КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Управління проєктами
(назва випускової кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР
(бакалавр, магістр)

на тему:

«Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес»

Васюк Артем Вікторович
Карпов Михайло Євгенович
(прізвище, ім’я та по батькові здобувача повністю)

Київ 2025р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Управління проєктами
(назва випускової кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри
Веренич О.В.
„___” _____2025року

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР
(бакалавр, магістр)

«Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес»
(назва)

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувачі Васюк Артем Вікторович,
Карпов Михайло Євгенович
(прізвище, ім’я та по батькові повністю)
Інформаційні системи та технології
(спеціальність)
Управління проєктами
(освітня програма)
Група ІСТ-УП-21
Керівник Войтенко О.С.
(прізвище та ініціали)
к.т.н., доцент
(вчене звання, науковий ступінь)
Рецензент_____
(прізвище та ініціали)
Ідентичність підтверджую

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій
Випускова кафедра: Управління проєктами
Ступінь вищої освіти: Бакалавр
Спеціальність: 126 Інформаційні системи та технології
Освітня програма: Управління проєктами

ЗАТВЕРДЖУЮ

Завідувач кафедри
Веренич О.В.
„___” _____ 2025року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Карпов Михайло Євгенович
Васюк Артем Вікторович
(прізвище, ім’я та по батькові здобувача)

1. Тема роботи Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес

затверджена наказом ректора КНУБА № 293/23/25 від «21» 01 2025 року

2. Керівник роботи Войтенко Олександр Степанович, кандидат технічних наук, доцент
(прізвище, ім’я та по батькові, науковий ступінь, вчене звання)

3. Термін подання здобувачем роботи до захисту 2025 р.

4. Зміст пояснювальної записки за розділами:

Р. 1. Аналіз предметної області дослідження
Р. 2 Планування проєкту розробки програмного забезпечення
Р. 3. Інструментальні засоби розробки програмного продукту «SMART PM CITY»

5. Графічний матеріал за розділами:

Р. 1. Рисунки моделей життєвих циклів розробки ПЗ
Р. 2 Рисунки аналізу середовища проєкту, показники проєкту в MS Project;
Таблиці аналізу середовища проєкту та фінансові показників проєкту
Р. 3. Рисунки розробки демонстраційної версії додатку «SMART PM CITY»;
Діаграми класів та дій;
Таблиця порівняння ігрових рушіїв;

6. Консультанти розділів кваліфікаційної випускної роботи:

Розділ	Прізвище, ініціали та посада	Перевірив	
		дата	підпис
Розділ 1.			
Розділ 2.			
Розділ 3.			

7. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1.	26.04.2025
Розділ 2.	09.05.2025
Розділ 3.	16.05.2025
Остаточне оформлення роботи	01.06.2025
Направлення роботи для перевірки на плагіат	04.06.2025
Попередній захист роботи	12.06.2025
Направлення роботи на рецензування	12.06.2025

8. Дата видачі завдання 21.02.2025 р.

Керівник (підпис) Войтенко О.С. (прізвище та ініціали)

Здобувач (підпис) Васюк А.В. (прізвище та ініціали)

(підпис) Карпов М.Є. (прізвище та ініціали)

РЕЗЮМЕ (SUMMARY) *до кваліфікаційної випускної роботи здобувача:*	Васюк Артем Вікторович, Карпов Михайло Євгенович Vasiuk Artem Viktorovych, Karpov Mykhailo Evhenovych *(ПІБ здобувача українською та англійською)*

ЗВО	Київський національний університет будівництва і архітектури		
Тема *(українською та англійською)*	«Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес» «Management of a software application development project to implement game elements into the educational process»		
Освітній ступінь	Бакалавр		
Факультет	Автоматизації і інформаційних технологій		
Випускова кафедра	Управління проєктами		
Спеціальність	126 Інформаційні системи та технології		
Освітня програма	Управління проєктами		
Керівник	Войтенко Олександр Степанович		
Обсяг роботи:	*пояснювальна записка, стор..*	*розділів*	*креслень формату А4*
	108	3	-
Розділ 1	Аналіз предметної області дослідження		
Розділ 2	Планування проєкту розробки програмного забезпечення		
Розділ 3	Інструментальні засоби розробки програмного продукту «SMART PM CITY»		
Висновки по роботі:	У межах виконання кваліфікаційної випускної роботи було створено навчальний застосунок «Smart PM City» з елементами гейміфікації для підвищення залученості студентів до вивчення управління проєктами. Проведено аналіз стану вищої освіти України в умовах сучасних викликів, обґрунтовано доцільність використання гейміфікації як ефективного інструменту модернізації навчального процесу. У процесі розробки було проаналізовано різні методології управління проєктами, серед яких застосовано методології Agile, Scrum та Scrumban, складено календарно-мережевий графік, фінансову модель, визначено ризики та ресурси. Особливу увагу приділено інструментальним засобам розробки ПЗ. Робота підтвердила доцільність поєднання освітніх і розважальних елементів, а отримані результати можуть бути впроваджені у навчальний процес та слугувати основою для подальших досліджень у сфері EdTech.		
Ключові слова: *Keywords:*	Управління проєктами, розробка програмних продуктів, гейміфікація в освіті, EdTech, Agile, інструменти розробки ПЗ, навчальний застосунок «Smart PM City» Project management, software development, gamification in education, EdTech, Agile, software development tools, educational application "Smart PM City"		

Керівник

(підпис)

Войтенко О.С.
(прізвище та ініціали)

Здобувач

(підпис)

Васюк А.В.
(прізвище та ініціали)

(підпис)

Карпов М.Є.
(прізвище та ініціали)

“___” _____20__

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 11
1.1. Аналіз вищої освіти в Україні 11
1.2. Аналіз методів та підходів, щодо управління розробки програмного забезпечення 14
1.2.1. Аналіз життєвих циклів, які використовуються в ІТ проєктах 14
1.2.2. Аналіз основних підходів, що використовуються у ІТ проєктах 25
1.2.3. Аналіз підходів, що використовуються у гнучкому підході 28
1.3. Аналіз сучасних технологій щодо реалізації проєктів з розробки програмного забезпечення 36
1.3.1. Аналіз технологій розробки програмного забезпечення 36
1.3.2. Аналіз технологій розробки відеоігор 40
1.3.3. Аналіз підходів щодо застосування методів гейміфікації в освітньому процесі 44
Висновки до розділу 1 48
РОЗДІЛ 2. ПЛАНУВАННЯ ПРОЄКТУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 49
2.1. Аналіз середовища проєкту розробки 49
2.2. Вибір моделі життєвого циклу проєкту 59
2.3. Статут та календарно-мережевий графік виконання проєкту (в MS Project) 61
2.3.1. Статут проєкту розробки ПЗ 61
2.3.2. Мережевий графік проєкту 62
2.4. Фінансові показники проєкту (в MS Project) 65
2.4.1. Матеріальні ресурси 65
2.4.2 Розрахунок заробітної плати 67
2.4.2 Вибір бізнес моделі 71
Висновки до розділу 2 72
РОЗДІЛ 3. ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ «SMART PM CITY» 73
3.1. Вибір середовища розробки 73

3.2. Робота з дизайном 77
3.3. Процес та механіки гри «SMART PM CITY» 80
3.4. Що було зроблено 84
3.5. Інтеграція навчальних елементів у гру 96
Висновки до розділу 3 96
ЗАГАЛЬНІ ВИСНОВКИ 98
СПИСОК ДЖЕРЕЛ 100
ДОДАТКИ 107
Додаток А. Статут проєкту 108
Додаток Б. Календарно-мережевий графік проєкту 117
Додаток В. UML діаграми класу 118
Додаток Г. Приклад коду гри. Скрипт GameField.cs 123
Додаток Д. Приклад коду гри. Скрипт PlayerHand.cs 126
Додаток Е. Приклад коду гри. Скрипт ProjectsField.cs 130
Додаток Ж. Презентація до КВР **Error! Bookmark not defined.**

ВСТУП

Сьогодні такі розваги, як настільні та комп’ютерні ігри, займають важливе місце в житті як молоді, так і дорослих поколінь людей. І з кожним роком кількість людей, які проводять свій вільний час за відеоіграми, зростає. Використання нових методів навчання через використання ігор або їх елементів може позитивно вплинути на рівень підготовки як учнів, так і студентів.

Актуальність даного дослідження зумовлена швидким розвитком як суспільства загалом, так і науки. З кожним роком кількість інформації збільшується в десятки разів в порівнянні з минулими роками і, відповідно, студенти та учні різних навчальних закладів мають слідкувати та встигати за сучасними технологіями та запам’ятовувати більше інформації. Тому пошук нових креативних методів для навчання набирає популярність з кожним роком все більше і більше. Одним з яких є гейміфікація – застосування ігрових елементів, технік або навчальних ігор в освітньому процесі для створення інтерактивного та мотивуючого середовища.

Об’єктом проєкту є методи та інструменти як для управління проєктом реалізації програмного забезпечення так і для самої розробки програмного забезпеченя, частиною якого є і реалізація відеоігор.

Предметом дослідження є програмний продукт, яким є навчальна гра «Smart PM City».

Мета дипломного проєкту полягає у встановленні та обґрунтуванні підходів до управління проєктами розробки програмного забезпечення та підходів щодо розробки самого програмного забезпечення. Також метою є дослідження середовища проєкту та розробити продукт, що буде відповідати всім вимогам та нормам.

Проєкт принесе значну цінність для системи освіти та суб’єктам цієї системи (університети, викладачі, студенти, МОН), бо на основі результатів майбутнього

впровадження, надасть можливість проаналізувати ефективність підходів гейміфікації у реаліях.

Продукт проєкту є унікальним завдяки поєднанню навчальних елементи з розважальними механіками, що є гейміфікацію навчання з фокусом на управління проєктами, що для ринку освіти України є незвичним. Також, сам проєкт має потенціал до розширення та охоплення більше навчальних дисциплін.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1. Аналіз вищої освіти в Україні

Для впровадження або розробки нових технологій для тої або іншої області необхідно для початку дослідити ринок цієї області, бо фінансові показники відображають зацікавленість суспільства у цьому напрямку.

Ринок освітніх послуг в Україні є динамічною та багатогранною системою, яка зазнає значних змін під впливом глобальних та внутрішніх факторів. Вища освіти в Україні має свої переваги та недоліки, що можуть бути обумовлені як внутрішніми, та і зовнішніми факторами.

Наприклад, українська система вищої освіти приваблює значну кількість іноземних студентів [1] завдяки конкурентним перевагам, таким як якісна підготовка за доступними цінами. Однак існують проблеми, пов'язані з організацією навчального процесу для іноземців, необхідністю покращення інформаційної підтримки та спрощення процедур легалізації документів.

Як написано у статті [2]: світовий ринок технологій у сфері освіти швидко зростає, і Україна не є винятком. Згідно з прогнозами, до 2025 року обсяг цього ринку досягне $404 млрд. В Україні спостерігається активний розвиток цього напрямку, що відкриває нові можливості для дистанційного навчання та інтеграції сучасних технологій в освітній процес.

Однак, воєнний конфлікт в Україні став однією з головних причин погіршення освіти загалом [3-4], що суттєво вплинуло на ринок освітніх послуг. Спостерігається зменшення попиту на вищу освіту через еміграцію українського населення та небезпеку на територіях, де розташовані навчальні заклади.

До того ж проблеми, які були в Україні ще задовго до конфлікту [5] нікуди не ділись, а саме:

– неефективне використання фінансових ресурсів. Україна витрачає на освіту близько 6% ВВП з державного бюджету та додатково 1% з приватних

фондів, що перевищує показники багатьох країн ЄС. Проте результати навчання не відповідають обсягу вкладених коштів, що свідчить про низьку ефективність витрат; нерівність у доступі до якісної освіти: Існує значна різниця в якості освіти між міськими та сільськими школами. Учні з великих міст мають більше можливостей навчатися в ліцеях, гімназіях та спеціалізованих школах, тоді як у сільській місцевості якість освіти часто є нижчою. Це призводить до освітньої нерівності;

– зниження якості вищої освіти та неефективне фінансування. Мережа вищих навчальних закладів є надмірно розгалуженою, з понад 300 університетами, тоді як кількість студентів зменшується. Чинна модель фінансування не сприяє покращенню якості освіти та потребує реформування для підвищення прозорості та ефективності;

– корупція та недостатня академічна доброчесність. Корупція залишається серйозною проблемою в українській освіті. Опитування показують, що 75% батьків вважають корупцію в університетах значною або дуже поширеною, а половина батьків відзначає її наявність у середній освіті. Це підриває довіру до освітньої системи та знижує якість навчання;

– система управління освітою в Україні страждає від надмірної бюрократії, дублювання функцій та перевантаженості звітними обов'язками. Це гальмує ефективне прийняття рішень та впровадження необхідних реформ;

– недостатня адаптація до сучасних освітніх тенденцій. Українська освіта повільно впроваджує інноваційні методики, такі як інтеграція ігрових елементів у навчальний процес, що вже широко застосовується в багатьох країнах для підвищення мотивації та залученості учнів.

Незважаючи на все, ринок освіти в Україні в останні роки зазнав значних змін, відображаючи глобальні тенденції та адаптуючись до нових викликів.

Основні напрямки розвитку включають:

– використання штучного інтелекту стало ключовим трендом, допомагаючи вчителям створювати індивідуальні плани уроків, адаптовані до потреб учнів, та оцінювати їхні досягнення;

– використання віртуальних лабораторії, де дозволяють учням проводити експерименти в онлайн-середовищі, долаючи обмеження фізичних ресурсів;

– міністерство освіти і науки оновило програму предмета «Захист України», включивши домедичну допомогу, тактичну медицину та інформаційну безпеку. Також було проведено навчання вчителів для впровадження цих змін, що покращує якість освіти та відповідає сучасним вимогам;

Як йдеться у звіті Міністерства фінансів України [6]: у 2020 році фінансування освіти в Україні значно зросло порівняно з попередніми роками. Згідно з проєктом Державного бюджету на 2020 рік, видатки на освіту збільшилися на 34 мільярди гривень. Середні видатки на навчання одного студента за державним замовленням становили 63 тис. гривень, що на 13% більше порівняно з попереднім роком.

Вище зазначені дані характеризують позитивну тенденцію, щодо загального розвитку ринку освіти в Україні.

Однак, функціонування та фінансування ринку освітніх послуг в Україні зазнає впливу соціально-економічних та технологічних змін. Аналіз показує необхідність адаптації освітніх установ до сучасних умов, зокрема через впровадження інноваційних технологій та оновлення управлінської політики.

1.2. Аналіз методів та підходів, щодо управління розробки програмного забезпечення

1.2.1. Аналіз життєвих циклів, які використовуються в IT проєктах

Життєвий цикл проєкту [7] – це послідовність фаз, через які проходить проєкт від моменту його виникнення до завершення. Зазвичай цей цикл включає такі етапи, як ініціація (визначення мети та обґрунтування проєкту), планування (розробка детального плану робіт, розподіл ресурсів, оцінка ризиків), виконання (реалізація запланованих завдань), моніторинг та контроль (відстеження прогресу, управління змінами) та, нарешті, закриття (передача результатів, аналіз отриманих уроків та оформлення документів про завершення).

Підбір оптимальної моделі життєвого циклу є однією з ключових задач менеджера проєктів, бо від правильності підбору буде залежати ефективність подальшої роботи у проєкті. В розробці IT продуктів використовується достатньо багато різних моделей життєвих циклів, кожна з яких має свою специфіку при використанні та переваги і недоліки.

Далі будуть розібрані моделі, які найчастіше використовуються в проєктах з розробки ПЗ, наведена їхня загальна характеристика та головні переваги і недоліки.

1.2.1.1. Водоспадна модель життєвого циклу

Водоспадна модель життєвого циклу (англ. «waterfall model») [8] – послідовний метод розробки програмного забезпечення, названий так через діаграму, схожу на водоспад (рис 1.1).

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві – областях, в яких зміни на пізніх етапах дуже дорогі або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів тощо). Зміни в проєкті фундаменту будинку після того, як покладений дах, коштують дуже дорого, тому перфекціонізм на початкових етапах проєктування просто необхідний.

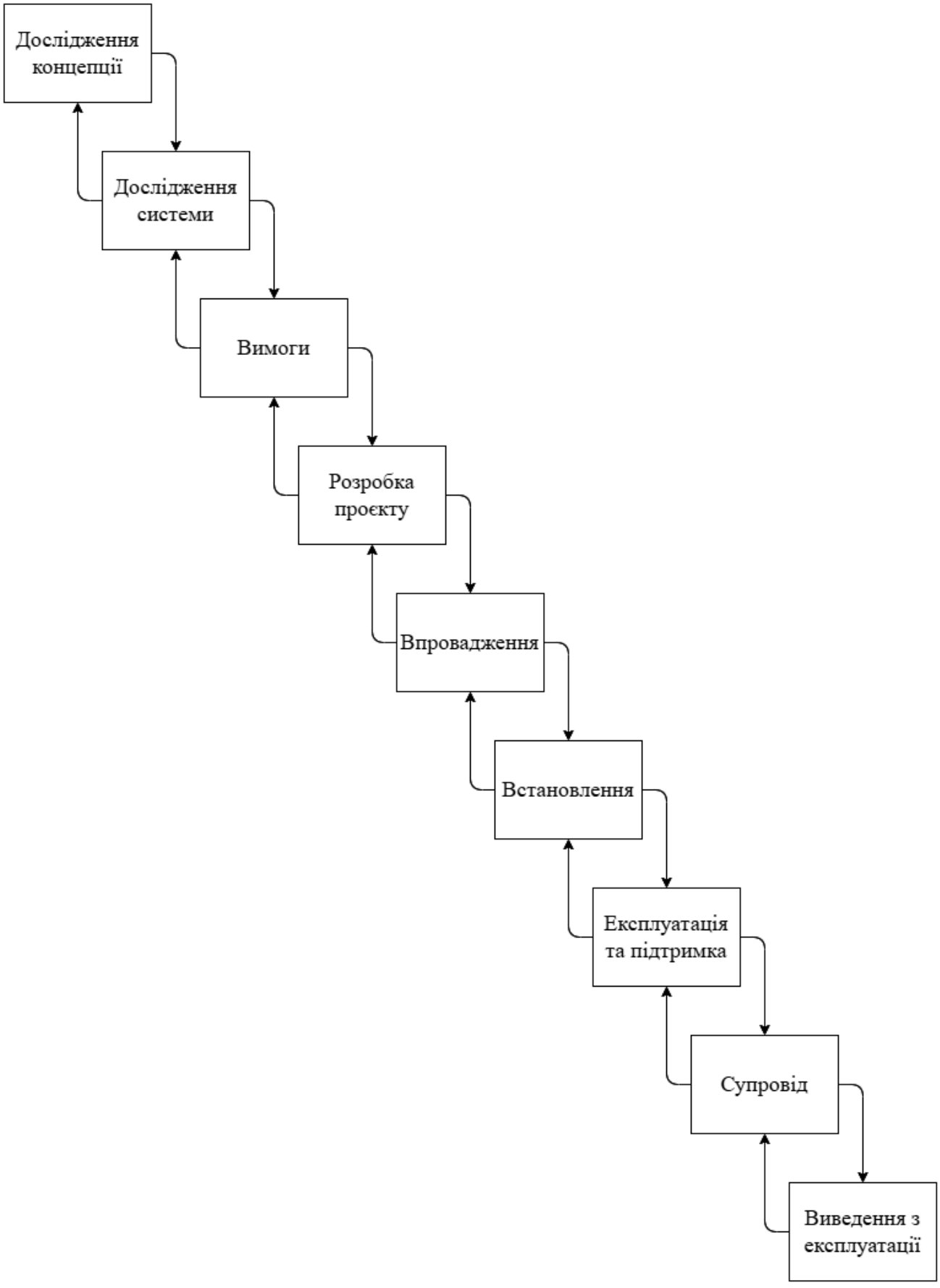


Рис. 1.1. Водоспадна модель життєвого циклу

Принципова особливість моделі – перехід на наступну стадію здійснюється тільки після повного завершення роботи на поточній стадії, повернення на пройдені стадії не передбачається. Кожна стадія закінчується одержанням результатів, що є вхідними даними для наступної стадії, та випуском повного комплекту документації. Вимоги до програмного забезпечення, визначені на стадії формування вимог, документуються у вигляді технічного завдання і фіксуються на весь час розроблення. Критерієм якості розробки за такої моделі є точність виконання специфікацій технічного завдання.

Головними перевагами даної моделі є:

– чітке визначення вимог на початкових етапах забезпечує мінімальну потребу в повторній розробці вже затверджених компонентів, що економить час і ресурси;

– ретельно оформлена специфікація стає основою для створення повної і зрозумілої документації, що сприяє подальшому ефективному супроводу продукту;

– логічна послідовність етапів дозволяє всім учасникам процесу легко орієнтуватися в проєкті та чітко розуміти свої завдання на кожному етапі.

З недоліків можна виділити:

– вимога до бездоганності на кожному кроці створює високий тиск на команду, і навіть невеликі недоліки можуть призвести до значних проблем у подальших фазах розробки;

– жорстка послідовність етапів не передбачає можливості гнучкого коригування вимог, що ускладнює адаптацію продукту до нових потреб або змін ринку;

– застосування надмірно детального планування може уповільнити розробку та збільшити витрати, оскільки багато ресурсів витрачається на створення документації, яка може виявитися непотрібною у майбутньому.

1.2.1.2. VEE подібна модель життєвого циклу

[9] VEE модель являє собою вдосконалену версію традиційної водоспадної моделі, де на кожному етапі здійснюється контроль за виконанням поточного процесу, щоб забезпечити можливість переходу до наступного рівня. У даній моделі тестування починається ще на стадії формування вимог, при цьому для кожного подальшого етапу встановлено свій рівень тестового покриття.

Назва моделі походить від форми латинської літери «V», що символізує логічну часову лінію, зігнуту навпіл у точці переходу від стадій визначення системи до етапів її реалізації та впровадження. Для програмних систем така модель може бути представлена, як показано на рис. 1.1, при цьому час розглядається як логічний показник проходження процесів, а не як реальний хід виконання робіт. Цей перелом ілюструє суть верифікації та валідації: створення окремих компонентів і всієї системи перевіряється за допомогою верифікації (перевірка відповідності формальним вимогам), а впровадження – через валідацію (перевірка задоволення вимог замовника).

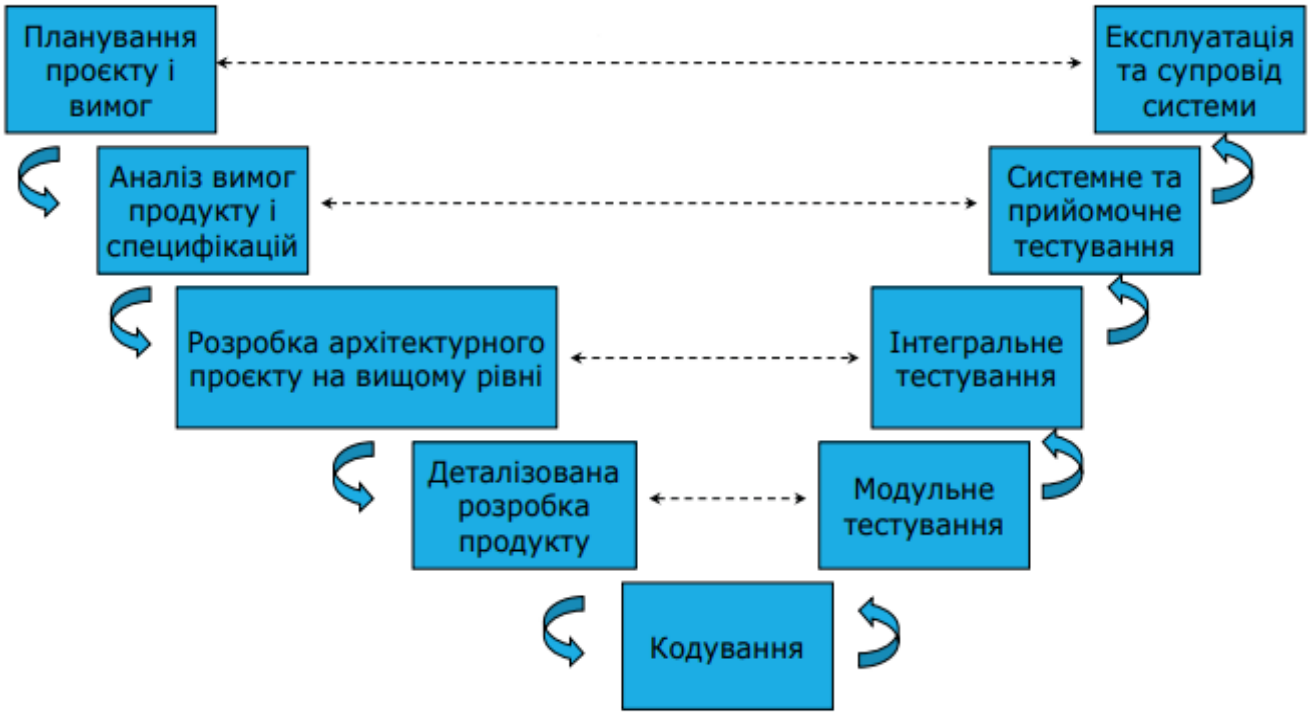


Рис. 1.2. VEE модель життєвого циклу

Для кожного рівня тестування розробляється окремий тест-план, що означає, що під час виконання тестування одного етапу паралельно формується стратегія для наступного. При створенні тест-планів визначаються очікувані результати тестування, а також встановлюються критерії входу та виходу для кожного етапу.

Головні переваги VEE моделі:

– модель розбиває процес розробки на чітко визначені фази, що забезпечує систематичне виконання завдань. Це дозволяє контролювати якість кожного етапу і уникати пропусків у виконанні робіт;

– завчасне визначення тестових сценаріїв і критеріїв якості дозволяє виявити потенційні проблеми ще до початку кодування. Це сприяє більш ефективному усуненню помилок і знижує витрати на їх виправлення;

– наявність проміжних точок контролю дозволяє краще розподіляти час і ресурси на кожну фазу проєкту. Це сприяє дотриманню графіка та своєчасному виявленню відхилень від плану;

– регулярне тестування після завершення кожної фази дозволяє виявляти дефекти на ранніх стадіях розробки. Це зменшує ризик накопичення помилок, що може призвести до значних витрат часу в кінцевих етапах проєкту.

Головні недоліки VEE моделі:

– жорстка структура не дозволяє вносити зміни у вимоги після затвердження кожного етапу. Це може бути критично важливо у випадках, коли ринкові або технологічні умови швидко змінюються;

– основна реалізація функціоналу починається лише після завершення попередніх фаз, що може призвести до виявлення проблем на пізнішій стадії. Це ускладнює внесення суттєвих змін у архітектуру продукту;

– модель не передбачає детального аналізу потенційних ризиків на початкових етапах розробки. Відсутність цього аналізу може призвести до несподіваних проблем у процесі розробки і вплинути на успішність проєкту;

– немає роботи з паралельними подіями і можливості динамічного внесення змін. VEE модель не дозволяє виконувати декілька процесів одночасно, що

обмежує адаптивність до змін. Це ускладнює реагування на непередбачені ситуації та швидке коригування плану проєкту.

1.2.1.3. Інкрементальна модель життєвого циклу

[10] Інкрементальна модель розробки програмного забезпечення передбачає побудову системи невеликими, керованими частинами – інкрементами. Кожен інкремент представляє собою одну частину функціональної системи, і після кожної іт інкременту надається робоча версія продукту (рис. 1.3). Ця модель дозволяє зменшити початкові витрати, зосереджуючись спочатку на критично важливих функціях, і забезпечувати гнучкість змін у вимогах протягом усього проєкту.

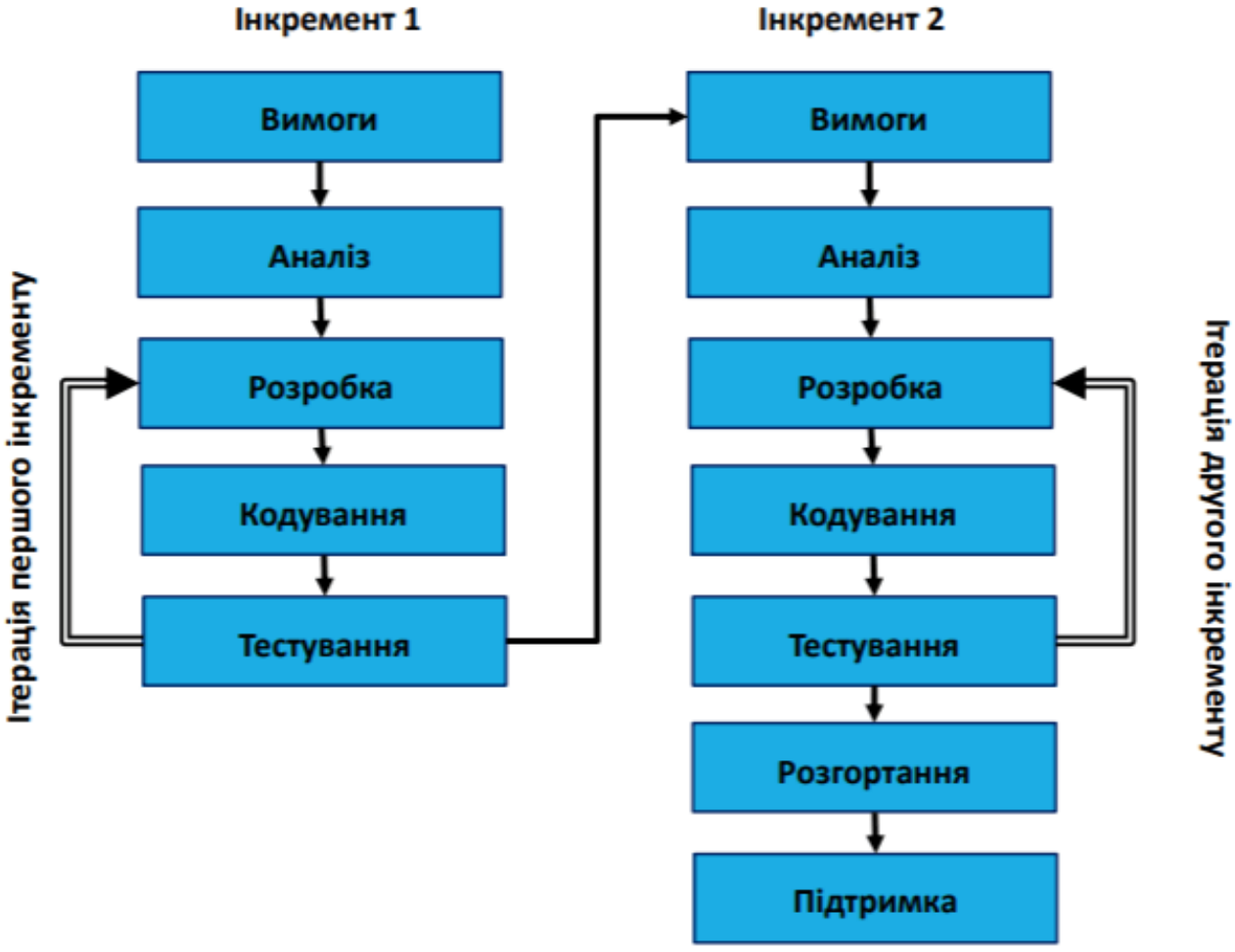


Рис. 1.3. Інкрементальна модель життєвого циклу

Інкрементальна модель зосереджується на поетапному впровадженні функціональних частин системи. Кожен інкремент додає нові функції або

вдосконалює наявні, дозволяючи клієнтам бачити конкретні результати на ранніх етапах розробки. Наприклад, FinTech (Financial Technology) компанія, яка займається розробкою мобільного банківського застосунку, може спочатку випустити базовий функціонал – створення облікового запису та відстеження балансу. А потім вже розробити додатковий функціонал, як історія транзакцій, оплата рахунків та інструменти для інвестування. Це забезпечує користувачам ранній доступ до ключових функцій, при цьому, додатковий функціонал поступово удосконалюється та інтегрується.

Інкрементальна модель життєвого циклу узагальнено представляють наступним чином:

– на старті проєкту проводиться комплексний аналіз для визначення вимог до системи та їх пріоритезації. Цей етап створює основу всього проєкту, виділяючи критично важливі функції, які будуть розроблятися першочергово;

– далі розробляється загальний дизайн системи, який забезпечує узгодженість усіх запланованих ітерацій. При цьому дизайн може змінюватися у процесі уточнення вимог;

– розробка системи здійснюється невеликими, контрольованими інкрементами, де кожен новий цикл включає етапи кодування, тестування та інтеграції нових можливостей;

– після кожного інкременту нові функції проходять етапи тестування, валідації та подальшого вдосконалення. Це дозволяє виявляти можливі проблеми на ранніх стадіях, що знижує ризик виникнення серйозних помилок у майбутньому.

– після завершення кожної ітерації готовий продукт передається клієнту, що забезпечує доступ до працездатних частин системи. Такий поетапний процес дозволяє замовнику відслідковувати прогрес розробки та надавати цінний зворотний зв’язок.

До переваг інкрементальної моделі відносять:

– доступ до продукту на ранніх етапах проєкту, що дає змогу користувачам отримати доступ до основних функцій ще до завершення повної розробки;

– зниження витрат дозволяє зосередитися на найважливіших функціях та рівномірно розподілити витрати;

– можливість бачити реальні результати та надавати зворотний зв’язок допомагає краще задовольняти потреби клієнтів;

– гнучкість та адаптивність реалізуються через зміни, легко вносити, та додавання нових функції в подальших інкрементах;

– розробка поетапно дозволяє ефективніше контролювати можливі ризики.

Проблеми та обмеження:

– незважаючи на гнучкість моделі, часті зміни можуть призвести до розширення меж проєкту та затримок у його реалізації;

– кожна частина системи розробляється окремо, що може створити труднощі при їх об’єднанні в єдину структуру. Чітка комунікація та ретельне планування можуть значно знизити ризики інтеграції;

– успіх моделі безпосередньо залежить від грамотної взаємодії між розробниками та зацікавленими сторонами, а також від точного планування кожного інкремента;

– для ефективного застосування інкрементальної моделі критично важливо активне залучення клієнта на всіх етапах розробки, що вимагає значних зусиль для організації процесу взаємодії.

1.2.1.4. Ітеративна модель життєвого циклу

[11] Ітеративна модель – життєвий цикл в якій використовується замість однієї тривалої послідовності дій, ряд окремих міні-циклів. Причому кожен з них складається із все тих же базових стадій моделі життєвого циклу. Назва цих міні-циклів – ітерації. У кожній з ітерацій відбувається розробка окремого компонента системи або проєкту, після чого цей компонент додається до вже раніше розробленого функціоналу (рис. 1.4).

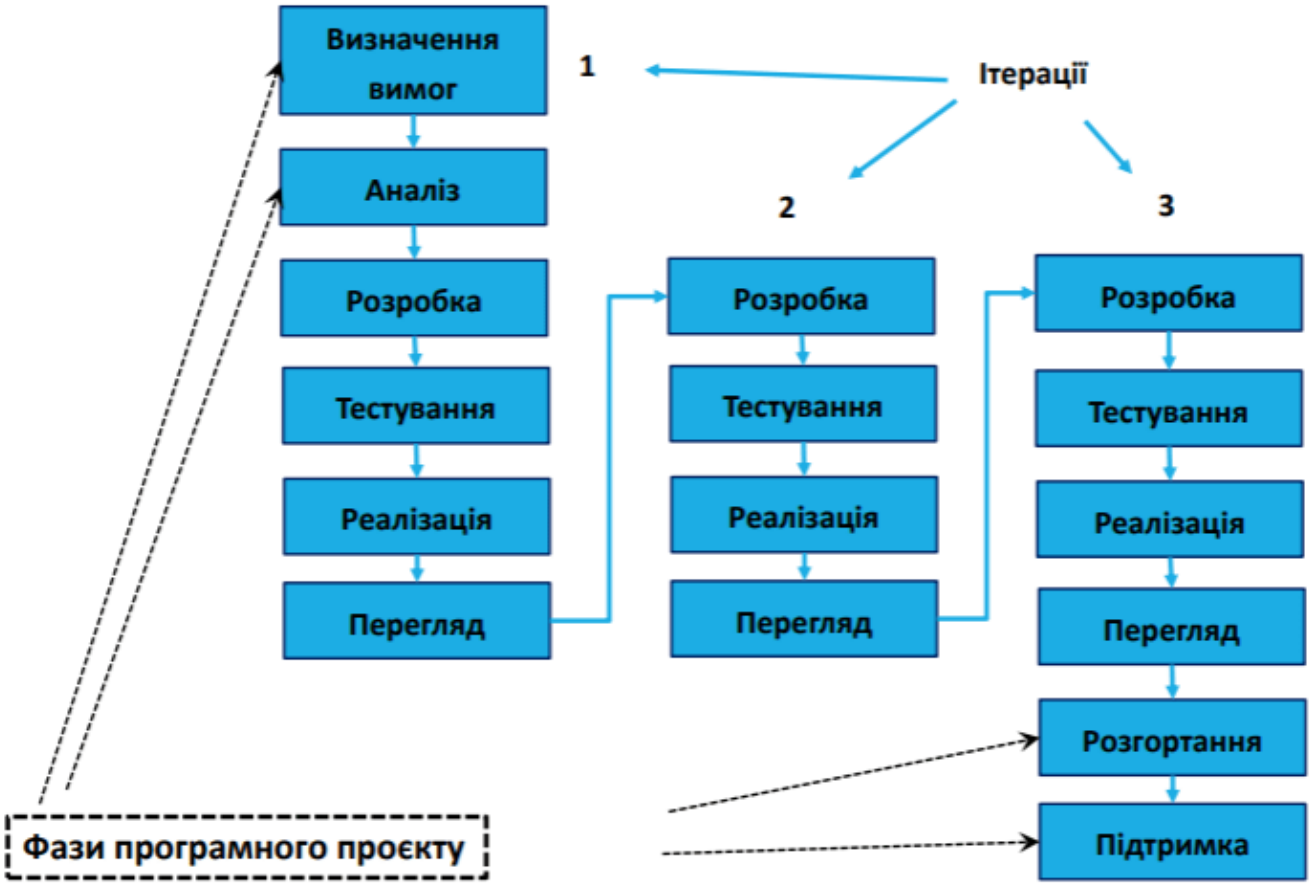


Рис. 1.4. Ітеративна модель життєвого

Ітеративна модель не передбачає певного обсягу вимог для початку робіт над продуктом. Розробка продукту може починатися з вимог до частини функціоналу, які можуть згодом розширятись і змінюватися. З кожним новим циклом, створюється нова версія продукту – реліз.

У простому варіанті, ітеративна модель складається з чотирьох основних стадій, які повторюються у кожній з ітерацій:

– визначення та аналіз вимог;

– дизайн та проєктування. Причому дизайн може як розроблятися окремо для даної функціональності, так і доповнювати вже існуючий;

– розробка і тестування нового компонента;

– перевірки, оцінка, перегляд поточних вимог та пропозиції щодо їх доповнення.

Після кожної ітерації приймається рішення чи будуть використані її результати для доповнення існуючої функціональності у проєкті і початку нової ітерації. Зрештою, досягається точка, в якій всі вимоги були втілені в продукт – відбувається реліз.

Важливим моментом є сувора верифікація вимог і ретельна валідація до розробки нових функцій в кожній з ітерацій.

Основні стадії процесу розробки в ітеративній моделі фактично повторюють модель водоспаду. У кожній ітерації створюється програмне забезпечення, що вимагає тестування на всіх рівнях.

Переваги ітеративної моделі:

– швидке створення працюючого ПЗ;

– готовність до зміни вимог на будь-якому етапі розробки;

– кожна ітерація може мати малий об’єм завдань, проаналізувати ризики та провести тести простіше, ніж для всього життєвого циклу продукту.

Недоліки ітеративної моделі:

– кожна ітерація самостійна і не накладаються одна на одну;

– можуть виникнути проблеми з реалізацією загальної архітектури системи, оскільки не всі вимоги відомі до початку проєктування.

1.2.1.5. Спіральна модель життєвого циклу

[12] Спіральна модель поєднує в собі ітеративну та каскадну модель. Суть її в тому, що весь процес створення продукту представлений у вигляді умовної площини, розбитої на 4 частини, Кожна з яких представляє окремий етап, а саме:

1. Визначення цілей;

2. Оцінка ризиків;

3. Розробка та тестування;

4. Планування нової ітерації.

У спіральної моделі життєвий шлях зображується у вигляді спіралі (рис. 1.5). Вона розпочинається на етапі планування та розкручується з проходженням

кожного наступного етапу. На виході з чергового витка (спіралі) повинні отримати готовий протестований прототип, який доповнює існуючий функціонал продукту та розробка якого, якщо задовольняє всім вимогам, може бути завершена.

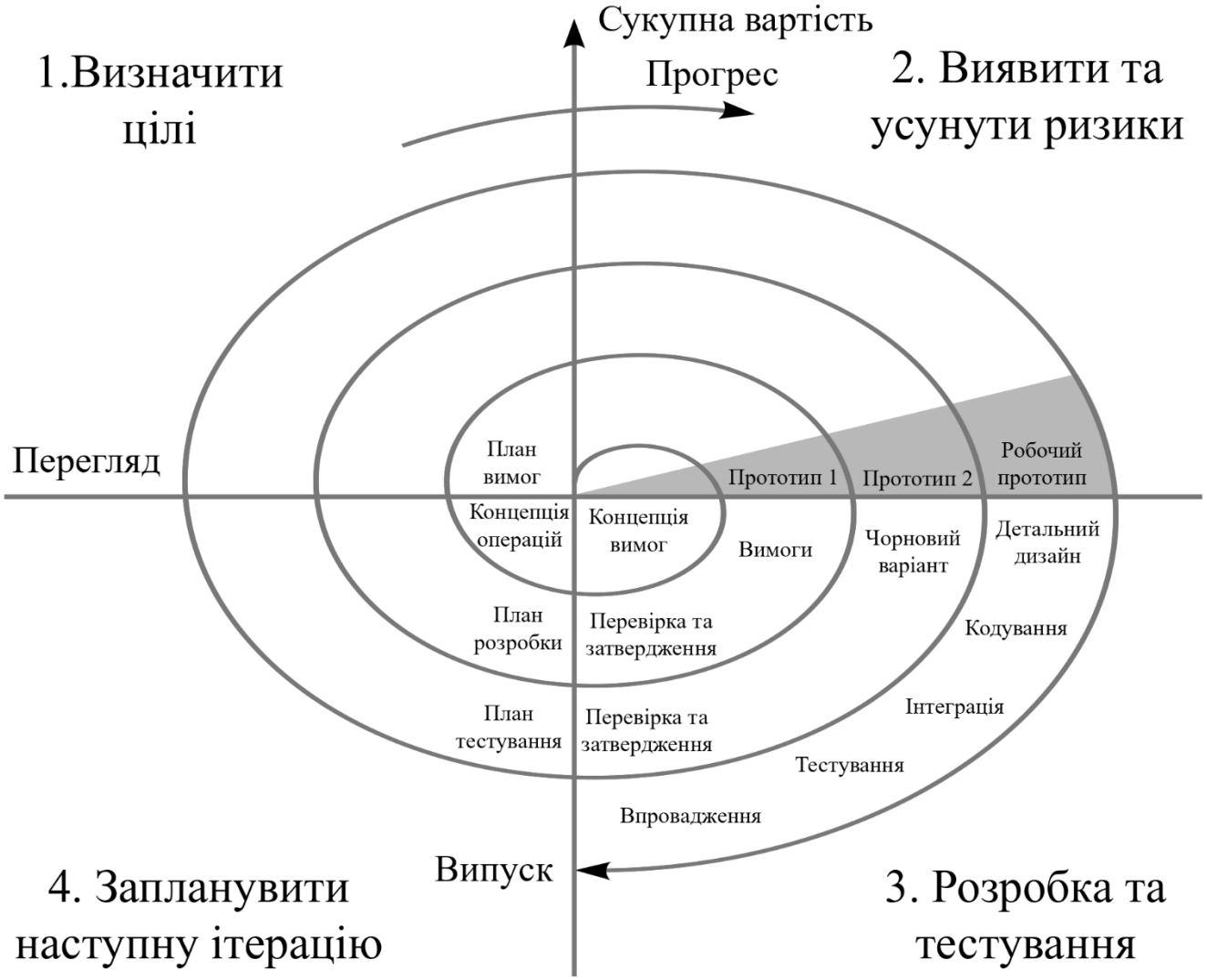


Рис. 1.5. Спіральна модель життєвого циклу

Переваги спіральної моделі:

– детальний аналіз ризиків;

– хороша документація процесу розробки;

– можливість внесення змін і додавання нової функціональності навіть на відносно пізніх етапах, що забезпечує гнучкість розробки;

– раннє створення робочих прототипів.

Недоліки даної моделі:

– може бути досить дорога у використанні;
– управління ризиками вимагає залучення висококласних фахівців;
– успіх процесу у певній мірі залежить від стадії аналізу ризиків;
– не підходить для малих проєктів.

Як видно з вищезазначених плюсах та мінусах, особливість спіральної моделі – концентрація на ризиках, що може, як зіграти на руку, так і бути недоліком майбутнього проєкту. Для оцінки ризиків навіть виділена відповідна фаза. Якщо віднести цю особливість до проєкту даного диплому, то основні типи ризиків, які можуть виникнути в процесі розробки програмного забезпечення це:

– нереалістичний бюджет і терміни;
– дефіцит фахівців;
– часта зміна вимог;
– надмірна оптимізація;
– низька продуктивність системи;
– невідповідність рівня кваліфікації фахівців.

1.2.2. Аналіз основних підходів, що використовуються у ІТ проєктах

Методологія в управлінні проєктами – це систематизований набір принципів, процесів, інструментів та практик, який встановлює структуру для планування, організації, виконання, моніторингу та завершення проєкту. Вона допомагає забезпечити послідовність дій, ефективне використання ресурсів, контроль за термінами, бюджетом та якістю, а також мінімізацію ризиків протягом усього життєвого циклу проєкту.

1.2.2.1. Методологія гнучкої розробки програмного забезпечення

[13] Гнучка розробка програмного забезпечення або Agile – методологія розробки програмного забезпечення, заснована на ітеративній розробці, де вимоги зіставляються завдяки співпраці між самоорганізаційними командами, що відповідальні за різні сторони проєкту.

Підходи методології Agile сприяють впорядкованому процесу управління проєктами [14], який передбачає часті перевірки, командну роботу, організацію і звітність. Підходи призначені для швидкого релізу високоякісного продукту. Гнучка розробка відноситься до будь-якого процесу розробки, що узгоджується з концепціями Agile маніфесту.

Через швидкий та частий реліз версій продукту, за допомогою цієї методології, можливо відразу виявити та усунути помилки на ранніх етапах розробки. Передбачається, що команда проєкту, яка використовує Agile, частіше взаємодіє із зацікавленими сторонами впродовж всього проєкту, залучаючи клієнта в кожен етап розробки. Своєчасне презентація робочого програмного продукту підвищує довіру між зацікавленими сторонами і спонукає їх більш активно брати участь в проєкті. Однак це також вимагає від клієнтів розуміння того, що вони бачать незавершену роботу.

Завдяки використанню фіксованого розкладу спринтів з періодом 1-4 тижні, нові функції для продукту надаються швидко і часто. Це також дає можливість випустити або провести тестування програмного забезпечення раніше, ніж планувалося, якщо це є необхідним. Кожен спринт має фіксовану тривалість, тому вартість передбачувана і обмежена обсягом робіт.

Загалом, можна сказати, що Agile надає можливість зосередитися на високоякісній розробці продукту, розбиваючи проєкт на керовані одиниці. Ці одиниці легко перевіряються та тестуються, що сприяє швидкому виявленню та виправленню дефектів.

1.2.2.2 Методологія Lean

Ощадлива (або бережлива) розробка програмного забезпечення (англ. «Lean software development») [15] – це методика орієнтована на постійне усунення усіх видів витрат.

Серед головних втрат під час розробки програмних продуктів виділяють надлишкові функції, непотрібні доопрацювання, незавершені роботи, а також створення дефектного продукту, що спричиняє витрати часу на виявлення та усунення помилок, які можуть складати до 40-50% загального часу розробки.

Ця методологія була запозичена з концепції ощадливого виробництва та використовує аналогічні методи, а у сфері ПЗ її впровадження виникло серед прихильників гнучких підходів. Основою бережливої розробки програмного забезпечення є реалізація ресурсозберігаючих та безвідходних виробничих процесів.

Бережлива розробка ПЗ ґрунтується на семи принципах, концептуально близьких до принципів ощадливого виробництва. Окрім постійного прагнення усунути всі види витрат, що не створюють цінності для кінцевого користувача, виділяють наступні принципи:

– основна увага на навчанні, підвищенні кваліфікації розробників. Це сприяє створенню оптимального середовища для розробки продукту, де замість накопичення зайвої документації основний акцент робиться на впровадженні нових ідеї функцій;

– рішення приймаються не на базі припущень чи прогнозів, а після отримання ключових фактів;

– швидка доставка продукту замовнику має бути однією з головних цілей. Завдяки коротким ітераціям компанії можуть оперативно реагувати на вимоги ринку;

– повинна бути мотивація команди, бо людей не можна розглядати виключно як ресурс;

– розробка проєкту має орієнтуватись на якість продукту, який повинен відповідати всім вимогам клієнта;

– повинно бути цілісне бачення проєкту, що означає стандартизацію процесів та встановлення міцних зв’язків між розробниками.

1.2.3. Аналіз підходів, що використовуються у гнучкому підході

1.2.3.1. Kanban

Kanban [16] – це підхід управління робочими процесами, який виник в Японії та використовувався в системі виробництва Toyota. Слово «kanban» в перекладі з японської означає «сигнальна картка» або «вивіска». Цей метод спрямований на візуалізацію процесів, щоб покращити ефективність та прозорість роботи.

Основні елементи Kanban включають в себе: дошку, списки, стовпці та картки. Дошка представляє проєкт або місце для відстеження інформації, де ви візуалізуєте всі робочі елементи. Це дозволяє членам команди бачити робочі процеси в режимі реального часу та керувати ними. Дошка має бути розділена мінімум на 3 стовпці, адже кожен процес можна розбити щонайменше на три етапи: «Запит», «Виконання», «Результат». Стовпчики містять картки, якими зображуються задачі. Картки переходь з одного стовпчику до наступного, таким чином зображаючи рух функції від ідеї до тестування та готового рішення. Картці може бути присвоєно відповідальних за неї розробників. Розробники можуть об'єднуватись в команди.

[17] Ще одним важливим елементом є ліміти на виконання робіт. Для досягнення високої ефективності команді необхідно знати, скільки завдань їй під силу виконати протягом певного робочого періоду.

Картка у Kanban – це візуальне зображення робочого елементу (завдання, можливості, ідеї). Вони містять відомості про завдання та його статус, такі як опис, термін, розмір, відповідальні особи тощо.

Kanban допомагає ефективно управляти командами різного розміру та розподіляти ресурси на виконання завдань, що допомагає вивести бізнес навіть із кризи [18].

1.2.3.2. Scrum

Ще один популярний підхід до управління проєктами, в IT-індустрії – Scrum. [19] Він спрямований на підвищення ефективності та гнучкості команд у проєктах. Визначення Scrum базується на принципах гнучкого управління, де проєкти виконуються в коротких ітеративних циклах, які називаються спринтами. Сам підхід призначений для швидкого реагування на зміни, мінімізації ризиків та регулярного випуску високоякісних продуктів.

Scrum складається з таких ключових елементів: ролі, артефакти та події. Вони створюють ефективний процес управління проєктами.

До ролей в Scrum відносять:

– власник продукту, який відповідає за максимізацію цінності продукту, визначення пріоритетів та забезпечує роботу команди над правильними завданнями;

– Scrum-майстер забезпечує правильне розуміння та застосування підходу Scrum. Scrum-майстер усуває перешкоди, підтримує команду та слідкує за процесами Scrum;

– команда розробників працюють над створенням продукту. Команда є самоорганізованою та багатофункціональною, тобто має всі навички, необхідні для виконання роботи.

Під артефактами мають на увазі:

– беклог (англ. «Backlog») продукту – список усіх функцій, удосконалень, виправлень та завдань, які необхідно виконати в проєкті;

– беклог спринту – набір завдань, вибраних з беклогу , які команда планує виконати в поточному спринті;

– інкремент – робочий продукт або його частина, що доставляється в кінці кожного спринту.

Scrum-події це:

– спринти, що є основною часовою одиницею в Scrum, зазвичай триває від одного до чотирьох тижнів;

– кожен спринт має бути спланований. Відбувається зустріч, на якій команда планує завдання, необхідні в майбутньому спринті;

– щоденні зустрічі, які ще називають щоденні scrum-meeting. Це коротка щоденна зустріч, на якій команда обговорює прогрес і планує роботу на наступний день;

– зустріч наприкінці спринту, де презентують завершений інкремент продукту зацікавленим сторонам, після чого збирають їхні відгуки;

– ретроспектива спринту – це зустрічі, на яких Scrum-команда оцінює спринт, визначає успіхи та елементи для вдосконалення.

Ці елементи створюють основу, яка допомагає командам ефективно управляти проєктами, створювати високоякісні продукти та постійно вдосконалювати свої процеси.

Як і будь-яка методологія управління проєктами, Scrum має переваги та недоліки, які варто враховувати перед впровадженням.

Перевагами Scrum є:

– підвищена гнучкість, що проявляється у можливості швидко реагувати на мінливі зміни у вимогах та умовах проєкту;

– регулярні зустрічі, як щоденні, так і ретроспективи спринту, що сприяють відкритому спілкуванню та співпраці всередині команди, що призводить до кращої координації діяльності;

– Scrum забезпечує регулярну доставку робочих етапів продукту, що дозволяє раніше і частіше надавати цінність клієнту;

– регулярні огляди спринтів та презентації етапів розвитку продукту сприяють постійному зворотному зв’язку зі стейкхолдерами та кращому узгодженню продукту з їхніми потребами.

З іншої сторони, недоліками Scrum є:

– повна віддача від усіх членів команди. Недостатня залученість або небажання працювати за цією методологією може призвести до неефективності;

– довгострокове планування може бути складним у Scrum, оскільки команда зосереджується на коротких спринтах;

– для ефективного впровадження Scrum потрібен досвідчений Scrum-майстер, який зможе керувати процесами і підтримувати команду;

– надмірна зосередженість на досягненні цілей спринту, а не на довгострокових цілях проєкту, що може призвести до ситуації, коли короткострокові цілі затьмарюють довгострокову стратегію;

– впровадження Scrum в організації може передбачати додаткові витрати на навчання та трансформацію процесів.

1.2.3.3. Scrumban

Як вказано в [17] Scrumban поєднує найкращі методи зі Scrum та Kanban, створюючи гібридну систему управління проєктами. Від Scrum у цьому підході використовується стабільна структура спринтів, зустрічей та ретроспектив. Kanban, в свою чергу, забезпечує наочне представлення процесів та обмежень щодо незавершених робіт. В результаті отримуємо дійсно гнучкий метод управління проєктами будь-якого розміру.

Спочатку інструмент Scrumban застосовувався для того, щоб команди могли легко переходити від Scrum до Kanban і навпаки, але тепер це розвинена система, що дозволяє справлятися з будь-якими проєктами.

Підхід Scrumban використовує Scrum, беручи такі його компоненти як спринти, щоденні зустрічі та зустрічі для ретроспективи. Зі сторони Kanban

використовуються дошка, завдання у вигляді карток та ліміти на виконання завдання.

Переваги методології Scrumban [17, 20]:

– підвищена гнучкість проєкту, що забезпечується релізом нової версії продукту після кожного спринту. Такий підхід дозволяє вносити зміни до проєкту навіть після його початку. При цьому вони не перешкоджають прогресу реалізації проєкту.

– робота представлена на дошці, а процес безперервний, що дозволяє командам постачати функції в міру їхнього завершення, не чекаючи закінчення спринту.

– зниження навантаження, адже існують обмеження виконуваної роботи відповідно до можливостей команди дозволяє просуватися до мети без емоційного вигоряння.

– розміщення карток на дошці забезпечує прозорість, необхідну командам Scrumban для покращення спільної роботи та оперативного вирішення виявлених проблем.

– здатність реалізовувати великі проєкти. Оскільки суть Scrum і Kanban полягає у постійному та поступовому покращенні, Scrumban дозволяє командам працювати над найскладнішими проєктами.

Однак, у підходу Scrumban є свої обмеження [17]:

– Scrumban – відносно новий підхід, тому відомостей щодо його реалізації не так багато. У зв'язку з цим можуть виникнути проблеми з пошуком рекомендацій чи передової практики.

– Scrumban немає традиційних scrum-ролей, учасники команди самостійно керують спринтами. Відсутність конкретного керівника може призвести до плутанини у розподілі обов'язків.

– складність у використання елементів двох методологій може спантеличити учасників команди, які користувалися не Agile, а іншою системою.

Використовувати Scrumban потрібно у тих випадках, коли можливостей Scrum чи Kanban недостатньо. Наприклад: проєкти з розробки програмного продукту зі змінними вимогами; проєкти з кількома паралельними ініціативами, коли триває робота над кількома проєктами одночасно, особливо якщо ними займаються одні і ті ж самі команди; стартапи або середовище, що швидко змінюється, коли будь-якої миті може виникнути нове завдання, а ресурси бувають обмежені.

1.2.3.4. eXtreme Programming

Екстремальне програмування або XP (англ. «eXtreme Programming») [21] – гнучкий підхід розробки програмного забезпечення. Як і в інших Agile підходах, у нього є особливі інструменти, процеси і ролі. В XP не придумали нічого нового, а взяли кращі практики гнучкої розробки і посилив до максимуму.

Мета підходу – впоратися з постійно мінливими вимогами до програмного продукту і підвищити якість розробки. Тому XP добре підходить для складних і невизначених проєктів. Він будується навколо чотирьох процесів: кодування, тестування, дизайну та огляду.

Усі учасники проєкту, при підході XP, працюють як одна команда. У неї обов'язково входить представник замовника, а якщо це реальний кінцевий користувач продукту то ще краще. Замовник висуває вимоги до продукту і розставляє пріоритети в реалізації функціоналу продукту. В цьому йому можуть допомагати бізнес-аналітики. З боку виконавців у команду входять розробники і тестувальники, інколи коуч (тренер), що направляє команду, і менеджер, який забезпечує команду ресурсами.

Планування в XP проводять у два етапи: планування релізу і планування ітерацій. На плануванні релізу команда програмістів зустрічається з замовником, щоб з'ясувати, який функціонал він хоче отримати до наступного релізу. Так як вимоги замовника часто розмиті, розробники конкретизують їх і дроблять на частини, реалізація яких займає не більше одного дня. Важливо, щоб замовник розбирався в операційному середовищі, в якому буде працювати продукт.

Задачі записуються на картки, а замовник визначає їх пріоритет. Далі розробники оцінюють, скільки часу піде на кожну задачу. Коли задачі описані і оцінені, замовник переглядає документацію, після чого повідомляє про початок розробки. Якщо команда не встигає виконати всі задачі до дати релізу, то реліз не відсувається.

За XP підходом версії випускаються часто, але з невеликим функціоналом. По-перше, маленький обсяг функціональності легко тестувати. По-друге, на кожну ітерацію замовник отримує частину функціоналу, яка має бізнес-цінність.

В XP будь-який розробник може редагувати будь-який шматок коду, так як код не закріплений за своїм автором. Кодом володіє вся команда. Це означає, що нові частини коду відразу ж вбудовуються в систему. Відразу видно, як останні зміни впливають на систему, тому можна одразу знайти помилку і виправити її. До того ж, коли новий функціонал відразу вбудовується у продукт, це означає, що команда завжди працює з останньою версією системи.

Коли кодом володіють всі, важливо прийняти єдині стандарти оформлення коду.

Одна з основних практик підходу – тести пишуться самими програмістами, причому до написання коду, який потрібно протестувати. При такому підході кожен шматок функціоналу буде протестований на 100%.

Ще одна практика XP підходу – парне програмування. Стара приказка «одна голова добре, а дві краще» відмінно ілюструє суть підходу. З двох варіантів вирішення проблеми вибирається найкращий, код оптимізується відразу ж, а помилки помічаються ще у процесі написання коду.

Ще однією практикою є «простий дизайн». Це означає робити тільки те, що потрібно зараз, не намагаючись вгадати майбутню функціональність. Простий дизайн і безперервний рефакторинг коду забезпечують просту та легкість оптимізації у майбутньому.

Рефакторинг (англ. «refactoring»), або перепроєктування коду – процес зміни внутрішньої структури програми, що не зачіпає її зовнішньої поведінки і має на меті полегшити розуміння її роботи. XP підхід передбачає постійні рефакторинг.

Переваги екстремального програмування:

– замовник отримує саме той продукт, який йому потрібен, навіть якщо на початку розробки сам точно не представляє його кінцевий вигляд;

– команда швидко додає нову функціональність за рахунок структурованого коду, частого планування і релізів;

– код завжди працює за рахунок постійного тестування, рефакторингу і безперервної інтеграції;

– код легко підтримується за рахунок дотримання єдиного стандарту і постійно рефакторингу;

– швидкий темп розробки за рахунок парного програмування, відсутність понаднормової роботи, присутності замовника у команді;

– знижуються ризики, пов'язані з розробкою, так як відповідальність за проєкт розподіляється рівномірно у команді. Найм або звільнення члена команди не зруйнує процес;

– витрати на розробку нижче, так як команда орієнтована на код, а не на документацію і збори.

Незважаючи на всі плюси, XP не завжди працює як очікується і має ряд слабких місць:

– успіх проєкту залежить від участі замовника в проєкті;

– важко передбачити витрати часу на проєкт, так як на початку немає повного списку вимог;

– успіх сильно залежить від рівня програмістів, підхід працює тільки з senior фахівцями;

– менеджмент негативно ставиться до парного програмування, не розуміючи необхідності забезпечувати заробітною платою двох програмістів замість одного;

– вимагає дуже сильних, відповідних підходу, змін в команді;

– через малу структурованість та нестачу документації не підходить для великих проєктів;

– гнучкі методології функціонально-орієнтовані, тому нефункціональні вимоги до якості продукту складно описати.

1.3. Аналіз сучасних технологій щодо реалізації проєктів з розробки програмного забезпечення

1.3.1. Аналіз технологій розробки програмного забезпечення

Як вказано в посібнику [11], програмне забезпечення (англ. «Software») – це комп’ютерні програми, а також документація й дані, що з ними пов’язані, які стосуються функціонування комп’ютерної системи. І, як правило, розробка популярної програми – складний і тривалий процес, в якому задіяна велика кількість людей. Тут, як і в будь-якому бізнесі, потрібні дослідження ринку, генерація ідей та маркетинг. А ще продукту потрібен хороший дизайн, документація, тестування та ін. Але без технічної складової всі ці речі марні.

Сучасні технології реалізації проєктів з розробки програмного забезпечення включає в себе розгляд основних інструментів, методологій та практик, які використовуються для оптимізації процесів розробки, покращення якості програмного продукту та ефективного управління проєктами і комунікацією в них. Важливими аспектами є використання систем контролю версій, інтегрованих середовищ розробки, платформ для управління проєктами, платформи для комунікацій у команді, технологій безперервної інтеграції та доставки, а також методів моніторингу та тестування. Роздивимось ці технології детальніше.

Системи контролю версій [22, 23] є ключовими інструментами для відстеження змін у коді, координації роботи між членами команди та забезпечення доступу до історії змін. Історія змін дозволяє повернутись до певної версії коду, яка є більш стабільною або просто потрібною на даний момент. Наприклад, GitHub та GitLab – це популярні платформи для хостингу репозиторіїв Git, що надають додаткові функції для спільної роботи, більш детально:

– Git [22] – розподілена система контролю версій, яка дозволяє відстежувати зміни в коді та співпрацювати з іншими розробниками. Підтримує

розгалуження та злиття, що спрощує роботу над різними функціями одночасно.Є основною платформою для інших розширень, приклади яких описані нижче в розділі;

– GitHub – веб-платформа для хостингу Git-репозиторіїв, яка надає можливості для спільної роботи, управління проєктами та інтеграції з іншими інструментами. [24] Окрім розміщення коду, учасники можуть спілкуватись, коментувати, редагування один одного, стежити за новинами знайомих, а також поєднувати свої репозиторії. Через свою простоту, ця платформа було обрана для використання у проєкті;

– GitLab [25] – альтернатива GitHub з додатковими можливостями, у вигляді інструментів автоматизації процесів тестування та розгортання.

Інтегровані середовища розробки [26] надають потужні інструменти для написання коду, рефакторингу, налагодження та інтеграції з іншими частинами розробки. Вони також мають підтримку безлічі мов програмування, що дозволяє розробникам працювати ефективно. Середовищ розробки багато, але найпопулярніші на сьогодні, це:

– Visual Studio Code [27] – текстовий редактор, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки програм. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості завдяки великій кількості як користувацьких так і офіційних розширень;

– Visual Studio [28] – це потужний інструмент, який можна використовувати для виконання всього циклу розробки додатку в одному місці. Використовується для запису, редагування та налагодження коду. Також включає компілятори, засоби завершення коду, керування версіями, розширення та багато інших функцій для покращення кожного етапу процесу розробки програмного забезпечення;

– IntelliJ IDEA – потужне середовище для розробки на Java з інтелектуальними функціями автозаповнення та рефакторингу.

Visual Studio Code був обраний для роботи з кодом у даному проєкті.

Для управління проєктами та завданнями команди існують платформи та інструменти, що дозволяють створювати, розподіляти та відстежувати задачі для команди чи окремих їх членів, окремо відстежувати їх виконання менеджерами та планувати етапи розробки на майбутнє. [29] Популярними є:

– JIRA – популярний в Agile-середовищах, де спринти та завдання можуть бути чітко сплановані, що дозволяє покращити продуктивність команди;

– Trello [30] – безкоштовна багатоплатформна система управління проєктами, розроблена Trello Enterprise. Вона використовує парадигму керування проєктами, відому як Kanban.

Графічний редактор – інструмент, що дозволяє створювати сучасні, інтуїтивно зрозумілі UI/UX дизайн для програмних додатків, які згодом інтегруються в процес розробки. Це дозволяє дизайнерам та розробникам ефективно співпрацювати. Серед найвикористовуваніших:

– Figma – хмарний інструмент для дизайну інтерфейсів та прототипування, який дозволяє командам співпрацювати в режимі реального часу, спрощуючи процес розробки дизайну та отримання зворотного зв'язку;

– Adobe Photoshop – багатофункціональний растровий графічний редактор, що розробляється та розповсюджується компанією Adobe Systems. Здебільшого працює з растровими зображеннями, проте має деякі векторні інструменти;

– CorelDRAW – це потужний і універсальний редактор векторної графіки. Крім того програма містить інші інструменти для роботи з не тільки векторною графікою.

При розробці додатку, основними редакторами були Adobe Photoshop та CorelDRAW.

Також варто згадати технологію CI/CD [31-32] (з англ. «continuous integration/continuous delivery or deployment», укр. «постійна інтеграція/ постійна доставка або розгортання»), що дозволяє автоматизувати процеси тестування, збірки та доставки програмного забезпечення, що значно прискорює вихід нових

релізів програмного забезпечення і знижує ймовірність помилок на етапах інтеграції та розгортання. Вони також дозволяють тестувати код на кожному етапі розробки, що підвищує якість кінцевого продукту. Технологія особливо корисна при частому внесенні малих змін у код, що підвищує частоту релізів. Можна зазначити, що вона відноситься до Agile-практик. Серед популярних платформ є Jenkins, GitLab CI/CD. В даному проєкті дана технологія розглядалась, але після аналізу вимог та наявних ресурсів, було вирішено не використовувати, через відсутність глобальних тестувань в кінці кожного релізу.

Успішна розробка програмного забезпечення неможлива без ефективної комунікації між членами команди. Зараз, коли команди можуть працювати віддалено або бути розподіленими по всьому світу, особливу роль відіграють сучасні інструменти для комунікації, які забезпечують зручний обмін інформацією, координацію завдань і спільну роботу над проєктами. Одними із найбільш поширених інструментів [33]:

– Slack – месенджер, який використовується в багатьох IT-компаніях. Він дозволяє створювати тематичні канали для обговорення різних аспектів проєкту, обмінюватися файлами та інтегруватися з іншими сервісами, такими як Jira, GitHub чи Google Drive. Slack зручний тим, що дозволяє команді швидко обговорювати поточні питання без перевантаження електронної пошти;

– Microsoft Teams – ще один важливий інструмент, який, крім текстових чатів, пропонує відеозустрічі, спільний доступ до файлів через OneDrive та інтеграцію з іншими продуктами Microsoft. Це ідеальне рішення для команд, які вже користуються екосистемою Microsoft, особливо для великих компаній і корпоративних проєктів;

– Discord [34] – спочатку створений для геймерів, потім став популярним серед розробників та IT-команд завдяки своїм функціональним можливостям, а саме: створення серверів, чатів, голосових кімнат, розміщення інформацію про проєкт у тематичних каналах. Його використовують молоді команди у гнучких проєктах або коли працюють над незалежними проєктоми. Особливо корисний для

стартапів та невеликих команд, де важлива швидкість комунікації та неформальна атмосфера. Завдяки цим перевагам Discord був обраний для комунікації у «Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес».

Загалом інструменти розробки ПЗ сьогодення значно підвищують ефективність та продуктивність команд і мають значні переваги. Вони дозволяють автоматизувати багато рутинних процесів, знижують ймовірність людських помилок, прискорюють розробку та виведення продукту на ринок. Крім того, ці технології сприяють кращій комунікації між членами команди, покращують моніторинг систем та забезпечують більшу прозорість процесів.

Однак є деякі недоліки, можна сказати ризики при впровадженні сучасних технологій. Наприклад, технології можуть вимагати значних витрат на навчання та адаптацію персоналу. Крім того, інструменти можуть бути складними для нових користувачів, а надмірне використання автоматизації може призвести до зниження гнучкості у прийнятті рішень або до надмірної складності налаштувань.

1.3.2. Аналіз технологій розробки відеоігор

Сьогодні існує багато інструментів для розробки відеоігор, і вибір залежить від жанру гри, платформи, рівня досвіду розробників та вимог до продуктивності. Основні категорії таких інструментів [35] включають ігрові рушії, мови програмування та зовнішні інструменти, до яких відносять: графічні редактори (Blender, Substance Painter, Adobe Photoshop, Corel DRAW), аудіоінструменти (Audacity, Ableton Live) для створення звуків та музики та для створення анімацій (Spine, Cascadeur).

Що ж до ігрового рушія, то стаття [36] описує його як програмне забезпечення, яке надає набір інструментів та середовищ для створення відеоігор. Основна мета ігрового рушія – спростити процес створення ігор, абстрагуючи складні технічні деталі та надаючи розробникам можливість зосередитися на творчих аспектах гри.

На сьогодні, при початку розробки гри, обираючи інструменти, в першу чергу звертають увагу саме на ігровий рушій, а не на мову програмування. Це пов’язано з тим, що від вибору рушія залежить швидкодія гри, наявність новітніх технологій, які рушій надає і спрощує саму розробку, та взагалі кінцевий вигляд гри. В той же час , вибір мови програмування не дає загалом ніяких переваг, бо на ігровому ринку переважають мови високого рівня, які між собою майже не відрізняються, якщо дивитись зі сторони продуктивності. Тільки мова C++, популярним представником якої є ігровий рушій Unreal Engine, має свої переваги у вигляді швидкості виконання команд, що в свою чергу може значно підвищити швидкодію гри і надати можливість використовувати більш важкі та комплексні вичислення, але в той же час підвищує ймовірність помилок, по типу витоку пам’яті (англ. «memory leak»), якщо код написаний недосвідченим програмістом. Це пов’язано з тим, що в C++ пам’ять керується в ручному режимі. Всі змінні, структури, об’єкти та інше, що не видаляються у коді вручну, залишаться у пам’яті, і у подальшому заважають роботі коду. Інші мови програмування, по типу C#, мають автоматичний «збирач сміття», що слідкує за даними і видаляє те що не знадобиться вже. Це спрощує розробку гри та знижує планку якості до самих програмістів, що з фінансової точки зору краще.

Ігровий рушій містить кілька ключових компонентів, що слугують причиною вибору того, чи іншого варіанту [37-38]:

– графічний рушій – відповідає за створення, редагування та відображення візуального контенту та елементів гри, 2D- та 3D-графіки, текстур, моделей, анімацій, ефектів освітлення та тіней;

– фізичний рушій – імітує фізичні закони для об’єктів у грі, такі як гравітація,рух, взаємодію між об'єктами, деформація;

– звуковий рушій – управляє відтворенням звуків, музики та звукових ефектів. Забезпечує відтворення просторове звучання;

– система штучного інтелекту (ШІ) – дозволяє створювати інтелектуальних, не ігрових персонажів з різною поведінкою. Включає алгоритми для пошуку шляху, прийняття рішень, слідкування та інше;

– інструменти для скриптів – дозволяють програмувати ігрову логіку та поведінку об'єктів, підтримувати різні мови програмування для гнучкості роботи, що може привабити більше коло людей до рушія.

Крім самого інструментарію, ігрові рушії обирають за такими загальними критеріями, які відносяться загалом до всього ПЗ:

– поріг входження – завдяки інтуїтивно зрозумілому інтерфейсу, використанню візуальних скриптів, можна досить швидко опанувати основи створення ігор. Це дозволяє новачкам зосередитися на творчому процесі, а не на складних технічних аспектах;

– швидкість розробки – використання інструментів, які вже виконують якусь частину ігрової логіки без написання коду, значно прискорює процес розробки. Замість написання тисяч рядків коду, розробники використовують готові шаблони та компоненти, що дозволяють швидко збирати гру з готових блоків. Це також корисно для прототипування, коли потрібно швидко перевірити ідею на практиці;

– мультиплатформний експорт – експорт на різні платформи, включаючи персональні комп’ютери, мобільні пристрої та веб-браузери. Це дає можливість створювати ігри, доступні для широкої аудиторії, без необхідності переписування коду для кожної платформи окремо;

– спільнота та підтримка – більшість ігрових рушіїв мають спільноту користувачів, які діляться своїм досвідом, порадами та готовими рішеннями. Це спрощує процес навчання та розв’язування проблем, а також надає доступ до великої кількості безплатних ресурсів та інструментів. Не завжди документація від самих розробників гри є зрозумілою або взагалі присутня.

Основними прикладами ігрових рушіїв на сьогодні [39] є Unity, Unreal Engine, CryENGINE, GameMaker:Studio та, набираючий популярність, Godot. Як претенденти для використанні у проєкті розглядались Unity, Unreal Engine і Godot.

Коротко роздивимось кожен обраний рушій [38, 40]. Unity відомий ігровий рушій, що використовується для створення як 2D, так і 3D ігор. Приклади ігор на Unity: Pokemon Go, Genshin Impact, Monument Valley 2. Цей ігровий рушій відомий своєю гнучкістю та широкими можливостями для розробників. Його основні переваги це: кросплатформність, що дозволяє розробляти одну гру на декілька платформ; велика спільнота, в якій люди допомагають один одному; магазин шаблонів, для більш швидкої розробки.

Однак з великими проєктами Unity важко, тому є вимога до оптимізації. До того ж є ліцензійні обмеження, бо безкоштовна версія має обмеження за доходом для продукту, після перевищення якого необхідно придбати платну ліцензію.

Unreal Engine – потужний ігровий рушій, відомий високоякісною графікою, що забезпечує реалістичну візуалізацію, та розширеними можливостями рендерингу. Він часто використовується для створення високобюджетних ігор. Як і Unity, цей рушій також кросплатформний. Особливістю Unreal Engine є система візуального програмування, що дозволяє створювати ігрову логіку без глибоких знань програмування. Також, треба зазначити, що компанія розробник рушія Epic Games бере 5% з розробників, які використовують у своїх додатках технології Unreal Engine і заробляють на цьому понад один мільйонів доларів на рік.

Однак високий поріг входження може відштовхнути багатьох, бо цей рушій складно вивчати порівняно з іншими. Також Unreal Engine потребує потужного обладнання для розробки та запуску проєктів. Найвідоміші ігри на цьому рушії: Fortnite, PUBG Mobile, Infinity Blade.

Godot – безкоштовний ігровий рушій з відкритим вихідним кодом, який підходить для створення як 2D, так і 3D ігор. Він відомий своєю гнучкістю та простотою у використанні та низьким порогом входження, через що і був обраний для даного проєкту. Завдяки відкритому вихідному коду, дозволяє розробникам змінювати рушій відповідно до своїх потреб. Сам рушій працює дуже швидко через його невеликий розмір. Godot зручний для початківців, бо має інтуїтивний та простий інтерфейс.Прикладами ігор на Godot можуть слугувати Hyper Light Drifter та Halls of Torment.

Але функції Godot є менш розвиненими, ніж у його конкурентів. Та і спільнота менша в порівнянні з Unity та Unreal Engine, що ускладнює пошук відповідей відносно рушія або його функцій.

Для проєкту ігровим рушієм був обраний Godot. Причини такого рішення вказані в розділі 3 пункті 3.1.

1.3.3. Аналіз підходів щодо застосування методів гейміфікації в освітньому процесі

Як світ, так і Україна робить перші кроки щодо впровадження ігрових технологій у навчальний процес як один із варіантів інноваційного підходу до навчання, що сприяє підвищенню зацікавленості учнів у освітньому процесі.

[38] Гейміфікація в освіті – це застосування ігрових елементів і технік у навчальному процесі для створення інтерактивного та мотивуючого середовища. На відміну від суто ігрового навчання (ігри або симуляції, створені спеціально для навчання), гейміфікація інтегрує окремі ігрові компоненти в реальні курси чи дисципліни. Вона використовується для покращення залученості та зацікавленості студентів, стимулювання їх до активної участі в навчанні та формування позитивного ставлення до навчального процесу. Ефективна гейміфікація передбачає наявність зрозумілих цілей, правил, системи прогресу, зворотного зв’язку та винагород, а також часто елементи співробітництва і змагання. У вищій освіті гейміфікація розглядається як інноваційний підхід для активізації навчання, який сприяє індивідуалізації освітнього досвіду та підвищенню якості навчання.

Ігрові технології дозволяють допомогти усвідомити складні концепти та принципи, а також навчитися розв'язувати проблеми та приймати рішення в нестандартних ситуаціях.

Треба зазначити, що термін «гейміфікація» різними дослідниками визначається по різному. В одному випадку, під цим терміном розуміють використання характерних для ігор елементів (механік, дизайну, правил) у навчальному процесі, що зазвичай не є грою. В [38] про гейміфікацію пишуть, що на відміну від суто ігрового навчання (ігри або симуляції, створені спеціально для

навчання), гейміфікація інтегрує окремі ігрові компоненти в реальні курси чи дисципліни, де метою є не розвага сама по собі, а підвищення зацікавленості студентів і покращення їхніх результатів шляхом використання природної мотивації від гри.

В іншому випадку до гейміфікації приписують як окремі ігрові елементи так і самі ігри з навчальними елементами (відеоігри чи настільні). Це пов'язано з тим, що ігри самі по собі приваблюють саме розважальною частиною, і виокремлення якогось елемента з системи гри, знецінює цей самий елемент, бо він працює тільки у зв'язці з іншими складовими ігрової системи. В [38] це називають «Серйозними іграми».

До прикладу можна навести бальну систему у закладах вищої освіти України, яку можна вважати як частину гейміфікації. Кількість балів відображає результативність. В іграх, за певну кількість балів, дають винагороду, як за працю яку гравець виконав і витратив час. В вищій освіті це проявлено в самій бальній системі, де, за певну кількість балів, студент може отримати грошову винагороду (стипендію). Тільки проблема в тому, що ця винагорода є чисто символічною і не мотивує студентів, бо, навіть, з фінансової точки зору вона замала. В іграх ця винагорода тісно пов'язана з подальшим розвитком гравця та гри і є певним рушієм. В системі освіти винагорода – це скоріш приємний бонус, який ні на що не впливає, бо незалежно від того, отримуєш ти її, чи ні – шлях завжди один для всіх.

Що ж до даної роботи, то під «гейміфікацією» буде розглянуто як елементи гри, так і навчальні ігри. Основні складові, які застосовуються в освітніх процесах, зазвичай, є елементами конкурування та заохочення, що включають [38]:

– бали та відзнаки, які учні отримують за виконання завдань, що стимулює їх до досягнення кращих результатів. Цінність винагороди безпосередньо впливає на зацікавленість учня. Прикладом може слугувати впровадження накопичувальної системи балів або очок, які учасник згодом може обміняти на реальний приз (звільнення від домашнього завдання на певний період, використання конспекту під час відповіді, вибір варіанту на контрольній роботі чи місця в класі тощо);

– прогресування через рівні або етапи надає учням відчуття досягнення та розвитку. Відстежування прогресу дозволяє контролювати ефективність навчального процесу, розуміти рівень засвоєння навчального матеріалу;

– взаємодія між студентами. Наприклад, змагання між учнями заохочує до навчання. Здорова конкуренція є зовнішнім стимулятором, який породжує певний азарт і сприяє розвитку лідерських якостей, наполегливості. Протилежно до цього є кооперація, що надає відчуття командної роботи та взаємодопомоги;

– використання сюжетних ліній та квестів перетворює освітній процес на більш цікаве, зрозуміле і залучене дійство. Перетворення учня зі спостерігача на учасника вигаданої чи реальної події створює ефект особливої значущості засвоєного матеріалу.

Однак сам підхід використання ігрових елементів має свої недоліки та переваги [42]. Із переваг можна визначити:

– ігри створюють емоційне занурення, привертають увагу та стимулюють бажання досягати результатів у навчанні;

– розвиток когнітивних навичок та вміння вирішувати проблеми. Ігрове навчання стимулює критичне мислення та аналітичні здібності, допомагаючи учням ефективно вирішувати завдання;

– ігрові методи сприяють соціалізації, розвитку комунікативних навичок та роботи в команді;

– ігрові підходи заохочують учнів до творчого мислення та нестандартних рішень.

Недоліки інтеграції ігрових технологій:

– ігрові елементи та ігри можуть відволікати від основного навчального матеріалу або взагалі демовувати студентів, через велику конкуренцію (приклад з балами);

– якщо гейміфікація реалізована поверхнево (лише як набір зовнішніх «призів»), вона ризикує стимулювати тільки зовнішню мотивацію заради нагород, а не знань;

– може вимагати значних технічних ресурсів та підтримки, як зі сторони того, хто навчається так і закладу;

– висока вартість розробки та придбання навчальних ігор;

– не всі ігри відповідають навчальним цілям, та навпаки, не всі предмети з учбової програми можна прив'язати до відеоігор;

– складність розробки ігор або інтеграції окремих ігрових елементів у вже існуючу систему освіти Україні, яка не передбачає у собі можливість інтеграції новітніх методів навчання;

– викладач має бути підготовленим для впровадження ігрових елементів у навчальний процес. І такі проблеми, як нестача часу, відсутність бажання, проблеми з керуванням групою або просто недостатня підготовка, стає перешкодою впровадженню ігрового навчання.

Існує декілька підходів до гейміфікації освітнього процесу [43]:

– структурна гейміфікація, що полягає в додаванні окремих ігрових елементів до наявного навчального контенту (система балів, нагород, рейтингів), не змінюючи сам зміст курсу;

– змістова гейміфікація – глибше інтегрує гру в сам навчальний матеріал, наприклад, подає матеріал у формі місій або сценаріїв, де засвоєння знань відбувається через проходження сюжету.

У практиці вищої школи наразі переважає структурна гейміфікація –системи балів і нагород зі звичайними завданнями, адже вона простіша у реалізації.

Незважаючи на заявлені переваги ігор, включення їх у навчальну програму в Україні є досить складним. Окрім вищезазначених проблем, існує також проблема відсутності конкретних методів інтеграції ігор у сам навчальний процес з урахуванням особливостей мови та культури.

Що ж до навчальних ігор, а не окремих їх елементів, то прикладом впровадження такого є відеогра Minecraft – відеогра в жанрі пісочниці у відкритому світі з поглядом від першої/третьої особи, розроблена та видана Mojang Studios у 2011 році. А саме версія «Minecraft: Education Edition», що є освітньою

версією гри, спеціально розробленою для використання у учбових класах і була розроблена пізніше. Має функції, створені спеціально для навчальних середовищ, для підтримки співпраці, оцінювання, програмування тощо.

[44] Лідером у використанні «Minecraft: Education Edition» є Джеймс Протеро, викладач у Уельсі. Він ініціював використання гри разом зі своїми колегами, розпочавши проєкт під назвою «Дивовижна архітектура», де студенти реалізують свої будівлі. Студенти досліджують будівлі своїх колег з погляду архітектури і дизайну та планують або покращують власні будівельні проєкти.

Дуже часто використовують ігри симулятори для навчання. Наприклад, [45] Медичний коледж Джорджії об’єдналася з ігровою компанією BreakAway, щоб створити симулятор, який дозволяє студентам-стоматологам вдосконалювати свої навички, не практикуючись на реальних пацієнтах.

Для інтеграції ігрового елементу в освітній процес можна використовувати вже існуючі інструменти та платформи. Приклади найпопулярніших з них: Duolingo, Classcraft, Kahoot!, плагіни Moodle, такі як Quizventure та Activities.

Висновки до розділу 1:

1. Було проведено аналіз вищої освіти України, зацікавленість в ній держави та представлені фінансові показники за минулі роки. Освіта в Україні постійно розвивається, адаптується та підтримується державою, тому реалізація проєкту в даній сфері діяльності є перспективною;

2. Існує багато різних життєвих циклів та підходів до розробки програмного забезпечення. У кожного з них є свої переваги і недоліки, тому важливо обрати правильний;

3. У даному пункті було розглянуто загальні технології для розробки програмного забезпечення та технології для розробки відеоігор. Із згаданих в розділі інструментів, в проєкті використовуються: GitHub, Visual Studio Code, Adobe Photoshop, CorelDRAW, Discord та ігровий рушій Godot. Також роздивились підходи гейміфікації для чіткого розуміння кінцевої мети проєкту розробки програмного додатку.

РОЗДІЛ 2.
ПЛАНУВАННЯ ПРОЄКТУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз середовища проєкту розробки

Середовище проєкту – це чинники, які діють на проєкт і його успішність. З джерела [46] виділяють зацікавлені сторони як чинника, що впливають на проєкт та мають певні очікування та роль. Для даного проєкту зацікавлені сторони представлені в табл. 2.1.

Табл. 2.1

Зацікавлені сторони проєкту та їхні очікування

Зацікавлена сторона	Очікування	Вплив на проєкт
Керівник проєкту (CEO)	Успішне управління, дотримання строків та бюджету, якість продукту	– контролює хід виконання, ухвалює рішення, забезпечує ресурсами
Команда розробки	Чіткі вимоги, злагоджена співпраця, зручний процес розробки	– розробку продукту; – якість продукту; – успішне впровадження і просування продукту; – тестує та налагоджує систему.
Консультант з освітніх елементів	Відповідність гри освітнім стандартам, ефективність навчання	– впливає на освітній зміст гри, механіки гейміфікації та освітню цінність;
Інвестори, грантові організації	Виконання проєкту в межах бюджету, конкурентний продукт	– забезпечують фінансування продукту; – додатковий моніторинг якості продукту;

Закінчення табл. 2.1

Зацікавлені сторони проєкту та їхні очікування

Зацікавлена сторона	Очікування	Вплив на проєкт
Користувачі (студенти, викладачі, учасники навчального процесу)	Цікава, корисна гра, зручність використання, адаптація до навчального процесу	– використовують продукт та надають відгуки; – залежно від відгуків, впливають за подальше успішне впровадження; – є фактором формування функціональних та нефункціональних вимоги до додатку, бо є цільовою аудиторією;
Навчальні заклади, освітні організації	Можливість інтеграції гри в навчальні програми	– визначають масштаб впровадження та поширення гри – перевірка відповідності продукту цілям – створює можливості для якісної реалізації продукту
Регуляторні органи, стандартизаційні комітети (ISO, IEEE, державні установи)	Відповідність стандартам якості, безпеки, авторських прав	– впливають через вимоги до сертифікації та відповідності нормативам

Як видно з таблиці, найбільший вплив на проєкт мають керівник проєкту, команда розробки, фінансуючи сторони. Частково сюди відносяться і користувачі, а саме викладачів, бо вони, у ролі консультантів, надають відгуки щодо впровадження або корегування освітніх елементів у грі.

Користувачі та навчальні заклади визначають успіх продукту на ринку та можливість його подальшого впровадження. Зі сторони користувачів проєкт отримує відгуки, а зі сторони навчальних закладів вищої освіти відгуки та реалізацію проєкту шляхом надання майданчику для розгортки та впровадження.

Регуляторні органи важливі для сертифікації та відповідності стандартам. Наявність документів про проходження сертифікації на відповідність вимогам якості продукту підвищує довіру до проєкту зі сторони користувачів та впевненість інвесторів в якості самого проєкту.

Також було проаналізовано рівень впливу та залученості кожної зі сторін у проєкт, що продемонстровано на діаграмі рис. 2.1.

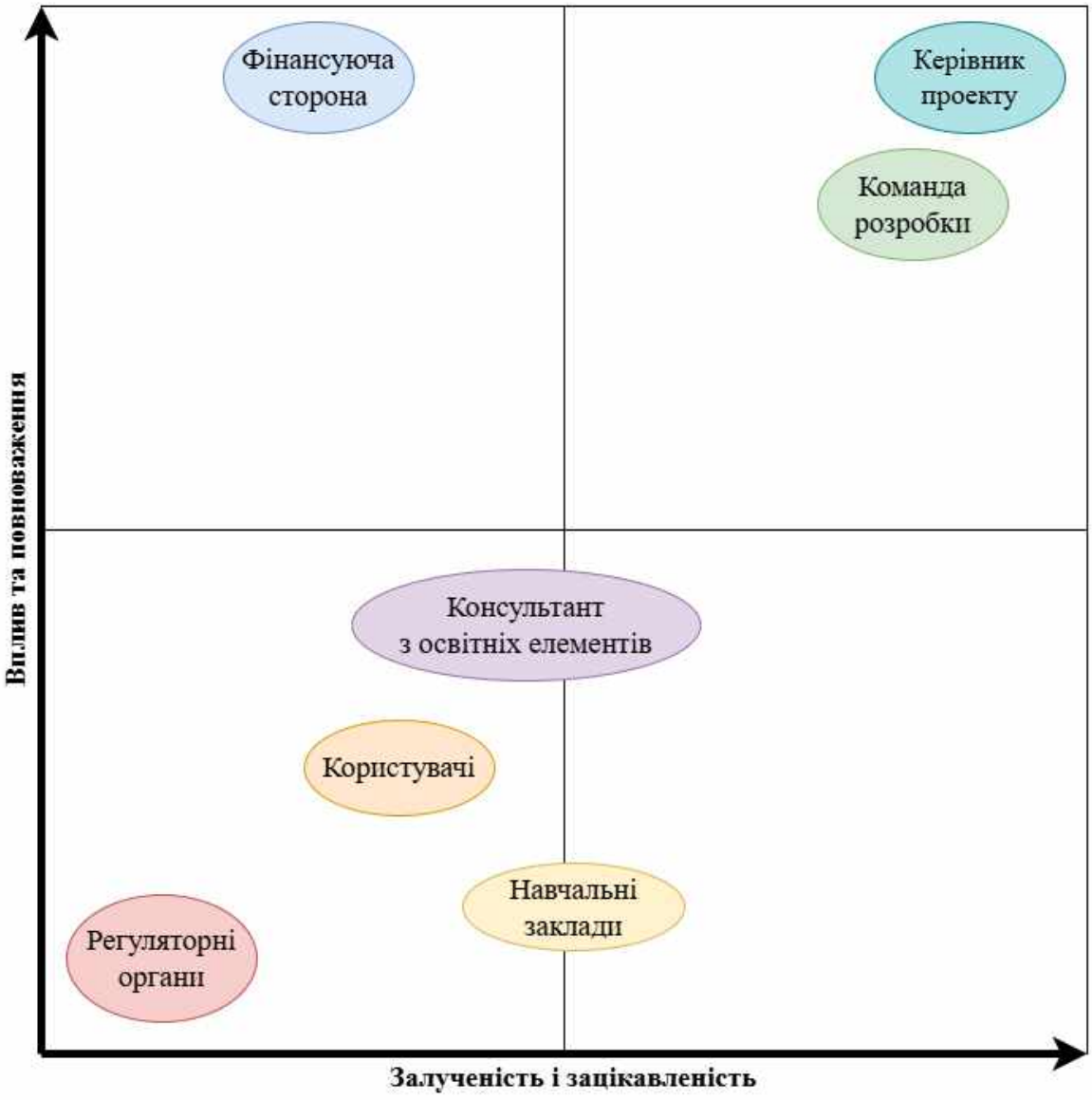


Рис. 2.1. Діаграма аналізу зацікавлених сторін.

Керівник проєкту залучений на всіх стадіях роботи проєкту. Він керує проєктом, має для цього повноваження та зацікавлений у результатах, бо успіх проєкту – його відповідальність.

Команда розробки це серце проєкту, завдяки якому досягаються основні цілі. Однак, команда має менші повноважень на відміну від керівника, бо основні рішення все ж таки приймає він.

Консультант з освітніх елементів, крім частини інтеграції освітніх функцій більше ніяк не впливає на проєкт. Сам він зацікавлений в успішності проєкту опосередковано, бо його основна ціль – вдала інтеграція освітніх елементів. Від цього залежить його репутація на ринку праці як фахівця.

Інвестори має високий вплив на проєкт. Від фінансування залежить якість проєкту і досяжність кінцевої мети. Зазвичай, зацікавленість інвесторів обмежується прибутком проєкту, тому, залученість у проєкт зводиться лише до дотримання зазначених вимог. Треба зазначити, що основним результатом проєкту є зміни в освітніх методиках навчання та підвищення зацікавленості студентів у процесі навчання. Тому, прибуток є другорядною ціллю, але в той же час не менш важливою.

Користувачі ж (студенти, викладачі, учасники навчального процесу) залучені у проєкт як непрофесійні тестувальники, де їхній вплив обмежується відгуками відносно продукту, під час його тестування.

Навчальні заклади та освітні організації не впливають на проєкт на етапах розробки, однак при подальшому впроваджені ЗВО зможуть консультувати проєкт для подальшого його розвитку та покращення, бо вони зацікавлені у впровадженні нових технологій та відповідальні за подальше впровадження і реалізацію проєкту у свій учбовий план.

Регуляторні органи і стандартизаційні комітети не впливають на проєкт, однак обмежують його через стандарти якості, які вони встановлюють. Проходження сертифікації на дотримання стандартів впевнить майбутніх користувачів та інвесторів у якості кінцевого продукту. Однак регуляторні органи мало зацікавлені у самому проєкті, бо їх основна мета – дотримання стандартів.

Серед інструментів аналізу середовища проєкту є інструменти стратегічного аналізу, які дозволяють оцінити середовище проєкту на стратегічному рівні та приймати зважені рішення, бо дозволяють виявити ризики, можливості, загрози та зовнішні впливи під час планування проєкту. Серед таких інструментів є PEST та SWOT аналізи.

PEST-аналіз [47] розшифровується як: P – Political (політичні фактори); E – Economic (економічні фактори); S – Social (соціальні фактори); T – Technological (технологічні фактори). PEST-аналіз оцінює зовнішнє середовище, яке не контролюється проєктною командою, але впливає на реалізацію проєкту. Цей аналіз дає розуміння зовнішніх загроз і можливостей, допомагає адаптувати стратегію до змін зовнішнього середовища та підвищує готовність до непередбачуваних обставин. PEST-аналіз представлений у табл. 2.2.

SWOT [48] розшифровується як: S – Strengths (сильні сторони); W – Weaknesses (слабкі сторони); O – Opportunities (можливості); T – Threats (загрози).

SWOT-аналіз же, крім зовнішнього середовища, аналізує ще і внутрішнє, щоб знайти баланс між можливостями та ризиками, з урахуванням внутрішніх сильних та слабких сторін. Він допомагає визначити сильні сторони проєкту для їх використання та слабкі для їх компенсування. Застосування SWOT-аналізу відкриває можливості для розвитку проєкту та створенню плану дій для уникнення загроз. SWOT-аналіз представлений у табл. 2.3.

Табл. 2.2

PEST аналіз проєкту

Фактор	Вплив
Політичні (Political)	– високий рівень корупції призведе до недостатнього фінансування проєкту для його успішного завершення; – податкова політика негативно впливає на розвиток малого та середнього бізнесу; – військові дії негативно впливають на кількість студентів ЗВО, що призводить до зменшення релевантності проєкту; – антимонопольне та трудове законодавство позитивно впливає на ринок IT продуктів, стимулюючи конкуренцію між компаніями; – зміни в законодавстві, що регулює правила роботи в галузі призведе до затримок у реалізації проєкту; – підтримка інноваційних компаній з боку держави підвищить якість та швидкість виконання проєкту.
Економічні (Economic)	– значний рівень інфляції призведе до збільшення незапланованих витрат на проєкт; – нестабільний курс основних валют призведе до проблем при плануванні бюджету проєкту; – високий рівень глобалізації та відкритості економіки допоможе просувати продукт на міжнародний ринок; – цінова конкуренція з боку зарубіжних компаній позитивно сприяє на розповсюдження продукту нашого проєкту; – позитивний інвестиційний клімат в галузі полегшить отримання фінансування для проєкту.

Закінчення табл. 2.2

PEST аналіз проєкту

Фактор	Вплив
Соціальні (Social)	– високий рівень підготовки молодих спеціалістів сприятиме полегшенню пошуку співробітників для реалізації проєкту; – високі вимоги до якості продукції та рівня сервісу призведуть до збільшення часових та грошових затрат на проєкт через необхідність детальнішого процесу контролю якості; – високий рівень еміграції серед українського населення негативно впливає на релевантність продукту проєкту через зменшення кількості потенційних студентів ЗВО; – розвиток релігії та інших вірувань мінімально впливає на наш проєкт; – підтримка вітчизняного виробника продуктів і послуг позитивно впливає на зацікавленість користувачів; – швидкі темпи росту населення позитивно вплинуть на ріст аудиторії нашого проєкту.
Технологічні (Technological)	– високий рівень інновації та технічного розвитку галузі покращить шанси на успішне фінансування проєкту; – висока ступінь інтеграції технологій допоможе спростити впровадження продукту у ЗВО; – доступ до новітніх технологій гейміфікації допоможе користувачам швидше опанувати дисципліну.

Табл. 2.3

SWOT аналіз проєкту

Фактор	Опис
Сильні сторони (Strength)	– навчальні елементи роблять гру корисною, а не лише розважальною; – гра працює повністю офлайн, що знижує технічні вимоги; – гнучка архітектура гри – використання Godot + C# дозволяє масштабувати та адаптувати проєкт у майбутньому; – гра збиратиме статистику ігрового процесу для покращення ігрових та освітніх механік; – простий та інтуїтивно зрозумілий інтерфейс; – невисокі системні вимоги дозволять запускати гру навіть на слабких ПК; – Вмотивована команда проєкту.
Слабкі сторони (Weaknesses)	– обмеженість ресурсів вплине на швидкість розробки; – відсутність фінансування на маркетинг або подальшого масштабування; – відсутність підтримки мобільних пристроїв (Android, iOS), що зменшує охоплення аудиторії; – недостатнє балансування освітніх та ігрових механік а) освітня гра повинна залишатися цікавою; б) потрібно забезпечити рівновагу між складністю і навчальною ефективністю. – гра не має мультиплеєра або інтеграції з соціальними мережами; – оптимізація гри під різні конфігурації Windows може викликати труднощі.

Продовження табл. 2.3

SWOT аналіз проєкту

Фактор	Опис
Можливості (Opportunities)	– запуск франшизи; – масштабування та доповнення продукту через додавання нових навчальних модулів, додаткових сценарії, карток, механік; – зростання попиту на освітні ігри розширить цільову аудиторії, що створює можливості для співпраці з навчальними закладами в Україні та за її кордоном.; – при достатній кількості ресурсів – випустити мобільну версію; – запропонувати гру як інструмент для тренінгів із управління проєктами; – потенційна співпраця з університетами або онлайн-курсами; – аналіз ігрових сесій дозволить покращити механіки та зробити гру ефективнішою; – використання статистики для доказу ефективності навчального підходу; – формування позитивної репутації команди; – можливість реклами через партнерські вузи.

Закінчення табл. 2.3

SWOT аналіз проєкту

Фактор	Опис
Загрози (Threats)	– плагіат контенту або гри цілком; – зміна в тарифних планах використовуваних інструментів (ПО); – важко розраховувати на органічне зростання популярності без маркетингової підтримки; – конкуренція в сфері освітніх ігор; – без якісного просування важко залучити користувачів; – освітні ігри викликають менший інтерес, ніж розважальні; – якщо механіки будуть занадто складними або нудними, гравці швидко втратять мотивацію; – відсутність онлайн-функціоналу обмежує можливість отримання відгуків від гравців; – Обмежена аудиторія через відсутність мобільних версій додатку; – затримка в розробці; – без активного тестування на аудиторії складно зрозуміти, що потрібно покращити; – можливі юридичні ризики: а) необхідно враховувати правила щодо збору та збереження ігрових даних; б) використання стороннього контенту (звуки, шрифти, графіка) може викликати проблеми з авторським правом.

2.2. Вибір моделі життєвого циклу проєкту

У пункті 1.2.1 першого розділу було проаналізовано різні моделі життєвих циклів для проєктів з розробки програмного продукту, а в пункті 1.2.2 основні методології. Серед проаналізованих моделей, для управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес була обрана методологія Agile, і, відповідно, ітеративна модель життєвого циклу. Обраним підходом у Agile став Scrum.

Основними критеріями для вибору Agile, були можливість створення працюючого додатку у короткі терміни та введення змін до вимог на будь-якому з етапів розробки. Завдяки цьому, перша версія працюючого продукту планується бути готовою до тестування вже після другої ітерації, а після четвертої повинно бути фінальне тестування.

Внесення змін до вимог або самого продукту після тестувань не буде викликати труднощі. На початку кожної ітерації проводився перегляд вимог до продукту з залученням зацікавлених сторін, а в кінці проводились тести розроблених в межах даної ітерації функціоналу, що дозволяло фокусуватись ще і на якості самого продукту. Самі ж ітерації поділені на спринти.

Що ж до підходу Agile Scrum – він представлений у проєкті наступними практиками: спринти, мова про яких йшла вище; артефактами, у вигляді беклогу продукту та беклогу спринтів; зустрічі, як щоденні, так і зустрічі для огляду спринтів та ітерацій продукту.

Загалом, розробка складається з п’яти ітерацій. Структура життєвого циклу проєкту містить наступні фази:

1. Аналіз ринку;
2. Ініціалізація;
3. Планування та проєктування;
4. Організація проєкту;
5. Розробка 1 версії додатку;
6. Розробка 2 версії додатку;
7. Тестування базової версії;

8. Розробка 3 версії додатку (введення змін на основі даних тестування);
9. Розробка 4 версії додатку;
10. Тестування кінцевої версії;
11. Розробка 5 версії додатку (введення змін на основі даних тестування);
12. Презентація продукту;
13. Завершення проєкту.

2.3. Статут та календарно-мережевий графік виконання проєкту (в MS Project)

2.3.1. Статут проєкту розробки ПЗ

Статут проєкту [13, 49] – це розроблений замовником та керівником проєкту документ, що формально затверджує існування проєкту та надає керівнику проєкту повноваження використовувати ресурси організації для операцій проєкту. Статут ініціює проєкт та встановлює відносини між виконавцем і замовником. Він описує потреби організації, допущення, обмеження, а також новий продукт або результат, який планують створити. Статут включає такі розділи [49]:

– мета проєкту або обґрунтування;

– попередній опис проєкту;

– вимоги проєкту, що виражаються в характеристиках продукту та його функцій для задоволення потреб та очікувань зацікавлених сторін;

– цілі проєкту, а точніше:

а) за змістом, що, описують запланований прибуток від проєкту;

б) часу, описують завершення проєкту;

в) витрат. Наприклад: витрати на проєкт не перевищить 7 500 000 грн.;

г) додаткові, які визначені організацією, наприклад цілі якості або безпеки.

– ризики високого рівня, що означають ризики на рівні проєкту, такі як наявність фінансування, технологій, ресурсів та інше;

– розклад за віхами – ключові дати в проєкті;

– сумарний бюджет, що виражається у передбачуваному діапазоні витрат для проєкту;

– список зацікавлених у проєкті сторін;

– вимоги щодо схвалення або затвердження проєкту;

– призначений керівник проєкту, та його відповідальність та рівень авторитету;

– назва та повноваження спонсора або особи, яка уповноважує проєкт статуту.

Для управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес статут представлений в «Додатку А».

2.3.2. Мережевий графік проєкту

Мережеве планування [50] – це одна з форм графічного відображення змісту робіт і тривалості виконання стратегічних планів і довгострокових комплексів проєктних, планових, організаційних та інших видів діяльності підприємств. Широко використовується при розробці планів, для контролю за виконанням проєктів, раціонального використання ресурсів, координації взаємодії між керівниками та виконавцями проєктів тощо. Методи мережевого планування та управління використовуються при плануванні складних комплексних проєктів, наприклад, при будівництві та реконструкції будь-яких об’єктів, виконанні науково-дослідних та конструкторських робіт, розробці відомчих програм, підготовці та освоєнні нових видів продукції, розробці родовищ корисних копалин, ремонті обладнання та інших.

Поряд з лінійними графіками та табличними розрахунками мережеві методи планування знаходять широке застосування при розробці перспективних планів та моделей створення складних виробничих систем та інших об'єктів довгострокового використання. Мережеві плани робіт підприємств по створенню нової конкурентоздатної продукції містять не тільки загальну тривалість всього комплексу проєктно-виробничої та фінансово-економічної діяльності, але й тривалість та послідовність здійснення окремих процесів чи етапів, а також потреби необхідних економічних ресурсів.

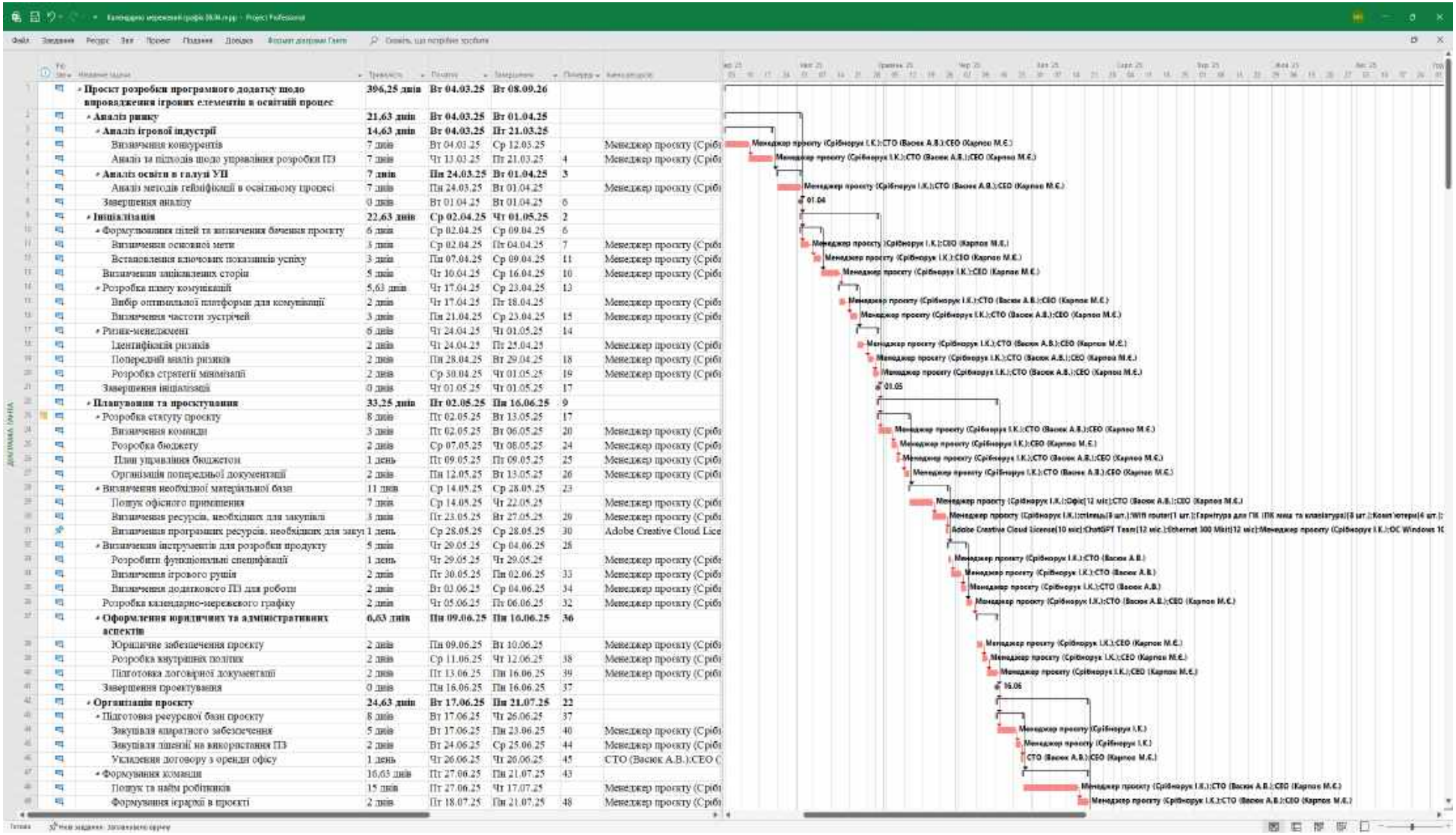
Мережева модель – це економіко-математична модель, що відображає комплекс робіт та подій та регламентує в просторі і часі їх реалізацію. Вона описує процес досягнення поставленої цілі в реалізації деякого проєкту через відображення технологічної та логічної послідовності подій та зв’язків між ними.

Важливим є поняття критичного шляху, який складається з дуг, що з’єднують події з нульовим резервом часу. Кожна мережева модель має хоча б один критичний шлях, який визначає мінімальний термін виконання проєкту.

Зменшення терміну виконання операцій, розташованих на критичному шляху, забезпечують зменшення тривалості виконання усього комплексу операцій.

Цю властивість критичного шляху використовують у плануванні і управлінні реалізацією комплексу операцій, коли, намагаючись зменшити тривалість комплексу, розподіляють ресурси передусім на виконання операцій критичного шляху, а також у режимі оперативного управління, де особливу увагу приділено операціям критичного шляху. На мережевих моделях розв’язуються задачі оптимізації термінів реалізації комплексу операцій, оптимізації розподілу і використання ресурсів. Мережева модель є хорошим інструментом імітації і використовується з метою прогнозування реалізації комплексу операцій, раціональних стратегій управління. Багатомережеві системи планування і управління дають змогу керувати основною виробничою діяльністю економічного об’єкта, виходячи з раціонального використання наявних ресурсів. У промисловості та будівництві багатомережеві системи відіграють важливу роль як засіб координації дій різноманітних учасників виробничого процесу.

Для нашого проєкту було розроблено календарно-мережевій графік у застосунку Microsoft Project, на якому показано повний перелік робіт, необхідний для успішного завершення проєкту та виділено критичний щлях проєкту.



	Назва задачі	Тривалість	Початок	Завершення	Попередники	Виконавці
1	**Проєкт розробки програмного додатку щодо впровадження ігрових елементів в освітній процес**	**396,25 днів**	**Вт 04.03.25**	**Вт 08.09.26**		
2	**Аналіз ринку**	**21,63 днів**	**Вт 04.03.25**	**Вт 01.04.25**		
3	**Аналіз ігрової індустрії**	**14,63 днів**	**Вт 04.03.25**	**Пт 21.03.25**		
4	Визначення конкурентів	7 днів	Вт 04.03.25	Ср 12.03.25		Менеджер проєкту (Срібн
5	Аналіз та підходів щодо управління розробки ПЗ	7 днів	Чт 13.03.25	Пт 21.03.25	4	Менеджер проєкту (Срібн
6	**Аналіз освіти в галузі УП**	**7 днів**	**Пн 24.03.25**	**Вт 01.04.25**	**3**	
7	Аналіз методів гейміфікації в освітньому процесі	7 днів	Пн 24.03.25	Вт 01.04.25		Менеджер проєкту (Срібн
8	Завершення аналізу	0 днів	Вт 01.04.25	Вт 01.04.25	6	
9	**Ініціалізація**	**22,63 днів**	**Ср 02.04.25**	**Чт 01.05.25**	**2**	
10	Формулювання цілей та визначення бачення проєкту	6 днів	Ср 02.04.25	Ср 09.04.25	6	
11	Визначення основної мети	3 днів	Ср 02.04.25	Пт 04.04.25	7	Менеджер проєкту (Срібн
12	Встановлення ключових показників успіху	3 днів	Пн 07.04.25	Ср 09.04.25	11	Менеджер проєкту (Срібн
13	Визначення зацікавлених сторін	5 днів	Чт 10.04.25	Ср 16.04.25	10	Менеджер проєкту (Срібн
14	Розробка плану комунікацій	5,63 днів	Чт 17.04.25	Ср 23.04.25	13	
15	Вибір оптимальної платформи для комунікації	2 днів	Чт 17.04.25	Пт 18.04.25		Менеджер проєкту (Срібн
16	Визначення частоти зустрічей	3 днів	Пн 21.04.25	Ср 23.04.25	15	Менеджер проєкту (Срібн
17	Ризик-менеджмент	6 днів	Чт 24.04.25	Чт 01.05.25	14	
18	Ідентифікація ризиків	2 днів	Чт 24.04.25	Пт 25.04.25		Менеджер проєкту (Срібн
19	Попередній аналіз ризиків	2 днів	Пн 28.04.25	Вт 29.04.25	18	Менеджер проєкту (Срібн
20	Розробка стратегії мінімізації	2 днів	Ср 30.04.25	Чт 01.05.25	19	Менеджер проєкту (Срібн
21	Завершення ініціалізації	0 днів	Чт 01.05.25	Чт 01.05.25	17	
22	**Планування та проєктування**	**33,25 днів**	**Пт 02.05.25**	**Пн 16.06.25**	**9**	
23	Розробка статуту проєкту	8 днів	Пт 02.05.25	Вт 13.05.25	17	
24	Визначення команди	3 днів	Пт 02.05.25	Вт 06.05.25	20	Менеджер проєкту (Срібн
25	Розробка бюджету	2 днів	Ср 07.05.25	Чт 08.05.25	24	Менеджер проєкту (Срібн
26	План управління бюджетом	1 день	Пт 09.05.25	Пт 09.05.25	25	Менеджер проєкту (Срібн
27	Організація попередньої документації	2 днів	Пн 12.05.25	Вт 13.05.25	26	Менеджер проєкту (Срібн
28	Визначення необхідної матеріальної бази	11 днів	Ср 14.05.25	Ср 28.05.25	23	
29	Пошук офісного приміщення	7 днів	Ср 14.05.25	Чт 22.05.25		Менеджер проєкту (Срібн
30	Визначення ресурсів, необхідних для закупівлі	3 днів	Пт 23.05.25	Вт 27.05.25	29	Менеджер проєкту (Срібн
31	Визначення програмних ресурсів, необхідних для закуп	1 день	Ср 28.05.25	Ср 28.05.25	30	Adobe Creative Cloud Lice
32	Визначення інструментів для розробки продукту	5 днів	Чт 29.05.25	Ср 04.06.25	28	
33	Розробити функціональні специфікації	1 день	Чт 29.05.25	Чт 29.05.25		Менеджер проєкту (Срібн
34	Визначення ігрового рушія	2 днів	Пт 30.05.25	Пн 02.06.25	33	Менеджер проєкту (Срібн
35	Визначення додаткового ПЗ для роботи	2 днів	Вт 03.06.25	Ср 04.06.25	34	Менеджер проєкту (Срібн
36	Розробка календарно-мережевого графіку	2 днів	Чт 05.06.25	Пт 06.06.25	32	Менеджер проєкту (Срібн
37	**Оформлення юридичних та адміністративних аспектів**	**6,63 днів**	**Пн 09.06.25**	**Пн 16.06.25**	**36**	
38	Юридичне забезпечення проєкту	2 днів	Пн 09.06.25	Вт 10.06.25		Менеджер проєкту (Срібн
39	Розробка внутрішніх політик	2 днів	Ср 11.06.25	Чт 12.06.25	38	Менеджер проєкту (Срібн
40	Підготовка договірної документації	2 днів	Пт 13.06.25	Пн 16.06.25	39	Менеджер проєкту (Срібн
41	Завершення проєктування	0 днів	Пн 16.06.25	Пн 16.06.25	37	
42	**Організація проєкту**	**24,63 днів**	**Вт 17.06.25**	**Пн 21.07.25**	**22**	
43	Підготовка ресурсної бази проєкту	8 днів	Вт 17.06.25	Чт 26.06.25	37	
44	Закупівля апаратного забезпечення	5 днів	Вт 17.06.25	Пн 23.06.25	40	Менеджер проєкту (Срібн
45	Закупівля ліцензії на використання ПЗ	2 днів	Вт 24.06.25	Ср 25.06.25	44	Менеджер проєкту (Срібн
46	Укладення договору з оренди офісу	1 день	Чт 26.06.25	Чт 26.06.25	45	СТО (Васюк А.В.);CEO (
47	Формування команди	16,63 днів	Пт 27.06.25	Пн 21.07.25	43	
48	Пошук та найм робітників	15 днів	Пт 27.06.25	Чт 17.07.25		Менеджер проєкту (Срібн
49	Формування ієрархії в проєкті	2 днів	Пт 18.07.25	Пн 21.07.25	48	Менеджер проєкту (Срібн

Рис. 2.2. Календарно-мережевий графік

2.4. Фінансові показники проєкту (в MS Project)

2.4.1. Матеріальні ресурси

Матеріальними ресурсами називають основні та оборотні засоби виробництва, які використовуються або можуть бути використані у виробничому процесі та формують його матеріально-речову базу. На рисунках 2.3 – 2.4 представлено розподілення витрат на всі ресурси проєкту. Окремо витрати представлені на табл. 2.4 на матеріальні ресурси, табл. 2.6 трудові ресурси та на табл. 2.7 разові витрати.

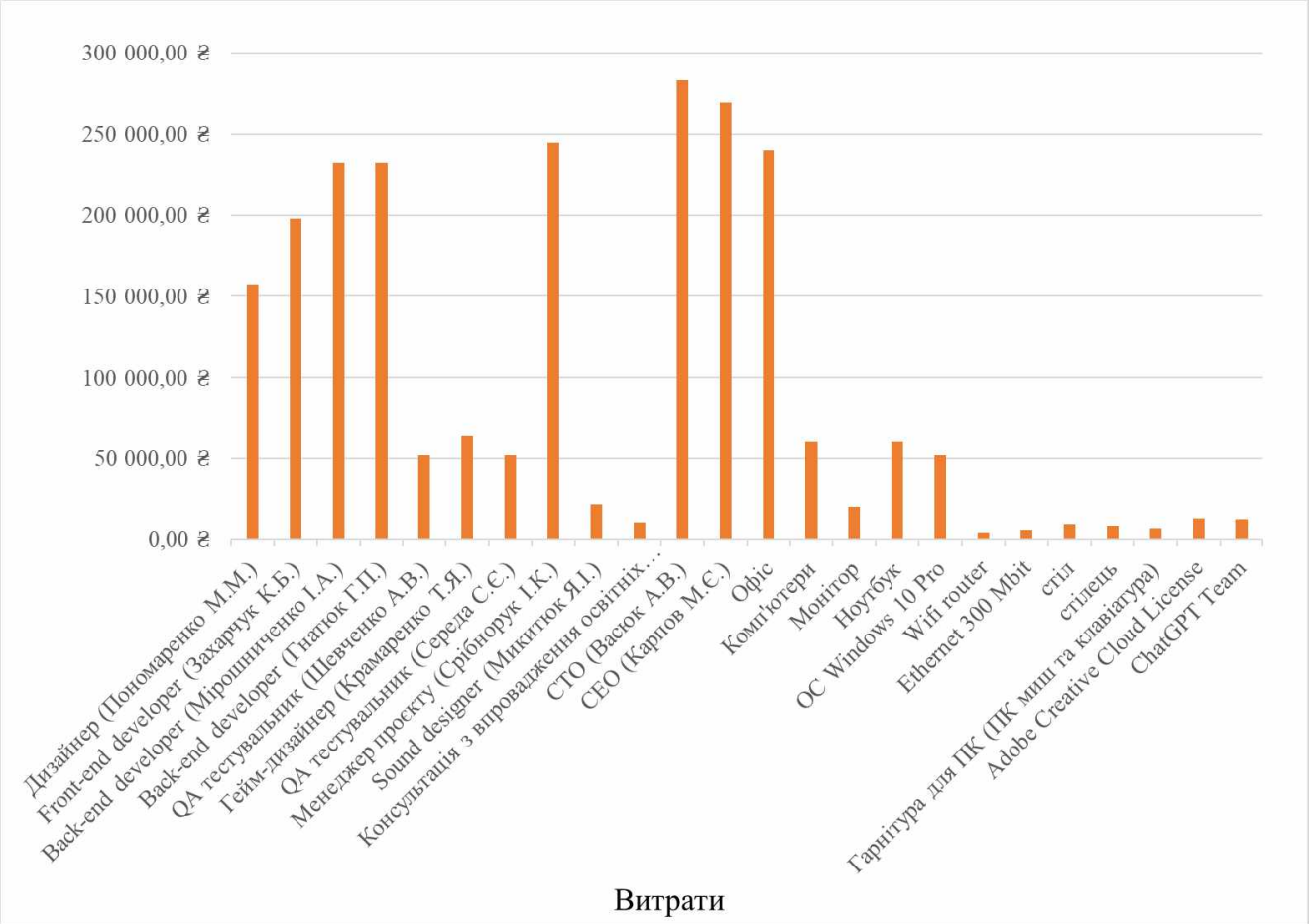


Рис. 2.3. Графік розподілення витрат на ресурси проєкту

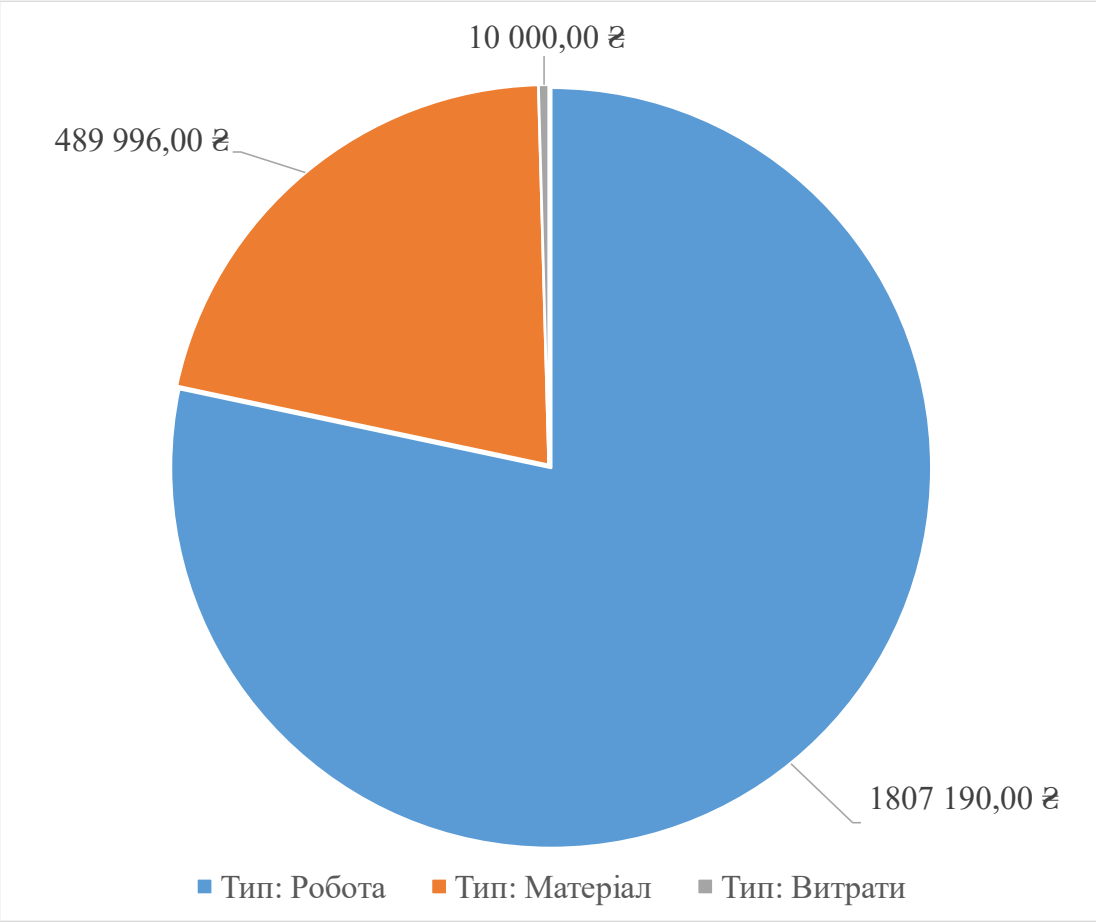


Рис. 2.4. Розподілення витрат на ресурси проєкту за категоріями

Таблиця 2.4

Перелік матеріальних ресурсів

№ п/п	Назва	од. виміру	Кількість	Ціна за одиницю, грн (₴)	Сума, грн (₴)
1	Офіс (оренда)	міс.	12	20 000,00	240 000
2	Комп'ютер	шт.	4	15 000,00	60 000
3	Монітор	шт.	4	5 000,00	20 000
4	Ноутбук	шт.	4	15 000,00	60 000
5	ОС Windows 10 Pro	копія	8	6 500,00	52 000
6	Wifi router	шт.	1	4 000,00	4 000
7	Ethernet 300 Mbit	міс.	12	450,00	5 400

Закінчення таблиці 2.4

Перелік матеріальних ресурсів

№ п/п	Назва	од. виміру	Кількість	Ціна за одиницю, грн (₴)	Сума, грн (₴)
8	Стіл	шт.	8	1 100,00	8 800
9	Стілець	шт.	8	1 000,00	8 000
10	Гарнітура для персонального комп'ютеру (миш та клавіатура)	шт.	8	800,00	6 400
11	Adobe Creative Cloud License	міс.	10	1 300,00	13 000
12	ChatGPT Team	міс.	12	1033,00	12 396,00
Всього					489 996

2.4.2 Розрахунок заробітної плати

Заробітна плата [51] – це винагорода або заробіток, обчислений у грошовому виразі, який за трудовим договором роботодавець сплачує працівникові за роботу, яку виконано або має бути виконано. Для трудових ресурсів, заробітна плата визначалась як середня за ринком праці.

Існує дві форми оплати праці [52]: відрядна та погодинна.

Системи відрядної форми оплати праці:

– пряма відрядна – оплата праці виконується за кожну одиницю випущеної продукції певної якості з урахуванням складності й умов праці;

– непряма відрядна – заробіток допоміжного персоналу, що визначається залежно від результатів праці основних працівників, яких він обслуговує;

– відрядно-прогресивна – заробіток розрахований за фактичний виробіток у межах норми та виробіток понад норму, що розраховується за підвищеними розцінками;

– відрядно-преміальна – крім заробітку по прямим відрядним розцінкам працівник додатково отримує премію за певні якісні та кількісні показники;

– акордна – розмір оплати праці встановлюється виходячи з діючих норм виробітку й відрядних розцінок на весь комплекс робіт;

– акордно-преміальна – акордна система, доповнена виплатою премій, встановлених за виконання й перевиконання кількісних/якісних показників.

Використовуються дві системи погодинної оплати:

– проста погодинна – сума заробітку визначається його тарифною ставкою та фактично відпрацьованим часом;

– погодинно-преміальна – понад заробіток по тарифній ставці працівник додатково отримує премію за виконання визначених показників.

Також є посадовий оклад – абсолютний розмір заробітної плати за місяць.

У проєкті використовується погодинна система оплати праці. Вона представлена двома ставками: звичайна погодинна та понаднормова.

Таблиця 2.5

Заробітна плата працівників

Людські ресурси	Стандартна ставка (грн/год)	Понаднормова ставка (грн/год)
Дизайнер	156	200
Front-end developer	156	200
Back-end developer	175	210
QA тестувальник	125	150
Гейм-дизайнер	156	200
Менеджер проєкту	218	350
Sound designer	125	150
CTO	220	350
CEO	220	350

Таблиця 2.6

Розрахунок витрат на трудові (людські) ресурси за стандартною ставкою

Людські ресурси	Розрахована робота (год)	Загальна витрати (грн)
Дизайнер	1 008	157 500
Front-end developer	1 264	197 500
Back-end developer-1	1 328	232 400
Back-end developer-2	1 328	232 400
QA тестувальник-1	416	52 000
QA тестувальник-2	416	52 000
Гейм-дизайнер	408	63 750
Менеджер проєкту	1 112	243 250
Sound designer	176	22 000
CTO	1 288	283 360
CEO	1 224	269 280
Всього	9968	1 805 440

Таблиця 2.7

Разові витрати

Разові витрати	Витрати (грн)
Консультація з впровадження освітніх елементів	10 000,00

Треба зазначити, що витрати за комунальні послуги (вода, опалення, електроенергія та ін.) враховано в оренду офісу. Витрати на утримання і експлуатацію обладнання, за необхідності, ремонт, входять у додаткові витрати.

На виконання проєкту розраховано 12 місяців та ще 6 місяців додатково як буфер. Всього 18 місяців.

Був розроблений календар для проєкту, представлений рис. 2.5.

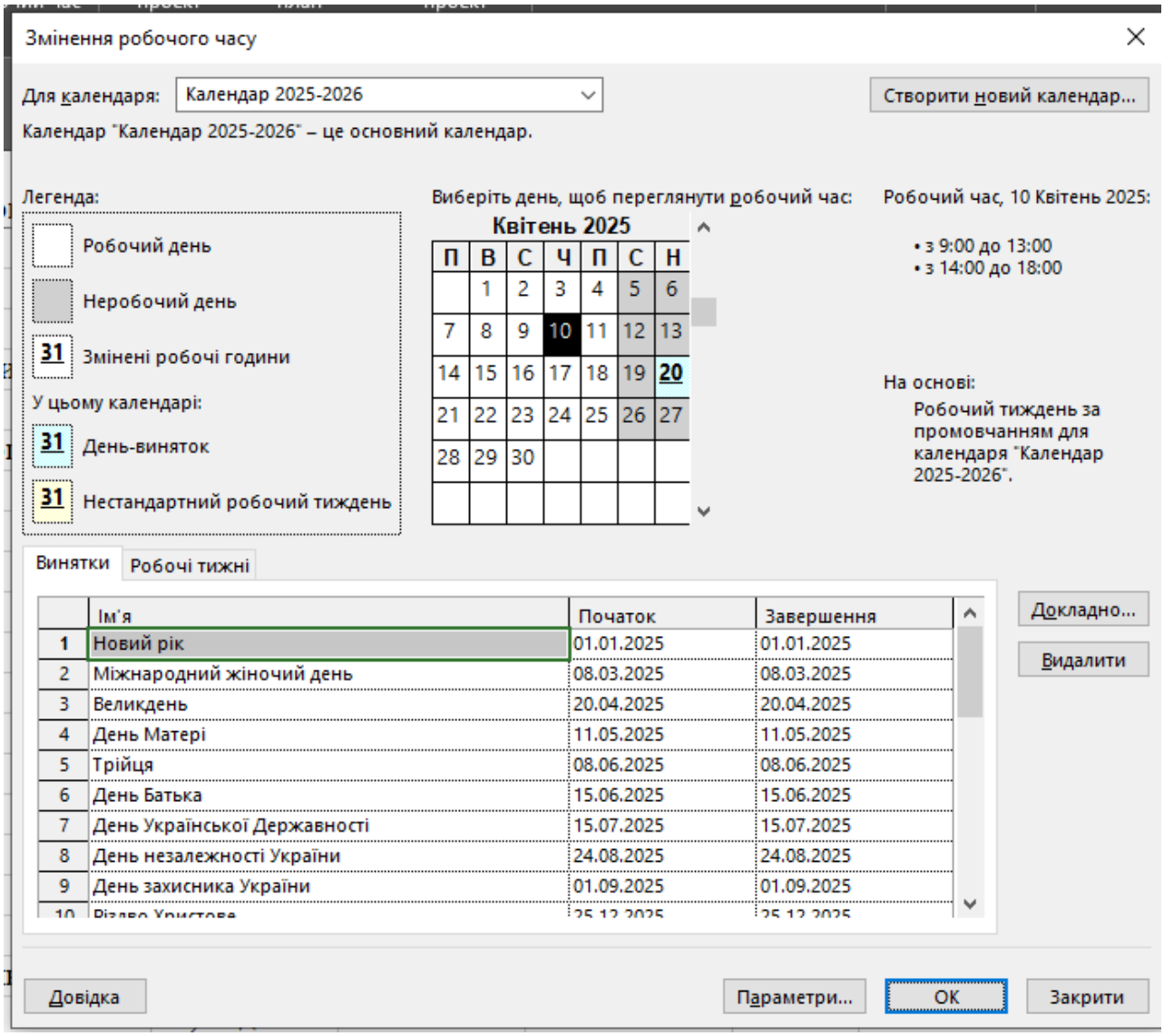


Рис. 2.5. Скріншот календаря проєкту

2.4.2 Вибір бізнес моделі

Проєкт, що представляє собою гру з навчальними елементами в області управління проєктами, спочатку розробляється в межах можливо грантового або освітнього фінансування, а згодом планується до масштабування або комерціалізації, можна розглядати декілька бізнес-моделей, залежно від цілей команди, цільової аудиторії та ринку.

Для даного проєкту розглядались багато моделей [53], але серед перспективних:

– Freemium. Базова версія гри – безкоштовна, що включає доступ до основних модулів, рівнів, уроків. Доступ до додаткових функцій, а саме додаткові сценарії, рівні, звіти, статистика – платний. Це популярний підхід для мобільних додатків, онлайн-сервісів та ігор. За цієї моделі легко привабити користувачів, адже вони можуть ознайомитись з базовою версією, а вже потім придбати додаткові функцій.

– B2B модель (Business-to-Business). Пропонуєте ліцензію для навчальних закладів, які хочуть впровадити інноваційні методи. Можна співпрацювати з університетами та онлайн-платформами. Модель має можливість включати технічну підтримку, кастомізацію гри під курси, аналітику результатів. Перевага – прогнозовані доходи.

– Підписка (Subscription). Користувачі регулярно сплачують фіксовану суму за доступ до продукту чи послуги. Ця модель забезпечує стабільний дохід і часто використовується в SaaS (Software as a Service), стрімінгових платформах та мобільних додатках. Наприклад, підписка для викладача з функціями аналітики – 299 грн/міс, а для закладу з доступом для більше 20 користувачів – 3800 грн/рік. Така модель дозволяє мати стабільний дохід та гнучко оновлювати продукт.

– Sponsorship або Partner-based. Пошук корпоративних або урядових партнерів для фінансування. Платформа є безкоштовною для користувачів, але за підтримки компаній, які зацікавлені в підготовці фахівців.

Після огляду чотирьох, описаних вище, бізнес-моделей, було сформовано стратегію розповсюдження проєкту с фінансової точки зору. Як було вказано спочатку розділу, на початкових етапах, проєкт фінансується грантами та впроваджується його пілотна версія в університеті КНУБА. Через 6-9 місяців після впровадження пілотної версії йде стадія масштабування, що передбачає реалізацію бізнес моделі у вигляді підписки. В довгостроковій перспективі буде використовуватись комбінована модель із моделей B2B та підписки.

Висновки до розділу 2:

1. Для проєкту проведений аналіз внутрішнього та зовнішнього середовища, а саме аналізи PEST та SWOT. Також проаналізовані зацікавлені сторони, їх вплив на проєкт та очікування. На даний момент користувачі мінімально впливають на хід розробки проєкту – тільки через глобальні тестування. Це негативно вплине на кінцевий результат, тому треба активніше залучати користувачів у розвиток та реалізацію проєкту;

2. Після аналізу моделей життєвого циклу та методологій з підходами у першому розділі, для проєкту розробки програмного додатку було обрано методологію Agile, яка реалізується через ітеративну модель життєвого циклу. Із розглянутих практик Agile обрано Scrum;

3. Для проєкту було розроблено статут та створено календарно-мережевий графік у MS Project відповідно до обраної моделі життєвого циклу;

Було прораховано фінансові показники проєкту за допомогою MS Project, а саме витрати на матеріальні ресурси, трудові ресурси (заробітна плата робітників) та разові витрати. В статуті вказано, що загальна сума витрат зіставляє 2 293 040,00 грн. Проаналізовано бізнес моделі для подальшого впровадження додатку. Для початкових етапів грантова фінансування, а в довгостроковій перспективі – модель B2B та підписка.

РОЗДІЛ 3.
ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ «SMART PM CITY»

3.1. Вибір середовища розробки

Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес являє собою розробку гри з навчальними елементами, тому вибір ігрового рушія – це важливе стратегічне рішення. Було обрано ігровий рушій Godot.

Godot Engine [54] – це багатофункціональний кросплатформний ігровий рушій для створення 2D та 3D ігор з єдиного інтерфейсу. Він надає повний набір інструментів для зосередження на створенні ігор. Експорт проєктів доступний на низку платформ: настільні платформи Linux, macOS та Windows; мобільні платформи Android та iOS; веб-платформи та консолі (через сторонніх постачальників або видавців проміжного програмного забезпечення).

Godot повністю безкоштовний з відкритим вихідним кодом, що працює за ліцензією MIT. Жодних зобов'язань чи обмежень. Розробка Godot є повністю незалежною та орієнтованою на спільноту, що допомагає розвивати та покращувати рушій.

У першому розділі пункту 1.3.2 розглядались ігрові рушії Godot, Unity та Unreal Engine. Крім зазначених джерел, проаналізовано окремо відгуки та порівняння самих користувачів цих рушіїв. Загальний огляд можливостей цих рушіїв представлений у таблиці 3.1.

Таблиця 3.1

Аналіз ігрових рушіїв за функціоналом/можливостями

Ігровий рушій / Функ-ціонал/ Можливість	Unity Engine	Unreal Engine	Godot Engine
Ціна	Freemium з платними тарифами Pro	На основі підписки, без сплати роялті для першого доходу в 1 мільйон доларів США	З відкритим вихідним кодом і абсолютно безкоштовним
Легкість використання	Зрозумілий і простий інтерфейс. Є багато документації та ресурсів для навчання. Вважається не важких рушієм однак потрібен час для навчання основному інструментарію	Є багато документації та ресурсів для навчання, але важкий для початківців. Однак це зумовлено потужним інструментарієм	Простий та інтуїтивно зрозумілий інтерфейс. Низький поріг входження. Є багато документації, однак кількість ресурсів для навчання тільки починає зростати
Можливості графічного рушія	Здатний створювати вражаючі візуальні ефекти, але вимагає знань та кваліфікованої оптимізації.	Відомий приголомшливою візуальною складовою та передовими технологіями рендерингу.	Може досягати пристойної візуальної складової, але бракує високоякісних графічних елементів.

Закінчення таблиці 3.1

Аналіз ігрових рушіїв за функціоналом/можливостями

Ігровий рушій / Функ-ціонал/ Можливість	Unity Engine	Unreal Engine	Godot Engine
3D-рушій	Велике сховище ресурсів із готовими моделями, анімацією та ефектами	Величезна бібліотека високоякісних ресурсів, часто надається за ліцензію	Зростаюча бібліотека ресурсів, але обмежена порівняно з більшими движками
2D-рушій	Імітація 2D через 3D-рушій, але є потужні можливості зі спеціалізованими інструментами та плагінами	Переважно зосереджений на 3D, але 2D-інструменти вдосконалюються	Вбудований 2D-рушій з потужними функціями та спеціальною мовою сценаріїв
Мільти-платформність	Інструменти для експорту на інші платформи є в кожному рушії		
Мови програмування	Переважно C# та власний UnityScript	Переважно C++, з візуальними скриптами Blueprint для початківців	C++, C# та GDScript (спеціальна мова для Godot)
Підтримка спільноти	Велика та активна спільнота, обширні онлайн-ресурси	Велика та активна спільнота, обширні онлайн-ресурси та професійна підтримка	Зростаюча та активна спільнота, онлайн-ресурси

Після короткого огляду кожного рушія треба описати основні причини вибору рушія Godot [38, 40]:

– Godot відомий завдяки розробці 2D ігор, бо має власний 2D-рушій. Також є 3D-рушій, який має менший функціонал та можливості в порівнянні з Unity та Unreal Engine. Unity та Unreal Engine, в свою чергу, не мають справжнього 2D-рушія, тому їх 2D-рушій це імітація від 3D. Це робить розробку складнішою і проблематичнішою, ніж потрібно. Тому Godot має найкращий 2D-рушій, через що і був обраний для проєкту;

– Godot також відомий завдяки низькому порогу входження. Інтуїтивний інтерфейс допомагає швидко розібратись у рушії і почати розробку гри. Для проєкту це було однією з головних причин, адже як Unity, так і Unreal Engine мають комплексний та складний інтерфейс. При детальному вивчені це дає свої плюси, але на це витрачається час. Спеціалісти Unity або Unreal Engine обійдуться дорожче;

– Godot дозволяє розробляти ігри для Windows, Linux, macOS, Android, iOS тощо. При вдалому запуску проєкту, розглядається подальша його реалізація на інші платформи для охоплення більшої аудиторії;

– Godot має зручну систему вузлів (англ. «node»), що сприяє модульності розробки та повторному використанню модулів (в рушії називаються сценами). Сама ж система вузлів представлена як дерево, де кожен вузол може мати свої підвузли, що в кінцевому результаті розгортається в зручну для навігації деревоподібну структуру;

– Godot має власну мову під назвою GDScript, яка схожа на Python. Вона була розроблена бути легкою та зручною для програмування ігор. До того ж можна використовувати C# та C++ в межах одного проєкту (гри). Це зручніше ніж лише C# у Unity або C++ у Unreal Engine. Підтримка різних мов програмування спрощує пошук спеціалістів для розробки і робить розробку більш гнучкою;

– з фінансової точки зору, проєкт зроблений у Godot повністю належить розробнику. Жодних комісій, жодних обмежень, жодних хвилювань щодо змін цін,

зміни власності тощо. Що не можна сказати про вже згадані Unity чи Unreal Engine. Про їхню цінову політику йшлось у пункті 1.3.2.

Godot має недоліки, але вони некритичні для проєкту. Серед недоліків, які спільнота знаходить найважливішими у рушії, це:

– набагато менший функціонал 3D-рушія ніж у Unity або Unreal Engine. Але через розробку саме 2D гри для проєкту, цей недолік нівелюється;

– Godot має менше уроків, документації та меншу спільноту ніж у популярних рушіїв. Однак спільнота активна, тому можна швидко отримати рішення для своєї проблеми;

– обмежені можливості у створенні pop-up вікон для інтеграції мобільної реклами, що в даному проєкті взагалі не використовується.

3.2. Робота з дизайном

Одним із основних методів роботи з дизайном у середовищі Godot є об’єкт «StyleBox» [55], що має різновиди «Empty», «Texture», «Flat» та «Line».

StyleBoxEmpty використовується для того, щоб зробити об’єкт невидимим.

StyleBoxTexture використовується для надання об’єкту користувацької текстури, яку потім редагують.

StyleBoxFlat надає об’єкту колір, обравши значення кольорових схем RGB, HSV, RAW. Також, колір можна задати через hex-код, який складається з шести символів. StyleBoxFlat також надає можливість задати контур, якому можна обрати товщину та колір, закруглити кути, додати тінь тощо.

StyleBoxLine перетворює вигляд об’єкту на лінію, якій можна задати товщину та колір.

Невід’ємною частиною створення дизайну є теми. Вони застосовуються до одного або декількох типів об’єктів, та надають спеціальні поля відповідно до типу обраного об’єкту. Теми дозволяють детально налаштовувати елементи інтерфейсу, у яких є багато станів. Прикладом такого об’єкту виступає кнопка, яка

характеризується такими станами: «normal», «pressed», «hover», «hover_pressed», «disabled» та ін. Скріншот прикладу налаштування теми зображений на рис. 3.1.

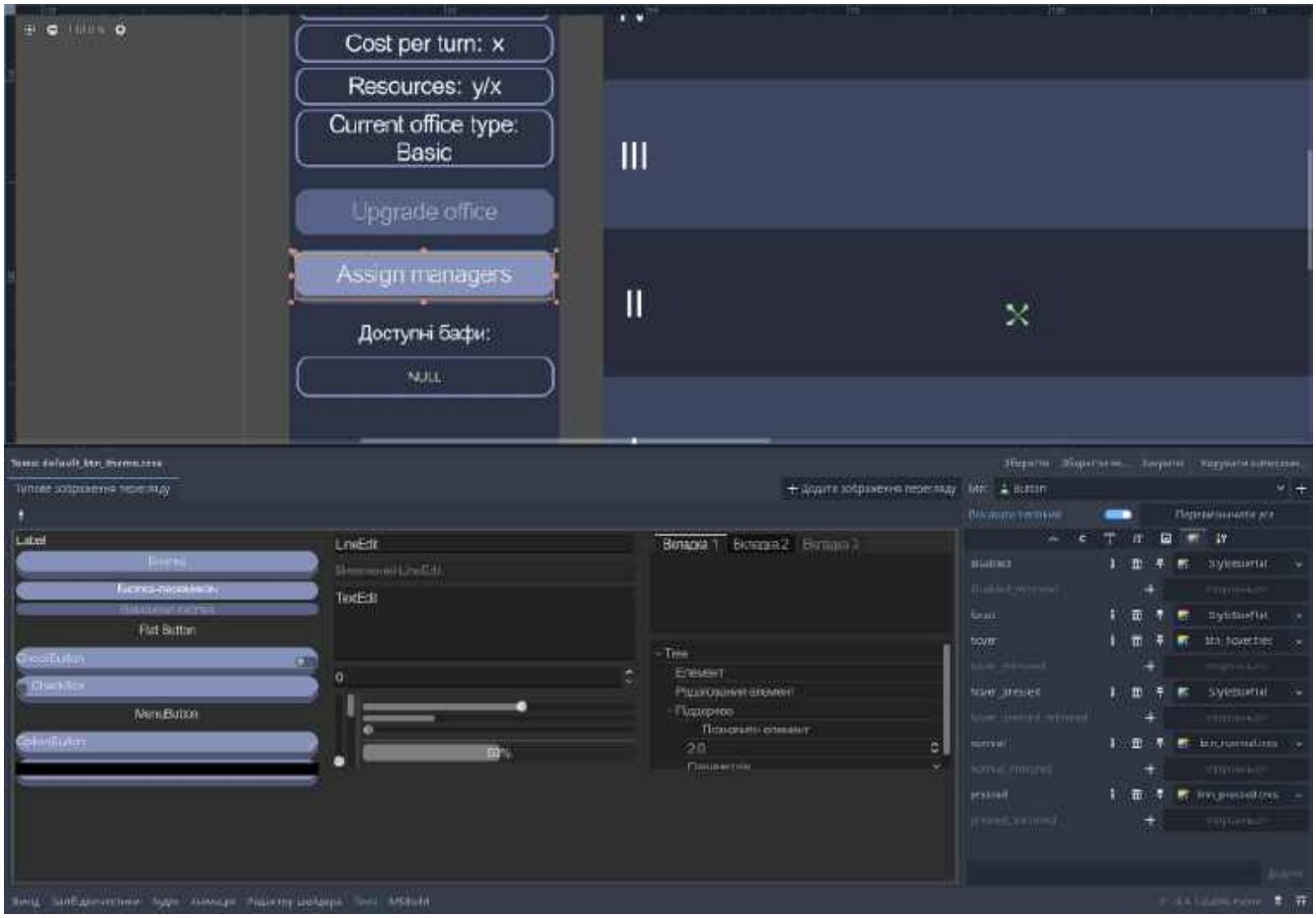


Рисунок 3.1 налаштування теми в середовищі Godot

Слід також зазначити, що як StyleBox, так і теми можна зберігати окремими файлами. Кожна тема має можливість закріплювати за собою набір StyleBox. Це дозволяє відразу застосовувати цілі колекції StyleBox при використанні одної, вже заздалегідь збереженої, теми до ігрового об’єкта.

Для загального дизайну гри був обраний простий векторний стиль, який допомагає передати всю необхідну інформацію користувачеві, без додавання перевантаженого оформлення. Також завдяки простоті стилю пришвидшується розробка його елементів та загального вигляду в цілому.

Важливим елементом UX/UI-дизайну, що напряму впливає на сприйняття, емоції та взаємодію користувача з продуктом є правильно підібрана кольорова палітра, яка створює унікальну візуальну ідентичність додатку. Добре підібрані

кольори спрощують навігацію, виділяють важливі елементи, зменшують візуальне навантаження. Контраст між кольорами забезпечує читабельність тексту, видимість кнопок тощо.

Більшість елементів інтерфейсу було зроблено завдяки вбудованим елементам та функціям рушія Godot, але для реалізації більш комплексних елементів було використано додаткове ПЗ для роботи з комп'ютерною графікою (див. розділ 1, пункт 3.1): CorelDRAW та Adobe Photoshop. CorelDRAW використовувався для створення карт робітників, та сорочки для карт. Adobe Photoshop використовувався для створення дизайн-концепції інтерфейсу, карт проєктів, логотипу гри та для загального редагування та вирізання зображень та іконок. На рис. 3.2 та 3.3 представлено роботу у середовищі CorelDRAW та Adobe Photoshop відповідно.

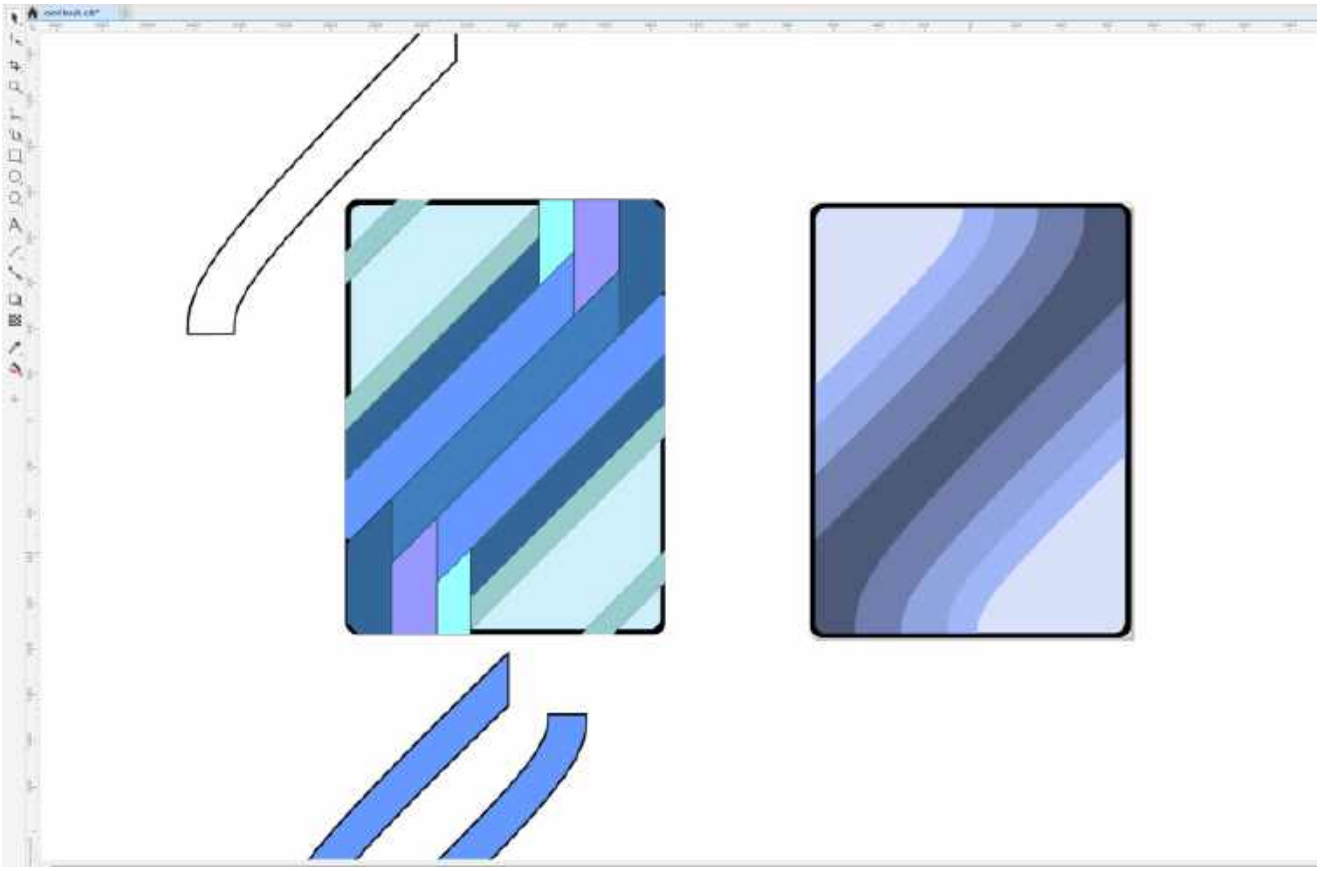


Рис. 3.2. Робота у середовищі CorelDRAW.

Рис. 3.3. Робота у середовищі Adobe Photoshop.

3.3. Процес та механіки гри «SMART PM CITY»

Розробка гри з освітніми механіками це складний процес. Крім ідеї, яка зацікавила б гравця (в даному випадку студента), потрібні механіки та елементи для освітнього аспекту.

Основною ідеєю для гри «SMART PM CITY» було з’єднання ідеї управління проєктами в ігровій манері з освітніми механіками, що представлені у вигляді тестів. Також важливим елементом є управління ресурсами, що надаються гравцю.

Головна ціль гри - досягнення необхідної кількості балів для побудови «розумного міста». Для цього треба виконувати проєкти різної спеціалізації. З розвитком міста, гравець може виконувати дорожчі проєкти і збільшувати свою команду для виконання цих самих проєктів.

Кожна карта проєкту має п’ять характеристик: гроші, які гравець отримає за виконання проєкту; кількість балів прогресу; час на виконання проєкту; кількість робітників для виконання проєкту; спеціалізація.

Картки ресурсів мають три характеристики: заробітна плата; значення часу, яке впливає на час виконання проєкту; спеціалізацію.

Місто має чотири рівні розвитку. При досягненні певної кількості балів прогресу, відкривається наступний рівень. З новим рівнем, гравцю відкриваються нові проєкти для виконання, які є складнішими, але більш прибутковими.

На ігровому полі по боках від поля вибору проєктів є панелі на яких зображено усі потрібні для гравця показники: рівень офісу, кількість ресурсів, рівень міста, кількість балів до наступного рівня, дні до дедлайну, баланс грошей та доступні бафи (англ. «Buff/Debuff» – це тимчасове посилення або послаблення, накладене на ігрового персонажа чи об’єкт).

У гру були додані посади менеджерів (рис. 3.4), а саме: менеджер проєктів, менеджер фінансів та HR-менеджер. В залежності від рангу ресурсу, якого назначають на посаду менеджера, гравець отримує бафи або дебафи, які впливають на ігровий процес.

Рис. 3.4. Скріншот вікна менеджерів у грі «SMART PM CITY»

Було реалізовано систему офісів (рис. 3.5) як елемент управління командою. На початку гри доступний офіс першого рівня. Після переходу на новий рівень міста, гравцю відкриється можливість покращити офіс. Офіс впливає на максимальну кількість робітників. Різні варіації дають свої бафи і дебафи. Чим

вищий рівень офісу, тим дорожча його оренда. З новим рівнем відкриваються нові посади менеджерів.

Рис. 3.5. Скріншот вікна налаштування офісів у грі «SMART PM CITY»

Правила гри для першого раунду:

– Гравець має початковий набір ресурсів: х грошей, базовий офіс, одна посада для менеджера проєктів, перший рівень міста, відкриті проєкти першого рівня, запас у 100 днів до настання дедлайну

– Гравцю видається 3 картки ресурсів. За бажанням, він може замінити 1 картку на іншу, випадкову з колоди. Гра починається коли гравець візьме картки.

Для раундів після першого:

– Гравцю видається 2 картки ресурсів. Обирається тільки одна або пропускається вибір. Під час вибору карт є можливість звільнити ресурс зі свого штату (руки). Ресурс потрапляє у колоду зброшених карток, які більше не з’являться у грі. Якщо було використано усі картки з загальної колоди, тоді ресурси оновлюються та з’являються в загальній колоді, але в іншому порядку.

– Гравець назначає картки ресурси на доступні для виконання проєкти. Дозволяється обирати декілька проєктів або жодного.

– При необхідності, гравець може назначити ресурс на посаду менеджера:

а) Обраний ресурс стає недоступний для виконання проєктів, але дає бафи/дебафи залежно від займаної посади.

б) Ресурс, що є менеджером рахується у штаті працівників і займає місце у офісі.

в) Після назначення на посаду, ресурс не може бути знятий з посади 3 раунди.

г) Якщо базова зарплатня ресурсу менше 70 одиниць, то зарплатня як менеджера буде дорівнювати 70. Якщо більше 70 одиниць, то до базової зарплатні додається 10 одиниць.

д) На першому рівні офісу, доступна одна посада менеджера проєктів. Для відкриття посади фінансового менеджера та HR-менеджера потрібно, відповідно, мати офіс другого та третього рівня.

е) На четвертому рівня міста, є можливість підвищити одного з менеджерів. Це збільшить його зарплатню, але і надасть більш значущі бафи.

– Для збільшення максимальної кількості ресурсів, гравцю надається на вибір різні поліпшення для офісу:

а) За офіс гравець сплачує оренду. Чим більший рівень офісу, тим більша оренда.

б) Є 3 рівня офісів. Кожен з яких стає доступним на відповідному рівні міста.

в) Відмінити поліпшення офісу не можна.

Кінець раунду:

– При умові, що гравцю вистачає грошей для оплати оренди за офіс, зарплатні працівникам, дедлайн не настав і вистачає ресурсів на всі обрані проєкти, гра закінчить хід та обрахує усі показники.

– При нестачі коштів, гравець закінчує гру програшем або перерозподіляє ресурси, щоб коштів вистачило.

– Перед початком наступного раунду обраховується проміжок через який буде задаватись питання гравцю. Проміжок залежить від днів, що залишились до

дедлайну та загальної кількості запитань. За замовчуванням у гру завантажено 20 тестових питань.

Так, обираючи проєкти для виконання, назначаючи менеджерів та поліпшуючи офіс, гравець будує місто. Після накопичення достатньої кількості балів прогресу на четвертому рівні, гра вважається закінченою - перемога.

Рис. 3.6. Скріншот ігрового процесу «SMART PM CITY»

3.4. Що було зроблено

Для огляду виконаної роботи, продукт буде розглянуто зі сторони дизайну (front-end) та ігрового коду (back-end).

Для демонстрації розробленої ігрової логіки буде використана UML (англ. Unified Modeling Language - уніфікована мова моделювання) діаграма класів та діаграма активності для розгляду робочих процесів у програмі.

Для побудови діаграми класів був використаний додаток Software Ideas Modeler. [56] Це інструмент та програмне забезпечення для створення діаграм, яке підтримує UML, SysML, ERD, BPMN, ArchiMate, блок-схеми, користувацькі історії

та інше. Додаток дозволяє генерувати готові діаграми на основі вже існуючого коду. Треба тільки обрати мову, яка використовуються в коді.

Згенеровані діаграми у спрощеному вигляді, для демонстраційної версії гри, представлені на рис. 3.3-3.5. Треба зазначити, що Software Ideas Modeler згенерував діаграму з великою кількістю відносин-залежностей, більшість яких було прибрано в кінцевому варіанті діаграми. Це пов’язано з тим, що у Godot не має жорсткого обмеження на доступ до класів і всі класи мають доступ один до одного. Це дозволяє вільно використовувати функції або атрибути інших класів. Наприклад, клас кнопки для завершення ходу, викликає функцію класу головного вузла гри для оновлення панелей з інформацією про прогрес гри.

Коли створюється новий скрипт для вузла, клас скрипту за замовчуванням відкритий, тобто має ідентифікатор доступу «public» – означає, що клас відкритий для всіх. Використання можливостей мови C# для обмеження доступу (інкапсуляція) дозволяється, але це збільшить час на проєктування та кодування загальної архітектури програми, що для малих команд потребуватиме більше ресурсів, що не вигідно. Однак «інкапсуляція» часто використовується для атрибутів та функцій класів ігрових об’єктів або в класах не пов’язаними з ігровими об’єктами, що відповідають за ігрові обчислення, які не відображаються у грі.

Всього три діаграми, але Global_1 та Global_2 це одна діаграма, просто розбита на частини, щоб можна було чітко роздивитись її у цій роботі. Друга діаграма Controllers. Вони названі так через саму структуру розташування скриптів. Всього є дві папки: «_Global» та «Controllers». У першій – всі скрипти не пов’язані з ігровими об’єктами на сцені, у другій всі класи об’єктів гри (вузлів). Класи вузлів мають доступ один до одного через дерево вузлів гри і мають доступ до інших класів, які у папці «_Global», бо там знаходяться менеджери, дані гри та інше. Однак класи, що не є вузлами не мають доступу до дерева вузлів гри, тому і впливати візуально на гру не можуть, тільки через класи-вузли. Більш детально, з функціями та атрибутами, діаграми представлені у Додатку Х-У. Діаграми додатку 1-2 – це розбита на частини діаграма, що представлена на рис. 3.4. Деякі класи

пусті, бо вони відображені в одній з частин діаграми, щоб не перевантажувати і так великі діаграми.

На діаграмах є класи виділені різними кольорами. Так позначені батьківські та дочірні класи. Згори до низу – від батьківського до дочірнього класу. Також є класи об’єднані у чотирикутні області – це 1 клас, що у коді поділений на різні файли. В C# такий клас має 1 назву, але має ключове слово «partial», що перекладається як «частка».

Загалом, код гри можна поділити на декілька частин, а саме класи:

– контроллери. У грі використовується багато об’єктів, що є дочірніми до базового класу середовища Godot – «Control», що зображено на рис. 3.3. Тобто це класи об’єктів гри, що відображаються на екрані. Програма не побудована по схемі «Model-View-Controller», тому назва класів не відноситься до цього;

– менеджери даних. Назва каже сама за себе – класи, що керують та зберігають дані ігрових об’єктів. На діаграмі та в коді це:

а) «BuffManager» – відповідає за управління бафами/дебафами;

б) «CardsMngr» – управління ігровими картками, що представленні у вигляді ресурсів та проєктів;

в) «Questions_Mngr» – управління тестовими питаннями, які з’являються у грі.

– класи об’єктів гри – класи, що описують основні об’єкти. Наприклад, проєкт має 4 атрибути: бали прогресу за виконання проєкту, кількість витрачених днів, кількість необхідних ресурсів та сума грошей, що видається за проєкт. Все це прописано в класі «ProjectCardInfo»;

– класи даних – класи, що зберігають заготовлений набір ігрових об’єктів для передачі їх до контроллерів для відображення у грі. Наприклад, при початку гри, клас «ProjectField», що відповідає за відображення карток проєктів, робить копію набору цих проєктів з класу «ProjCardArr» та зберігає їх у глобальних змінних «GlobalVar», після чого малює картки на екрані.

При розробці програмного коду використовувався шаблон проєктування «singleton». На основі нього написаний скрипт «BuffManager». Шаблон гарантує, що в додатку буде єдиний екземпляр класу, який надає глобальну точку доступу до цього екземпляру.

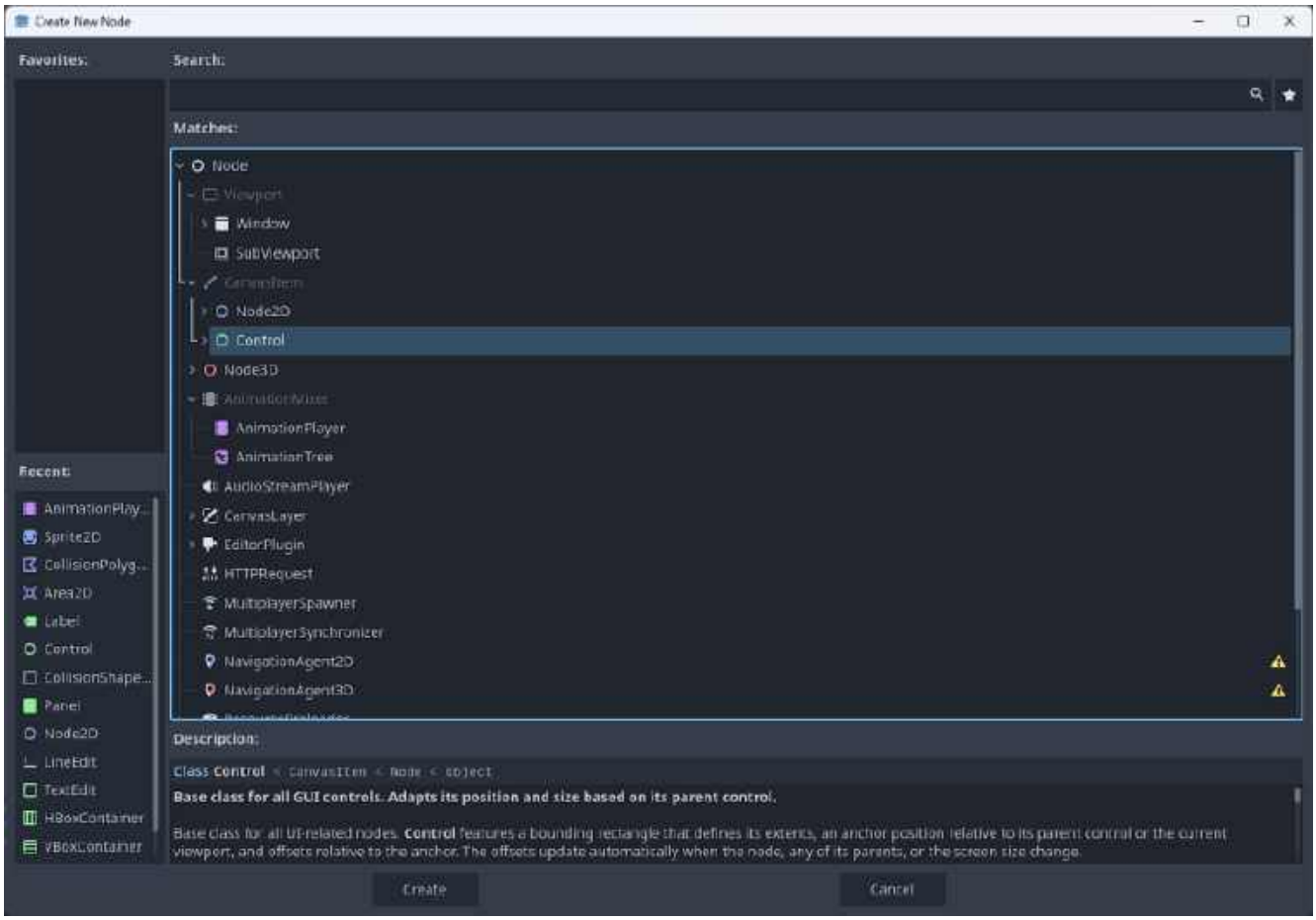


Рис. 3.4. Вікно створення вузлів

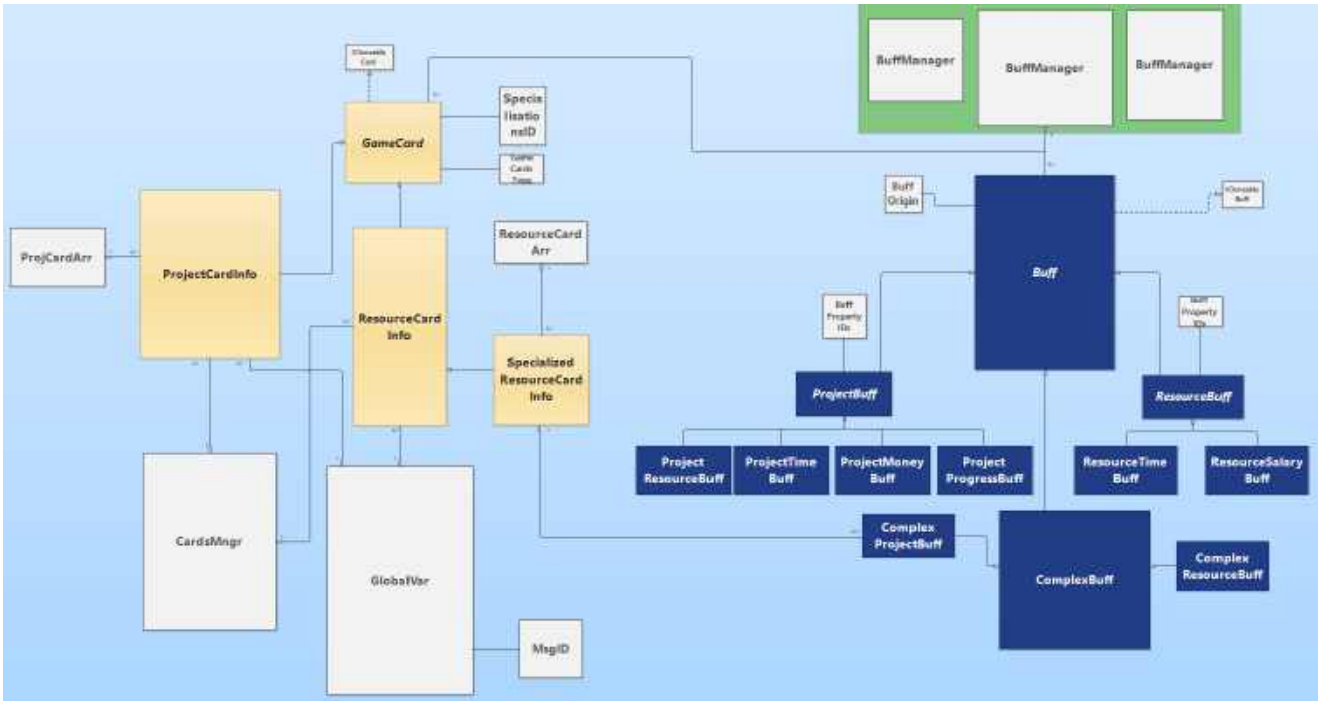


Рис. 3.5. Global_1

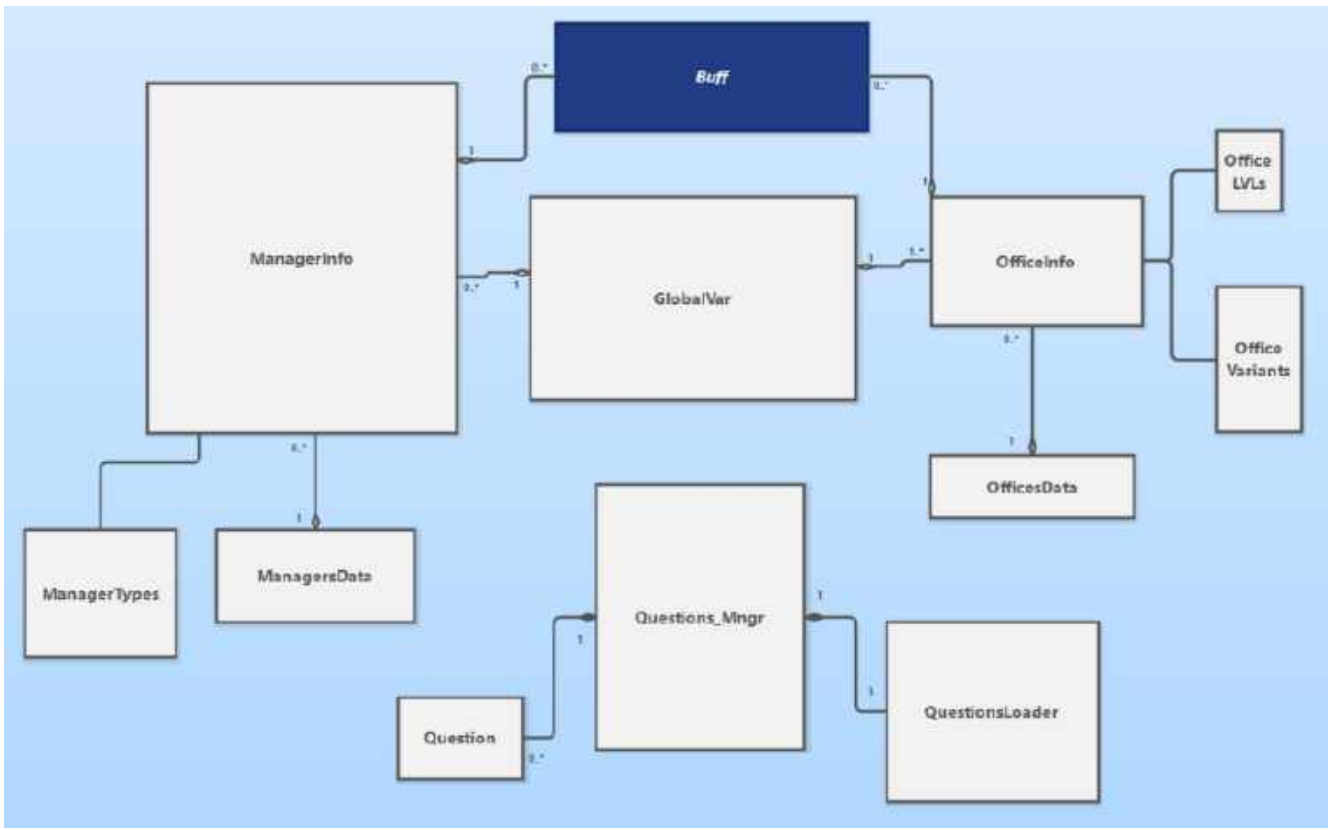


Рис. 3.6. Global_2

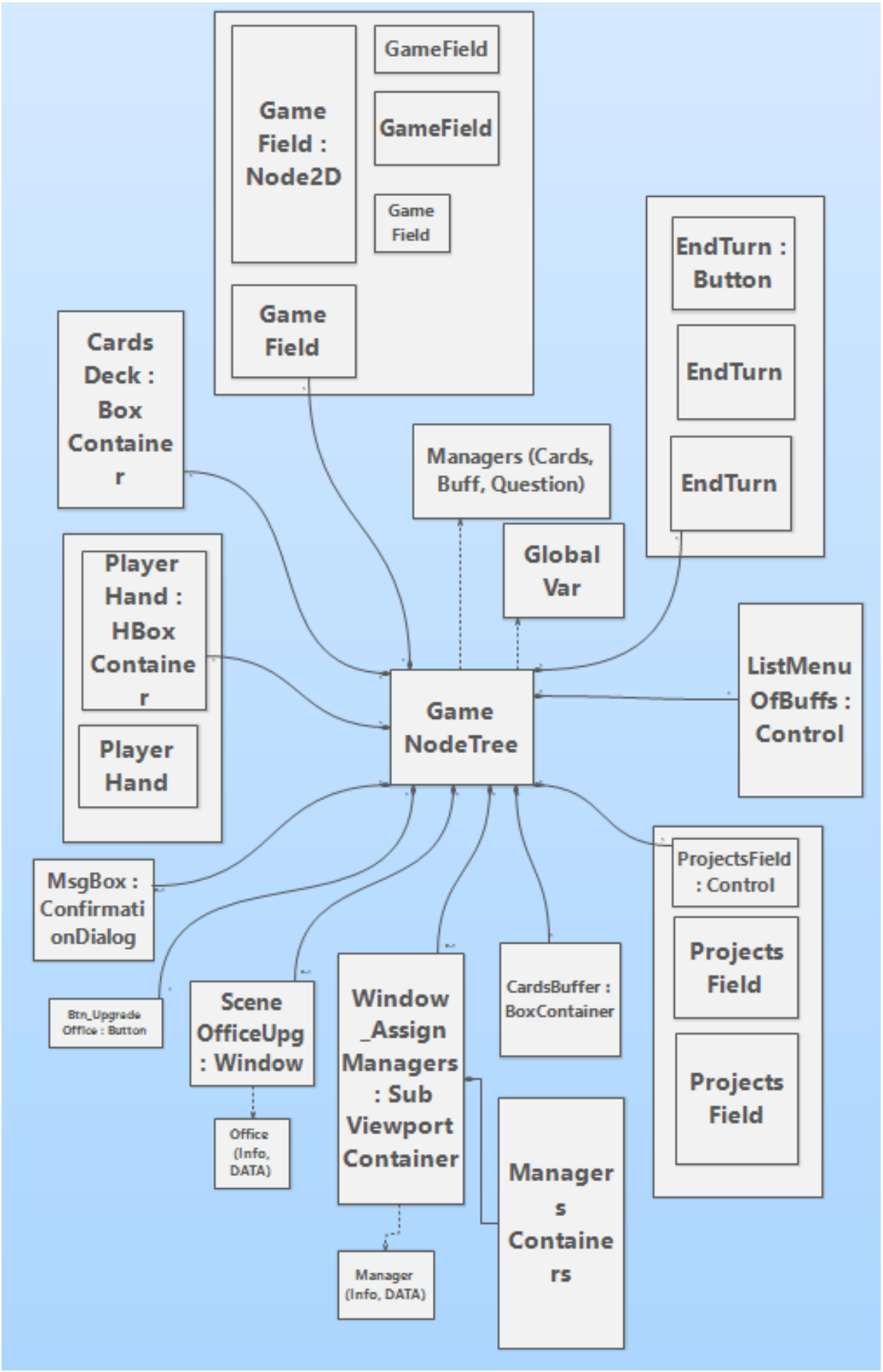


Рис. 3.7. Controllers

Для діаграми активностей використовувався веб-застосунок для створення діаграм Visual Paradigm. Діаграми представлені на рис. 3.8-3.9. Вони відображають можливі дії у меню гри (див. рис 3.8) та у самій грі (див. рис 3.9). Для меню всі дії пояснюються можливим вибором пункту головного меню: «гра наодинці», «вихід», «титри», «параметри». Для гри це дії, що може виконати гравець та процеси обчислення.

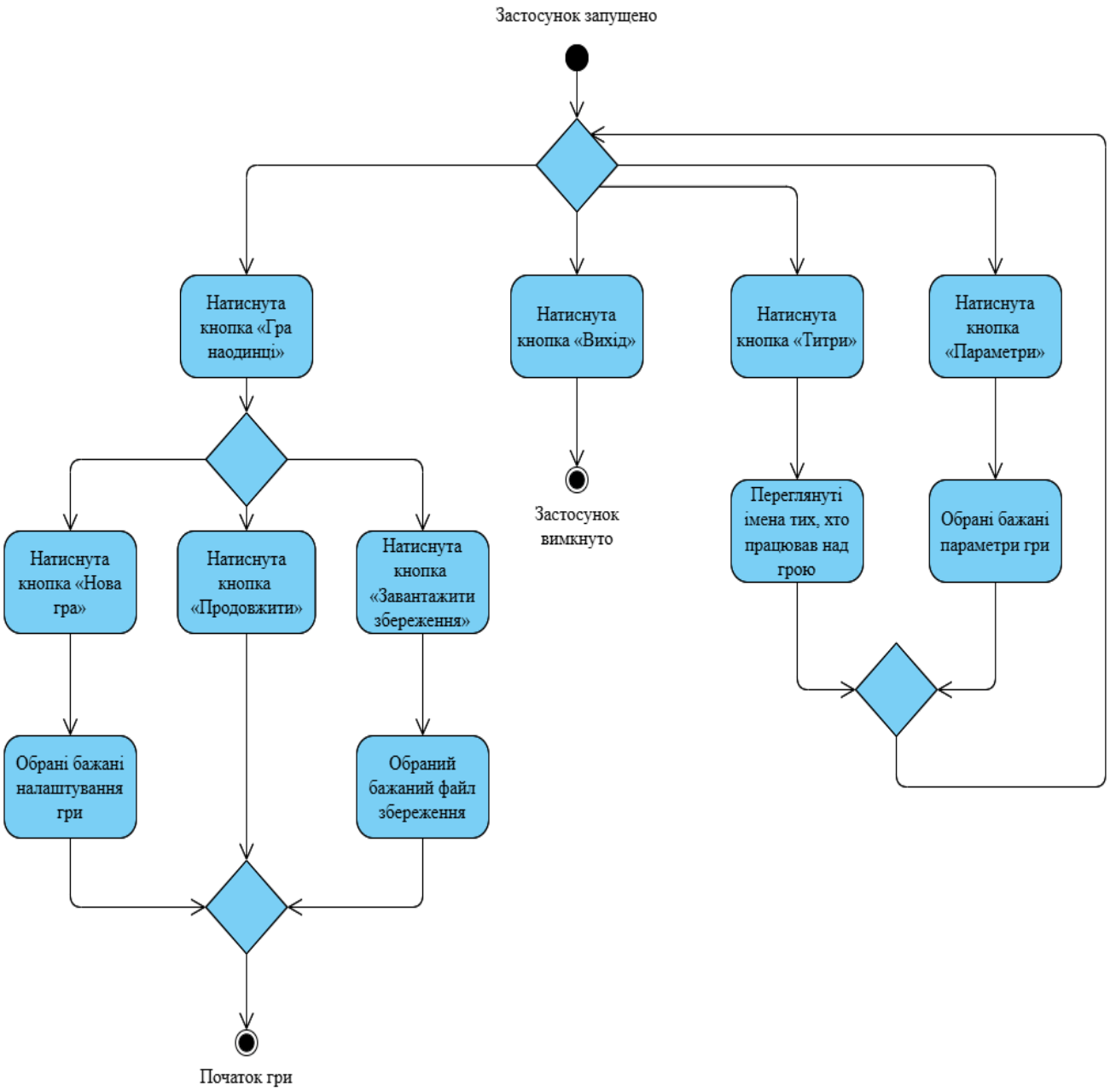


Рис. 3.8. Діаграма можливих дій у головному меню

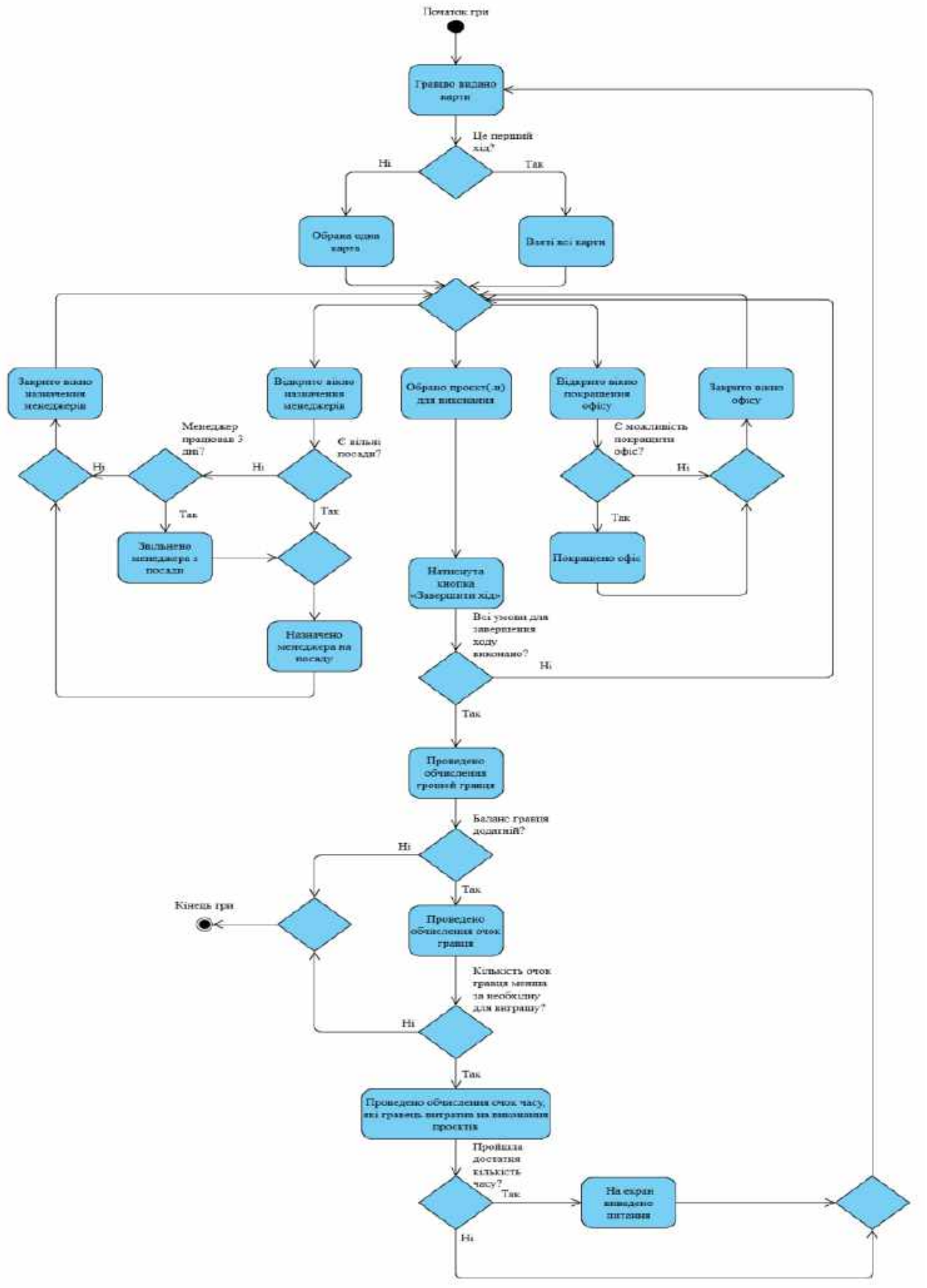


Рис. 3.9. Діаграма дій під час гри безпосередньо

Перед початком роботи над дизайном та інтерфейсом додатку важливо розробити загальне бачення, тобто те, що треба отримати в кінцевому результаті. Це включає: загальний дизайн застосунку, загальний вигляд основних вікон, окремі елементи дизайну, кольорову палітру тощо. Щоб розробити один або декілька варіантів дизайну використовують дизайн-концепцію. Це допомагає вносити зміни вже на ранньому етапі реалізації додатку, підтримуючи зв’язок з замовником. Перед початком роботи над проєктом було створено дизайн-концепцію головного вікна гри, що зображена на рис. 3.10 і яка використовувалася впродовж перших етапів розробки.

Рис. 3.10. Дизайн-концепція головного вікна гри

Наступним кроком розробки дизайну був вибір підходящої палітри кольорів для застосунку. Як йшлося в пункті 3.2 третього розділу: колір відіграє важливу роль у сприйнятті користувачем продукту. Тому після перегляду ряду кольорів, головним було обрано синій колір під назвою «ebony clay» (укр. «ебонітова глина»), та його відтінки.

Синій колір має низку переваг, які є важливими для дизайну кінцевого продукту: синій колір асоціюється зі спокоєм і стабільністю, що допомагає

користувачам зосередитись на навчанні. Синій колір гармонійно комбінується з білим, сірим, зеленим і помаранчевим, що полегшує створення контрастного та водночас гармонійного інтерфейсу. Світлі відтінки синього на темному фоні забезпечують хорошу читабельність тексту та не напружує очі. Багато EdTech-компаній використовують синій колір (наприклад, Coursera, Projector, Київська Школа Економіки), що створює враження звичності та сучасності.

В якості головного шрифту гри було обрано «Arial». Цей шрифт є одним з базових шрифтів сучасних операційних систем та підтримує велику кількість мов. Завдяки цьому інтерфейс гри можна легко адаптувати під будь-яку мову. Також цей шрифт добре поєднується з простим векторним стилем гри. Для дрібного шрифту на картах ресурсів та проєктів було вирішено додати обведення для покращення видимості.

Після визначення всіх базових елементів дизайну було розпочато роботу над інтерфейсом у середовищі рушія Godot. Спочатку було визначено розташування всіх головних елементів користувацького інтерфейсу. Далі почалась робота над дизайном цих елементів: надання їм кольору, вибір розміру шрифту, додання додаткового оформлення.

Після завершення роботи над основними елементами було розпочато працю над другорядними елементами, в тому числі і додатковими вікнами, такими як: вікно для призначення менеджерів на посади, вікно для покращення офісу та вікно з тестовими питаннями. При створенні вікон було пройдено всі ті ж етапи, що й при створені головних елементів. Було зроблено декілька дизайн-концепцій, після чого обрано кращу, обрано кольори, визначено розташування елементів та розроблено дизайн.

Далі розроблявся дизайн карт проєктів та ресурсів. Спочатку було розроблено загальний макет оформлення для картки. Після цього було обрано колір для кожної спеціалізації карток, рис 3.11. Оформлення карток проєктів було зроблено таким чином, що рамка картки знаходиться окремо від ілюстрації до цієї картки, а вже в середовищі рушія вони об’єднуються в один об’єкт. Це допомагає

зменшити місце, що займає програма шляхом використання однієї і тої ж самої рамки для декількох картинок карт проєктів.

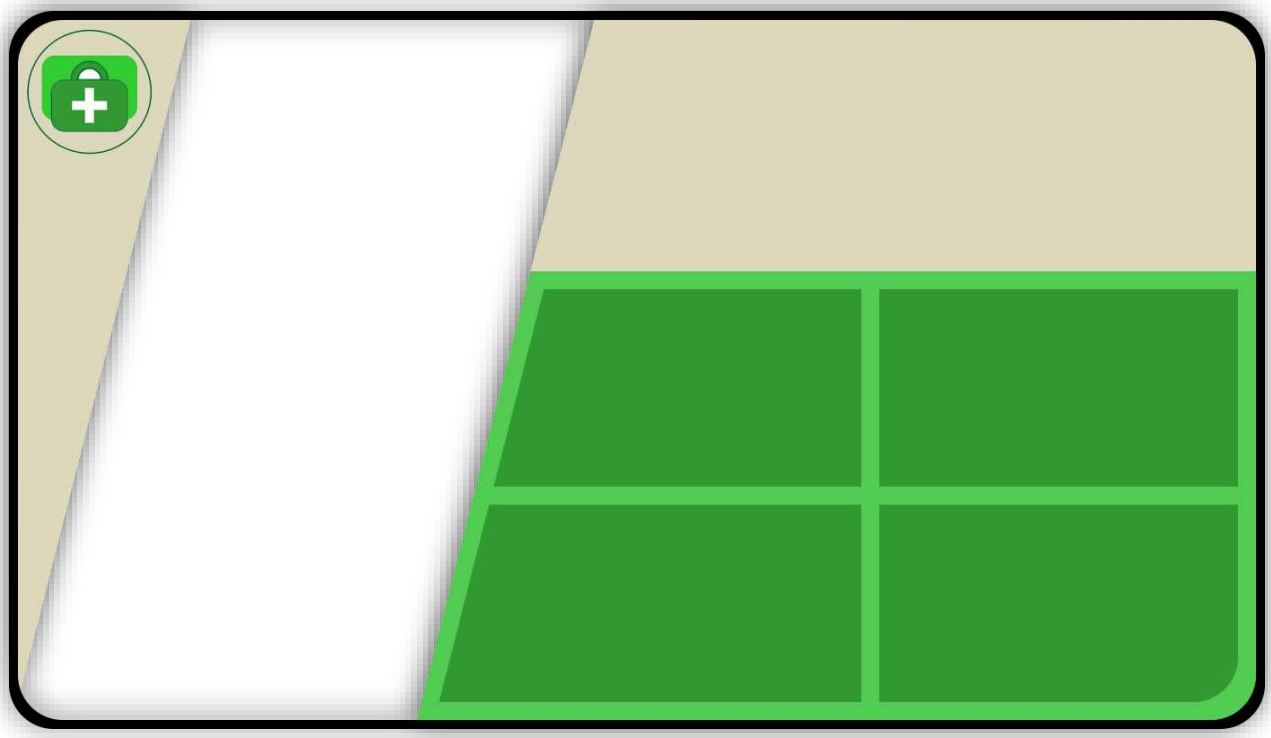


Рис. 3.11. Рамка для карток медичних проєктів

Для створення зображень було використано чат-бот ChatGPT. Для карт ресурсів було вирішено використати персонажів зі слов'янської міфології та зобразити їх в умовах сучасного світу, рис. 3.12. Використовувався опис персонажу з літератури або його художнє зображення. Основою для проєктів слугували різні будівлі та системи, що використовуються у містах, наприклад: електронний уряд, інститут, пошта та інше. Таким чином було створено 54 ілюстрації для карт ресурсів та 48 для карт проєктів.

Останнім кроком у роботі над користувацьким інтерфейсом стала розробка головного меню гри та суміжних вікон, наприклад, вікна налаштувань гри. Щоб підкреслити орієнтованість гри на побудову міста було прийняте рішення зобразити в головному меню малюнок міста з хмарочосом на передньому плані, а кнопки розмістити у вікнах цього хмарочоса (рис. 3.13).

Рис. 3.12. Перун у діловому костюмі

Рис. 3.13. Головне меню гри

3.5. Інтеграція навчальних елементів у гру

Проєкт націлений на розробку гри з навчальними елементами, тому, як і для будь-якої гри, їй притаманні компоненти прогресу та система балів. Це є основою для першої освітньої складової – менеджмент ресурсів.

Студент, у ролі гравця, має обмежену кількість ресурсів та можливостей на початку. Під ресурсами мається на увазі людські – робітники, та фінансові – гроші для виплати заробітної плати, оренду офісу та найму менеджерів. При виконанні доступних проєктів малого рангу (на які потрібно менше ресурсів, днів та за які дають менше прогресу), студент відкривається можливість виконання проєктів вищого рангу і, відповідно, доступ до більшої кількості ресурсів. Якщо студент буде невідповідально ставитись до менеджменту ресурсами, він не буде не зможе продовжити гру, через нестачу критично важливих компонентів для подальшої гри.

Друга ж складова – це тести з запитаннями про управління проєктами. Є як тести вже завантаження у гру за замовчуванням, так і є можливість завантажити користувацькі тести, наприклад створені викладачем. Для адаптації навчального процесу під студента, гра надає можливість надати кожному питанню свій рівень складності. Він задається окремо в налаштуваннях. Від обраної кількості тестів, залежить частота їх появи під час гри. Також, як нагорода, якщо студент відповідає на питання правильно, підвищується шанс отримання студентом більш робітника високої кваліфікації на умовному ігровому ринку праці.

Висновки до розділу 3:

1. Було виконано порівняльну характеристику популярних рушіїв на ринку за їх функціоналом та можливостями. Наведено переваги Godot завдяки яким було обрано цей рушій;
2. Описано основні засоби роботи з дизайном в середовищі Godot, такі як StyleBox і тема. Розглянуто загальні положення про створення дизайну;
3. Описано правила гри та основні її механіки;
4. Розглянуто виконану роботу над демонстраційною версією гри. Для огляду back-end частини додатку було створено UML діаграми класів та

активностей. Описано структуру скриптів та типів на які було їх умовно поділено. Описано шлях створення дизайну та інтерфейсу гри. Від перших концептів до розробки окремих елементів. Обґрунтовано вибір кольорової палітри;

5. Описано два основі навчальні елементи, що були інтегровані у гру: менеджмент ресурсами та тестові запитання, що стосуються управління проєктами.

ЗАГАЛЬНІ ВИСНОВКИ

У межах виконання кваліфікаційної випускної роботи на тему «Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес» було досягнуто поставленої мети та вирішено всі основні завдання, що підтверджує практичну та теоретичну цінність проєкту.

У процесі дослідження було проведено ґрунтовний аналіз сучасного стану вищої освіти в Україні, зокрема в умовах викликів, зумовлених глобальними змінами та військовими діями. Встановлено, що система освіти потребує оновлення не лише на структурному та організаційному рівні, а й у підходах до навчального процесу. Одним із ефективних інструментів модернізації виявилась гейміфікація — використання ігрових елементів для підвищення мотивації, залученості та продуктивності студентів.

В межах роботи було детально проаналізовано моделі життєвого циклу проєктів та сучасні методології управління розробкою програмного забезпечення, серед яких виокремлено та обґрунтовано доцільність застосування підходів Agile, Scrum та Scrumban. Це дозволило забезпечити гнучкість управління проєктом, прозорість етапів розробки, ефективне планування та контроль виконання завдань.

Здійснено розробку календарно-мережевого графіка, фінансової моделі проєкту, визначено ресурси, ризики та розраховано витрати на реалізацію. Використання MS Project забезпечило візуалізацію та контроль графіку робіт. Було також створено статут проєкту, який визначив основні цілі, завдання, ролі, обмеження та очікувані результати.

Особливу увагу приділено інструментальним засобам розробки: середовищу програмування Visual Studio Code, графічним редакторам Adobe Photoshop і CorelDRAW, а також платформі GitHub для контролю версій. Завдяки цим інструментам вдалося створити функціональний прототип мобільного додатку «Smart PM City».

Результатом проєкту є навчальний застосунок з елементами гейміфікації, розроблений для підвищення залученості студентів до вивчення управління проєктами. Застосунок містить освітній контент, ігрові механіки, рівні, бали, місії та інтерактивні елементи, що створюють мотивуюче навчальне середовище. Реалізація продукту продемонструвала можливість ефективного поєднання освітніх цілей з розважальним контекстом, що є інноваційним підходом для українського освітнього ринку.

Таким чином, проведене дослідження та реалізований проєкт підтвердили гіпотезу щодо доцільності використання гейміфікації в освіті. Результати роботи можуть бути впроваджені в навчальний процес вищих навчальних закладів, а сам застосунок — адаптований для інших освітніх дисциплін. Отримані знання, досвід та результати можуть слугувати основою для подальших досліджень у галузях цифрової трансформації освіти, EdTech та управління IT-проєктами.

СПИСОК ДЖЕРЕЛ

1. Ринок освітніх послуг в україні: конкурентні переваги та сучасні виклики для іноземних студентів. Наукове фахове видання. [Електронний ресурс] // Режим доступу: https://science.iea.gov.ua/wp-content/uploads/2020/06/1_Londar_18_2020_5_22.pdf (дата звернення: 30.04.2025).

2. Філіпов І. Перспективи та напрямки зростання EdTech в Україні. Europian Business Association. [Електронний ресурс] // Режим доступу: https://eba.com.ua/perspektyvy-ta-napryamky-zrostannya-edtech-v-ukrayini/ (дата звернення: 02.05.2025).

3. Дослідження тенденцій розвитку ринку вищої освіти в умовах війни. Academic Journals and Conferences [Електронний ресурс] // Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2023/sep/31290/menedzhment-303-316.pdf (дата звернення: 30.04.2025).

4. Фінансування освіти україни з бюджетів різних рівнів в умовах війни. Наукове фахове видання. [Електронний ресурс] // Режим доступу: https://science.iea.gov.ua/wp-content/uploads/2023/10/3_Totska_Tytarenko_324_2023_34-44.pdf (дата звернення: 30.04.2025).

5. Kukharska, L.V. (2020). Educational services market in ukraine: realities and prospects. Social and Legal Studios, 3(3), 184-191. https://doi.org/10.32518/2617-4162-2020-3-184-191 (дата звернення: 30.04.2025).

6. Видатки на навчання одного студента за держзамовленням на 2020 рік. Міністерство Фінансів України. [Електронний ресурс] // Режим доступу: https://mof.gov.ua/uk/news/minfin_u_2020_rotsi_seredni_vidatki_na_navchannia_odnogo_studenta_za_derzhzamovlenniam_virosli_na_13_ta_stanovili_63_tis_griven-3201 (дата звернення: 30.04.2025).

7. Добровська Л. М. Управління ІТ-проєктами навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського ; уклад.: Л. М.

Добровська, О. С. Коваленко, О. А. Аверьянова. – Київ : КПІ ім. Ігоря Сікорського, 2022. – 284 с. (дата звернення: 25.04.2025).

8. Життєвий цикл розробки програмного забезпечення. | TheUpperCode. High-quality software for your business. [Електронний ресурс] // Режим доступу: https://theuppercode.com/blogs/software-development-life-cycle (дата звернення: 30.04.2025).

9. Програмна інженерія в системах управління. Лекції. Автор і лектор: Олександр Пупена | ProgIngContrSystems [Електронний ресурс] // Режим доступу: https://pupenasan.github.io/ProgIngContrSystems/Лекц/17_lyfecycle.html (дата звернення: 02.05.2025).

10. Інкрементальна модель життєвого циклу розробки ПЗ: ключові переваги та етапи | Custom Web & Mobile Development Company – New Line Technologies. [Електронний ресурс] // Режим доступу: https://newline.tech/inkrementalna-model-zhittyevogo-ciklu-rozrobki-programnogo-zabezpechennya-uk/ (дата звернення: 02.05.2025).

11. Дегтярьова Л.М. Навчальний посібник з дисципліни «Технології розробки програмного забезпечення» для студентів спеціальності 123 «Комп'ютерна інженерія» / Гроза П.М., Сомов С.В. – Полтава: ПолтНТУ, 2017. – 218 с.

12. Шаблони процесу розробки ПЗ – спіральна модель | Qalight. [Електронний ресурс] // Режим доступу: https://qalight.ua/baza-znaniy/cpiralna-model-spiral-model/ (дата звернення: 02.05.2025).

13. Project Management Institute. (2021). A guide to the project management body of knowledge (PMBOK guide)—seventh edition. The standard for project management.

14. Chen, Lu-Jui & Ho, Tzu-Ping & Lee, Han & Lin, Chu-Qiao. (2025). How can projects succeed? An exploration through agile project management. Journal of management research. 17. 28. 10.5296/jmr.v17i1.22769. (дата звернення: 30.04.2025).

15. Теоретико-методичні засади реалізації концепції ощадливого виробництва в практиці діяльності промислових підприємств. | Інституційний репозитарій Хмельницького національного університету. [Електронний ресурс] //

Режим доступу: https://elar.khmnu.edu.ua/server/api/core/bitstreams/dbf147ef-367c-4188-b47a-88fe325d5a92/content (дата звернення: 02.05.2025).

16. Керівництво до канбан методу. | Kanban University [Електронний ресурс] // Режим доступу: https://kanban.university/wp-content/uploads/2022/03/The-Official-Kanban-Guide_Ukrainian_A4.pdf

17. What is Agile? | Atlassian. Atlassian. [Електронний ресурс] // Режим доступу: https://www.atlassian.com/en/agile (дата звернення: 30.04.2025).

18. Dennis, P. (2005). Andy & Me: Crisis and transformation on the LEAN journey. [Електронний ресурс] // Режим доступу: https://books.google.ca/books?id=uOtAnt4XIo8C (дата звернення: 30.04.2025).

19. Кен Ш. Посібник зі Скраму: правила гри / Джефф С. – Scrum Guides. 2020 [Електронний ресурс] // Режим доступу: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Ukrainian.pdf (дата звернення: 30.04.2025).

20. Banijamali, Ahmad & Dawadi, Research & Ahmad, Muhammad Ovais & Similä, Jouni & Oivo, Markku & Liukkunen, Kari. (2017). Empirical investigation of scrumban in global software development. 10.1007/978-3-319-66302-9_12 (дата звернення: 30.04.2025).

21. All about Agile: Extreme Programming. Cape Project Management Inc. [Електронний ресурс] // Режим доступу: https://www.agileprojectmanagementtraining.com/wp-content/uploads/grassblade/7280-pmi-acp-module-2-3/presentation_content/external_files/PMI_ACP-SS-Mod-2-3.pdf (дата звернення: 30.04.2025).

22. Git – Про систему контролю версій | Git. [Електронний ресурс] // Режим доступу: https://git-scm.com/book/uk/v2/Вступ-Про-систему-контролю-версій (дата звернення: 02.05.2025).

23. Rehman, R. U., & Paul, C. (2003). The Linux development platform: configuring, using, and maintaining a complete programming environment. [Електронний ресурс] // Режим доступу: https://ptgmedia.pearsoncmg.com /imprint_downloads/informit/perens/0130091154.pdf (дата звернення: 30.04.2025).

24. Git – Створення та налаштування облікового запису | Git. [Електронний ресурс] // Режим доступу: https://git-scm.com/book/uk/v2/GitHub-Створення-та-налаштування-облікового-запису (дата звернення: 02.05.2025).

25. Contributor T. What is GitLab? | Definition from TechTarget. WhatIs. [Електронний ресурс] // Режим доступу: https://www.techtarget.com/whatis/definition/GitLab (дата звернення: 02.05.2025).

26. Інструменти розробника. Integrated Development Environment (IDE). Уроки для початківців. W3Schools українською. [Електронний ресурс] // Режим доступу: https://w3schoolsua.github.io/hyperskill/ide.html#gsc.tab=0 (дата звернення: 02.05.2025).

27. Microsoft. Visual Studio Code – Code Editing. Redefined. [Електронний ресурс] // Режим доступу: https://code.visualstudio.com/ (дата звернення: 02.05.2025).

28. Contributors to Wikimedia projects. Visual Studio – Wikipedia. Wikipedia, the free encyclopedia. [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Visual_Studio (дата звернення: 02.05.2025).

29. Найкращі програми для управління командою у 2024 році. Worksection. [Електронний ресурс] // Режим доступу: https://worksection.com/ua/blog/10-best-team-management-software.html (дата звернення: 02.05.2025).

30. What is Trello: Learn features, uses & more | Trello. Capture, organize, and tackle your to-dos from anywhere | Trello. [Електронний ресурс] // Режим доступу: https://trello.com/en/tour (дата звернення: 02.05.2025).

31. Jani, Yash. (2023). Implementing continuous integration and continuous deployment (ci/cd) in modern software development. International Journal of Science and Research (IJSR). 12. 2984-2987. 10.21275/SR24716120535.

32. Інструменти безперервної інтеграції: порівняння 7 найкращих продуктів. Atlassian. [Електронний ресурс] // Режим доступу: https://www.atlassian.com/en/continuous-delivery/continuous-integration/tools (дата звернення: 02.05.2025).

33. Підбірка найкращих інструментів для віддаленої роботи. eSputnik. [Електронний ресурс] // Режим доступу: https://esputnik.com/uk/blog/korysni-instrumenty-dlya-komandnoi-roboty (дата звернення: 02.05.2025).

34. Порівняння популярних корпоративних месенджерів | HURMA. [Електронний ресурс] // Режим доступу: https://hurma.work/blog/porivnyannya-populyarnih-korporativnih-mesendzheriv/ (дата звернення: 02.05.2025).

35. Essential tools for game design and development. Freelancer Blog. [Електронний ресурс] // Режим доступу: https://www.freelancermap.com/blog/essential-tools-for-game-design-and-development/ (дата звернення: 02.05.2025).

36. What is a gaming engine? Arm | The Architecture for the Digital World. [Електронний ресурс] // Режим доступу: https://www.arm.com/glossary/gaming-engines (дата звернення: 02.05.2025).

37. Salama, Ramiz & Elsayed, Mohamed. (2018). Basic elements and characteristics of game engine. Global Journal of Computer Sciences: Theory and Research. 8. 126-131. 10.18844/gjcs.v8i3.4023.

38. Tas, Nurullah & Bolat, Yusuf & Publication, Istes. (2023). Digital games and gamification in education.

39. The best gaming engines for 2024. Incredibuild. [Електронний ресурс] // Режим доступу: https://www.incredibuild.com/blog/top-gaming-engines-you-should-consider (дата звернення: 02.05.2025).

40. Yadav A. Godot vs Unity vs Unreal. Medium. [Електронний ресурс] // Режим доступу: https://medium.com/@amit25173/godot-vs-unity-vs-unreal-775fa7d91b2c (дата звернення: 02.05.2025).

41. Unity Vs Unreal Vs Godot – Comparison, Pros, Cons. AI, Games, Blockchain Development & Outsourcing Company. | imeta.io. [Електронний ресурс] // Режим доступу: https://imetatech.io/blog/unity-unreal-godot-comparison (дата звернення: 02.05.2025).

42. "Smart PM city" game development for educational purposes (Теза)

43. Khaldi, A., Bouzidi, R. & Nader, F. Gamification of e-learning in higher education: a systematic literature review. Smart Learn. Environ. 10, 10 (2023). [Електронний ресурс] // Режим доступу: https://doi.org/10.1186/s40561-023-00227-z дата звернення

44. Educators across the globe are using Minecraft: Education Edition for remote learning. education.minecraft.net. [Електронний ресурс] // Режим доступу: https://education.minecraft.net/en-us/blog/educators-across-the-globe-are-using-minecraft-education-edition-for-remote-learning (дата звернення: 02.05.2025).

45. Ways universities are using video games to learn. TeachThought. [Електронний ресурс] // Режим доступу: https://www.teachthought.com/technology/ways-universities-are-using-video-games-in-learning/ (дата звернення: 02.05.2025).

46. Лобунець Т.В. Конспект лекцій з дисципліни «Основи бізнес-проєктування» для студентів спеціальності 073 «Менеджмент» факультету аграрного менеджменту НУБіП / Т.В. Лобунець – К.:Вид. центр НУБіП, 2017. – 150 с.

47. Dukhnovskyi, Ivan. PEST-analysis as a tool for forming an enterprise development strategy. Jan. 2021, doi:10.32847/BUSINESS-NAVIGATOR.63-12.

48. Gürel, Emet & Tat, Merba. (2017). SWOT Analysis: A theoretical review. Journal of International Social Research. 10. 994-1006. 10.17719/jisr.2017.1832.

49. Управління проєктами: навч. посіб. / О.С. Войтенко. – Київ: КНУБА, 2020. – 276 с.

50. Конспект лекції. Мережеве планування та керування. Факультет управління фінансами та бізнесу. [Електронний ресурс] // Режим доступу: https://financial.lnu.edu.ua/wp-content/uploads/2019/09/ME-lektsiia-9.pdf (дата звернення: 02.05.2025).

51. Лисак В.Ю. Сутність заробітної плати та її значення в процесі економічного управління підприємством // Кам'янець-Подільський національний університет імені івана огієнка. – Улектронний журнал «Економіка та суспільство». 2015 – С.264-269.

52. Форми та системи оплати праці. Податки & бухоблік, № 50, Червень, 2016 | Factor. iFactor. [Електронний ресурс] // Режим доступу: https://i.factor.ua/ukr/journals/nibu/2016/june/issue-50/article-18968.html (дата звернення: 02.05.2025).

53. Бізнес-моделі: Від теорії до практики| BizMag [Електронний ресурс] // Режим доступу: https://bizmag.com.ua/biznes-model/

54. Godot Engine – Free and open source 2D and 3D game engine. Godot Engine. [Електронний ресурс] // Режим доступу: https://godotengine.org/ (дата звернення: 02.05.2025).

55. StyleBox. Godot Engine documentation. [Електронний ресурс] // Режим доступу: https://docs.godotengine.org/en/stable/classes/class_stylebox.html (дата звернення: 02.05.2025).

56. Diagram CASE Tool for Software Modeling & Analysis – UML, BPMN, ERD. Diagram CASE Tool for Software Modeling & Analysis – UML, BPMN, ERD. [Електронний ресурс] // Режим доступу: https://www.softwareideas.net/ (дата звернення: 02.05.2025).

ДОДАТКИ

Додаток А. Статут проєкту

Назва проєкту: Управління проєктом розробки програмного додатку щодо впровадження ігрових елементів в освітній процес

Спонсор проєкту: -

Дата розробки: 01.02.2025

Керівник проєкту: Карпов М.Є., Васюк А.В..

Замовник проєкту: -

Мета проєкту або обґрунтування:

Розробка ПЗ (гри) з використанням освітніх елементів для подальшого впровадження в освітній процес з метою підвищення рівня освіти та рівня зацікавленості до освітнього процесу зі сторони клієнтів (студентів).

Опис проєкту:

Програмний додаток (гра) спрямований на спеціальність пов'язану з менеджментом ресурсів різного характеру. Кінцевою метою гри є побудова ігрового міста, для якого треба виконувати проєкти різної направленості. Для виконання ж проєктів потрібні ресурси, що представленні у вигляді робітників різної спеціалізації та кваліфікації. Кожен проєкт та ресурс мають свої характеристики, які можуть змінюватись в залежності від дій гравця. Тому для досягнення основної мети, гравцю треба обирати оптимальні стратегії для успішної побудови ігрового міста. Крім менеджменту ресурсів, у гру-додаток інтегровані тести задля перевірки рівня знань гравця в області УП.

Вимоги високого рівня:

1 Функціональні вимоги:

- Розробка ігрової механіки з освітніми елементами.
- Автономна робота без підключення до інтернету.
- Збір ігрових даних для подальшого аналізу.
- Зручний інтерфейс для взаємодії з грою.

2 Нефункціональні вимоги:

- Підтримка на Windows із стабільною продуктивністю.
- Збереження ігрових даних локально або у вигляді файлу для експорту.

- Мінімальні апаратні вимоги для плавної роботи.
- Простий процес встановлення та запуску.
- Фокус на освітніх механіках без складних економічних чи соціальних елементів.

3 Технічні вимоги:

1. Мінімальні системні вимоги:

- ОС: Windows 10 (64-bit) або новіша
- Процесор: Intel Core i3 (4 покоління) або аналог AMD
- Оперативна пам’ять: 4 GB RAM
- Відеокарта: Вбудоване графічне ядро (Intel HD Graphics 4000 або вище)
- Місце на диску: 2 Гб вільного місця
- Роздільна здатність екрану: 1280x720 або вище

2. Вимоги до архітектури гри:

- Ігровий двигун: Godot (C#)
- Збереження даних: Локальні файли (JSON або XML)
- Інтерфейс: Адаптивний UI для різних роздільних здатностей

3. Вимоги до безпеки та збереження даних:

- Локальне збереження прогресу: Автоматичне та ручне збереження
- Формат файлів даних захищений від ручного редагування (опціонально)
- Можливість експорту даних: У CSV або JSON для аналізу

4 Бізнес-вимоги:

– Економічна ефективність: Обґрунтованість інвестицій, план окупності та можливість подальшого розвитку продукту.

– Відповідність ринку: Урахування потреб освітнього процесу, адаптація до змін у методиках навчання та вимог замовників.

– Легкість підтримки та оновлення: Забезпечення можливості швидкого реагування на зворотний зв’язок користувачів та внесення оновлень.

5 Організаційні вимоги

– Розробка системи та створення супутньої документації виконуються на основі стандарту ISO.

Ризики високого рівня:

Технічні ризики:

– Невизначеність вимог щодо ігрових механік та їх ефективності для освітнього процесу може призвести до перерозробок.

– Можливі проблеми з оптимізацією продуктивності на різних версіях Windows.

– Збереження та обробка даних без доступу до інтернету може ускладнити аналіз.

– Неможливість підтримки різних роздільних здатностей екранів.

– Можливе нерозуміння ігрової механіки користувачами.

Управлінські ризики:

– Невизначеність вимог: Часті зміни або нечіткість вимог замовника можуть призвести до затримок у розробці та збільшення бюджету.

– Неправильна оцінка термінів та ресурсів: Недостатня оцінка складності проєкту може вплинути на строки виконання та якість кінцевого продукту.

– Комунікація між зацікавленими сторонами: Неефективна взаємодія між розробниками, замовниками та кінцевими користувачами може спричинити непорозуміння та затримки.

– Відсутність чіткого визначення ефективності освітніх механік.

Бізнес-ризики:

– Низька залученість користувачів: Якщо кінцеві користувачі (студенти та викладачі) не приймуть або не зацікавляться новим підходом, інвестиції можуть не окупитися.

– Конкуренція: Висока конкуренція на ринку освітніх технологій може вплинути на комерційну успішність продукту.

– Зміни в законодавстві: Можливі зміни в нормативній базі (наприклад, щодо захисту даних) можуть вимагати додаткових ресурсів для адаптації продукту.

– Недостатня мотивація гравців для проходження гри.

Юридичні ризики:

– Необхідність дотримання законів про авторські права.

Таблиця А.1

Цілі проєкту

Цілі проєкту	**Критерії успіху**	**Особа, яка затверджує**
Розробити зручний та зрозумілий дизайн/UI за	Інтерфейс додатку розроблено відповідно до вимог, тести пройдено успішно. Можна прописувати бізнес логіку додатку	Дизайнер та Front-end dev.
Розробити ігрову логіку для додатку з можливістю тестування та налагодження.	Ігрові механіки працюють стабільно, Unit тести пройдено успішно. Ігрові та освітні елементи успішно інтегровані	Back end Devs
Впровадити в додаток освітні елементи відповідно до нефункціональних вимог.	Освітні елементи успішно інтегровані у додаток. Створено список тестових питань.	Консультант з впровадження освітніх елементів (викладач)
Розробити ефективний план тестування додатку цілком та його окремих компонентів.	Всі тести відповідають вимогам якості проєкту.	Front-end dev. Back end Devs Проджект менеджер

Збір ігрових даних для аналізу результативності процесу навчання.	Розроблено систему збору та аналізу даних кожної партії.	Консультант з впровадження освітніх елементів (викладач), Back end Devs.

Проєктна діяльність обмежена у часі: 16-18 місяців

Таблиця А.2

Заробітна плата працівників

Людські ресурси	Заробіня плата на 1 працівника, Стандартна ставка (грн/год)	Заробіня плата на 1 працівника, Понаднормова ставка (грн/год)
Дизайнер	156,25	200
Front-end developer	156,25	200
Back-end developer	175	210
QA тестувальник	125	150
Гейм-дизайнер	156,25	200
Менеджер проєкту	218,75	350
Sound designer	125	150
CTO	220	350
CEO	220	350

Таблиця А.3

Разові витрати

Разові витрати	
Консультація з впровадження освітніх елементів	10 000,00

Таблиця А.4

Розрахунок витрат на людські ресурси

Людські ресурси	Розрахована робота (г)	Загальна витрати (грн) (за стандартною ставка)
Дизайнер	1 008	157 500,00
Front-end developer	1 264	197 500,00
Back-end developer	1 328	232 400,00
Back-end developer	1 328	232 400,00
QA тестувальник	416	52 000,00
QA тестувальник	416	52 000,00
Гейм-дизайнер	408	63 750,00
Менеджер проєкту	1 112	243 250,00
Sound designer	176	22 000,00
CTO	1 288	283 360,00
CEO	1 224	269 280,00
ВСЬОГО:	9968	1 805 440

Таблиця А.5

Матеріальні ресурси

№ п/п	Назва	од. виміру	Кількість	Ціна за одиницю, грн (₴)	Сума, грн (₴)
1	Офіс (оренда)	міс	12	20 000,00	240 000
2	Комп'ютер	шт.	4	15 000,00	60 000
3	Монітор	шт.	4	5 000,00	20 000
4	Ноутбук	шт.	4	15 000,00	60 000
5	ОС Windows 10 Pro	копія	8	6 500,00	52 000
6	Wifi router	шт.	1	4 000,00	4 000
7	Ethernet 300 Mbit	міс	12	450,00	5 400

8	Стіл	шт.	8	1 100,00	8 800
9	Стілець	шт.	8	1 000,00	8 000
10	Гарнітура для персонального комп'ютеру (миш та клавіатура)	шт.	8	800,00 ₴	6 400
11	Adobe Creative Cloud License	міс	10	1 300,00 ₴	13 000
Всього					477600

Таблиця А.6

Загальні витарти:

Ресурс	Витрати (грн)
Матеріальні ресурси	477 600,00
Разові витрати	10 000,00
Людські ресурси	1 805 440,00
Всього без непередбачувані витрат	2 293 040,00
Непередбачувані витрати (10%)	229 304
Всього	2 468 344

Таблиця А.7

Віхи проєкту

Основні віхи проєкту	Строк (дата)
(Початок)Розробка концепцію проєкту	01.02.2025
Організація проєкту	01.03.2025
Планування і проєктування проєкту	01.04.2025
Визначення параметрів модульного/рівневого дизайну додатку	01.05.2025

Формування команди	01.07.2025
Розробка демонстраційної версії додатку	01.08.2025
Тестування	01.12.2025
Внесення корективів	01.01.2026
Розробка кінцевої версії продукту	01.02.2026
Тестування	01.02.2026
Презентація продукту	01.05.2026
Завершити проєкт	01.06.2026

Ймовірний бюджет (estimated): Діяльність має обмеження бюджету (2 500 000 грн.) для виконання робіт та найму ресурсів

Таблиця А.8

Зацікавлені сторони

Зацікавлена сторона	Роль
Викладачі	• Відповідають за подальше впровадження. • Є фактором формування функціональних та нефункціональних вимоги до додатку.
Студенти	• Використовують продукт та надають відгуки. • Є фактором формування функціональних та нефукціональних вимоги до додатку.
Керівництво університету	• Перевірка відповідності продукту цілям • Створює можливості для якісної реалізації продукту • Забезпечує фінансування продукту
Команда розробки	• Забезпечує розробку продукту • Забезпечує якість продукту • Забезпечує успішне впровадження і просування продукту • Тестує та налагоджує систему.

Додаток Б. Календарно-мережевий графік проєкту

Рис. Б.1. Календарно-мережевий графік

Рис. Б.2. Календарно-мережевий графік (продовження)

Рис. Б.3. Календарно-мережевий графік (продовження)

Рис. Б.4. Календарно-мережевий графік (продовження)

Рис. Б.5. Календарно-мережевий графік (продовження)

Рис. Б.6. Календарно-мережевий графік (продовження)

Рис. Б.7. Календарно-мережевий графік (продовження)

Додаток В. UML діаграми класу

Рис. В.1-а. Діаграма класів «Global»

Рис. В.1-б. Діаграма класів «Global» (продовження)

Рис. В.2. Діаграма класів «Global» (продовження)

Рис. В.3. Діаграма класів «Global» (продовження)

Рис. В.4. Діаграма класів «Controllers»

Додаток Г. Приклад коду гри. Скрипт GameField.cs

```
using Godot;

public partial class GameField : Node2D
{
    PackedScene BufferCard =
ResourceLoader.Load<PackedScene>("res://Scenes/Buffer.tscn");
    PackedScene MsgBoxScene =
ResourceLoader.Load<PackedScene>("res://Scenes/AlertBox.tscn");

    public override void _Ready()
    {
        StartNEWGame();
        Questions_Mngr.QuestionsLoader.LoadQuestions();
    }

    public void StartNEWGame(){

        SetDefauldGameDATA();
        SetSceneToTakeNewCards();
        TakeCardsFromDeck(3);   //defauld 3
        UpdateInfoPanels();
    }


    public void StartNEXTTurn()
    {

        GlobalVar.CurrentRound++;

       {GlobalVar.CurrentRound}");

        ResetFields();
        RedrawAllCards();
        SetSceneToTakeNewCards();
        TakeCardsFromDeck(2);
        UpdateInfoPanels();
    }


    private void SetDefauldGameDATA(){
        GlobalVar.ResetUIDGiver();
        GlobalVar.ResetProgression();
        GetNode<PlayerHand>("Player_hand").ClearHand();
        GetNode<ProjectsField>("Container_MiddleField/Projects").MakeProjectsActive(G
lobalVar.CurrentLVL);
    }
```

```
    public void SetSceneToTakeNewCards(){
        SetScene();
    }

    public void SetSceneToContinueGame()
    {
        SetScene(false);
        SetDeckCount();
    }


    private void SetScene(bool isEndTurn = true){
        GetTree().Root.GetNode<Control>("Node2D/Container_MiddleField/Projects").Visi
ble = !isEndTurn;
        GetTree().Root.GetNode<BoxContainer>("Node2D/Container_MiddleField/CardBuffer
").Visible = isEndTurn;
        GetTree().Root.GetNode<CardsBuffer>("Node2D/Container_MiddleField/CardBuffer"
).BtnTurn(isEndTurn);

        GetTree().Root.GetNode<Control>("Node2D/Market&Discard/Discard").Call("Discar
dCardActive", isEndTurn);
        GetTree().Root.GetNode<Button>("Node2D/Btn_EndTurn").Visible = !isEndTurn;
    }

    public void TakeCardsFromDeck(int CountOfCardsToTake)
    {
        GetTree().Root.GetNode<CardsBuffer>("Node2D/Container_MiddleField/CardBuffer"
).SpawnResourceCards(CountOfCardsToTake);
    }

    public void TakeCardsToHand()
    {
        GetNode<PlayerHand>("Player_hand").ReSpawnCards();
        SetSceneToContinueGame();

    }

    public void ResetFields()
    {
        ResetBuffs();
        GetTree().Root.GetNode<PlayerHand>("Node2D/Player_hand").ReSpawnCards();
    }
    private void ResetBuffs()
    {

        BuffManager.Instance.ProgressTemporaryBuffTimer();
        BuffManager.Instance.ResetBuffs();
        GetNode<ListMenuOfBuffs>("%BuffsList").ReDrawBuffs();
    }
```

```
    public void RedrawAllCards()
    {
        GetTree().Root.GetNode<PlayerHand>("Node2D/Player_hand").ReDrawAllCards();
        GetTree().Root.GetNode<ProjectsField>("Node2D/Container_MiddleField/Projects"
).RedrawAllCards();
        UpdateInfoPanels();
    }

    public void ToMainMenu(bool SaveGame)
    {

        PackedScene Menu =
ResourceLoader.Load<PackedScene>("res://Scenes/Title_screen_scene.tscn");
        GetTree().Root.GetNode<PlayerHand>("Node2D/Player_hand").ClearHand();

        if (SaveGame)
        {
            //TODO: Save game and load Main menu
            // Reset all Parameters
            GetTree().ChangeSceneToPacked(Menu);
        }
        else
        {
            GetTree().ChangeSceneToPacked(Menu);
        }
    }

    public bool IsCardOnBuffer(int CardID)
    {
        return CardsMngr.GetCardData(CardID).IsOnBuffer;
    }
    public void CardRemovedFromBuffer(int CardID)
    {
        CardsMngr.GetCardData(CardID).IsOnBuffer = false;
    }
    public int GetCardsArrLength()
    {
        return GlobalVar.CardArrLength;
    }

}
```

Додаток Д. Приклад коду гри. Скрипт PlayerHand.cs

```
using Godot;
using System;
using System.Collections.Generic;
using System.Linq;

public partial class PlayerHand : HBoxContainer
{
    // Called when the node enters the scene tree for the first time.

    readonly PackedScene card =
ResourceLoader.Load<PackedScene>("res://Scenes/Cards/Card.tscn");

    /// <summary>
    /// Called by GDScript
    /// </summary>
    public bool IsCardOnHand(int CardID)
    {

        var Card = GetChildren();
        for (int i = 0; i < Card.Count; i++)
        {
            if (Card[i] is Container && ((int)Card[i].Get("CardID") == CardID))
            {
                return true;
            }
        }
        return false;
    }

    public void MoveCardsToDiscard(params int[] CardID)
    {


        foreach (int item in CardID){
            CardsMngr.Remove(item);
        }



        var Cards = GetChildren();
        for (int i = 0; i < Cards.Count; i++)
        {
            for (int j = 0; j < CardID.Length; j++)
            {
                if (Cards[i] is Container && ((int)Cards[i].Get("CardID") ==
CardID[j]))
                {
```

```
                    Cards[i].QueueFree();
                }
            }

        }

        GetTree().Root.GetNode<BoxContainer>("Node2D/Market&Discard/Discard").Call("D
iscardCardActive", false);
    }

    public void ReDrawAllCards()
    {
        var GetCardsContainers = GetChildren();

        for (int i = 0; i < GetCardsContainers.Count; i++)
            GetCardsContainers[i].Call("DrawLabels");

        if (GetCardsContainers.Count > 7)
        {
            var Separation = (GetCardsContainers.Count - 5) * -7;
            this.AddThemeConstantOverride("separation", Separation);
        }

    }


    public void PlaySelectAnimation(params ResourceCardInfo[] ResCards)
    {
        var Cards = GetChildren();
        List<int> CardIDs = [.. ResCards.Select(x => x.CardID)];

        for (int i = 0; i < Cards.Count; i++)
            if (Cards[i].Get("CardID").Obj != null &&
CardIDs.Contains((int)Cards[i].Get("CardID")))
                Cards[i].Call("AnimSelect");


    }
    public void PlayDeselectAnimation(params ResourceCardInfo[] ResCards)
    {
        var Cards = GetChildren();
        List<int> CardIDs = [.. ResCards.Select(x=>x.CardID)];

        for (int i = 0; i < Cards.Count; i++)
            if (Cards[i].Get("CardID").Obj != null &&
CardIDs.Contains((int)Cards[i].Get("CardID")))
                Cards[i].Call("AnimDeselect");

    }
```

```
    public void ReSpawnCards()
    {
        PackedScene Card =
ResourceLoader.Load<PackedScene>("res://Scenes/Cards/Card.tscn");
        var CardsToRespawn = CardsMngr.CardsInHandIDNow();

        while (GetChildren().Count > 0)
        {
            GetChildren()[0].Free();
        }
        for (int i = 0; i < CardsToRespawn.Count; i++)
        {
            var NewCard = Card.Instantiate();
            NewCard.Set("CardID", CardsToRespawn.ElementAt(i));
            AddChild(NewCard);
        }
    }

    public void ReSpawnCard(int ResCardID)
    {
        var Card = CardsMngr.GetCardData(ResCardID);
        var Cards = GetChildren();

        for (int i = 0; i < Cards.Count; i++)
            if ((int)Cards[i].Get("CardID") == Card.CardID &&
Cards[i].Get("ConnectedToProjID").Obj != null)
            {
                Cards[i].Call("DisConnectCard");
            }
    }

    public void ResetResourceCardsArray()
    {
        ResourceCardArr.CopyTo(GlobalVar.CardResourceArr);
    }


    public void ClearHand()
    {
        CardsMngr.Clear();
        ReSpawnCards();
    }

    public void CardAdded(int CardID, int ConnectedProjectID)
    {
        CardsMngr.Add(ConnectedProjectID, CardID);

        if (ConnectedProjectID != 0 && GetTree().Root != null)
        {
```

```
            BuffManager.Instance.ApplyBuffsToCard(CardsMngr.GetProjectData(ConnectedP
rojectID));
            BuffManager.Instance.ApplyBuffsToCard(CardsMngr.GetCardData(CardID));

            BuffManager.Instance.ActivateBuffs();
            GetTree().Root.GetNode<GameField>("Node2D").RedrawAllCards();
            GetTree().Root.GetNode<ListMenuOfBuffs>("Node2D/%BuffsList").ReDrawBuffs(
);
        }
        GetTree().Root.GetNode<GameField>("Node2D").UpdateInfoPanels();


    }

    public void CardRemoved(int CardID, int ConnectedProjectID)
    {

        CardsMngr.Remove(CardID, ConnectedProjectID);

        if (ConnectedProjectID != 0 && GetTree().Root != null)
        {
            BuffManager.Instance.RemoveBuffsFromCard(CardsMngr.GetCardData(CardID));
            BuffManager.Instance.RemoveBuffsFromCard(CardsMngr.GetProjectData(Connect
edProjectID));

            GetTree().Root.GetNode<GameField>("Node2D").RedrawAllCards();
            GetTree().Root.GetNode<ListMenuOfBuffs>("Node2D/%BuffsList").ReDrawBuffs(
);
        }
    }

    public bool CanTakeMoreCards()
    {

        if (CardsMngr.AllCardInHandCount() <
GlobalVar.CurrentOfficeLVL.Last().MaxCountOfResources)
        {
            return true;
        }
        else
        {
            GetTree().Root.GetNode<GameField>("Node2D").CallMessageBox("#Забагато
ресурсів", "Досягнутий макс. ліміт ресурсів. Більше найняти неможливо. Для більшого
штату потрібен кращій офіс");
            return false;
        }
    }

}
```

Додаток Е. Приклад коду гри. Скрипт ProjectsField.cs

```
using System;
using System.Linq;
using Godot;

public partial class ProjectsField : Control
{
    public override void _Ready()
    {
        .Log($"_____ProjectCard.LoadProjectCards()_____");  CardsProject =
LoadProjectCards(GetChildren());
        SetProjectCards();
    }

    public void SetProjectCards()
    {
        LoadProjectsTextures();
        MakeProjTransperent();
        OnMouseProjectScaling();
    }

    public void RedrawAllCards()
    {
        for (int i = 0; i < CardsProject.Length; i++)
        {
            if (CardsProject[i].GetChildCount() == 0)
                continue;

            var ProjectCard = CardsProject[i].GetChild(0) as Control;
            int ProjID = (int)CardsProject[i].GetMeta("ProjID");
            FillProjLabels(ProjID, ProjectCard);
        }

    }

    public void MakeProjectsActive(int LVL)
    {

        if (LVL < 1 && LVL > 4)
            return;

        for (int i = 1; i <= LVL; i++)
        {
            MakeProjsActive(i);
        }

    }
```

```
    private void MakeProjsActive(int LVL)  // ffffff78 halfTransperent : ffffff
nonTransperent
    {
        foreach (var item in CardsProject)
        {

            if ((int)item.GetMeta("ProjLVL") == LVL)
            {
                int ProjID = (int)item.GetMeta("ProjID");

                var ProjCard = CardsMngr.GetProjectData(ProjID);

                if (ProjCard.isActive)
                    return;

                ProjCard.isActive = true;
                item.Modulate = Color.Color8(255, 255, 255, 255);
                item.GetChild<Control>(0).GetNode<BoxContainer>("EventHandler").Mouse
Entered += () =>
                {
                    GlobalVar.MouseOverProject = true;
                    GlobalVar.LastSelectedProjectID = ProjID;
                };
                item.GetChild<Control>(0).GetNode<BoxContainer>("EventHandler").Mouse
Exited += () =>
                {
                    GlobalVar.MouseOverProject = false;
                    GlobalVar.LastSelectedProjectID = 0;
                };
            }
        }
    }
    public void CloseFinishedProj(int[] FinishedProjID)
    {
        for (int j = 0; j < FinishedProjID.Length; j++)
        {
            for (int i = 0; i < CardsProject.Length; i++)
            {
                if ((int)CardsProject[i].GetMeta("ProjID") == FinishedProjID[j])
                {
                    CardsProject[i].GetChild<Control>(0).QueueFree();

                    for (int k = 0; k < GlobalVar.CardProjectArr.Length; k++)
                        GlobalVar.CardProjectArr[k].ProjSelected = false;
                }
            }
        }
    }
}
```