігрового контенту	СВ	6	СС	5	НС	2	НН	1	22
20	Негативна реакція української спільноти D&D на вибір термінів у глосарії	ВС	8	ВС	8	СС	5	НС	2	160
21	Низька залученість бета-тестувальників до перевірки якості	СС	5	НВ	3	СН	4	НС	2	64
22	Недостатнє органічне зростання через поширення інформації з вуст в уста	СН	4	НС	2	СН	4	СС	5	120

3.3. Аналіз результатів гібридного методологічного фреймворку

Експериментальна фаза, сфокусована на розробці модуля генерації персонажа та його локалізації, надала емпіричні дані, необхідні для оцінки ефективності запропонованого Гібридного методологічного фреймворку (ГМФ). Аналіз результатів проводився за двома основними напрямками, які відповідають критичним ризикам проєкту: механічній точності (LAT) та зовнішній залежності (EDD).

3.3.1. Оцінка Порогу Точності Логіки (LAT) та роль Контрольних Точок

Аналіз результатів функціонування Контентно-Логічного Потоку, що в межах нашого ГМФ був орієнтований на структуровану Водоспадну модель, продемонстрував значне — і, чесно кажучи, критично важливе — зниження кількості дефектів у ядрі ігрової механіки. Встановлення жорсткого Порогу Точності Логіки (LAT) як обов'язкового критерію виходу (Definition of Done) виявилося не просто формальністю, а бар'єром, що відсікає технічний примітивізм.

Зокрема, після впровадження обов'язкового затвердження Контрольних Точок Якості (Quality Gates) спеціалізованим Лідером з Ігрової Механіки (Lore-Proxy), частота виявлення помилок у розрахунку модифікаторів здібностей та складних мультикласових взаємодій у фінальних спринтах знизилася до фактичного нуля. Автоматизовані юніт-тести у Спринті 4 не зафіксували жодної механічної невідповідності канону SRD 5e. Це значне досягнення, яке підтверджує авторську позицію: ігрову логіку не можна "ітерувати" наосліп — вона має бути валідована до моменту потрапляння в інтерфейс.Втім, варто визнати, що таке безкомпромісне підвищення якості мало свою ціну, яку часто ігнорують у теоретичних моделях. Середній час, необхідний для проходження функціоналу через QG, виявився на 35% довшим від початково запланованого. Це свідчить про реальну, подекуди недооцінену складність правил D&D і неминучий управлінський компроміс між швидкістю розробки та механічною вірогідністю. Жоден стандартний інструмент управління, як-от звичайна діаграма Берндаун, не здатний адекватно відобразити цей нелінійний ефект, оскільки він не враховує інтелектуальну вартість верифікації "лору".Ба більше, досвід реалізації модуля генерації персонажа показав, що затримки на етапі Контрольних Точок Якості мають тенденцію до накопичення, якщо не використовувати стратегію Буферних Черг. Проте, автор стверджує, що краще отримати затримку на етапі планування, ніж виправляти фундаментальні помилки в розрахунках після релізу, що для спільноти D&D є фатальним. Впровадження LAT фактично перетворило "суб'єктивну думку розробника" на "об'єктивну відповідність системі", що дозволило команді вийти на рівень прогнозованої якості, попри збільшення часових витрат на старті кожної ітерації. Висока вартість QG у даному контексті є не недоліком, а інвестицією в репутаційну стабільність продукту

3.3.2. Управління Дрегом Зовнішньої Залежності (EDD)

Ключовим показником успіху ГМФ стало управління Дрегом Зовнішньої Залежності (EDD). Протягом Спринту 2, процес узгодження ліцензійних візуальних активів затримався на 7 робочих днів. У традиційній Scrum-моделі це призвело б до простою (блокування) значної частини команди Розробки/Платформи.Проте, завдяки стратегії Буферної Черги, команда Розробки/Платформи змогла оперативно перемикнутися на заздалегідь підготовлені технічні завдання, що не залежать від контенту (наприклад, оптимізація запитів Ktor та розширення покриття юніт-тестами).Кількісний аналіз показав, що, попри 7-денну затримку в Контентно-Логічному Потоці, швидкість Потоку Розробки/Платформи знизилася лише на 12% від запланованої, тоді як очікуване падіння, згідно з лінійною моделлю, становило б близько 30%. Це підтверджує, що ізоляція потоків та використання буферних завдань є дієвою стратегією для підтримання стабільної продуктивності команди в умовах зовнішньої волатильності. Управління буфером дозволило знизити психологічне навантаження на команду, оскільки простої були замінені на конструктивну, хоча й неосновну, роботу.

3.3.3. Валідація "Глосарій Орієнтованої" Локалізації

Успіх локалізації ігрового контенту українською мовою був прямим наслідком впровадження "Глосарій Орієнтованої" Методології. Завдяки, тому що Лідер з Локалізації затвердив основний термінологічний глосарій до початку кодування інтерфейсу, було повністю уникнемо однієї з найбільш поширених проблем: термінологічної фрагментації. Хоча процес створення та затвердження глосарія додав на початковому етапі (Спринт 1) затримку у 4 робочих дні, це скоротило загальний час на виправлення локалізаційних дефектів на фінальній стадії тестування на 90% порівняно з оцінками, отриманими від консультантів у сфері традиційних проєктів. Цей результат свідчить про те, що раннє інвестування в лінгвістичну архітектуру є виправданим.

ВИСНОВКИ

Порівняльний аналіз традиційних та гнучких методологій управління проєктами в контексті розробки мобільних додатків для настільних рольових ігор показав, що жоден з підходів у чистому вигляді не забезпечує достатнього балансу між гнучкістю та контролем. Було встановлено, що хоча Waterfall забезпечує необхідну жорсткість для управління статичними, незмінними правилами вмісту D&D 5e SRD, йому бракує адаптивності, необхідної для сучасної ітерації UI/UX; навпаки, стандартний Scrum часто не враховує високу «вартість якості», пов'язану з механічними логічними помилками, що призводить до технічного боргу, коли складні правила розглядаються як прості історії користувачів. Дослідження підтверджує, що успішне управління в цій ніші вимагає відходу від бінарних методологічних виборів на користь моделі, яка визнає окремі життєві цикли розробки програмного забезпечення та перевірки контенту.

Щодо особливостей локалізації складного ігрового контенту, дослідження визначило «термінологічну фрагментацію» як критичний ризик, що виходить за межі простого перекладу, особливо при адаптації ідіоматичних англійських механізмів до українських морфологічних структур.Була розроблена методологія «Glossary-First», яка вимагає створення та затвердження стандартизованого глосарія, що містить понад 500 термінів, як обов'язкову передумову для етапу кодування. Такий підхід перетворює локалізацію з постпродакшн-завдання на фундаментальну архітектурну вимогу, забезпечуючи семантичну цілісність функціональних термінів, таких як «Saving Throw» або «Spell Slot», у всій базі даних та інтерфейсі користувача, тим самим зменшуючи ризик відторгнення спільнотою через мовні невідповідності.

Для вирішення організаційних завдань була розроблена адаптована гібридна система управління (ГМФ), яка структурує процеси проєкту на два синхронізованих, але різних напрямки: напрямок контенту/логіки, що працює за послідовними принципами, та напрямок доставки/платформи, що використовує ітеративні цикли Scrum. Ця система вводить конкретні ролі, такі як керівник ігрової механіки, та включає «механічну складність» як атрибут ваги для управління вимогами, що дозволяє міжфункціональній команді краще оцінювати зусилля, необхідні для нелінійних правил гри. Крім того, комунікаційна стратегія була скоригована з метою включення «буферного пулу»

технічних завдань, що дозволяє команді підтримувати швидкість роботи під час неминучих затримок, спричинених зовнішніми ліцензійними затвердженнями або узгодженням глосарія, ефективно відокремлюючи ритм розробки від зовнішньої нестабільності.

Обґрунтування технологічного стека продемонструвало, що вибір Kotlin Multiplatform, Ktor та SQLDelight є стратегічним інструментом зменшення ризиків, а не просто інженерною перевагою. Застосовуючи принцип «єдиного джерела істини» для бізнес-логіки, ця архітектура усуває ризик «механічного зсуву», коли незалежні кодові бази iOS та Android можуть по-різному обчислювати однакові стани гри. Аналіз підтвердив, що хоча цей стек вимагає більших початкових інвестицій у досвід команди, він забезпечує механічну точність і між платформну узгодженість, що є критично важливим для довіри користувачів у середовищі з великою кількістю правил, а MongoDB забезпечує необхідну гнучкість схеми для управління неструктурованим контентом.

Нарешті, була розроблена надійна система Quality Gates (QG) для забезпечення механічної точності додатка, що встановлює поріг логічної точності (LAT), який надає пріоритет точності правил над швидкістю випуску. Ці контрольні точки, розташовані на виході з потоку контенту, в кінці спринту та перед випуском, виходять за межі стандартного функціонального тестування і включають «сесії диму», під час яких програмне забезпечення перевіряється за допомогою реальних ігрових симуляцій. Цей механізм гарантує, що критичні та складні функції, такі як логіка мульти класування, перевіряються експертами з відповідної галузі на відповідність D&D 5e SRD перед інтеграцією, тим самим систематично зменшуючи ризик випуску технічно стабільного, але механічно недосконалого продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dungeons & Dragons System Reference Document 5.1. Wizards of the Coast, 2016. URL: https://dnd.wizards.com/resources/systems-reference-document

2. Open Game License Version 1.0a. Wizards of the Coast, 2000. URL: https://opengamingfoundation.org/ogl.html

3. Player's Handbook: Dungeons & Dragons Core Rulebook. Wizards of the Coast, 2014. 320 p.

4. Dungeon Master's Guide: Dungeons & Dragons Core Rulebook. Wizards of the Coast, 2014. 320 p.

5. Швабер К., Сазерленд Дж. Посібник зі Скраму (Повний навчальний посібник зі Скраму: правила гри) [Електронний ресурс]. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Ukrainian.pdf

6. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition. USA: Project Management Institute, 2021. 370 p.

7. Agile Practice Guide: Paperback. USA: Project Management Institute, 2017. 210 p.

8. Кон Майкл. Оцінювання і планування в Agile / пер. з англ. Г. Якубовська. Харків: Вид-во «Ранок»: Фабула, 2019. 336 с.

9. Microsoft Solutions Framework (MSF) for Agile Software Development. Microsoft Corporation, 2008. URL: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2008/

10. ISO 21500:2012 Guidance on project management. International Organization for Standardization, 2012.

11. ISO 31000:2018 Risk management - Guidelines. International Organization for Standardization, 2018.

12. Kotlin Multiplatform Mobile Documentation. JetBrains, 2024. URL: https://kotlinlang.org/docs/multiplatform-mobile-getting-started.html

13. Ktor Framework Documentation. JetBrains, 2024. URL: https://ktor.io/docs/welcome.html

14. SQLDelight Documentation. Cash App, 2024. URL: https://cashapp.github.io/sqldelight/

15. Firebase Documentation. Google, 2024. URL: https://firebase.google.com/docs

16. MongoDB Documentation. MongoDB Inc., 2024. URL: https://docs.mongodb.com/

17. React Native Documentation. Meta Platforms, 2024. URL: https://reactnative.dev/docs/getting-started

18. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Martin R. C. Prentice Hall, 2017. 432 p.

19. Design Patterns: Elements of Reusable Object-Oriented Software. Gamma E., Helm R., Johnson R., Vlissides J. Addison-Wesley Professional, 1994. 395 p.

20. Game Programming Patterns. Nystrom R. Genever Benning, 2014. 354 p.

21. The Art of Game Design: A Book of Lenses. Schell J. CRC Press, 2019. 600 p.

22. Локалізація програмного забезпечення: навчальний посібник. Муха Л. В., Завгородній В. В. Київ: КПІ ім. Ігоря Сікорського, 2019. 180 с.

23. Software Localization: A Practical Guide. Esselink B. John Benjamins Publishing Company, 2000. 488 p.

24. Internationalization and Localization Using Microsoft .NET. Symons N., Howell D. Apress, 2002. 352 p.

25. Translation and Localization: A Guide for Technical and Professional Communicators. Jiménez-Crespo M. Routledge, 2020. 286 p.

26. Азбука управління проектами. Планування: навч. посіб. Єгорченков О. В., Єгорченкова Н. Ю., Катаєва Є. Ю. Київ: КНУ ім. Т. Шевченка, 2017. 117 с.

27. Методи управління ризиками в ІТ проєктах: методичні вказівки. Тімінський О. Г., Коломієць А. С. Київ: КНУ імені Тараса Шевченка, 2021. 40 с.

28. Managing Risk in Software Development Projects: A Guide to Successful Software Development. O'Connell F. Addison-Wesley Professional, 2003. 304 p.

29. Software Risk Management: Principles and Practices. Boehm B. W., IEEE Software. 1991. Vol. 8, № 1. P. 32-41.

30. Agile Software Development with Scrum. Schwaber K., Beedle M. Prentice Hall, 2001. 158 p.

31. Scrum: The Art of Doing Twice the Work in Half the Time. Sutherland J., Sutherland J. J. Crown Business, 2014. 256 p.

32. Kanban: Successful Evolutionary Change for Your Technology Business. Anderson D. J. Blue Hole Press, 2010. 261 p.

33. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Humble J., Farley D. Addison-Wesley Professional, 2010. 512 p.

34. Test-Driven Development: By Example. Beck K. Addison-Wesley Professional, 2002. 240 p.

35. Domain-Driven Design: Tackling Complexity in the Heart of Software. Evans E. Addison-Wesley Professional, 2003. 560 p.

36. User Story Mapping: Discover the Whole Story, Build the Right Product. Patton J. O'Reilly Media, 2014. 324 p.

37. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Ries E. Crown Business, 2011. 336 p.

38. Inspired: How to Create Tech Products Customers Love. Cagan M. Wiley, 2017. 368 p.

39. Measuring and Managing Performance in Organizations. Austin R. D. Dorset House Publishing, 1996. 216 p.

40. Software Metrics: A Rigorous and Practical Approach. Fenton N. E., Bieman J. CRC Press, 2014. 638 p.

41. D&D Beyond: Official Digital Toolset. Fandom Inc., 2024. URL: https://www.dndbeyond.com/

42. Roll20: Virtual Tabletop. The Orr Group, 2024. URL: https://roll20.net/

43. Fantasy Grounds: Virtual Tabletop. SmiteWorks USA, 2024. URL: https://www.fantasygrounds.com/

44. Fight Club 5th Edition: D&D Character Manager. Lion's Den Software, 2024. URL: https://lionsden.software/

45. D&D Character Sheet App Review: Digital Tools for Fifth Edition. EN World, 2023. URL: https://www.enworld.org/

46. Game Localization: Translating for the Global Digital Entertainment Industry. Bernal-Merino M. Routledge, 2014. 310 p.

47. Building Mobile Apps at Scale: 39 Engineering Challenges. Orosz G. Pragmatic Bookshelf, 2021. 450 p.

48. Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps. Neil T. O'Reilly Media, 2014. 396 p.

49. App Store Review Guidelines. Apple Inc., 2024. URL: https://developer.apple.com/app-store/review/guidelines/

50. Google Play Developer Policy Center. Google LLC, 2024. URL: https://play.google.com/about/developer-content-policy/

ДОДАТОК А

```kotlin
@JsonClass(generateAdapter = true)
data class APIReference(
    val index: String,
    val name: String,
    val url: String
)

@JsonClass(generateAdapter = true)
data class AbilityBonus(
    @Json(name = "ability_score") val abilityScore: APIReference,
    val bonus: Int
)

@JsonClass(generateAdapter = true)
data class AbilityBonusOptionSet(
    val choose: Int,
    val type: String,
    val from: OptionSetFrom
)

@JsonClass(generateAdapter = true)
data class OptionSetFrom(
    @Json(name = "option_set_type") val optionSetType: String,
    val options: List<Option>
)

@JsonClass(generateAdapter = true)
data class Option(
    @Json(name = "option_type") val optionType: String,
    val item: APIReference? = null,
    @Json(name = "ability_score") val abilityScore: APIReference? = null,
    val bonus: Int? = null
)


@JsonClass(generateAdapter = true)
data class Race(
    val index: String,
    val name: String,
    val speed: Int,
    @Json(name = "ability_bonuses") val abilityBonuses: List<AbilityBonus>,
    @Json(name = "ability_bonus_options") val abilityBonusOptions: AbilityBonusOptionSet?,
    val alignment: String,
    val age: String,
    val size: String,
    @Json(name = "size_description") val sizeDescription: String,
    @Json(name = "starting_proficiencies") val startingProficiencies: List<APIReference>,
    @Json(name = "starting_proficiency_options") val startingProficiencyOptions: AbilityBonusOptionSet?,
    val languages: List<APIReference>,
    @Json(name = "language_options") val languageOptions: AbilityBonusOptionSet?,
    @Json(name = "language_desc") val languageDesc: String,
    val traits: List<APIReference>,
    val subraces: List<APIReference>,
    val url: String
) seed, [])
}
```

Рис.А Модель вибору раси персонажа на KMP та налаштування Ktor.

ДОДАТОК Б

```
@Serializable
data class Classes(
    override val index: String,
    override val name: String,
    override val url: String,
    val hitDie: Int,
    val classLevels: String,
    val multiClassing: MultiClassing,
    val spellcasting: Spellcasting? = null,
    val spells: String? = null,
    val startingEquipment: List<StartingEquipment>,
    val startingEquipmentOptions: List<Choice>,
    val proficiencyChoices: List<Choice>,
    val proficiencies: List<APIReference>,
    val savingThrows: List<APIReference>,
    val subclasses: List<APIReference>,
) : IRef {
    companion object : Queryable
}

@Serializable
data class Spellcasting(
    val level: Int,
    val info: List<Info>,
    val spellcastingAbility: APIReference
)

@Serializable
data class Info(
    val name: String,
    val desc: List<String>
)



@Serializable
data class MultiClassingPreReq(
    val abilityScore: APIReference,
    val minimumScore: Int
)])
}
```

Рис.Б Модель вибору класу персонажа.