

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

САВЧУК Дмитро Володимирович

**Мобільний застосунок для створення та проходження AR-квестів із
використання геолокації / Mobile Application for Creating and Performing AR
Quests Using Geolocation**

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
Д. В. Савчук

Науковий керівник
к.т.н., доцент П. Є. Биковий

Кваліфікаційну роботу допущено до
захисту

«___»_____ 20___ р.

Завідувач кафедри Н. В. Дзюбановська

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____ 20__р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
САВЧУКУ Дмитру Володимировичу**

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи (укр. / англ. мовами)

Мобільний застосунок для створення та проходження AR-квестів із використанням геолокації / Mobile Application for Creating and Performing AR Quests Using Geolocation

керівник роботи к.т.н., доцент, Биковий Павло Євгенович,
затверджені наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу, науково-технічна література з принципів побудови систем доповненої реальності (AR) та геоінформаційних систем; офіційна документація середовища розробки Unity та супутніх програмних інтерфейсів (SDK).

4. Основні питання, які потрібно розробити:

- аналіз предметної області, огляд існуючих аналогів та обґрунтування вибору технологій розробки;
- проектування архітектури системи та структури даних для зберігання інформації про квести й точки інтересу;
- розроблення мобільного застосунку для створення та проходження AR-квестів із використанням геолокації;
- проведення функціонального тестування та оцінювання працездатності розробленого програмного забезпечення.

5. Перелік графічного матеріалу у роботі:

- схема алгоритму функціонування системи;
- схема алгоритму підсистеми конструювання;
- схема ігрового алгоритму;
- схема алгоритму обміну;
- інтерфейси аналогів;
- UML-діаграма класів основної бізнес-логіки;
- UML-діаграма станів графічного інтерфейсу користувача;
- структурні схеми та скріншоти роботи програми.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студент

(підпис)

Савчук Д. В.

(прізвище та ініціали)

Керівник роботи

(підпис)

Биковий П. Є.

(прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Мобільний застосунок для створення та проходження AR-квестів із використанням геолокації» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 86 сторінок і містить 24 ілюстрації, 8 додатків та 33 використаних джерела.

Метою роботи є проєктування та розроблення мобільної програмної системи для створення, редагування та проходження просторових квестів із використанням технологій геолокації та доповненої реальності.

Методами розроблення обрано методи об'єктно-орієнтованого проєктування (для побудови архітектури системи), методи просторового моделювання (для розрахунку географічних координат та дистанцій), методи реляційної алгебри (для організації локальної бази даних) та методи тестування програмного забезпечення (для оцінювання працездатності AR-модулів та графічного інтерфейсу).

Внаслідок виконання роботи обґрунтовано раціональний вибір програмних засобів для реалізації AR-систем з геолокацією, спроектовано децентралізовану архітектуру та розроблено мобільний застосунок, який дає змогу користувачам конструювати просторові сценарії, обмінюватися ними та проходити квести з відображенням 3D-об'єктів у реальному світі.

Результати дослідження можуть бути використані в туристичній галузі для створення інтерактивних екскурсій, в освітніх закладах для проведення краєзнавчих заходів, а також у сфері розважальної індустрії.

Ключові слова: ДОПОВНЕНА РЕАЛЬНІСТЬ, ГЕОЛОКАЦІЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, КВЕСТ, UNITY, ПРОСТОРОВЕ МОДЕЛЮВАННЯ.

ANNOTATION

Qualification work on the topic «Mobile Application for Creating and Performing AR Quests Using Geolocation» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 86 pages and it contains 24 figures, 8 annexes and 33 sources.

The purpose of the work is to design and develop an autonomous mobile software system for creating, editing, and playing quests using geolocation and augmented reality technologies.

Research methods are object-oriented design methods (to build system architecture), spatial modeling methods (to calculate geographic coordinates and distances), relational algebra methods (to organize the local database), and software testing methods (to estimate the functionality of AR modules and graphic interface).

As a result of the work the rational approach to the choice of software tools for implementing Outdoor AR systems was substantiated. Also, the decentralized architecture was designed and the mobile application, which allows users to autonomously construct spatial scenarios, exchange them and play quests with 3D models displaying in the real world, was developed.

The results of the research can be used in the tourism industry to create interactive excursions, in educational institutions for local history events, and in the entertainment industry.

Keywords: AUGMENTED REALITY, GEOLOCATION, MOBILE APPLICATION, QUEST, UNITY, SPATIAL MODELING.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної області створення та проходження AR-квестів	9
1.1 Опис предметної області створення та проходження AR-квестів.....	9
1.2 Огляд і аналіз існуючих аналогів.....	14
1.3 Постановка задачі дослідження	19
2 Алгоритмічне та інформаційне забезпечення системи	21
2.1 Загальна структура проєктованої системи.....	21
2.2 Алгоритмічне забезпечення	24
2.3 Інформаційне забезпечення.....	30
3 Програмно-технологічне забезпечення системи.....	36
3.1 Реалізація програмного забезпечення	36
3.2 Інтерфейс користувача	43
3.3 Тестування розробленого мобільного застосунку	47
Висновки	51
Список використаних джерел	53
Додаток А Схема алгоритму функціонування системи.....	56
Додаток Б Схема алгоритму модуля конструювання квестів	57
Додаток В Схема алгоритму ігрової системи	58
Додаток Г Схема алгоритму модуля обміну	59
Додаток Д UML-діаграма класів основної бізнес-логіки.....	60
Додаток Е UML-діаграма станів графічного інтерфейсу	61
Додаток Ж Лістинг програмного коду.....	62
Додаток И Апробація отриманих результатів	81

ВСТУП

Розвиток мобільних технологій та технологій доповненої реальності надають нові інтерактивні можливості для взаємодії людини з навколишнім середовищем. Сучасні мобільні пристрої є досить потужними та оснащені засобами, що дозволяють визначати пристрій на місцевості та в просторі, що є дуже корисним для створення додатків з доповненою реальністю.

Вже зараз дозвілля та освітні заходи переходять у цифровий простір, а квести на основі AR-геолокації є зручним та цікавим інструментом, здатним покращити досвід учасників як у розважальній, так і в освітній сферах. Подібні рішення допомагають легко поєднувати ігровий процес та фізичну активність з навчанням, що дуже актуально для неформального навчання, туризму та міських заходів.

Хоча існуючі проекти є досить популярними, більшість з них є закритими або орієнтовані на розробників і вимагають наявності навичок програмування. У мережі бракує інструментів, які могли б дати звичайному користувачу можливість створювати власні квести, налаштовувати їх параметри та ділитися ними з іншими. Важливим завданням є створення системи, яка б поєднувала в собі технології комп'ютерного зору (на базі Niantic Lightship), геопросторові дані (ArcGIS) та ефективне управління локальними даними.

Мета роботи – розробити мобільний застосунок для створення та проходження інтерактивних квестів із використанням технологій доповненої реальності та прив'язкою до географічних координат на базі Unity.

Об'єкт дослідження – процеси візуалізації об'єктів доповненої реальності та обробка геопросторових даних у мобільних системах.

Предмет дослідження – методи та програмні засоби поєднання AR-технологій, картографічних сервісів і локальних баз даних для створення зрозумілих систем, орієнтованих на звичайного користувача.

Для досягнення поставленої мети були визначені такі завдання:

- аналіз сучасних технологічних рішень поєднання AR та геолокації у мобільних додатках;

- проєктування архітектури системи та моделі бази даних для зберігання параметрів квестів;
- розробка модуля візуалізації AR-об'єктів з використанням платформи Niantic Lightship;
- проєктування та створення інтерфейсу додатка та відображення міні-карти, використовуючи інструменти Unity та сервісу ArcGIS;
- розробка модуля обміну квестами між користувачами;
- тестування функціональних можливостей та стабільності роботи додатка.

Методи дослідження. Для вирішення поставлених задач використано методи об'єктно-орієнтованого програмування, принципи проєктування реляційних баз даних, алгоритми обробки GPS-даних, а також методи відображення AR-об'єктів у просторі.

Створення готової системи дозволить швидко та легко створювати інтерактивні квести, що може використовуватись як інструмент для розробки туристичних маршрутів, освітніх заходів чи змагальних ігор з використанням технологій доповненої реальності.

Результати кваліфікаційної роботи апробовано на I Міжнародній науково-практичній конференції студентів і молодих вчених «Інтелектуальні обчислювальні системи та управління проєктами (ICSPM 2026)» (м. Тернопіль, 21–22 травня 2026 р.). За матеріалами дослідження опубліковано тези доповіді «Мобільний застосунок для створення та проходження AR-квестів із використанням геолокації».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СТВОРЕННЯ ТА ПРОХОДЖЕННЯ AR-КВЕСТІВ

1.1 Опис предметної області створення та проходження AR-квестів

У наш час поєднання реального та цифрового світу стає все досконалішим. Сучасні технології вже зараз дозволяють доповнити фізичний простір цифровою інформацією і надати людині спосіб взаємодіяти з нею. Головною причиною розвитку є сильне розповсюдження доволі потужних мобільних телефонів. Сучасні гаджети оснащені такими модулями, як: GPS, акселерометри, гіроскопи та інші сенсори. Серед них найважливішими є високоякісні камери, котрі здатні чітко передавати зображення навіть у русі. Такі можливості сучасних гаджетів є ідеальним підґрунтям для розвитку таких сфер, як: геоінформаційні сервіси (Location-Based Services, LBS) та технологій доповненої реальності (Augmented Reality, AR).

Предметною областю даної кваліфікаційної роботи є сфера проектування та створення додатків, які поєднують навігацію у реальному просторі та інтерактивні елементи віртуального середовища, розраховані на компактні гаджети, такі як телефон чи планшет. Подібне називають доповнена реальність з прив'язкою до місцевості або “Location-based AR” [1, 2]. Доповнена реальність має велику відмінність від класичних віртуальних світів (VR), які повністю ізолюють користувачів в цифровому просторі, без жодної взаємодії з реальним оточенням. AR-технології доповнюють фізичний світ цифровим контентом, який часто залежить від положення користувача в просторі [2, 3].

Доповнена реальність – це простір, де комбінуються фізичні та віртуальні об'єкти в реальному часі. Класичні AR-системи повинні поєднувати реальне з віртуальним, взаємодіяти у реальному часі та працювати у тривимірному просторі (3D). Для роботи AR-системи використовується відеопотік з камери пристрою. Цей потік відеоданих доповнюється AR-контентом і демонструється користувачу через екран смартфона чи планшета [1].

Для функціонування AR-систем використовуються такі технології, як комп'ютерний зір (Computer Vision) та алгоритми SLAM (Simultaneous Localization

and Mapping – одночасна локалізація та побудова карти). SLAM дозволяє визначити положення пристрою та формувати карту оточення, на основі якої можуть визначатись площини та положення пристрою відносно них. Подібна інформація є дуже важливою, адже дозволяє контролювати віртуальні об'єкти (3D-моделі, вказівники) і не прив'язуватись до конкретних QR-кодів чи зображень-маркерів. Саме на таких алгоритмах побудовані сучасні SDK, такі як ARKit, ARCore чи Niantic Lightship, що дозволяють створювати AR-системи [4].

Не менш важливими для створення AR-квестів є геоінформаційні системи (ГІС). Перш за все йдеться про обробку просторових даних. Мобільний пристрій майже завжди має доступ до супутникових систем навігації (GPS, Galileo тощо), які визначають широту та довготу гаджета. Також ГІС використовується для створення елементів інтерфейсу, таких як міні-карта.

Використання звичайного GPS без додаткової обробки не завжди є достатнім для створення AR-квестів [5]. Зазвичай у містах точність GPS-сигналу має похибку 5-15 метрів, що може бути багато, якщо в системі потрібне точне розташування віртуальних об'єктів. В такому випадку доцільно використовувати декілька методів навігації одночасно, наприклад поєднувати GPS для глобального позиціонування та візуальне позиціонування (VPS) для точного відображення об'єкта перед користувачем [6]. Для кращої навігації користувача по точках інтересу (Point of Interest, POI) мобільний застосунок повинен мати карту, яка інформує гравця про об'єкти довкола нього та дозволяла планувати маршрут. Для реалізації подібного використовують картографічні сервіси, як ArcGIS, Google Maps, OpenStreetMap та інші.

Подача інформації в ігровій формі є чудовою практикою, яка допомагає легше доносити важку або нудну інформацію. AR-квести є одним із найкращих методів застосування гейміфікації в навчанні чи побутових сферах. Доповнення звичного простору віртуальними об'єктами підвищує інтерес у людей [5]. Успішним прикладом поєднання гейміфікації з AR є гра Pokémon GO, яка поєднує звичайні AR-об'єкти з локаціями реального світу. Головна ідея – пошук персонажів на

реальній місцевості [7]. Подібний застосунок спонукає людей займатися прогулянкою, подаючи її в ігровій формі.

AR-квести також застосовують в інших сферах, наприклад:

- Освіта. Навчальні заклади здатні подавати інформацію через доповнену реальність. Цифрові об'єкти доносять учням необхідну інформацію та допомагають відобразити її у графічному вигляді [8].

- Туризм. Відображення історичних постатей і локацій через віртуальні об'єкти. Також проєктування маршрутів екскурсій на певній місцевості [9].

- Маркетинг. Використання міток та інформації, яка необхідна покупцю в певній сфері, наприклад 3D-модель та інформація автомобіля чи маршрут до певного товару в магазині.

Розвиток сучасних технологій в сфері GPS та AR надає безліч можливостей для створення будь-якого квесту, який буде поєднувати віртуальні об'єкти та реальне розташування. Проте звичайна людина, яка не має жодного відношення до програмування, не має можливості самостійно створювати подібні рішення.

Для вирішення цієї проблеми та збільшення контенту, створеного користувачами (User Generated Content), пропонується створення додатків-редакторів, які дозволять легко створювати AR-квест з прив'язкою до місцевості. Подібний інструмент повинен давати людині можливість самостійно розміщувати точки на карті, прив'язувати до них AR-об'єкти, придумувати сценарії проходження та мати можливість поділитись створеним контентом. Інтерфейс (UI/UX) подібного додатку повинен бути інтуїтивно зрозумілим, оскільки ним користуватимуться люди без досвіду програмування. Система повинна мати надійні модулі обробки GPS та AR-даних і продуману структуру баз даних (SQLite, MySQL та інші).

Отже, для створення додатків, пов'язаних з AR-квестами, потрібне розуміння алгоритмів обробки просторових даних, методів розміщення AR-об'єктів, структури баз даних та апаратних можливостей мобільних пристроїв.

1.1.1 Моделювання процесів функціонування системи та виділення функціональних складових системи

Основним процесом, що реалізується в системі, є повний цикл створення та використання AR-контенту, прив'язаного до координат місцевості. Головним завданням є поєднання процесів створення AR-контенту, його візуалізація та використання в одній системі, яка буде працювати в реальному часі.

Для моделювання системи слід використовувати транзакційну та аналітичну складові процесу функціонування системи.

У межах проєктованої системи, транзакційна складова має виконувати наступні функції:

- збір даних, де система отримує координати (довгота, широта) точки при розміщенні на карті;
- реєстрація сутностей в базі даних, коли при створенні квесту чи додаванні точки, система повинна оновлювати чи створювати інформацію в базі даних;
- обмін даними між користувачами, експорт та імпорт файлів, які містять інформацію про квест (назва, точки, координати та інше);
- фіксація прогресу, де здійснюється постійний моніторинг змін під час проходження та оновлення відповідної інформації в базі даних;
- валідація даних, коли відбувається перевірка коректності введених користувачем даних перед їх збереженням у системі. Якщо є якісь порушення, система повинна або інформувати користувача або не давати ввести некоректні дані.

Аналітична складова відповідає за інтерпретацію просторових даних, необхідних для функціонування системи. До такого слід віднести аналіз розташування, а саме розрахунок відстані між об'єктами та користувачем на основі геодезичних розрахунків. Не менш важливим в системі, пов'язаній з AR-квестами, є обробка та відображення AR-контенту. Система забезпечує обробку даних, отриманих від AR-модуля. Ця частина відповідає за коректне розташування AR-об'єктів відповідно до заданих координат.

Наступна діаграма дерева функцій (рисунок 1.1) демонструє необхідний функціонал для реалізації досліджуваного процесу.



Рисунок 1.1 – Дерево функцій

У результаті моделювання можна виділити функції, які система повинна виконувати обов'язково:

- функції управління даними, що здатні повністю контролювати інформаційні об'єкти системи (створення, редагування, видалення, збереження);
- функції навігації, що містять методи візуалізації контенту, пов'язаного з орієнтацією на місцевості (карти, дистанція) і використовуються для навігації користувача;
- функції візуалізації, що використовуються для забезпечення інтерфейсу та методів взаємодії з об'єктами доповненої реальності.

1.2 Огляд і аналіз існуючих аналогів

У наш час існує досить багато проєктів, які використовують AR або геоінформаційні сервіси (LBS). Різноманітність цих систем дозволяє проаналізувати переваги та недоліки їхніх модулів, що дозволить використовувати найкраще у власній розробці. Для дослідження було проаналізовано декілька програмних продуктів, які мають подібний функціонал до розроблюваної системи.

Найкращим прикладом для власної системи може стати проєкт Pokémon GO від компанії Niantic, Inc. Даний продукт є одним з найкращих додатків, які використовують доповнену реальність та подають її в ігровій формі. Він також використовує розташування, щоб направляти гравця до точок інтересу.

Pokémon GO пропонує користувачам шукати мітки на карті, по якій рухається їхній аватар. Карта наповнена різними точками інтересу, які гравець може відвідати. Ці мітки прив'язані до певних координат реальності до яких користувач повинен фізично дістатися. Карта стилізована під ігровий світ і відображається через ізометричну проекцію, що дозволяє дивитись на картинку, наче вона 3D [7, 10].

Головними точками інтересу в грі є PokéStops та Gyms, які відображають розташування покемонів або зали. Для різних ситуацій, використовуються різні моделі, а саме: сині та фіолетові мітки, для місць з покемонами та червоно-білі вежі для залів тренерів [7]. Різні моделі та кольори дозволяють гравцю легко розуміти, що саме потрібно шукати і куди йти.

Коли людина знаходиться поруч з ціллю, гра пропонує спіймати покемона. У цей момент гра переходить до AR-сцени, демонструючи модель покемона перед гравцем. Щоб спіймати істоту, потрібно влучити покеболом, який також є моделькою доповненої реальності. В режимі AR, гравець має змогу сфотографувати покемона. Для коректного відображення AR-об'єктів, застосунок користується гіроскопом та камерою гаджета [7].

З 2017 року в грі активно з'являлись оновлення, які дозволяли гравцям проходити квести разом, через рейди і тренувальні зали [7]. Подібна ідея спонукає людей залучати своїх друзів до гри, роблячи гру популярнішою і цікавішою.

Основні екранні форми додатку (рисунок 1.2) [11]:

- головний екран (карта), що відображає місцевість, точки інтересу, аватар користувача та навігаційні елементи. Даний розділ є головним екраном взаємодії системи з користувачем;
- екран взаємодії з об'єктом, що відображає 3D-сцену з об'єктом, елементи керування (покебол, меню). Використовується для взаємодії з 3D-контентом;
- екран профілю/інвентарю, що містить інформацію про гравця та покемонів.



Рисунок 1.2 – Зображення екранів додатку Pokémon GO [11]

Найпопулярніший продукт Niantic демонструє, як слід використовувати 3D об'єкти та карту в ігрових додатках. Карта повинна демонструвати усі необхідні місця інтересу та допомагати користувачу з навігацією. 3D повинен бути стабільним та точно відповідати даним на карті. Також Pokémon GO чудово демонструє приклад гейміфікації і як слід зацікавлювати користувачів.

Не менш інформативним прикладом для розроблювальної системи може стати проект FitHub, який спеціалізується на створенні інтерактивних квестів. Хоча застосунок не використовує доповнену реальність, він все ще має безліч прикладів побудови квестів, їхнього оформлення та взаємодію з користувачами.

Перш за все **Wb** дозволяє створити власний квест (**Quest**) та поділитись ним. Інші гравці можуть оцінювати квест і просувати його в топи рейтингу. Мобільний застосунок пропонує широкий набір типів завдань, таких як вікторини, пошук локацій, пошук та сканування QR-кодів, завантаження різних відео та фото [12, 13].

Для створення свого контенту використовується окрема вебсторінка де можна створити квест. Цей інструмент має багато інтуїтивно зрозумілих вікон, які допомагають налаштовувати елементи квесту, так як потрібно автору. Користувач, який створює квест може вибрати багато різних типів завдань на різних етапах проходження, наприклад просте запитання чи знаходження певної локації на карті. Також є такі завдання, як намалювати чи сфотографувати щось. Все залежить від автора, а **Wb** надає змогу реалізувати це [12, 13].

Коли гравець хоче пройти якийсь квест, він використовує мобільний застосунок. Проходження можливе лише через телефон чи планшет. Для проходження квестів пов'язаних з геолокацією, **Wb** має навігацію через стрілки, які показують напрямок руху до потрібних координат.

Основні екранні форми додатку:

- Телефон: Інтерфейс додатка подібний до списку задач. Є декілька вікон в яких можна вибрати тип квесту для проходження (пошук місця, вікторина).
- Вебсторінка: виглядає просто, без зайвої деталізації. Є вікна для створення питань та поля для вибору налаштувань.

Інтерфейс вікон додатку зображено на рисунку 1.3 та 1.4

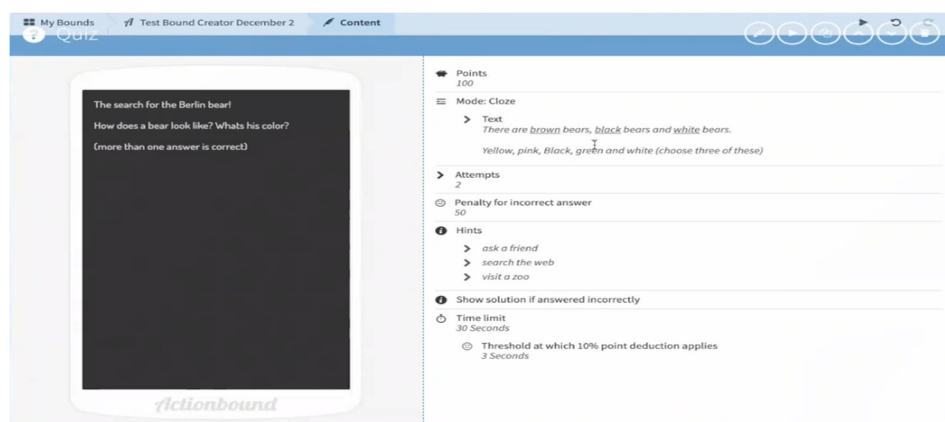


Рисунок 1.3 – Інтерфейс Веб-сторінки **Wb**

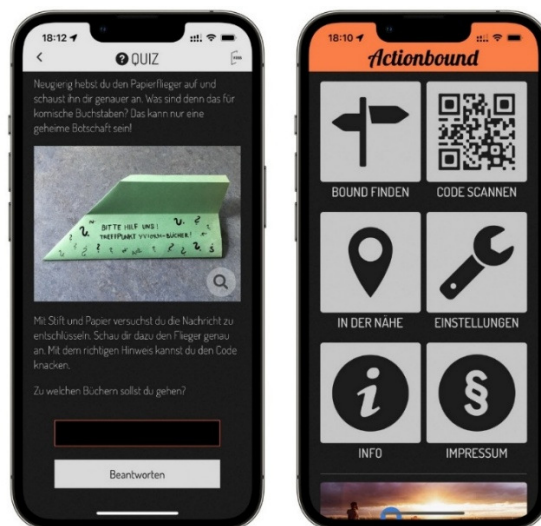


Рисунок 1.4 – Інтерфейс мобільного додатку Actionbound [14]

Проект Actionbound демонструє, як повинні виглядати розділи створення та проходження. Перший має давати користувачу створити квест, який бажає автор, а другий повинен бути зрозумілим та простим. Режим знаходження місця дуже поєднується з розроблюваною системою і ефективним рішенням є використання напрямних стрілок, які допомагають знайти потрібне місце.

Доволі корисним для розуміння роботи з картами може стати проект Geocaching від компанії Groundspeak. Головна ідея проекту полягає в пошуку скарбів, які люди залишають у певній місцевості. Хоча цей застосунок також не пов'язаний з доповненою реальністю, він є чудовим прикладом для модулів навігації та роботи з картою.

Головний функціонал системи це карта та мітки схованок на ній. Коли гравець вибирає цікаву йому мітку, застосунок дає додаткову інформацію про мітку (складність, розмір, ландшафт) [15, 16]. Мітки можна відфільтрувати за типом, розміром, складністю та подібним. Користувач також може відкрити сторінку схованки з детальнішою інформацією про об'єкт.

Цікавим рішенням є журнал, який дозволяє відмічати знайдені схованки, залишати коментарі або звернутись до власника сховку. Також даний застосунок дозволяє отримати підказку (фото, опис) [15]. Основні екранні форми додатку (рисунок 1.5):

- головний екран (карта). Інтерфейс додатку це перш за все карта з мітками. При натисканні на мітку з'являється додаткова інформація;
- інформація мітки. Екран з додатковою інформацією це проста сторінка з текстом і декількома кнопками.

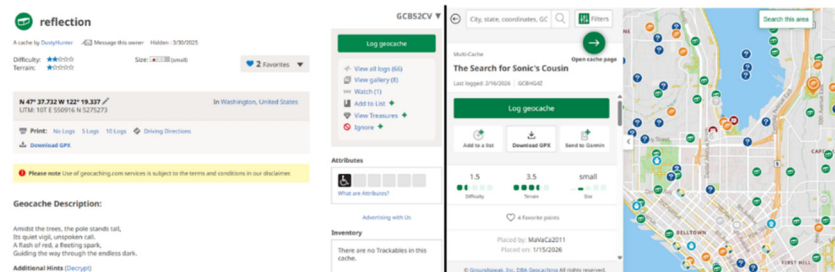


Рисунок 1.5 -Інтерфейс вікон Gcaching

Проект Gcaching демонструє роботу з картами та мітки на них. Також цікавою ідеєю для проєктованого додатку є журнал, тобто зберігання прогресу та історії у базі даних.

Для аналізу інших карт та використання їх підходить проєкт Ingress from від компанії Niantic, Inc. Основна ідея це захоплення порталів та створення зв'язків між ними. Як в додатку це самі портали, а карта реалізована в темних тонах з неоновими елементами (Рисунок 1.6). Цікавим в додатку є зв'язки між точками інтересу. Така візуалізація може бути корисною для проєктованої системи [17].

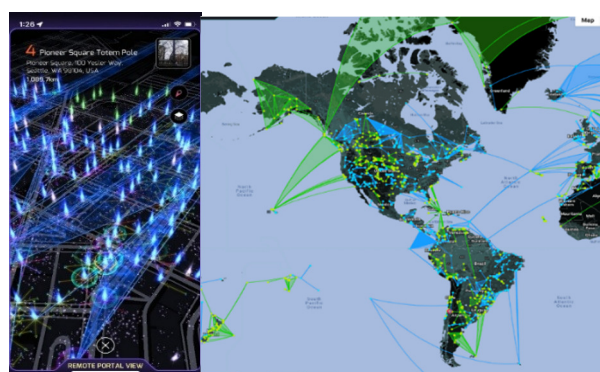


Рисунок 1.6 – Карти Ingress from [18]

Чудовим прикладом реалізації роботи SL є мобільний застосунок Portal. Проєкт дозволяє розмістити віртуальний об'єкт перед собою. Portal демонструє

маркерну та безмаркерні технології розміщення □□. Елементи доповненої реальності часто є інтерактивні та можуть використовуватись в міні-іграх [19].

Приклад розміщеного об'єкта видно на рисунку 1.7

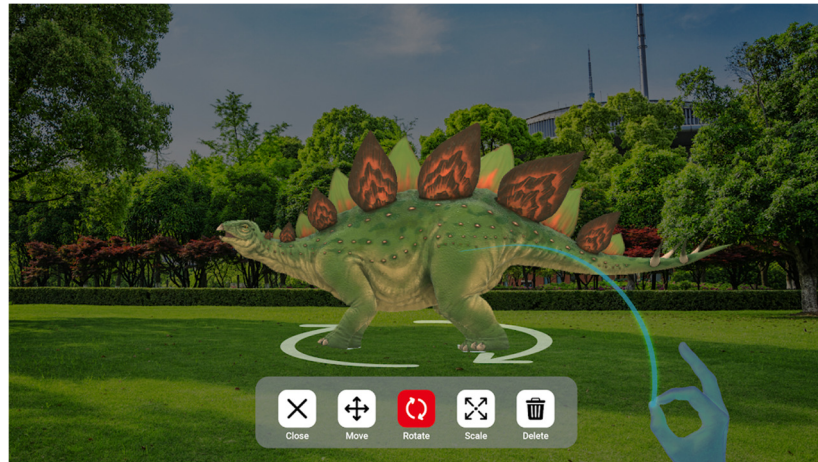


Рисунок 1.7 – □□ в □rl□□□□[20]

Аналіз існуючих аналогів показав, як сучасні системи поєднують картографічні сервіси, □□-технології та елементи гейміфікації. Для розроблюваної системи доцільно використовувати наступне:

- Використання карти, як основний елемент навігації та додаткові вказівники (□□-стрілки).
- Розробка зрозумілого інтерфейсу створення та проходження квестів та елементи збереження та відображення сюжету (опису точок інтересу, журнал).
- Реалізація можливості взаємодії з □□-об'єктами.

1.3 Постановка задачі дослідження

Аналіз існуючих аналогів показав, що більшість систем орієнтовані на використання готового контенту, або потребують спеціальних знань для створення власних сценаріїв. Відсутні зручні інструменти, які дозволяють користувачу створювати власні □□-квести з прив'язкою до географічних координат.

Метою дослідження є розробка автономного мобільного додатку, який дозволить користувачу створювати власні сценарії (екскурсія або квест) з віртуальними об'єктами, які зафіксовані на певних GPS-координат.

Для створення подібної системи потрібно в одній архітектурі об'єднати роботу з картографічними сервісами, комп'ютерний зір та обчислення відстаней. Також розробити структуру бази даних для зберігання інформації з різних модулів та спроектувати зрозумілий інтерфейс.

Для розроблюваної системи сформульовано наступні вимоги:

1. Управління даними. Система повинна мати автономне сховище для зберігання інформації про квест (назва, опис, масив точок) та прогрес проходження. Необхідна розробка функцій, які дозволять контролювати зміни в системі, такі як: створення, оновлення, видалення квестів. Слід розробити модуль, який дозволить користувачам ділитись квестами між собою (імпорт/експорт).

2. Підсистема створення квестів. Головним функціоналом цієї частини є створення квесту та наповнення його точками інтересу. Дана частина повинна надавати можливість повністю налаштовувати параметри кожної точки: введення текстового опису, підказок, вибір 3D-моделі точки. Важливим елементом підсистеми є навігація по карті та засоби визначення місцезнаходження користувача.

3. Ігрова підсистема. Підсистема повинна забезпечувати постійну перевірку місцезнаходження гравця та розраховувати відстань до точок інтересу. Також важливою вимогою даної підсистеми є зрозуміла навігація, а саме міні-карта та 3D-вказівники. Саме цей модуль відповідає за розташування віртуальних об'єктів на певних координатах і повинен дозволити взаємодіяти з точками інтересу.

4. Інтерфейс та надійність. Візуальна частина даного додатку повинна бути інтуїтивно зрозумілою і чітко розділяти режим проходження та режим створення. Також система повинна мати захисні механізми, які попереджатимуть користувача про зникнення підключення до мережі Інтернет та вимкненні GPS-модуля.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна структура проєктованої системи

Для вирішення поставленої задачі пропонується створення комплексної архітектури, яка дозволить взаємодіяти таким базовим компонентам: геопросторовому модулю, підсистемі генерації контенту та модулю візуалізації доповненої реальності. Зважаючи на основне призначення системи – створення та проходження квестів з геолокацією, система повинна працювати в реальному часі та швидко реагувати на зміну фізичних координат користувача. Для цього архітектура поділена на модулі з чітким розділенням бізнес-логіки, даних та інтерфейсу. З точки зору функціональної структури, дані компоненти поділяються на п'ять підсистем: управління даними, конструювання та редагування, навігації, □□ та обміну даними.

Для реалізації загальної структури системи та моделювання її частин було обрано об'єктно-орієнтований підхід. Саме він надає достатньо можливостей для опису предметної області (квести, точки, прогрес, маркери та інше) через об'єкти, які взаємодіють між собою. Використання об'єктно-орієнтованого підходу дозволяє зробити архітектуру гнучкою, що дозволить легко додавати покращення в окремі частини системи або впроваджувати новий функціонал (наприклад, додавання нових типів завдань чи 3D-моделей).

Функціональна структура розроблюваної системи зображена на рисунку 2.1

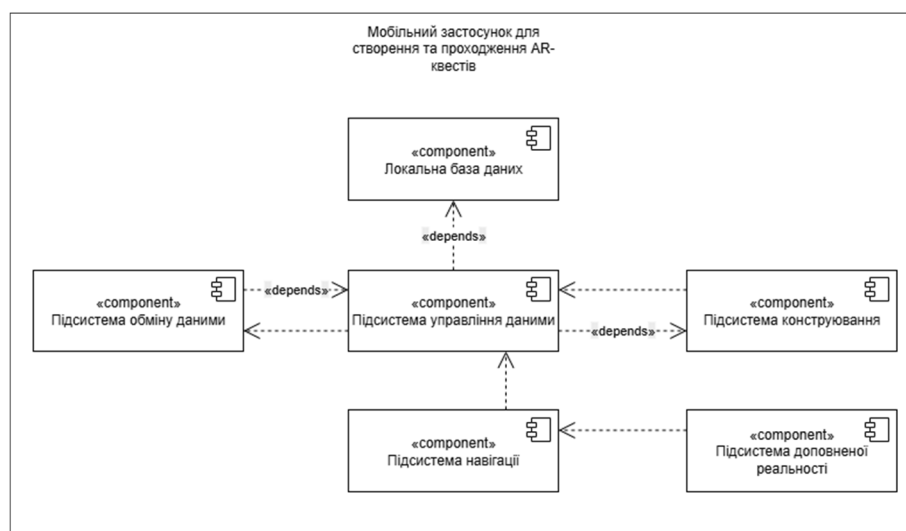


Рисунок 2.1 – Функціональна структура системи

1. Підсистема управління даними.
 - Відповідає за локальне зберігання всієї інформації з різних модулів без постійного підключення до мережі.
 - Функції: виконання CRUD операцій для записів про квести, точки інтересу (POI), їх метадані (координати, опис, підказка, тип, 3D-модель) та прогресу користувача.
2. Підсистема конструювання та редагування.
 - Надає автору необхідні інструменти для створення квесту та наповнення його точками інтересу.
 - Функції: отримання координат з інтерактивної карти, введення метаданих (опис, підказки), класифікація точок (послідовні чи постійні), вибір 3D-моделі та редагування вибраних налаштувань.
3. Підсистема навігації в просторі.
 - Основний інструмент, який дозволяє користувачу орієнтуватися на місцевості через карту та вказівники.
 - Функції: отримання поточних GPS-координат гравця, розрахунок відстаней між користувачем та віртуальними об'єктами, генерація маркерів на міні-карті та управління віртуальними вказівниками.
4. Підсистема доповненої реальності (AR).
 - Основний модуль який відповідає за візуалізацію віртуальних об'єктів та накладання їх на відеопотік.
 - Функції: аналіз простору навколо користувача, відстеження переміщень пристрою, рендер об'єктів на заданих координатах, контроль видимості 3D-об'єктів через відстань та обробка взаємодії гравця з віртуальними об'єктами.
5. Підсистема обміну даними.
 - Окремий модуль для експорту та імпорту даних квестів.
 - Функції: формування об'єктів передачі даних (DTO), перетворення даних з реляційної бази даних в текстовий формат, перевірка даних при імпорті та перевірка на наявність дублікатів.

Усі ці модулі взаємодіють між собою, передаючи інформацію чи запити до інших модулів, утворюючи єдиний інформаційний простір. Розроблювана система має такі інформаційні зв'язки між функціональними підсистемами:

Модулі конструювання та управління даними мають двосторонній зв'язок. Коли гравець завершує налаштування квесту та його метаданих (назви, координати точок, текст точок, ідентифікатори 3D-моделей та інше) підсистема створення квесту відправляє ці дані для запису в базу даних. І навпаки, коли користувачу потрібно відредагувати існуючий квест, модуль управління даних надсилає потрібну інформацію для їхнього редагування чи видалення.

Зв'язок також є між модулями управління даними та підсистемою навігації. Навігаційний модуль отримує масив точок з бази даних, після чого починає обраховувати відстань між гравцем та точками інтересу. Після цього дана інформація передається в підсистему доповненої реальності.

Передача відстаней до точок створює зв'язок між модулями AR-візуалізації та навігації. На основі цих відстаней підсистема доповненої реальності приймає рішення відображати 3D-об'єкти чи приховати їх.

Не менш важливим є інформаційний зв'язок між підсистемою AR та підсистемою управління даними. Коли гравець взаємодіє з точкою інтересу, відбувається процес збереження прогресу. Інформація про знайдену точку передається в базу даних, після чого оновлює стан квесту, вмикаючи наступну сюжетну точку.

Модуль обміну даних має зв'язок лише з базою даних. При експорті даних модуль отримує необхідну інформацію про квест та перетворює її в текстовий формат після чого відбувається передача файлу. Під час імпорту квесту, модуль обміну зчитує інформацію з вхідного файлу та передає запит до бази даних для запису.

Для функціонування системи існують такі інформаційні зв'язки із зовнішніми джерелами інформації:

Дані, отримані через апаратні сенсори мобільного пристрою (GPS, гіроскоп, камера), надходять постійно і є критично необхідними для функціонування модулів навігації та доповненої реальності.

Для коректного відображення карти в додатку використовується зв'язок із зовнішнім геоінформаційним сервісом. Дані з нього використовуються в модулі конструювання та під час проходження квесту (міні-карта).

Також зовнішнім джерелом є файлова система пристрою, яка взаємодіє з модулем обміну даними.

Фізична архітектура та інформаційні зв'язки із зовнішніми джерелами зображено на рисунку 2.2.

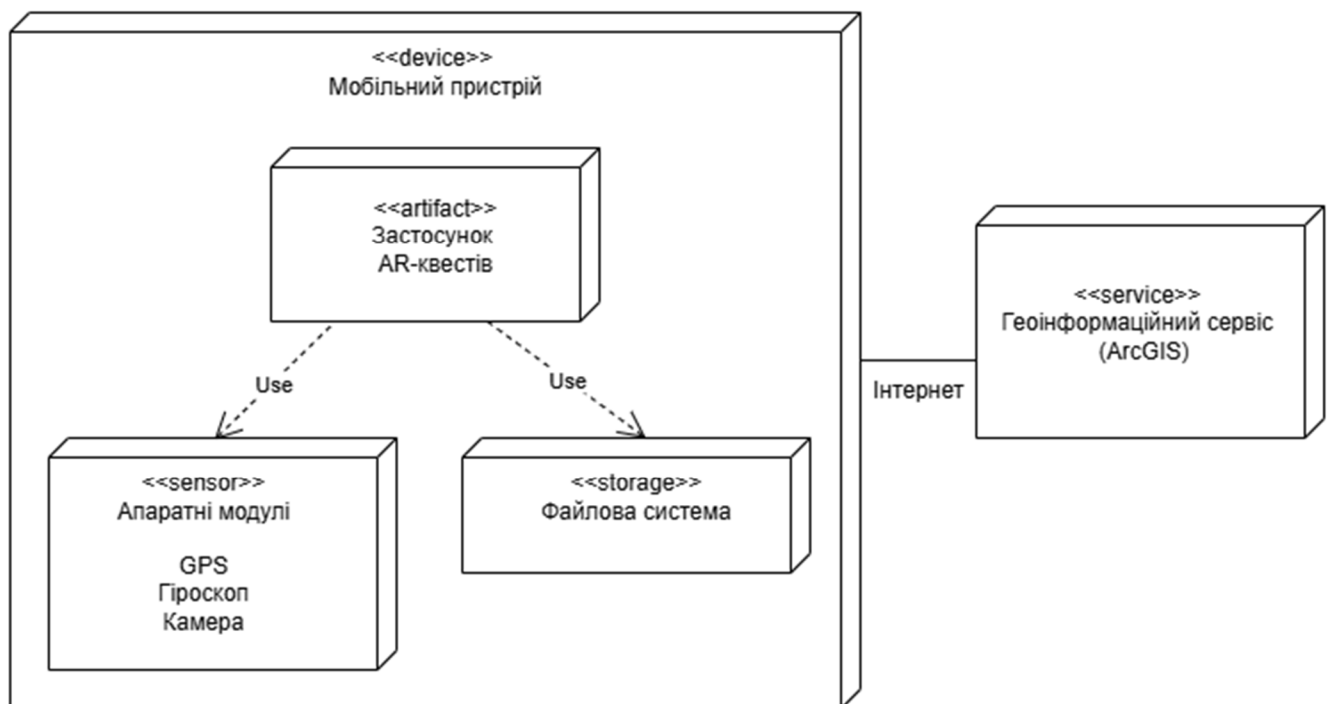


Рисунок 2.2 – Diagram of the physical architecture

2.2 Алгоритмічне забезпечення

Розроблювана система складається з декількох частин, які постійно взаємодіють між собою та часто передбачають повернення до попередніх станів. Дана архітектура має багато дрібних алгоритмів, які є частиною двох основних, що вирішують поставлену задачу. Головними в даній системі є алгоритми конструювання (створення або редагування сценарію) та проходження квесту. Перед початком роботи система проходить етап ініціалізації і дає можливість вибрати, подальший режим роботи (конструювання чи проходження).

2.2.1 Алгоритм функціонування системи

Під час старту додатка система виконується сценарій, який ініціалізує базові системи. Перш за все запускається модуль, який перевіряє підключення мобільного пристрою до мережі Інтернет та активність GPS-модуля. У разі відсутності хоча б однієї з умов з'являється повідомлення, яке інформує користувача та блокує доступ до основного функціоналу програми. Після запуску модуля перевірки створюється єдиний глобальний екземпляр (Singleton) підсистеми управління базою даних, який має доступ до файлу SQLite, гарантуючи цілісність даних. Ці два компоненти існують та працюють протягом усього життєвого циклу програми.

Після проходження перевірок користувач потрапляє в головне меню, яке виконує роль маршрутизатора і пропонує три основні напрямки:

1. Підсистема конструювання (створення та редагування квестів).
2. Підсистема проходження (ігровий процес).
3. Глобальне очищення (алгоритм повного видалення усієї інформації з бази даних.)

Також користувач може закрити застосунок, завершивши роботу алгоритму. Весь алгоритм функціонування системи зображено в Додатку А.

2.2.2 Алгоритмічне забезпечення підсистеми конструювання квестів

Головним завданням підсистеми створення та редагування квестів є надання можливості користувачу створити власний AR-квест з прив'язкою до геолокації. Даний алгоритм дозволяє автору створити новий сценарій, задати його метадані (назва, опис) та розмістити точки інтересу на карті. Після розміщення є можливість їх налаштувати (назва, опис, тип, 3D-модель) або видалити.

Після переходу до розділу створення користувачеві пропонується меню з чотирма варіантами: повернутися назад, створити новий квест, редагувати існуючий квест або перейти до модуля обміну квестами. Перед створенням нового квесту автор повинен ввести його назву та, за бажанням, опис, після чого відкривається сцена розміщення точок інтересу. Якщо автор вибере редагування вже створеного сценарію, то з бази даних завантажується ідентифікатор квесту, після чого користувач перейде до сцени розміщення точок.

Під час завантаження редактора точок для певного квесту, система звертається до бази даних та завантажує розставлені точки інтересу з їх налаштуваннями, якщо такі є. Дана сцена складається з інтерактивної карти, яка дозволяє розмістити приціл у потрібному місці, кнопки розміщення точки, кнопки редагування існуючих точок інтересу та двох кнопок виходу (зберегти та відкинути зміни). Після натискання кнопки розміщення точки відкривається панель вводу інформації про точку (назва, опис, тип, 3D-модель), після збереження якої дані точки разом з її координатами записуються у тимчасовий список. Панель редагування дозволяє видалити точку або змінити її метадані.

Коли користувач налаштував потрібний йому сценарій, він має два варіанти виходу: вийти без збереження, що відхиляє всі зміни і повертає в головне меню, або зберегти, що перезаписує усі дані з тимчасового списку у базу даних та повертає до головного меню. Якщо при виході квест залишається порожнім, то система автоматично видаляє його. В результаті роботи алгоритму створення користувач отримує готовий AR-квест з прив'язкою до геолокації, який можна завантажити для проходження.

Детальна схема алгоритму створення та редагування квесту зображена в Додатку Б.

Важливою частиною підсистеми конструювання є створення карти, відображення точок інтересу на ній та отримання координат вибраного місця. Для реалізації карти використовується геоінформаційний сервіс, за допомогою якого завантажуються тайли та будується цілісна карта. Для оптимізації система зменшує роздільну здатність тих фрагментів, які залишаються поза видимістю користувача.

Для керування прицілом та отримання координат використовується точка спостереження, яка розташована над картою. У звичайних умовах вказівник синхронізується з даними GPS-модуля пристрою, але під час дотику до екрана користувач може рухати точку спостереження, прицілюючись на карті. Координати точки визначаються за положенням прицілу на карті, після чого записуються в тимчасовий список.

Відображення маркерів точок на карті реалізується через їхнє відносне зміщення. Використання складних тригонометричних функцій для розрахунку

зміщення кожної точки в реальному часі є затратним та непотрібним. Для цього пропонується використання плоскої апроксимації, де за основу береться константа довжини одного градуса широти (приблизно 111320 метрів), а довгота визначається динамічно через косинус широти [21, 22]:

$$\begin{aligned}d_y &= \Delta\varphi * 111\,320 \\d_x &= \Delta\lambda * 111\,320 * \cos(\varphi_1)\end{aligned}\tag{2.1}$$

де φ – широта в градусах;

φ_1 – широта прицілу (точки спостереження);

λ – довгота в градусах.

Після переведення градусів у метри відстань обчислюється за теоремою Піфагора: $d = \sqrt{d_x^2 + d_y^2}$. Отримане значення є наближеною відстанню між об'єктом та користувачем, але є достатньо точним для масштабу карти. Такі перетворення мають меншу обчислювальну складність порівняно з повною сферичною моделлю.

2.2.3 Алгоритмічне забезпечення підсистеми проходження

Розділ проходження надає користувачу можливість пройти AR-сценарій, створений у модулі конструювання. Алгоритм цієї підсистеми безперервно відстежує цілі, керує відображенням доповненої реальності та фіксує ігрові досягнення і прогрес гравця.

Коли гравець вибирає розділ “пройти” у головному меню, система звертається до бази даних для генерації списку усіх доступних квестів. Після вибору потрібного квесту відбувається процес ініціалізації сценарію, а саме:

1. Завантаження даних квесту (усі точки квесту, їхні метадані), передача координат точок у модуль візуалізації.

2. Звернення до бази даних для перевірки прогресу користувача. Якщо користувач раніше проходив цей квест, алгоритм відновить проходження з моменту виходу гравця. Модуль пропустить усі знайдені точки та увімкне потрібну,

відновить журнал, заповнивши його інформацією знайдених точок, оновить мітки на карті та значення лічильників.

3. Починається відстеження координат гравця та розрахунок відстаней до активних точок інтересу. За допомогою цих відстаней здійснюється контроль видимості елементів доповненої реальності.

4. На основі координат пристрою ініціалізується міні-карта та розміщуються мітки.

Після ініціалізації підсистема продовжує обчислювати відстань між користувачем та активними цільовими точками. На основі цих даних здійснюється контроль видимості та доступу. Для гравця встановлено два порогових значення – радіус видимості та радіус взаємодії. Коли відстань менша за поріг видимості, система починає візуалізувати 3D-об'єкт; а при наближенні на відстань, меншу за поріг взаємодії, активується подія доступу, яка дозволяє гравцю взаємодіяти з точкою інтересу.

Після ініціалізації підсистема переходить у режим очікування дій користувача, паралельно відстежуючи координати пристрою. Користувачеві доступний пошук точок інтересу, дві кнопки підказок, взаємодія з 3D-об'єктами (при наближенні до них), перегляд журналу та перехід до меню налаштувань з кнопками повернення до головного меню та очищення прогресу.

Під час взаємодії з об'єктом алгоритм відображає інформаційне вікно точки, перезаписує прогрес у базі даних та оновлює інформацію на екрані (лічильник, карта, видимість об'єкта). Якщо ця точка є сюжетна, також відбувається перевірка її порядку. У випадку, коли точка остання, з'являється інформація про знаходження всіх сюжетних точок.

Для полегшення навігації користувач може натиснути кнопку пошуку точки, після чого вмикається вказівник та відображається відстань до точки. Гравцю доступні дві кнопки – пошук сюжетної точки та пошук постійної точки (яка вказуватиме на найближчу мітку). При повторному натисканні кнопки вказівник вимикається.

Коли гравець вибирає перегляд журналу, підсистема відображає інформацію точок у тій послідовності, у якій вони були знайдені, проте при відновленні прогресу відбувається сортування для групування сюжетної інформації.

Перейшовши до налаштувань, користувачеві доступні три варіанти: повернення до головне меню (збереження прогресу та вихід), очищення прогресу (видалення запису прогресу для цього квесту) та повернення до гри (закриває вікно налаштувань).

Схема алгоритму ігрового модуля наведена в Додатку В. На відміну від системи конструювання, де відстані використовувалися лише для міток на карті, у модулі проходження для визначення дистанції між користувачем та 3D-об'єктами необхідна підвищена точність визначення відстаней, тому використовується формула Гаверсинуса [23], яка враховує сферичну форму Землі та дозволяє розрахувати найкоротшу відстань по дузі кола:

$$d = R * c \quad (2.2)$$

де центральний кут c обчислюється за формулами

$$c = 2 \arctan \left(\frac{\sqrt{a}}{\sqrt{1-a}} \right);$$
$$a = \sin^2 \left(\frac{\Delta\varphi}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\Delta\lambda}{2} \right);$$
$$\Delta\varphi = \varphi_2 - \varphi_1;$$
$$\Delta\lambda = \lambda_2 - \lambda_1$$

де φ – широта в радіанах;

λ – довгота в радіанах;

R – радіус Землі (~6371000 м);

c – центральний кут між двома точками на сфері в радіанах;

d – відстань у метрах

2.2.4 Алгоритм обміну даними

Щоб надати користувачам можливість обмінюватись своїми квестами, розроблено алгоритм обміну даними. У розділі конструювання автор може обрати функцію обміну квестами, після чого відкривається панель з вибором експорту чи імпорту.

Експорту починається з формування списку доступних квестів. Після вибору потрібного сценарію ініціалізується процес перетворення даних. Система зчитує дані потрібного квесту (назва, опис, масив точок та їх метадані), формуючи об'єкти передачі даних (DTO). Ці об'єкти серіалізуються у структурований текстовий формат JSON. Далі користувачеві надається можливість вибрати директорію на пристрої для збереження файлу. Під час збереження також відбувається перевірка та очищення назви від заборонених символів та заміни пробілів на нижнє підкреслення.

Імпорт виконує зворотні дії, зчитуючи текстовий файл. Після отримання інформації відбувається валідація структури даних. Далі система порівнює назву завантаженого квесту зі збереженими в базі даних для пошуку дублікатів. Якщо знайдено збіг, система блокує збереження та виводить повідомлення користувачеві. В іншому випадку квест додається до бази даних. Схему алгоритму обміну квестами зображено в Додатку Г.

2.3 Інформаційне забезпечення

2.3.1 Опис вхідної та вихідної інформації

Система створення та проходження AR-квестів постійно отримує дані ззовні для функціонування певних модулів. Цю інформацію можна класифікувати на такі групи: геопросторові дані, метадані, введені користувачем, керуючі команди, зовнішні файлові дані.

Геопросторові дані – це координати (широта та довгота), які система отримує від апаратного модуля GPS. Надходження цих даних відбувається в моменти, коли гравець проходить квест або розставляє точки інтересу на карті. Ця інформація є необхідною для функціонування систем навігації та AR-модуля.

Метаданими, які вводить автор, можна вважати всю інформацію що задається під час створення квесту. Серед них: назва та опис квесту, координати точок інтересу та налаштування точок (назва, текст точки, тип, 3D-модель).

Керуючі команди – це сигнали, які отримує система при взаємодії користувача з інтерфейсом. До них належать підтвердження взаємодії користувача з точкою інтересу, команди збереження, видалення та скидання прогресу.

Вхідною інформацією також вважається JSON-файл під час імпорту квесту, дані з камери та тайли карти, отримані із геоінформаційного сервісу.

Після опрацювання вхідної інформації система надає вихідні дані, які також можна розділити на декілька груп. Серед них: картографічна інформація, AR-інформація, ігрові дані та дані алгоритму експорту.

До картографічної інформації належать створені карти, які будуються через поєднання даних із зовнішнього геоінформаційного сервісу та координат гравця. Також до цієї групи належать маркери, які накладаються на карту, та відстань до точки інтересу у метрах.

AR-дані – це уся доповнена інформація, яка відображається під час проходження квесту. Сюди належать 3D-моделі точок та вказівники, які накладаються на відеопотік.

Інформація, яка надається гравцю під час проходження, належить до ігрових даних. Тут є журнал, який зберігає інформацію знайдених точок, лічильники прогресу (кількість знайдених точок певного типу) та системні повідомлення (попередження про зникнення зв'язку, підтвердження збереження, повідомлення про помилки, інформація після закінчення проходження).

Згенерований JSON-файл після виконання експорту також належить до вихідної інформації системи.

2.3.2 Концептуальне інфологічне проєктування

Після аналізу бізнес-процесів створення та проходження квестів було виділено три основні сутності:

Сценарій квесту (Quest). Головна сутність, яка відповідає за створені квести та їх метадані (назва та опис). Атрибути – унікальний ідентифікатор, назва (заголовок) та текстовий опис.

Точка квесту (QuestPoint). Сутність, яка визначає фізичну локацію на карті та містить метадані цієї точки. Її атрибутами є унікальний ідентифікатор, географічні координати (широта та довгота), назва точки (для найменування локацій або ідентифікації користувачем), опис (або підказка, залежить від задуму автора), тип видимості (постійна чи послідовна) та візуальний тип 3D-моделі. Кожна точка прив'язана лише до одного квесту.

Прогрес (QuestProgress). Сутність, яка відображає стан проходження певного квесту. Тут зберігається інформація про поточну сюжетну (послідовну) точку та перелік уже знайдених постійних міток.

Між сутностями Quest та QuestPoint існує зв'язок “один до багатьох”. Один квест може мати безліч точок інтересу, але кожна точка належить лише одному квесту. У разі видалені квесту всі його точки також видаляються.

Сутності Quest та QuestProgress мають зв'язок “один до одного”. Для кожного квесту існує лише один запис прогресу користувача.

2.3.3 Проектування глобальної даталогічної моделі даних

У розроблюваній системі даталогічна модель складається з трьох взаємопов'язаних відношень (рисунок 2.3).

Таблиця “Quest” (зберігає загальну інформацію про квест).

- QuestId (Первинний ключ, PK) – унікальний ідентифікатор квесту.
- Name – назва квесту.
- Description – опис.

Таблиця “QuestPoint” (зберігає координати та параметри точок інтересу).

- PointId (Первинний ключ, PK) – унікальний ідентифікатор локації.
- QuestId (Зовнішній ключ, FK) – ідентифікатор квесту, до якого належить точка. Використовується для зв'язування з таблицею “Quest”.

- Latitude – географічна широта.
- Longitude – географічна довгота.

- `InfoText` – інформація про точку (назва).
 - `HintText` – текст точки (підказка або опис).
 - `IsAlwaysVisible` – логічний прапорець (boolean), що визначає тип точки (постійна чи послідовна).
 - `ModelType` – ідентифікатор тривимірної моделі.
- Таблиця “QuestProgress” (зберігає стан ігрової сесії).
- `QuestId` (Первинний ключ, зовнішній ключ, PK, FK) – Ідентифікатор квесту. Одночасно виступає як зв’язок із таблицею “Quest”.
 - `CurrentSequenceIndex` – цілочисельний індекс поточної сюжетної (послідовної) точки.
 - `CollectedPermanentIds` – серіалізований текстовий рядок, який містить ідентифікатори (`PointId`) вже знайдених постійних точок, розділених комою.

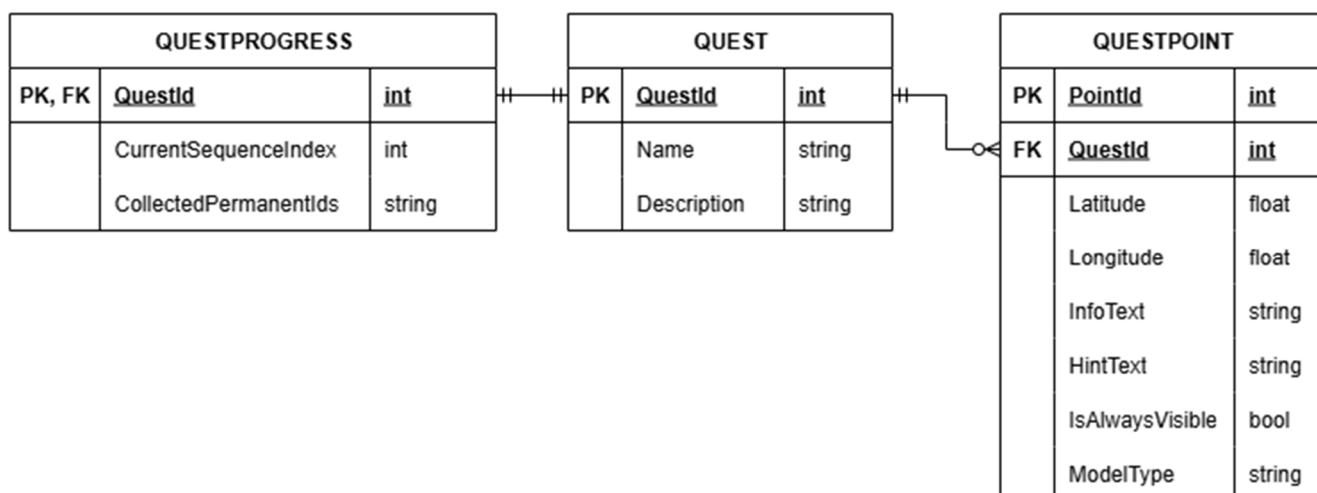


Рисунок 2.3 – ER-діаграма реляційної бази даних

2.3.4 Проектування фізичної моделі даних

Оскільки архітектура розроблюваного додатка не передбачає реєстрації, створення облікових записів та авторизації користувачів, розгортання клієнт-серверного бекенду (наприклад `MySQL` чи `PostgreSQL`) є недоцільним. Використання виділеного сервера потребує постійної підтримки та налаштувань, що ускладнює розгортання проєкту. Для обміну квестами був реалізований окремий

алгоритм шляхом створення текстового файлу та його збереження на пристрої, тому потреба у централізованому сховищі відпадає.

Саме тому для фізичної реалізації було обрано локальну СУБД SQLite [24], яка дозволяє створити ізольований мобільний застосунок. База даних зберігається у вигляді єдиного локального файлу в пам'яті телефону.

Фізичні типи даних SQLite, що були використані для реалізації даталогічної моделі:

- Ідентифікатори (QuestId, PointId) та індекси (CurrentSequenceIndex) реалізовані типом INTEGER із властивістю автоматичного інкременту для первинних ключів.
- Географічні координати (Latitude, Longitude) реалізовані типом REAL, що необхідно для збереження багатьох знаків після коми.
- Текстові дані (Name, Description, InfoText, HintText, ModelType, CollectedPermanentIds) реалізовані типом TEXT.
- Логічний прапорець (IsAlwaysVisible) також реалізований як INTEGER (де 0 – false, 1 – true).

2.3.5 Програмна реалізація бази даних

Для програмної реалізації управління даними використовується технологія ORM (Object-Relational Mapping), яка дозволяє проєктувати реляційні таблиці бази даних через об'єктно-орієнтовані класи мови програмування. Це дозволяє уникнути написання низькорівневих SQL-запитів для ініціалізації структури бази даних.

Взаємодія з фізичним файлом відбувається на основі патерну Singleton [25], коли в системі існує лише одне з'єднання з базою даних. Такий підхід запобігає конфліктам доступу при одночасному читанні та записі даних різних модулів програми.

Коли застосунок запускається вперше, створюється файл бази даних, який зберігається у захищеній директорії (persistentDataPath) [26]. Завдяки цьому файл бази даних не видаляється під час оновлення додатка. Після створення файлу відбувається процес ініціалізації, який виконується через механізм ORM,

аналізуючи класи моделей, генерує та виконує SQL-команди для створення потрібних таблиць.

Для обробки інформації використовується гібридний підхід. Для виконання звичайних CRUD-операцій використовуються об'єктні методи ORM-бібліотеки. Проте для складніших операцій, таких як каскадне очищення всіх точок перед оновленням квесту, використовуються прямі SQL-запити.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Реалізація програмного забезпечення

Програмна реалізація проєктованої системи виконана у вигляді мобільного додатка для операційної системи Android. Для розробки використовувався ігровий рушій Unity, а основна логіка написана об'єктно-орієнтованою мовою програмування C#. Для реалізації деяких основних функцій використовувалися спеціалізовані програмні інтерфейси: Niantic Lightship ARDK для реалізації доповненої реальності, ArcGIS Maps SDK для роботи з картографічними даними та СУБД SQLite для зберігання даних.

3.1.1 Обґрунтування вибору програмних засобів реалізації

Зважаючи на вимоги до розроблюваного продукту, а саме мобільний застосунок, який здатний працювати з геолокаційними координатами, відображати віртуальні об'єкти в доповненій реальності та функціонувати незалежно від централізованого сервісу, було обрано такий стек технологій:

Unity використовується як основне середовище розробки додатка [27]. На відміну від середовищ, призначених для розробки мобільних додатків (наприклад, Android Studio), де реалізація доповненої реальності потребує використання спеціалізованих бібліотек або низькорівневих API, Unity має готові інструменти для роботи з 3D-сценами та AR. Порівняно з Unreal Engine, Unity зазвичай забезпечує менші вимоги до апаратних ресурсів, що корисно для пристроїв із середніми технічними характеристиками.

Головною мовою написання системи стала C#, яка є базовою для середовища Unity. C# також використовує автоматичне керування пам'яттю, що спрощує процес розробки та зменшує ймовірність виникнення помилок.

Niantic Lightship ARDK обрано як основний фреймворк для роботи з доповненою реальністю [28]. На відміну від стандартних рішень (ARCore/ARKit) [29, 30], Lightship дозволяє легко реалізувати доволі точне закріплення 3D-об'єктів на певних координатах.

□r□G□S □aps – це основний картографічний сервіс, який використовується у системі. Цей сервіс має зручний плагін в Unity [31], через який можна легко та безкоштовно інтегрувати реальні географічні дані (тайли карт, координати) у тривимірний простір Unity. Він дозволяє легко отримувати координати місцевості та будувати точну карту.

СУБД SQLite використовується як локальне сховище даних. Структура даних квестів (координати, метадані) не є складною і не потребує паралельного багатокористувацького доступу. Саме тому використання віддаленого SQL-сервера є економічно та технічно недоцільним. SQLite ідеально задовольняє потреби проєктованого додатка, зменшуючи навантаження на мережу.

3.1.2 Структурна схема програмної системи

Для відображення поділу функцій системи між окремими програмними вузлами, які в Unity реалізовані як незалежні сцени, використовується структурна схема. Вона також демонструє інформаційні зв'язки та потоки передачі даних між модулями. Структурна схема програмної системи зображена на рисунку 3.1.

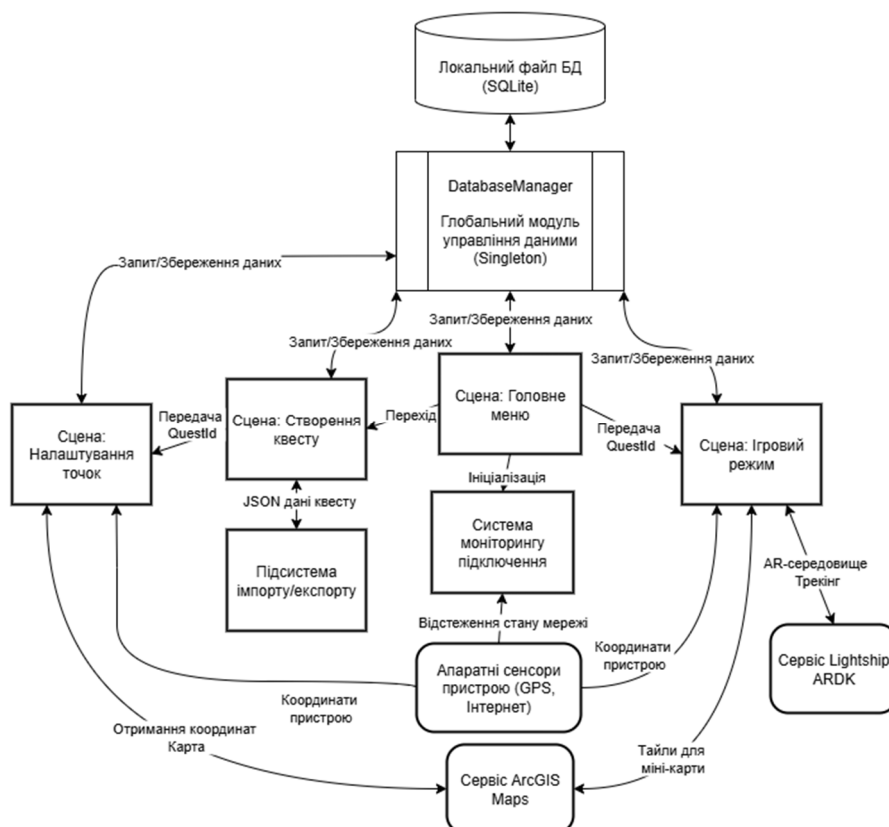


Рисунок 3.1 – Структурна схема додатку

Програмна система поділена на п'ять основних компонентів (один глобальний клас-менеджер та чотири функціональні сцени):

Глобальний модуль управління даними (DatabaseManager). Основний компонент, який має доступ до локального файлу бази даних. Реалізований за патерном Singleton [25], для створення єдиного активного підключення до файлу БД. Цей елемент ініціалізується під час запуску додатка та існує протягом усього його життєвого циклу.

Сцена “Головне меню”. Стартовий вузол, який вмикається при запуску додатка. Саме тут відбувається ініціалізація систем, таких як DatabaseManager та система моніторингу підключення. Головне меню є маршрутизатором, який дозволяє користувачу перейти у вузли створення або проходження квестів.

Сцена “Створення квесту”. Модуль, який відображає списки існуючих квестів або дає можливість створити новий маршрут. Під час вибору квесту для редагування або створення нового, цей модуль отримує або створює унікальний ідентифікатор сценарію (QuestId) і передає його до наступної сцени. Цей вузол також містить функціонал імпорту та експорту квестів.

Сцена “Налаштування точок”. Під час завантаження сцена отримує QuestId, після чого звертається до DatabaseManager для отримання уже налаштованих точок. Цей модуль дозволяє розмістити точки інтересу на карті, створеній через інтеграцію з ArcGIS, та налаштувати їхні метадані, після чого дані зберігаються в базу даних через модуль управління даними. Сервіс ArcGIS Maps надає можливість створити карту та зчитувати координати з потрібного місця.

Сцена “Ігровий режим”. Модуль, створений для проходження квестів. Після вибору квесту користувачем, “Головне меню” передає QuestId для завантаження потрібних даних з бази даних. Ця сцена взаємодіє одразу з двома зовнішніми сервісами: ArcGIS для міні-карти та Lightship ARDK для доповненої реальності. Модуль звертається до DatabaseManager для отримання точок квесту та для записування прогресу користувача.

Сцени взаємодіють між собою, передаючи ідентифікатори квестів. Кожна сцена завантажує потрібні дані незалежно через звернення до глобального модуля управління даними.

3.1.3 UML-діаграма класів основної бізнес-логіки

Класи, які реалізують архітектуру розробленого додатка, можна поділити на три основні групи: класи моделей даних, глобальні класи управління та класи контролери сцен (контролюють інтерфейс та логіку), які взаємодіють із зовнішніми сервісами.

Класи моделей даних та обміну (Models, DTO). Ці класи є структурами даних, які не успадковують базовий клас Unity MonoBehaviour. Використовуються для ORM-мапінгу бази даних SQLite та для перетворення даних при імпорті та експорті. До них належать такі класи: Quest (сутність сценарію), QuestPoint (сутність локації), QuestProgress (стан проходження), QuestTransferData та QuestPointTransferItem (об'єкти передачі даних, DTO).

Глобальні класи управління. Класи, які реалізуються за патерном Singleton [25] для створення єдиного екземпляру та не знищуються під час переходу між сценами. До них належать: DatabaseManager (клас для підключення до бази даних та маніпуляцій з даними, використовує Models) та SystemMonitor (перевірка підключення пристрою до мережі та активність GPS-модуля).

Класи контролери сцен. Класи, які успадковують MonoBehaviour та контролюють роботу сцени. Відповідають за обчислення, реакції на дії користувача, зміну елементів інтерфейсу та взаємодію із зовнішніми SDK. До них належать:

- Модуль Головного меню. QuestSelectionManager (ініціалізація сесії, навігація між елементами головного меню), AppStartupManager (взаємодія з ОС Android для отримання апаратних дозволів).

- Модуль Редактора квестів. QuestEditor (відповідає за процес створення нового квесту, генерацію ідентифікатора та передачу його до іншої сцени), QuestListManager (відповідає за функцію редагування квестів, формуючи список та передаючи потрібний ідентифікатор наступній сцені), FileSharingManager (відповідає за функціонал імпорту та експорту, використовує DTO).

- Модуль Налаштування точок. PointEditor_ArcGIS (контролер, який відповідає за усі дії в сцені налаштування точок. Генерує карту через сервіс ArcGIS, отримує та записує координати та решту метаданих точок в тимчасовий список, реагує на дії користувача, оновлює зміни в налаштуваннях точок інтересу),

MapDragHandler (відповідає за зміну стану відстеження карти, передаючи стан головному контролеру).

- Ігровий модуль. ARQuestGameController (головний клас, який виконує усі дії ігрового модуля, пов'язані з доповненою реальністю та ігровим процесом. Використовує Lightship ARDK), QuestUIManager (відповідає за створення міні-карти та міток на ній через ArcGIS. Також відповідальний за функцію Журналу).

Усі класи, окрім моделей даних та DTO-об'єктів, успадковуються від базового класу Unity MonoBehaviour. Для збереження чіткості на діаграмі класів позначені лише ключові атрибути та функціональні методи. Другорядні елементи, які відповідають за інтерфейс та не впливають на логіку програми, на схемі не зазначаються. Детальна UML-діаграма класів основної бізнес-логіки зображена в Додатку Д.

3.1.4 UML-діаграма станів графічного інтерфейсу

Графічний інтерфейс користувача (GUI) програми поділений за принципом, де кожна Unity-сцена – це окремий складений стан, який має декілька підстанів. Кожен підстан GUI реалізований як панель або діалогове вікно. UML-діаграма станів графічного інтерфейсу зображена в Додатку Е.

Діаграма станів поділена на чотири основні складені стани та один глобальний перериваючий стан.

1. Складений стан “Головне меню”. Початковий стан системи після ініціалізації додатка. Складається з таких підстанів:

- Стан “Очікування вибору”: базовий стан, де відображаються основні кнопки навігації.

- Стан “Перегляд доступних квестів”: активується кнопкою “Грати”. Відображає список існуючих сценаріїв.

- Стан “Деталі квесту”: спливаюче вікно з інформацією про квест та кнопкою “Почати”. Дозволяє перейти до складеного стану “Ігровий процес”.

- Стан “Підтвердження очищення БД”: спливаюче вікно із запитом на повне очищення даних БД.

2. Складений стан “Управління квестами”. Інтерфейс управління контентом. Складається з таких підстанів:

- Стан “Редактор даних квесту”: базовий стан, вікно з вибором дій (заповнення форми чи перехід до інших станів). Після заповнення форми та натискання “Створити” переходить до складеного стану “Конструктор точок”.

- Стан “Панель списку квестів”: список усіх квестів з бази даних. Активується кнопкою “Редагувати” та надає можливість змінити інформацію чи перейти до складеного стану “Конструктор точок”.

- Стан “Панель редагування назви та опису”: форма для зміни даних квесту. Активується іконкою редагувати.

- Стан “Панель обміну”: панель з вибором дії (імпорт чи експорт) та інформаційним повідомленням. Активується іконкою обміну.

3. Складений стан “Конструктор точок”. Інтерфейс просторового редагування з використанням ArcGIS. Складається з таких підстанів:

- Стан “Інтерактивна карта”: карта з прицілом та кнопки, які активують інші стани. Містить кнопки збереження, встановлення нової точки інтересу та кнопку списку тимчасових точок. Після натискання виходу відбувається перехід до складеного стану “Головне меню”.

- Стан “Панель вводу метаданих”: активується кнопкою додати точку. Відкриває панель із формою введення параметрів точки.

- Стан “Панель списку ”: активується кнопкою списку. Містить набір усіх точок інтересу, дозволяє видалити точку або перейти до редагування даних.

4. Складений стан “Ігровий процес”. Стан, який дозволяє пройти квест, поєднує інтерфейси карти та доповненої реальності. Складається з таких підстанів:

- Стан “AR навігація”: базовий стан, демонструє відео з віртуальними об’єктами, міні-карту, лічильник точок та ігрові кнопки.

- Стан “Панель інформації точки”: при наближенні до точки інтересу вмикається кнопка взаємодії, через яку активується цей стан. Відображає панель з інформацією точки (опис).

- Стан “Екран завершення”: активується при взаємодії з останньою сюжетною точкою. Інформаційне вікно з текстом про закінчення проходження.
- Стан “Панель Журналу”: активується кнопкою. Містить опис усіх знайдених точок.
- Стан “Налаштування”: активується кнопкою. Містить панель з кнопками повернутись, вийти та очистити прогрес. Кнопка вийти переходить до складеного стану “Головне меню”.

5. Глобальний перериваючий стан “Апаратне блокування”. У будь-який момент роботи системи, якщо пропаде підключення, вмикається панель блокування, яка інформує про зникнення підключення. Після відновлення підключення панель закривається і система повертається до попереднього стану.

3.1.5 Опис лістингу програми та вихідного коду

Лістинг програмного коду основних модулів розробленого мобільного застосунку наведено в Додатку Ж. До нього включено лише фрагменти вихідного коду, які реалізують ключову функціональність системи та відображають особливості програмної реалізації запропонованого рішення.

Зокрема, у додатку наведено фрагменти класу `DatabaseManager`, що забезпечує взаємодію з локальною базою даних `SQLite`, створення та редагування квестів, збереження прогресу користувача, а також імпорт і експорт даних у форматі `JSON`.

Також наведено основні методи класу `ARQuestGameController`, який реалізує логіку проходження квесту в режимі доповненої реальності, керування послідовністю сюжетних точок, обчислення відстані між географічними координатами та взаємодію користувача з AR-об'єктами.

Для демонстрації процесу створення квестів наведено фрагменти класу `PointEditor_ArcGIS`, який забезпечує роботу з картографічним сервісом `ArcGIS`, створення та редагування точок інтересу, визначення координат користувача за допомогою `GPS` та збереження маршрутів.

Окремо подано методи класу FileSharingManager, що реалізують механізми імпорту та експорту квестів між мобільними пристроями шляхом серіалізації даних у формат JSON.

Службові частини програмного коду, пов'язані з налаштуванням графічного інтерфейсу користувача, обробкою допоміжних подій та елементів керування, до додатку не включалися через їх значний обсяг та відсутність впливу на основні алгоритми функціонування системи.

3.2 Інтерфейс користувача

Оскільки мобільний застосунок передбачає використання доповненої реальності на вулиці, ключовими вимогами до дизайну стали контрастність елементів та мінімізація перекриття основних елементів сцен.

У всіх сценах додатка використовуються стандартизовані елементи. Наприклад, списки для елементів чи інформаційні вікна виглядають майже однаково в кожному розділі. Важливим елементом є підтверджуючі панелі, які викликаються при критичних змінах (видалення прогресу або очищення бази даних), захищаючи користувача від випадкових натискань.

Головне меню є маршрутизатором, тому містить лише кнопки переходу до інших розділів та кнопку очищення бази даних. Основні навігаційні кнопки згруповані по центру та збільшені, тоді як кнопка очищення БД винесена подалі. Інтерфейс головного меню зображено на рисунку 3.2.

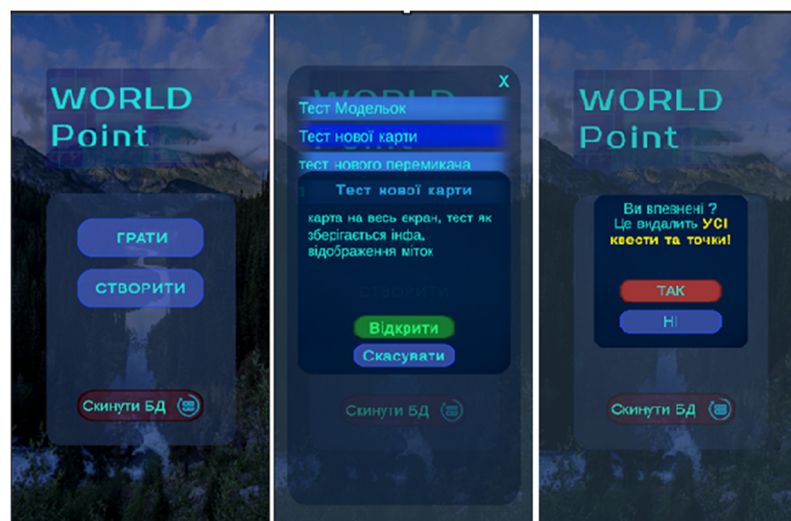


Рисунок 3.2 – Вікна розділу “Головне меню”

Екран створення квесту відкривається після натискання кнопки “СТВОРИТИ” в меню. Інтерфейс візуально поділений на дві частини: форму створення нового квесту по центру та допоміжну панель управління знизу. Нижня панель містить три кнопки: повернутись (іконкою), “РЕДАГУВАТИ” та обмін (іконкою).

Кнопка “РЕДАГУВАТИ” відкриває список наявних квестів. Кожен елемент списку складається з кнопки переходу до редагування точок інтересу та ще трьох допоміжних кнопок: інформація, редагування та видалення. Кожна з цих кнопок відкриває свою панель.

Кнопка обміну відкриває панель імпорту/експорту з вибором дії та інформаційним текстом. Вибір функції експорту попередньо відкриває список квестів для вибору користувачем, після чого запускається процес збереження файлу у вибране місце на пристрої. Вікна розділу створення та управління квестами зображено на рисунку 3.3.



Рисунок 3.3 – Вікна розділу створення квестів

Головним компонентом сцени розміщення точок є карта, яка займає увесь екран. На неї накладені кнопки взаємодії: додавання точки, списку точок, кнопки збереження та центрування карти за GPS. Кнопка додавання точок відкриває панель вводу метаданих, а кнопка списку точок генерує панель з тимчасовими точками та

надає можливість змінити метадані або видалити точку. Спроба вийти без збереження викликає панель підтвердження. Панелі сцени розміщення точок зображено на рисунку 3.4.



Рисунок 3.4 – Панелі розділу створення точок інтересу

Інтерфейс ігрової сцени оптимізовано під режим доповненої реальності. Елементи інтерфейсу розташовані максимально компактно по краях екрана, щоб не перекривати огляд реального світу.

Головними компонентами є міні-карта та панель внизу екрана. Панель містить лічильники точок та кнопки пошуку об'єктів. Також передбачені кнопки журналу та взаємодії (остання з'являється при наближенні за межу взаємодії). На краю екрана розташована панель налаштувань з кнопками виходу, повернення та очищення прогресу. Панелі ігрового режиму зображено на рисунку 3.5 та рисунку 3.6.

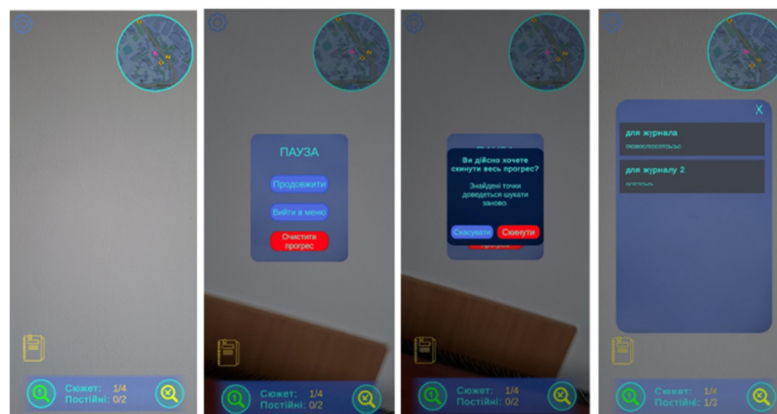


Рисунок 3.5 – Панелі ігрового режиму



Рисунок 3.6 – Панелі ігрового режиму, пов'язані з □□

Окремо створена глобальна система безпеки, яка перевіряє підключення до Інтернету й GPS і вмикає панель блокування у разі зникнення з'єднання. Панель блокування зображена на рисунку 3.7

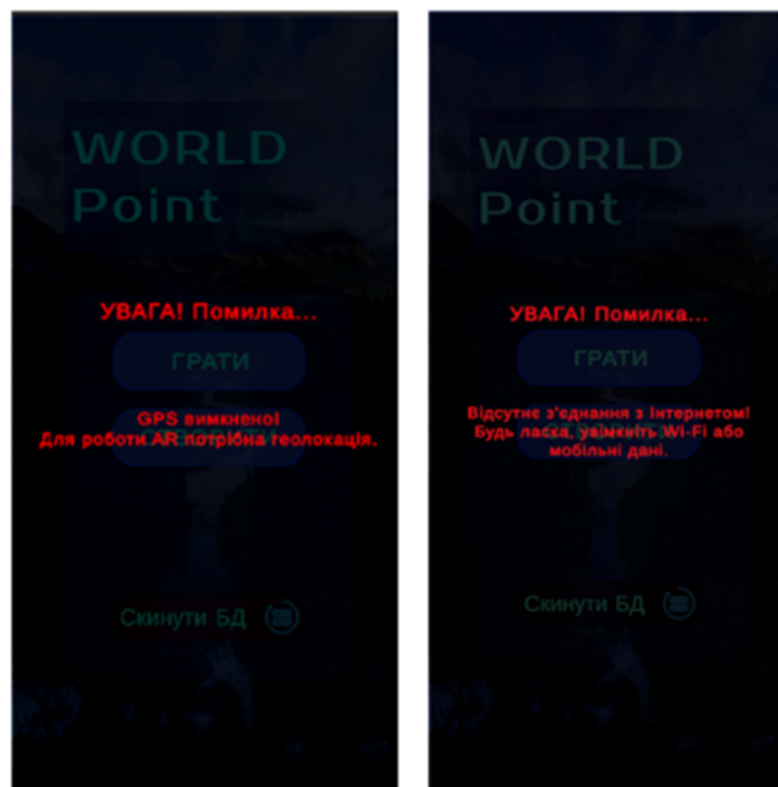


Рисунок 3.7 – Глобальна панель блокування

3.3 Тестування розробленого мобільного застосунку

Для перевірки роботи мобільного застосунку було проведено комплексне тестування. Оскільки він пов'язаний з доповненою реальністю та глобальним позиціюванням, тому однією з головних вимог є правильне закріплення віртуальних об'єктів на певних координатах. Для цієї частини відбувалася перевірка розміщення об'єктів у визначених місцях, відображення декількох об'єктів одночасно (постійних), розрахунку відстані, та контролю видимості та взаємодії:

1. Розміщення об'єктів на певних координатах (Рисунок 3.8).

Очікуваний результат: віртуальні об'єкти знаходяться там, де їх було розміщено на карті.

Фактичний результат: об'єкти розміщені на тих самих місцях з невеликою похибкою в кілька метрів (викликана апаратною точністю вбудованого модуля GPS)



Рисунок 3.8 – Тестування розміщення □□-об'єктів

2. Відображення декількох об'єктів разом (Рисунок 3.9).

Очікуваний результат: постійні точки відображаються відразу усі, сюжетні по черзі.

Фактичний результат: постійні точки видно обидві, сюжетні лише перша.



Рисунок 3.9 – Тестування поділу елементів на постійні та сюжетні

3. Контроль видимості (Рисунок 3.10)

Очікуваний результат: Модель не видно, якщо відстань більша за п'ятдесят метрів

Фактичний результат: Модель не видно при відстані 51 метр та видно при відстані 49 метрів.

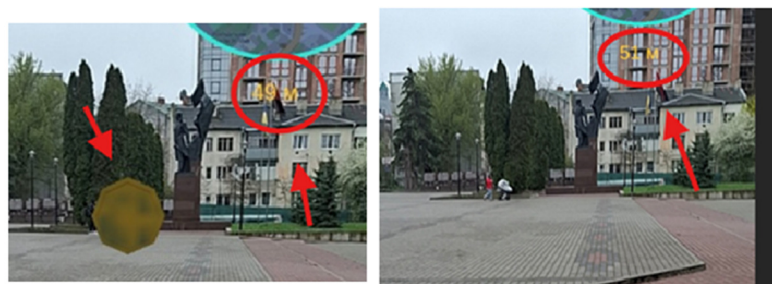


Рисунок 3.10 – Тестування видимості

4. Контроль взаємодії (Рисунок 3.11)

Очікуваний результат: якщо відстань до об'єкта менша за 20 метрів, вказівник та відстань зникають, з'являється кнопка взаємодії.

Фактичний результат: навігація зникає, кнопка взаємодії з'являється.

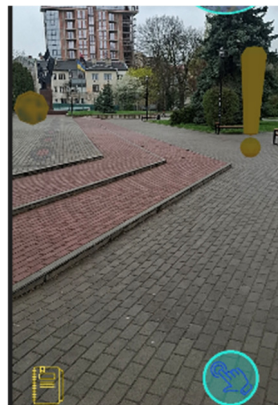


Рисунок 3.11 – Тестування взаємодії

5. Тестування вказівника (Рисунок 3.12)

Очікуваний результат: вказівник показує на вибраний об'єкт (постійний або сюжетний).

Фактичний результат: вказівник коректно показує на вибраний об'єкт.

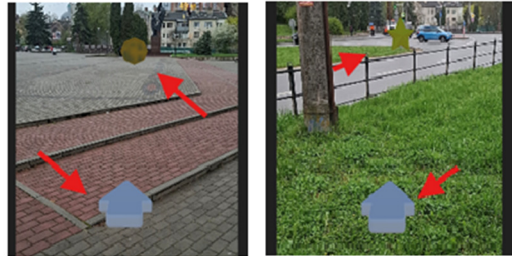


Рисунок 3.12 – Тестування вказівника

Також важливою частиною є процес створення квесту та функція імпорту/експорту. Для цієї частини було проведено повний цикл створення нового квесту, наповнення його даними, імпорту та експорту з перевіркою на дублікат.

Створено новий квест з 4 точками різних типів та моделей. Після збереження в списку редагування з'явився новий квест. Вибравши його дані відновилися в тих самих місцях. Зміна типу моделі та назви точки зберіглися та відновилися після повторного перезапуску. Тестування створення зображено на рисунку 3.13.

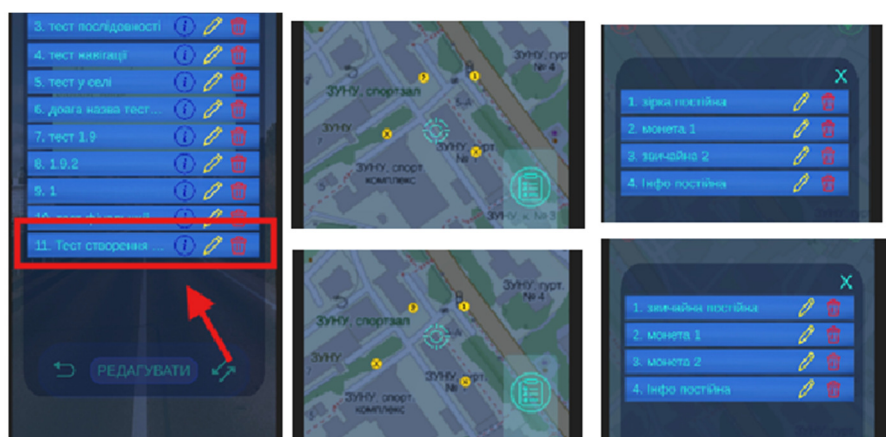


Рисунок 3.13 – Тестування створення та редагування квесту

Новий квест було експортовано на пристрій, після чого його було видалено. Далі файл було імпортовано назад та перевірено вміст (рисунок 3.14). При повторному імпорті, система повідомляє про дублювання та не зберігає.

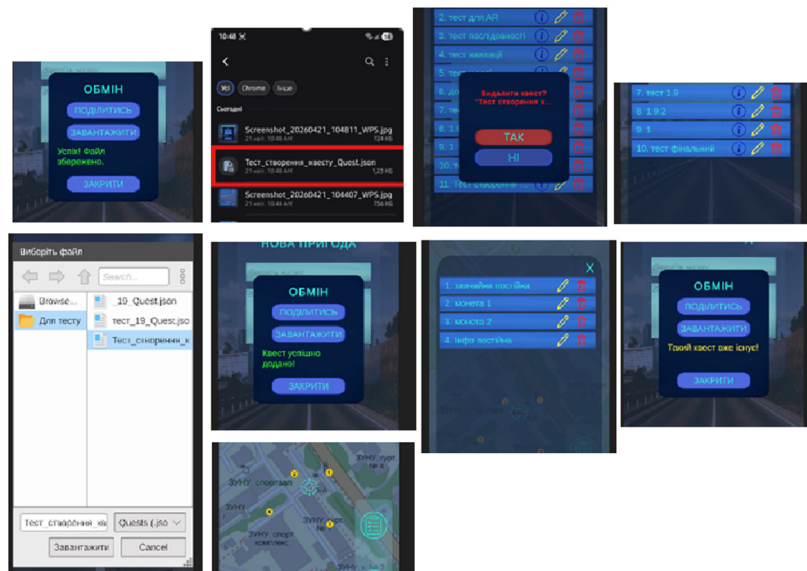


Рисунок 3.14 – Тестування імпорту та експорту

Також було проведено тестування інтерфейсу: перевірено роботу кожної кнопки, появу кожної панелі та введення некоректних даних. Усі компоненти працюють правильно, а некоректні дані система або не зберігає, або не дає ввести.

В результаті тестування було перевірено роботу компонентів системи. Дані в базі даних зберігаються коректно, карта та мітки на ній генеруються правильно, віртуальні об'єкти закріплюються на заданих координатах, алгоритм імпорту та експорту працює безпомилково.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне практичне завдання щодо проектування та розроблення мобільного додатка для створення, редагування та проходження геолокаційних квестів із використанням технології доповненої реальності. За результатами дослідження можна зробити такі висновки:

1. Спроектовано та розроблено архітектуру програмної системи. Використано ігровий рушій Unity, мову C# та локальну реляційну СУБД SQLite для збереження інформації про сценарії на пристрої. Додано механізми серіалізації та десеріалізації об'єктів у формат JSON для функції обміну квестами. Розроблений застосунок не потребує розгортання та оренди хмарних сервісів.

2. Розроблено модуль картографії та навігації. Для розробки використовувався сервіс ArcGIS Maps, а саме його плагін в Unity, програмна реалізація математичної сферичної моделі (формулу Гаверсінуса) та плоску апроксимацію (для UI-міток). Плагін дозволив створити карту та надав можливість отримувати координати місцевостей. Формула Гаверсінуса дозволила визначати дистанцію між користувачем та віртуальними об'єктами для активації тригерів в ігровій сцені.

3. Реалізовано ігровий модуль з використанням доповненої реальності. Для позиціонування віртуальних об'єктів використовувався Niantic Lightship ARDK та координати отримані з карти (через ArcGIS). Цей підхід дозволяє легко та безкоштовно закріплювати 3D-об'єкти за вибраними координатами. Відстань тут використовується для контролю видимості та області взаємодії з віртуальними об'єктами.

4. Спроектовано зручний інтерфейс користувача. Він має контрастні елементи, зрозумілі кнопки та панелі. Використовуються модальні діалогові вікна для захисту від критичних змін (наприклад, очищення прогресу) та єдиний стиль для подібних елементів. Додано фоновий сервіс апаратного моніторингу для відстеження підключення та інформування користувача при їх зникненні. Ігровий HUD реалізований за принципом мінімалізму, не перекриваючи відео з камери.

На відміну від аналогів (наприклад, Pokémon GO або інші платформи для геокешингу), які обмежують користувачів лише контентом від розробників, розроблена система дозволяє самому створювати квести з використанням доповненої реальності. Користувач в одному додатку може бути автором (завдяки інструментам конструювання) так і гравцем (у режимі проходження).

Розроблений продукт має високий ступінь готовності до використання і являє собою повноцінний MVP (Minimum Viable Product). Масштаби його використання охоплюють сфери туризму (створення інтерактивних екскурсій містом), освітньої діяльності (проведення історичних чи краєзнавчих квестів для студентів/учнів) та розважального сегмента.

Рекомендації та можливі покращення. Доцільним є впровадження інших алгоритмів комп'ютерного зору, які дозволять розпізнавати реальні об'єкти (будівлі) як додаткові тригери або перешкоди для відображення. Також можна розширити логіку конструктора, додаючи розгалуження до сценаріїв або інші типи взаємодії з віртуальними об'єктами. Важливим доповненням може стати створення бібліотеки квестів із системами рейтингів та відгуків.

Основні результати дослідження пройшли апробацію на Міжнародній науково-практичній конференції "ICSPM 2026" (Додаток II).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

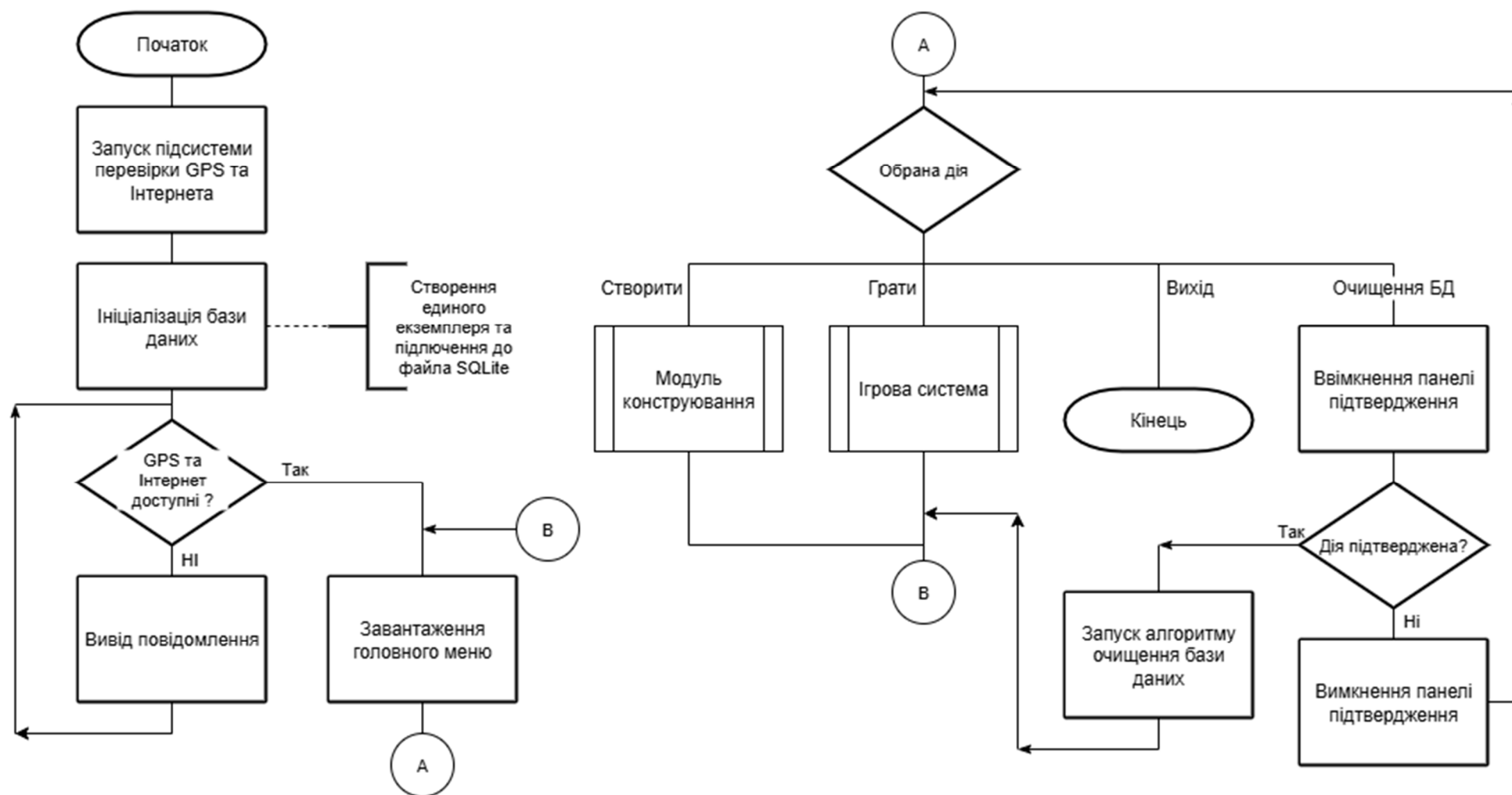
1. Аналіз технології та інструментів для реалізації систем доповненої реальності. *Оптико-електронні інформаційно-енергетичні технології*. 2024. № 47(1). С. 66–77. URL: <https://doi.org/10.31649/1681-7893-2024-47-1-66-77> (дата звернення: 06.02.2026).
2. Wikitude. *Wikipedia*. URL: <https://en.wikipedia.org/wiki/Wikitude> (дата звернення: 12.02.2026).
3. Методи та програмні засоби формування доповненої реальності. URL: <https://ela.kpi.ua/items/6616966e-49c0-4000-afc8-be952b57b217> (дата звернення: 06.02.2026).
4. Застосування алгоритмів одночасної локалізації та картографування SLAM при формуванні доповненої реальності. *Інформаційні системи та технології: матеріали конференції*. 2018. С. 26. URL: <https://ipi.kpi.ua/wp-content/uploads/2018/05/%D0%A2%D0%B5%D0%B7%D0%B8%20%20%D0%86%D0%9E%D0%A2-2018.pdf> (дата звернення: 09.02.2026).
5. *Impact of Geolocation Data on Augmented Reality Usability: A Comparative User Test*. arXiv. URL: <https://arxiv.org/abs/2308.13544> (дата звернення: 10.02.2026).
6. Методи та програмні засоби позиціонування об'єктів доповненої реальності. URL: <https://ela.kpi.ua/items/5f12d66b-8b5e-40b7-9eca-830a8a204f45> (дата звернення: 10.02.2026).
7. Pokémon GO. URL: https://en.wikipedia.org/wiki/Pok%C3%A9mon_Go (дата звернення: 12.02.2026).
8. Модель та засоби системи навігації між корпусами закладу освіти з використанням технології доповненої реальності. *Ukrainian Journal of Information Technology*. 2024. Том 6, № 1. URL: <https://science.lpnu.ua/uk/ujit/vsi-vypusky/tom-6-nomer-1-2024/model-ta-zasoby-systemy-navigaciyi-mizh-korpusamy-zakladu-osvity> (дата звернення: 12.03.2026).
9. *Use of Augmented Reality in Human Wayfinding: A Systematic Review*. arXiv. URL: <https://arxiv.org/abs/2311.11923> (дата звернення: 12.02.2026).

10. Що таке правило ізометричної проекції. URL: <https://dopom.911.v.ua/ukraincyam/shho-take-pravilo-izometrichnoi-proekcii.html> (дата звернення: 13.02.2026).
11. Rotolo T. The UX of Pokemon Go. *Trymana Blog*. URL: <https://trymata.com/blog/pokemon-go-ux/> (дата звернення: 13.02.2026).
12. *Actionbound*. URL: <https://de.wikipedia.org/wiki/Actionbound> (дата звернення: 13.02.2026).
13. Beginner's course – Actionbound Basics. URL: https://youtu.be/t_sQkCCjm8c (дата звернення: 13.02.2026).
14. Actionbound – interaktive Schnitzeljagden selbst erstellen. *iphone-ticker.de*. URL: <https://www.iphone-ticker.de/actionbound-interaktive-schnitzeljagden-selbst-erstellen-184859/> (дата звернення: 13.02.2026).
15. *Geocaching* (App). URL: [https://geocaching.fandom.com/wiki/Geocaching_\(App\)](https://geocaching.fandom.com/wiki/Geocaching_(App)) (дата звернення: 16.02.2026).
16. Geocaching Official Map. URL: <https://www.geocaching.com/play/map> (дата звернення: 16.02.2026).
17. Ingress (video game). *Wikipedia*. URL: [https://en.wikipedia.org/wiki/Ingress_\(video_game\)](https://en.wikipedia.org/wiki/Ingress_(video_game)) (дата звернення: 16.02.2026).
18. Ingress (video game). *Wikipedia*. URL: <https://ingress.com/> (дата звернення: 16.02.2026).
19. ARLOOPA – Augmented Reality Platform. URL: <https://www.arloopa.com/> (дата звернення: 16.02.2026).
20. ARLOOPA – Augmented Reality. *Google Play*. URL: <https://play.google.com/store/apps/details?id=com.arloopa.arloopa&hl=uk> (дата звернення: 16.02.2026).
21. How Many Meters Are in a Degree Latitude. URL: <https://sage-advice.com/how-many-meters-are-in-a-degree-latitude/> (дата звернення: 26.02.2026).
22. Equirectangular Projection. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Equirectangular_projection (дата звернення: 26.02.2026).
23. Формула гаверсинуса. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Формула_гаверсинуса (дата звернення: 26.02.2026).

24. SQLite Documentation. URL: <https://www.sqlite.org/> (дата звернення: 20.02.2026).
25. Singleton Design Pattern. URL: <https://refactoring.guru/design-patterns/singleton> (дата звернення: 20.02.2026).
26. Application.persistentDataPath. URL: <https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html> (дата звернення: 21.02.2026).
27. Unity Manual. URL: <https://docs.unity3d.com/Manual/index.html> (дата звернення: 23.02.2026).
28. Niantic Spatial SDK Documentation. URL: https://www.nianticspatial.com/docs/nsdk/xr_settings/?platform=unity (дата звернення: 02.03.2026).
29. ARCore Developer Documentation. URL: <https://developers.google.com/ar/develop> (дата звернення: 25.02.2026).
30. AR Foundation Documentation. URL: <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.4/manual/> (дата звернення: 26.02.2026).
31. ArcGIS Maps SDK for Unity. URL: <https://developers.arcgis.com/unity/maps/> (дата звернення: 25.02.2026).
32. Савчук Д.В., Биковий П.Є. Мобільний застосунок для створення та проходження AR-квестів із використанням геолокації. *ICSPM 2026: Інтелектуальні обчислювальні системи та управління проектами: зб. тез доповідей І міжнародної науково-практичної конференції студентів і молодих вчених (21-22 травня 2026р. Тернопіль).* 2026. С. 45-47.
33. Комар М. П., Саченко А. О., Васильків Н. М., Гладій Г. М., Коваль В. С., Ліп'яніна-Гончаренко Х. В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль : ЗУНУ, 2024. 52 с.

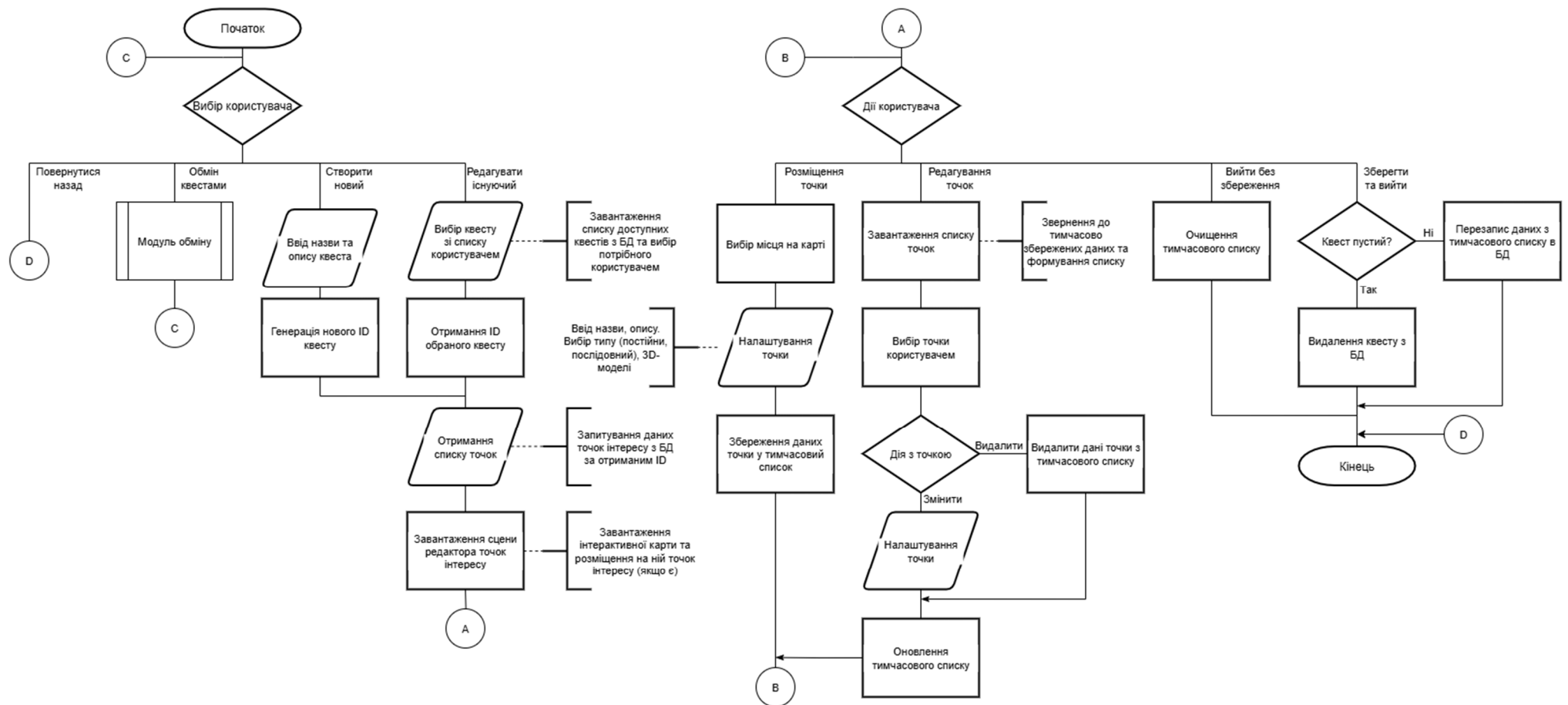
Додаток А

Схема алгоритму функціонування системи



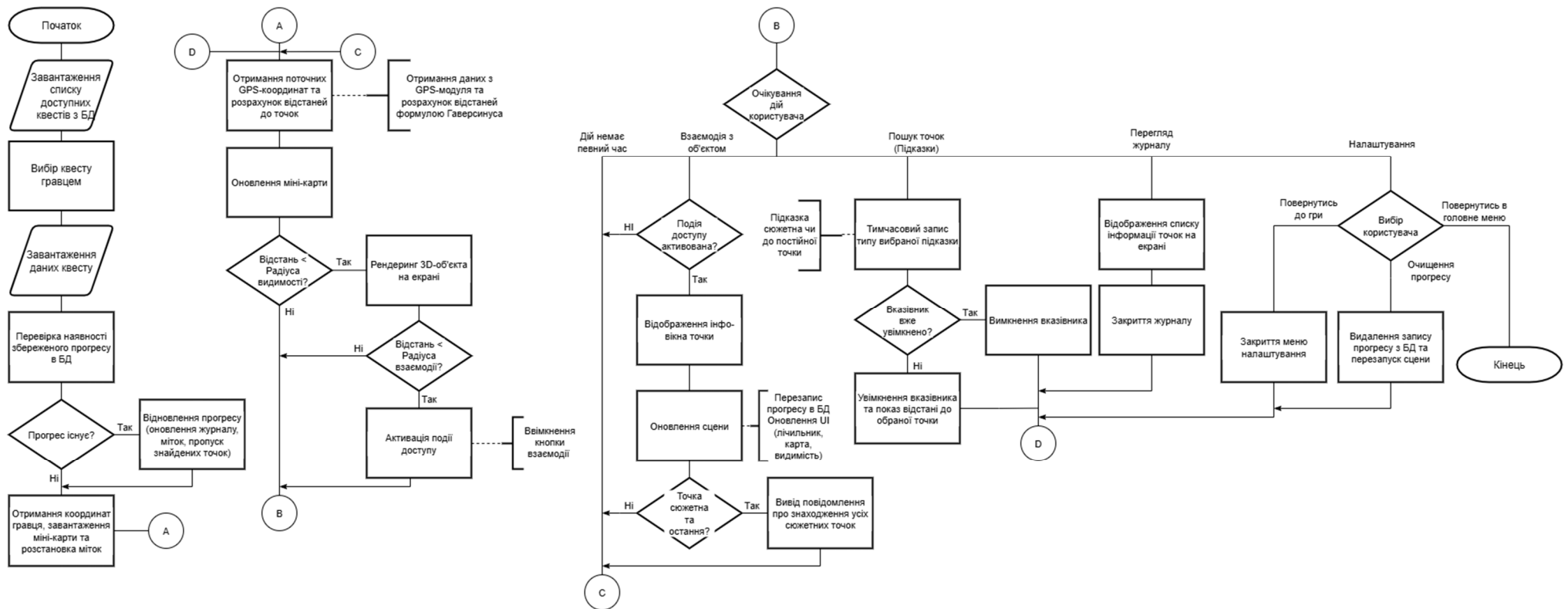
Додаток Б

Схема алгоритму модуля конструювання квестів



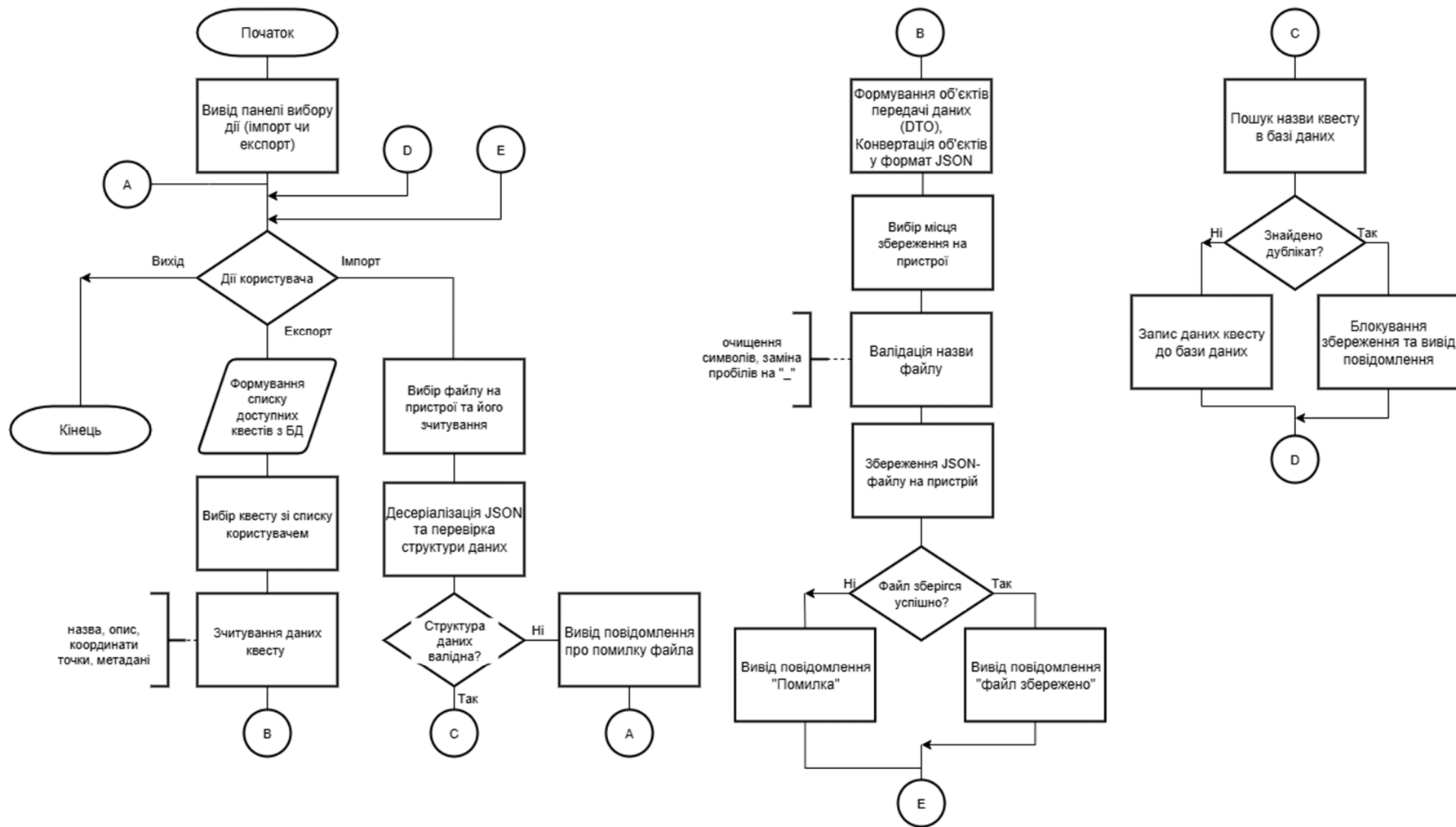
Додаток В

Схема алгоритму ігрової системи



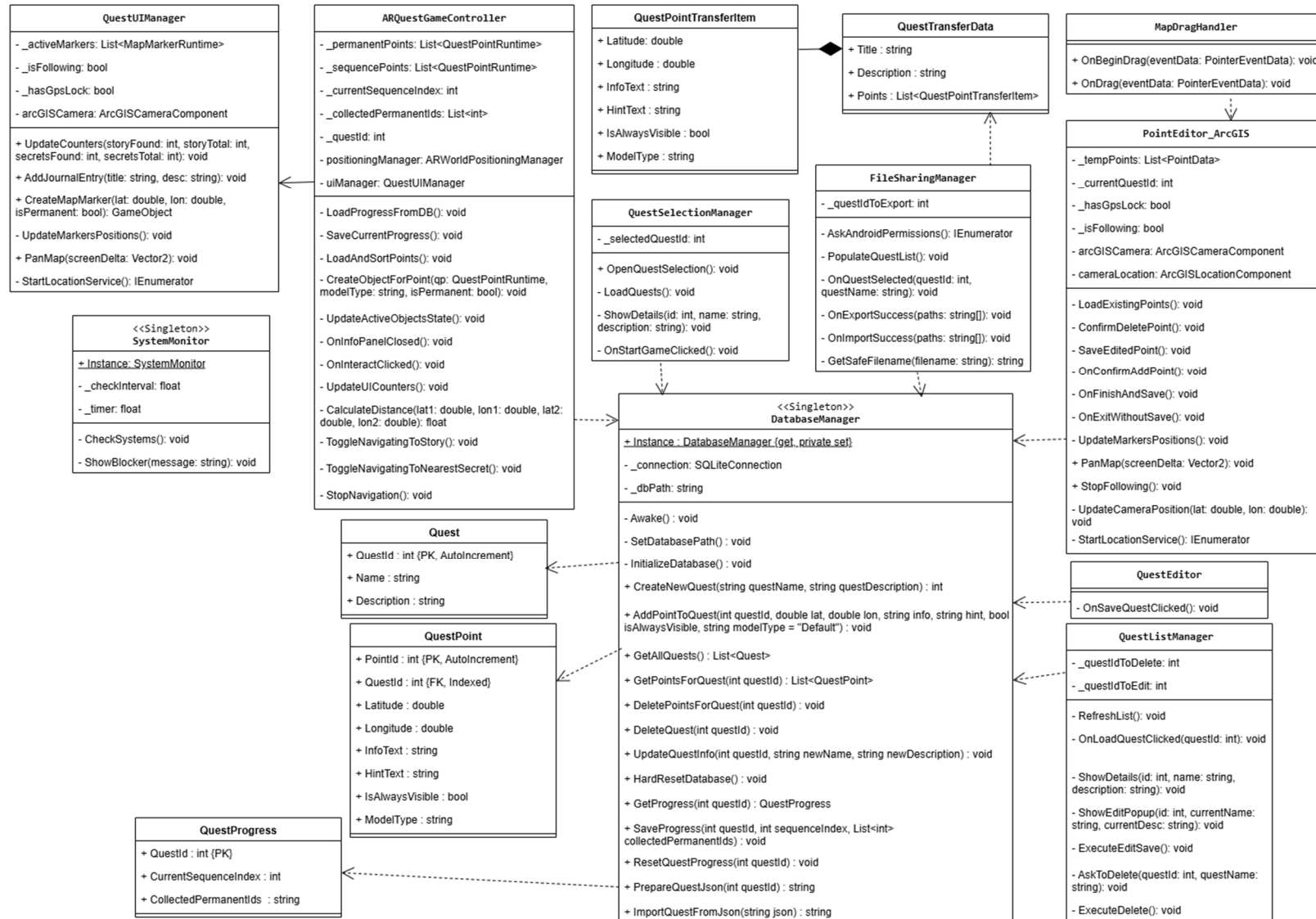
Додаток Г

Схема алгоритму модуля обміну



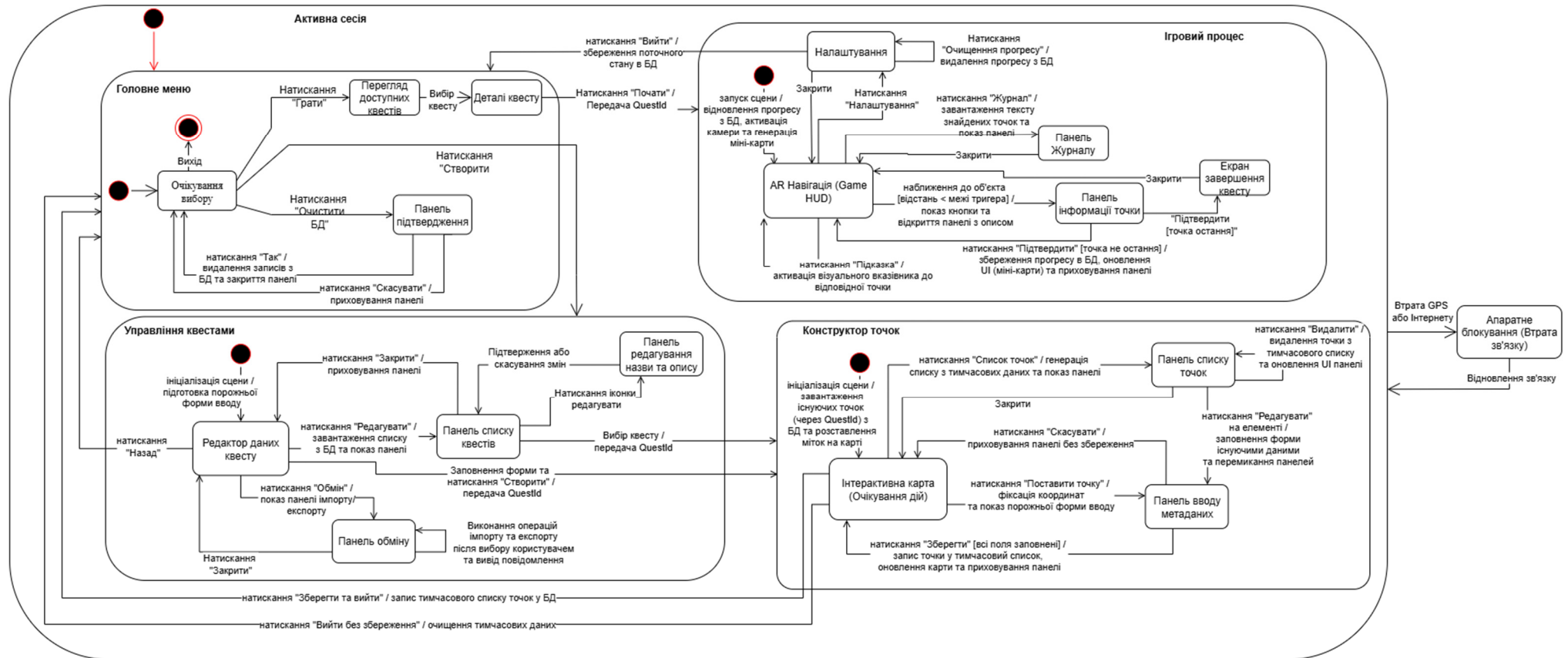
Додаток Д

UML-діаграма класів основної бізнес-логіки



Додаток Е

U□L-діаграма станів графічного інтерфейсу



Модуль DatabaseManager.cs

```
using UnityEngine;
using SQLite;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System;

/// <summary>
/// Глобальний менеджер управління локальною базою даних SQLite. Реалізує патерн
Singleton.
/// </summary>
public class DatabaseManager : MonoBehaviour
{
    public static DatabaseManager Instance { get; private set; }

    private SQLiteConnection _connection; // З'єднання з БД
    private string _dbPath;

    private void Awake()
    {
        // Singleton паттерн
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject); // Знищення дубліката
        }
        else
        {
            Instance = this;
            DontDestroyOnLoad(gameObject); // Не знищувати при переході між сценами

            // Шлях до БД
            SetDatabasePath();
            // Ініціалізація БД
            InitializeDatabase();
        }
    }

    private void SetDatabasePath()
    {
        // Розташування файлу БД
        #if UNITY_EDITOR
            // В редакторі
            _dbPath = Path.Combine(Application.dataPath, "Quests.db");
        #else
            // На реальному пристрої persistentDataPath
            _dbPath = Path.Combine(Application.persistentDataPath, "Quests.db");
        #endif
    }

    private void InitializeDatabase()
    {
        // Створення з'єднання
        _connection = new SQLiteConnection(_dbPath);
    }
}
```

```

    // Створення таблиць
    _connection.CreateTable<Quest>();
    _connection.CreateTable<QuestPoint>();
    _connection.CreateTable<QuestProgress>();
}

/// <summary>
/// Створює новий квест у базі даних.
/// </summary>
/// <param name="questName">Назва квесту.</param>
/// <param name="questDescription">Опис квесту.</param>
/// <returns>ID щойно створеного квесту.</returns>
public int CreateNewQuest(string questName, string questDescription)
{
    var newQuest = new Quest
    {
        Name = questName,
        Description = questDescription
    };

    _connection.Insert(newQuest);

    return newQuest.QuestId;
}

/// <summary>
/// Додає нову просторову точку до існуючого квесту.
/// </summary>
/// <param name="questId">Ідентифікатор квесту, до якого належить точка.</param>
/// <param name="lat">Географічна широта.</param>
/// <param name="lon">Географічна довгота.</param>
/// <param name="info">Текстова інформація (назва точки).</param>
/// <param name="hint">Підказка або опис локації.</param>
/// <param name="isAlwaysVisible">Прапорець, що визначає, чи є точка секретною
(постійно видимою).</param>
/// <param name="modelType">Тип 3D-моделі для відображення в AR.</param>
public void AddPointToQuest(int questId, double lat, double lon, string info, string
hint, bool isAlwaysVisible, string modelType = "Default")
{
    var newPoint = new QuestPoint
    {
        QuestId = questId,
        Latitude = lat,
        Longitude = lon,
        InfoText = info,
        HintText = hint,
        IsAlwaysVisible = isAlwaysVisible,
        ModelType = modelType
    };

    _connection.Insert(newPoint);
}

// Отримати точки конкретного квесту (щоб завантажити їх на карту)
public List<QuestPoint> GetPointsForQuest(int questId)
{
    return _connection.Table<QuestPoint>().Where(p => p.QuestId == questId).ToList();
}

// Видалити всі точки квесту (викликаємо перед збереженням відредагованого квесту)
public void DeletePointsForQuest(int questId)
{
    _connection.Execute("DELETE FROM QuestPoint WHERE QuestId = ?", questId);
}

```

```

    }

    /// <summary>
    /// Виконує незворотне очищення бази даних із видаленням та перестворенням усіх
таблиць.
    /// </summary>
    public void HardResetDatabase()
    {
        // Видаляємо існуючі таблиці
        _connection.DropTable<QuestPoint>();
        _connection.DropTable<Quest>();
        // Створюємо заново
        _connection.CreateTable<Quest>();
        _connection.CreateTable<QuestPoint>();
    }

    /// <summary>
    /// Отримання прогресу квесту. Або створення нового
    /// </summary>
    /// <param name="questId">ID квесту</param>
    public QuestProgress GetProgress(int questId)
    {
        var progress = _connection.Find<QuestProgress>(questId);
        if (progress == null)
        {
            // Якщо запису немає - створюємо новий
            progress = new QuestProgress
            {
                QuestId = questId,
                CurrentSequenceIndex = 0,
                CollectedPermanentIds = ""
            };
            _connection.Insert(progress);
        }
        return progress;
    }

    /// <summary>
    /// Зберігає поточний стан проходження квесту гравцем.
    /// </summary>
    /// <param name="questId">Ідентифікатор активного квесту.</param>
    /// <param name="sequenceIndex">Індекс поточної сюжетної точки.</param>
    /// <param name="collectedPermanentIds">Список ідентифікаторів знайдених секретних
точок.</param>
    public void SaveProgress(int questId, int sequenceIndex, List<int>
collectedPermanentIds)
    {
        // Перетворюємо список ID у рядок "1,2,3"
        string idsString = string.Join(",", collectedPermanentIds);

        var progress = new QuestProgress
        {
            QuestId = questId,
            CurrentSequenceIndex = sequenceIndex,
            CollectedPermanentIds = idsString
        };
        _connection.InsertOrReplace(progress);
    }

    /// <summary>
    /// Формує JSON-рядок із повною структурою квесту для експорту.
    /// </summary>

```

```

/// <param name="questId">Ідентифікатор квесту для серіалізації.</param>
/// <returns>Сформований JSON-рядок або null у разі помилки.</returns>
public string PrepareQuestJson(int questId)
{
    var quest = _connection.Find<Quest>(questId);

    if (quest == null) return null;

    var dbPoints = GetPointsForQuest(questId);
    var transferPoints = new List<QuestPointTransferItem>();

    foreach (var p in dbPoints)
    {
        transferPoints.Add(new QuestPointTransferItem
        {
            Latitude = p.Latitude,
            Longitude = p.Longitude,
            InfoText = p.InfoText,
            HintText = p.HintText,
            IsAlwaysVisible = p.IsAlwaysVisible,
            ModelType = p.ModelType
        });
    }

    QuestTransferData data = new QuestTransferData
    {
        Title = quest.Name,
        Description = quest.Description,
        Points = transferPoints
    };

    return JsonUtility.ToJson(data, true);
}

/// <summary>
/// Десеріалізує JSON-рядок та записує новий квест у базу даних, виконуючи перевірку
на дублікати.
/// </summary>
/// <param name="json">Вміст файлу у форматі JSON.</param>
/// <returns>Статус виконання операції: "SUCCESS", "EXISTS" або "ERROR".</returns>
public string ImportQuestFromJson(string json)
{
    try
    {
        QuestTransferData data = JsonUtility.FromJson<QuestTransferData>(json);

        if (data == null || string.IsNullOrEmpty(data.Title))
            return "ERROR_EMPTY";

        var existingQuest = _connection.Table<Quest>()
            .Where(q => q.Name == data.Title)
            .FirstOrDefault();

        if (existingQuest != null)
        {
            return "EXISTS";
        }

        Quest newQuest = new Quest
        {
            Name = data.Title,
            Description = data.Description
        };
    }
}

```

```

        _connection.Insert(newQuest);

        int newQuestId = newQuest.QuestId;

        if (data.Points != null)
        {
            foreach (var p in data.Points)
            {
                QuestPoint newPoint = new QuestPoint
                {
                    QuestId = newQuestId,
                    Latitude = p.Latitude,
                    Longitude = p.Longitude,
                    InfoText = p.InfoText,
                    HintText = p.HintText,
                    IsAlwaysVisible = p.IsAlwaysVisible,
                    ModelType = p.ModelType
                };
                _connection.Insert(newPoint);
            }
        }

        return "SUCCESS";
    }
    catch (Exception e)
    {
        return "ERROR";
    }
}
}

```

Модуль ARQuestGameController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using Niantic.Experimental.Lightship.AR.WorldPositioning;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using TMPPro;
using Esri.GameEngine.Geometry;

/// <summary>
/// Головний контролер ігрового процесу, що поєднує логіку квесту з модулем доповненої
/// реальності (Lightship ARDK).
/// </summary>
public class ARQuestGameController : MonoBehaviour
{
    [Serializable]
    public struct ModelConfig
    {
        public string typeName;
        public GameObject prefab;
    }

    [SerializeField] private List<ModelConfig> availableModels;
    [SerializeField] private GameObject defaultPrefab;
    [SerializeField] private ARWorldPositioningManager positioningManager;
    [SerializeField] private ARWorldPositioningObjectHelper objectHelper;
    [SerializeField] private QuestUIManager uiManager;

```

```

[Header("НАВИГАЦІЯ")]
[SerializeField] private GameObject navigationArrowPrefab;
[SerializeField] private Transform arrowPivot;
[SerializeField] private Transform arCameraTransform;

private class QuestPointRuntime
{
    public int RealDatabaseId;
    public GameObject Obj;
    public GameObject MinimapMarker;
    public double Lat;
    public double Lon;
    public string Title;
    public string Description;
    public bool IsCollected;
}

private List<QuestPointRuntime> _permanentPoints = new List<QuestPointRuntime>();
private List<QuestPointRuntime> _sequencePoints = new List<QuestPointRuntime>();
private int _currentSequenceIndex = 0;
private List<int> _collectedPermanentIds = new List<int>();
private int _questId;
private QuestPointRuntime _currentInteractionTarget;
private GameObject _currentArrowObj;
private QuestPointRuntime _navigationTarget;

void Start()
{
    _questId = EditorContext.CurrentEditingQuestId;

    // Створюємо стрілку
    if (navigationArrowPrefab)
    {
        _currentArrowObj = Instantiate(navigationArrowPrefab);
        _currentArrowObj.transform.SetParent(arCameraTransform, false);
        arrowPivot = _currentArrowObj.transform.Find("ArrowPivot");

        if (arrowPivot == null)
        {
            Debug.LogError("ArrowPivot НЕ знайдено в префабі!");
        }
        _currentArrowObj.SetActive(false);
    }
    if (distanceText) distanceText.text = "";

    LoadProgressFromDB();
    if (uiManager) uiManager.UpdateCounters(0, 0, 0, 0);
    LoadAndSortPoints();
    UpdateActiveObjectsState();
    RestoreJournalHistory();
    UpdateUICounters();
}

// --- MAIN UPDATE ---

void Update()
{
    if (positioningManager == null) return;
    if (infoPanel.activeSelf) return;

    // --- ПЕРЕВІРКА СТАТУСУ ---
    bool isStable = positioningManager.IsAvailable;

```

```

if (!isStable)
{
    if (statusText) statusText.text = "Пошук GPS... !";
    return;
}
else
{
    if (statusText) statusText.text = "Все ОК...";
}
double camLat = positioningManager.DefaultCameraHelper.Latitude;
double camLon = positioningManager.DefaultCameraHelper.Longitude;

// --- ОНОВЛЕННЯ СТІЛКИ НАВІГАЦІЇ ---
if (_navigationTarget != null && _currentArrowObj != null &&
_currentArrowObj.isActiveSelf)
{
    if (arCameraTransform != null)
    {
        _currentArrowObj.transform.localScale = arrowScale;

        Vector3 targetPosition = arCameraTransform.position +
(arCameraTransform.forward * arrowDistanceInFront);
        targetPosition.y += arrowHeightOffset;

        Vector3 localTargetPos =
            Vector3.forward * arrowDistanceInFront +
            Vector3.up * arrowHeightOffset;

        _currentArrowObj.transform.localPosition = Vector3.Lerp(
            _currentArrowObj.transform.localPosition,
            localTargetPos,
            Time.deltaTime * arrowSmoothSpeed
        );

        if (_navigationTarget.Obj != null)
        {
            Vector3 dir = _navigationTarget.Obj.transform.position -
_currentArrowObj.transform.position;
            dir.y = 0f;

            if (dir.sqrMagnitude > 0.001f)
            {
                arrowPivot.rotation = Quaternion.LookRotation(dir, Vector3.up);
            }
        }
    }

    float distToTarget = CalculateDistance(camLat, camLon, _navigationTarget.Lat,
_navigationTarget.Lon);
    if (distanceText) distanceText.text = $"{Mathf.RoundToInt(distToTarget)} м";
    if (distToTarget < interactionDistance) StopNavigation();
}

// --- ЛОГІКА ПОШУКУ ТОЧОК ---
QuestPointRuntime closestPoint = null;
float minDist = float.MaxValue;

// ПОСТІЙНІ
foreach (var p in _permanentPoints)
{
    if (p.IsCollected) continue;
    if (p.MinimapMarker != null && !p.MinimapMarker.isActiveSelf)
p.MinimapMarker.SetActive(true);
}

```

```

float dist = CalculateDistance(camLat, camLon, p.Lat, p.Lon);
bool visible3D = dist <= viewDistance;

// Якщо об'єкт вже створений - керуємо видимістю
if (p.Obj != null)
{
    // Додаткова перевірка: не показувати 3D, якщо GPS різко змінився
    var renderer = p.Obj.GetComponentInChildren<Renderer>();
    if (renderer != null) renderer.enabled = visible3D;
}

if (visible3D && dist <= interactionDistance && dist < minDist)
{
    minDist = dist;
    closestPoint = p;
}
}

// СЮЖЕТНІ
if (_currentSequenceIndex < _sequencePoints.Count)
{
    var currentP = _sequencePoints[_currentSequenceIndex];
    float dist = CalculateDistance(camLat, camLon, currentP.Lat, currentP.Lon);
    bool visible3D = dist <= viewDistance;

    if (currentP.Obj != null)
    {
        var renderer = currentP.Obj.GetComponentInChildren<Renderer>();
        if (renderer != null) renderer.enabled = visible3D;
    }

    if (visible3D && dist <= interactionDistance && dist < minDist)
    {
        minDist = dist;
        closestPoint = currentP;
    }
}

// КНОПКА
if (closestPoint != null)
{
    _currentInteractionTarget = closestPoint;
    if (!actionButton.activeSelf) actionButton.SetActive(true);
}
else
{
    _currentInteractionTarget = null;
    if (actionButton.activeSelf) actionButton.SetActive(false);
}
}

// --- ІНШІ МЕТОДИ ---
private void RestoreJournalHistory()
{
    if (uiManager == null) return;
    for (int i = 0; i < _currentSequenceIndex && i < _sequencePoints.Count; i++)
    {
        var p = _sequencePoints[i];
        uiManager.AddJournalEntry(p.Title, p.Description);
    }
    foreach (var id in _collectedPermanentIds)
    {

```

```

        var p = _permanentPoints.Find(point => point.RealDatabaseId == id);

        if (p != null) uiManager.AddJournalEntry(p.Title, p.Description);
    }
}

private void LoadProgressFromDB()
{
    var progress = DatabaseManager.Instance.GetProgress(_questId);
    _currentSequenceIndex = progress.CurrentSequenceIndex;
    _collectedPermanentIds.Clear();
    if (!string.IsNullOrEmpty(progress.CollectedExceptionIds))
    {
        var ids = progress.CollectedExceptionIds.Split(new[] { ',' },
StringSplitOptions.RemoveEmptyEntries);
        foreach (var id in ids)
        {
            if (int.TryParse(id, out int parsedId))
                _collectedPermanentIds.Add(parsedId);
        }
    }
}

/// <summary>
/// Завантажує точки з БД, сортує їх на сюжетні та постійні, а також перевіряє статус
їх збору.
/// </summary>
private void LoadAndSortPoints()
{
    var dbPoints = DatabaseManager.Instance.GetPointsForQuest(_questId);
    dbPoints.Sort((a, b) => a.PointId.CompareTo(b.PointId));

    foreach (var p in dbPoints)
    {
        QuestPointRuntime qp = new QuestPointRuntime
        {
            RealDatabaseId = p.PointId,
            Lat = p.Latitude,
            Lon = p.Longitude,
            Title = p.InfoText,
            Description = p.HintText,
            IsCollected = false
        };

        if (p.IsAlwaysVisible)
        {
            if (_collectedPermanentIds.Contains(p.PointId))
            {
                qp.IsCollected = true;
                _permanentPoints.Add(qp);
                continue;
            }
            CreateObjectForPoint(qp, p.ModelType, true);
            _permanentPoints.Add(qp);
        }
        else
        {
            CreateObjectForPoint(qp, p.ModelType, false);
            _sequencePoints.Add(qp);
        }
    }
}
}

```

```

private void CreateObjectForPoint(QuestPointRuntime qp, string modelType, bool
isPermanent)
{
    GameObject prefab = GetPrefabByName(modelType);
    GameObject newObj = Instantiate(prefab);

    objectHelper.AddOrUpdateObject(newObj, qp.Lat, qp.Lon, 0, Quaternion.identity);
    qp.Obj = newObj;

    // Ховаємо ТІЛЬКИ Renderer
    var renderer = newObj.GetComponentInChildren<Renderer>();
    if (renderer != null) renderer.enabled = false;

    if (uiManager != null)
    {
        GameObject marker = uiManager.CreateMapMarker(qp.Lat, qp.Lon, isPermanent);
        marker.SetActive(false);
        qp.MinimapMarker = marker;
    }
}

private void UpdateActiveObjectsState()
{
    for (int i = 0; i < _sequencePoints.Count; i++)
    {
        var pt = _sequencePoints[i];

        if (i < _currentSequenceIndex)
        {
            pt.IsCollected = true;
            if (pt.Obj != null)
            {
                var renderer = pt.Obj.GetComponentInChildren<Renderer>();
                if (renderer != null) renderer.enabled = false;
            }
            if (pt.MinimapMarker != null) pt.MinimapMarker.SetActive(false);
        }
        else if (i == _currentSequenceIndex)
        {
            pt.IsCollected = false;
            if (pt.MinimapMarker != null) pt.MinimapMarker.SetActive(true);
        }
        else
        {
            pt.IsCollected = false;
            if (pt.Obj != null)
            {
                var renderer = pt.Obj.GetComponentInChildren<Renderer>();

                if (renderer != null) renderer.enabled = false;
            }
            if (pt.MinimapMarker != null) pt.MinimapMarker.SetActive(false);
        }
    }
}

/// <summary>
/// Обробляє подію взаємодії користувача із знайденим AR-об'єктом.
/// </summary>
private void OnInteractClicked()
{
    if (_currentInteractionTarget == null) return;

```

```

// Перевіряємо, чи є хоч якийсь текст
bool hasTitle = !string.IsNullOrEmpty(_currentInteractionTarget.Title);
bool hasDesc = !string.IsNullOrEmpty(_currentInteractionTarget.Description);

if (!hasTitle && !hasDesc)
{
    // Якщо тексту нуль – збираємо точку без повідомлення.
    OnInfoPanelClosed();
}
else
{
    // Якщо текст є – показуємо панель
    ShowInfoPanel(_currentInteractionTarget);
}
}

/// <summary>
/// Оновлює стан об'єктів у сцені після закриття інформаційної панелі, зараховує
прогрес та записує дані в журнал.
/// </summary>
private void OnInfoPanelClosed()
{
    infoPanel.SetActive(false);

    if (_currentInteractionTarget != null)
    {
        _currentInteractionTarget.IsCollected = true;
        if (_currentInteractionTarget.Obj != null)
        {
            var renderer =
            _currentInteractionTarget.Obj.GetComponentInChildren<Renderer>();
            if (renderer != null) renderer.enabled = false;
        }
        if (_currentInteractionTarget.MinimapMarker != null)
            _currentInteractionTarget.MinimapMarker.SetActive(false);

        if (uiManager) uiManager.AddJournalEntry(_currentInteractionTarget.Title,
            _currentInteractionTarget.Description);

        if (_permanentPoints.Contains(_currentInteractionTarget))
        {
            if
            (!_collectedPermanentIds.Contains(_currentInteractionTarget.RealDatabaseId))
                _collectedPermanentIds.Add(_currentInteractionTarget.RealDatabaseId);
        }
        else if (_sequencePoints.Contains(_currentInteractionTarget))
        {
            if (_sequencePoints.IndexOf(_currentInteractionTarget) ==
            _currentSequenceIndex)
            {
                _currentSequenceIndex++;
                UpdateActiveObjectsState();

                if (_currentSequenceIndex >= _sequencePoints.Count)
                {
                    if (questCompletePopup) questCompletePopup.SetActive(true);
                }
            }
        }
        SaveCurrentProgress();
        UpdateUICounters();
    }
}

```

```

        if (_navigationTarget == _currentInteractionTarget) StopNavigation();
        _currentInteractionTarget = null;
    }

    /// <summary>
    /// Обчислює дистанцію між двома географічними координатами за сферичною формулою
    Гаверсинуса.
    /// </summary>
    /// <param name="lat1">Широта першої координати.</param>
    /// <param name="lon1">Довгота першої координати.</param>
    /// <param name="lat2">Широта другої координати.</param>
    /// <param name="lon2">Довгота другої координати.</param>
    /// <returns>Фізична відстань у метрах.</returns>
    private float CalculateDistance(double lat1, double lon1, double lat2, double lon2)
    {
        const double R = 6371000;
        double dLat = Mathf.Deg2Rad * (float)(lat2 - lat1);

        double dLon = Mathf.Deg2Rad * (float)(lon2 - lon1);
        double a = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) +
            Math.Cos(Mathf.Deg2Rad * (float)lat1) *
            Math.Cos(Mathf.Deg2Rad * (float)lat2) *
            Math.Sin(dLon / 2) * Math.Sin(dLon / 2);
        double c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
        return (float)(R * c);
    }
}

```

Модуль PointEditor_ArcGIS.cs

```

using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;
using System.Collections;
using System.Collections.Generic;
using Esri.ArcGISMapsSDK.Components;
using Esri.GameEngine.Geometry;
using Esri.ArcGISMapsSDK.Utills.GeoCoord;
using System;
using System.Linq;

/// <summary>
/// Контролер просторового конструювання квестів. Інтегрує картографічний сервіс ArcGIS
/// для візуалізації та розстановки точок.
/// </summary>
public class PointEditor_ArcGIS : MonoBehaviour
{
    [System.Serializable]
    public class PointData
    {
        public double Latitude;
        public double Longitude;
        public string Info;
        public string Hint;
        public bool IsAlwaysVisible;
        public string ModelType;
        public GameObject MarkerObj;
    }

    [Header("Markers & Map")]

```

```

[SerializeField] private GameObject markerPrefab;
[SerializeField] private Transform markerContainer;
[SerializeField] private RectTransform mapRectTransform;
[SerializeField] private float edgePadding = 20f;

[Header("ArcGIS Dependencies")]
[SerializeField] private ArcGISCameraComponent arcGISCamera;
[SerializeField] private ArcGISLocationComponent cameraLocation;

[Header("Settings")]
[SerializeField] private float panSpeedMultiplier = 0.0005f;

// Підібрані еталонні данні
[SerializeField] private float baseRadarZoom = 5f;
[SerializeField] private float baseMapSize = 2026f;
[SerializeField] private float verticalCorrection = 1.0f;

// Яку саме точку зараз збираємось видаляти
private int _pointToDeleteIndex = -1;

private List<PointData> _tempPoints = new List<PointData>();
private int _currentQuestId;
private bool _hasGpsLock = false;
private bool _isFollowing = true;
private double _lastLat = 0;
private double _lastLon = 0;
private const double _locationThreshold = 0.00001;

// Індекс точки, яку зараз редагуємо (-1 означає ніяку)
private int _editingIndex = -1;

void Start()
{
    _currentQuestId = EditorContext.CurrentEditingQuestId;
    // ЗАВАНТАЖУЄМО ТОЧКИ З БАЗИ
    LoadExistingPoints();
    StartCoroutine(StartLocationService());
}

// --- ЗАВАНТАЖЕННЯ ТОЧОК ---
private void LoadExistingPoints()
{
    _tempPoints.Clear(); // Чистимо список перед завантаженням

    // Беремо дані з бази
    var dbPoints = DatabaseManager.Instance.GetPointsForQuest(_currentQuestId);

    if (dbPoints != null && dbPoints.Count > 0)
    {
        foreach (var dbP in dbPoints)
        {
            PointData p = new PointData
            {
                Latitude = dbP.Latitude,
                Longitude = dbP.Longitude,
                Info = dbP.InfoText,
                Hint = dbP.HintText,
                IsAlwaysVisible = dbP.IsAlwaysVisible,
                ModelType = dbP.ModelType
            };
            // Якщо в базі було пусто, ставимо дефолт
            if (string.IsNullOrEmpty(p.ModelType)) p.ModelType = "Default";
            CreateMarkerObject(p); // Створюємо візуал
        }
    }
}

```

```

        _tempPoints.Add(p);
    }
}

// Метод викликається, якщо гравець натиснув "ТАК" у вікні підтвердження
private void ConfirmDeletePoint()
{
    // Перевіряємо, чи індекс адекватний (чи не дорівнює -1 і чи не виходить за межі
    списку)
    if (_pointToDeleteIndex >= 0 && _pointToDeleteIndex < _tempPoints.Count)
    {
        // 1. Видаляємо маркер з карти
        if (_tempPoints[_pointToDeleteIndex].MarkerObj != null)
        {
            Destroy(_tempPoints[_pointToDeleteIndex].MarkerObj);
        }
        // 2. Видаляємо з даних
        _tempPoints.RemoveAt(_pointToDeleteIndex);
        // 3. Оновлюємо список
        OpenPointList();
    }

    // Закриваємо вікно і скидаємо індекс
    if (deleteConfirmPopup) deleteConfirmPopup.SetActive(false);
    _pointToDeleteIndex = -1;
}

/// <summary>
/// Зчитує координати центру екрана (або поточного місцеположення) та створює новий
об'єкт точки в оперативній пам'яті.
/// </summary>
private void OnConfirmAddPoint()
{
    if (cameraLocation == null) return;

    // Координати беремо з прицілу (центр екрану / поточна позиція камери)
    double lat = cameraLocation.Position.Y;
    double lon = cameraLocation.Position.X;

    // Зчитуємо обрану модель з Dropdown
    string selectedModel = "Default";
    if (addModelDropdown != null && addModelDropdown.options.Count > 0)
    {
        selectedModel = addModelDropdown.options[addModelDropdown.value].text;
    }

    bool isVisible = false; // За замовчуванням false
    if (pointTypeToggle != null)
    {
        isVisible = !pointTypeToggle.IsStoryActive; // Якщо вибрана Секретна (не
        Сюжетна), то IsAlwaysVisible = true
    }

    PointData newPoint = new PointData
    {
        Latitude = lat,
        Longitude = lon,
        Info = infoTextInput.text,
        Hint = hintTextInput.text,
        IsAlwaysVisible = isVisible,
        ModelType = selectedModel // Записуємо обрану модель!
    }
}

```

```

        //IsAlwaysVisible = alwaysVisibleToggle != null && alwaysVisibleToggle.isOn,
    };

    CreateMarkerObject(newPoint);
    _tempPoints.Add(newPoint);

    // Закриваємо панель і повертаємося до карти
    CloseAddPanel();
}

/// <summary>
/// Виконує пакетне збереження всіх створених точок у базу даних, попередньо видаливши
старі записи для цього квесту.
/// </summary>
private void OnFinishAndSave()
{
    if (_tempPoints.Count == 0)
    {
        // Відкриваємо вікно попередження для гравця
        if (emptyQuestWarningPopup) emptyQuestWarningPopup.SetActive(true);
        return;
    }
    // 1. ВИДАЛЯЄМО СТАРІ ТОЧКИ
    DatabaseManager.Instance.DeletePointsForQuest(_currentQuestId);

    // 2. ЗАПИСУЄМО ВСІ ТОЧКИ ЗІ СПИСКУ
    foreach (var p in _tempPoints)
    {
        DatabaseManager.Instance.AddPointToQuest(
            _currentQuestId,
            p.Latitude,
            p.Longitude,
            p.Info,
            p.Hint,
            p.IsAlwaysVisible,
            p.ModelType
        );
    }
    SceneManager.LoadScene("MainMenu_Scene");
}

// --- (методи оновлення маркерів, GPS і PanMap) ---

void Update()
{
    UpdateMarkersPositions();

    if (!_hasGpsLock || !_isFollowing || Input.location.status !=
    LocationServiceStatus.Running) return;

    double lat = Input.location.lastData.latitude;
    double lon = Input.location.lastData.longitude;
    if (Math.Abs(lat - _lastLat) > _locationThreshold || Math.Abs(lon - _lastLon) >
    _locationThreshold)
    {
        _lastLat = lat;
        _lastLon = lon;
        UpdateCameraPosition(lat, lon);
    }
}

/// <summary>

```

```

    /// Динамічно розраховує та оновлює локальні координати (екранні пікселі) для маркерів
точок,
    /// базуючись на їх географічних координатах та поточному рівні масштабування радара.
    /// </summary>
private void UpdateMarkersPositions()
{
    if (cameraLocation == null || mapRectTransform == null) return;

    float mapHalfWidth = (mapRectTransform.rect.width / 2f) - edgePadding;
    float mapHalfHeight = (mapRectTransform.rect.height / 2f) - edgePadding;

    // --- ДИНАМІЧНИЙ РОЗРАХУНОК ЗУМУ ---
    float currentMapSize = mapRectTransform.rect.height;
    // Вираховуємо правильний зум за пропорцією
    float radarZoom = baseRadarZoom * (currentMapSize / baseMapSize);
    // -----

    double userLat = cameraLocation.Position.Y;
    double userLon = cameraLocation.Position.X;

    const double metersPerLat = 111320.0;
    double metersPerLon = 111320.0 * Math.Cos(userLat * Mathf.Deg2Rad);

    int sequentialIndex = 1;

    foreach (var point in _tempPoints)
    {
        if (point.MarkerObj == null) continue;

        double dLat = point.Latitude - userLat;
        double dLon = point.Longitude - userLon;

        double distY_Meters = dLat * metersPerLat;
        double distX_Meters = dLon * metersPerLon;

        float pixelX = (float)(distX_Meters * radarZoom);
        float pixelY = (float)(distY_Meters * radarZoom);

        pixelY *= verticalCorrection;

        Vector2 targetPos = new Vector2(pixelX, pixelY);

        bool insideX = Mathf.Abs(targetPos.x) <= mapHalfWidth;
        bool insideY = Mathf.Abs(targetPos.y) <= mapHalfHeight;

        if (insideX && insideY)
        {
            point.MarkerObj.transform.localPosition = targetPos;
        }
        else
        {
            Vector2 dir = targetPos.normalized;
            float tX = (dir.x != 0) ? (dir.x > 0 ? mapHalfWidth : -mapHalfWidth) /
dir.x : float.MaxValue;
            float tY = (dir.y != 0) ? (dir.y > 0 ? mapHalfHeight : -mapHalfHeight) /
dir.y : float.MaxValue;
            float t = Mathf.Min(Mathf.Abs(tX), Mathf.Abs(tY));

            point.MarkerObj.transform.localPosition = dir * t;
        }

        var textComp = point.MarkerObj.GetComponentInChildren<TMP_Text>();
        if (textComp != null)

```

```

        {
            if (point.IsAlwaysVisible) textComp.text = "X";
            else { textComp.text = sequentialIndex.ToString(); sequentialIndex++; }
        }
    }
}

/// <summary>
/// Виконує переміщення камери на карті за допомогою жестів (свайпів), відключаючи при цьому автоматичне слідування за GPS.
/// </summary>
/// <param name="screenDelta">Вектор зміщення сенсорного вводу в пікселях.</param>
public void PanMap(Vector2 screenDelta)
{
    StopFollowing();
    float speed = 500f * panSpeedMultiplier;

    Vector3 moveVector = (arcGISCamera.transform.right * screenDelta.x) +
        (arcGISCamera.transform.up * screenDelta.y);

    arcGISCamera.transform.position -= moveVector * speed;
}

/// <summary>
/// Переміщення камери (прицілу) в 3D-сцені.
/// </summary>
private void UpdateCameraPosition(double lat, double lon)
{
    if (cameraLocation == null) return;
    cameraLocation.Position = new ArcGISPoint(lon, lat, 500,
ArcGISSpatialReference.WGS84());
    cameraLocation.Rotation = new ArcGISRotation(0, 0, 0);
}

```

Модуль FileSharingManager.cs

```

using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System.IO;
using System.Collections;
using SimpleFileBrowser;
using System.Text.RegularExpressions;

/// <summary>
/// Менеджер обміну файлами, що відповідає за взаємодію з файловою системою пристрою (Android) та операції імпорту/експорту.
/// </summary>
public class FileSharingManager : MonoBehaviour
{
    private int _questIdToExport;

    void Start()
    {
        // --- Налаштування для Android ---
        FileBrowser.SetFilters(true, new FileBrowser.Filter("Quests", ".json"));
        FileBrowser.SetDefaultFilter(".json");
        // Дозволяємо читати/писати
        FileBrowser.AddQuickLink("Users", "Users", null);
    }

    /// <summary>

```

```

    /// Здійснює генерацію JSON-даних та їх фізичний запис у файлову систему після
    підтвердження шляху користувачем.
    /// </summary>
    /// <param name="paths">Масив шляхів до вибраної директорії збереження.</param>
    private void OnExportSuccess(string[] paths)
    {

        if (paths.Length == 0) return;
        string path = paths[0];

        // 1. Генеруємо JSON
        string json = DatabaseManager.Instance.PrepareQuestJson(_questIdToExport);

        // 2. ПЕРЕВІРКА НА ПУСТОТУ
        if (string.IsNullOrEmpty(json) || json == "{}")
        {
            if (statusText)
            {
                statusText.text = "Помилка: JSON пустий! Файл не створено.";
                statusText.color = Color.red;
            }
            return;
        }

        // 3. ЗАПИС ФАЙЛУ (ЧЕРЕЗ HELPER ПЛАГІНА)
        try
        {
            FileBrowserHelpers.WriteTextToFile(path, json);

            if (statusText)
            {
                statusText.text = "Успіх! Файл збережено.";
                statusText.color = Color.green;
            }
        }
        catch (System.Exception e)
        {
            if (statusText) statusText.text = "Помилка запису: " + e.Message;
        }
    }

    /// <summary>
    /// Зчитує вміст обраного JSON-файлу та передає його до менеджера бази даних для
    розпакування.
    /// </summary>
    /// <param name="paths">Масив шляхів, що містить шлях до обраного файлу.</param>
    private void OnImportSuccess(string[] paths)
    {
        if (paths.Length == 0) return;
        string path = paths[0];

        // Читаємо через Helper плагіна
        string json = FileBrowserHelpers.ReadTextFromFile(path);
        string result = DatabaseManager.Instance.ImportQuestFromJson(json);

        if (result == "SUCCESS")
        {
            if (statusText) { statusText.text = "Квест успішно додано!"; statusText.color
= Color.green; }
            if (exportPanel.activeSelf) PopulateQuestList();
        }
        else if (result == "EXISTS")
        {

```

```

        if (statusText) { statusText.text = "Такий квест вже існує!"; statusText.color
= Color.yellow; }
    }
    else
    {
        if (statusText) { statusText.text = "Помилка даних."; statusText.color =
Color.red; }
    }
}

/// <summary>
/// Очищає рядок від недопустимих символів за допомогою регулярних виразів для
безпечного формування назви файлу.
/// </summary>
/// <param name="filename">Оригінальна назва квесту.</param>
/// <returns>Безпечна назва файлу, придатна для використання у файлових
системах.</returns>
private string GetSafeFilename(string filename)
{
    // 1. Замінюємо пробіли на підкреслення
    string safe = filename.Replace(" ", "_");

    // 2. Фільтруємо:
    // \p{L} - будь-яка літера (українська 'і', 'ї', 'є', англійська тощо)
    // \p{N} - будь-яка цифра
    safe = Regex.Replace(safe, @"^[^\\p{L}\\p{N}_-]", "");
    return safe;
}
}

```

Додаток И

Апробація отриманих результатів

Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління



Матеріали

I Міжнародної науково-практичної конференції студентів і молодих вчених
«**ICSPM 2026: Intelligent Computing Systems and Project Management /
Інтелектуальні обчислювальні системи та управління проєктами**»
21–22 травня 2026 р.



м. Тернопіль - 2026

УДК 004.8:005.8](063)

Присвячено 60-річчю Західноукраїнського національного університету

ICSPM 2026: Інтелектуальні обчислювальні системи та управління проектами : збірник тез доповідей I Міжнародної науково-практичної конференції студентів і молодих вчених (Тернопіль, 21–22 травня 2026 року). – Тернопіль : ЗУНУ, 2026. – 190 с.

До збірника увійшли тези доповідей учасників I Міжнародної науково-практичної конференції студентів і молодих вчених «ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проектами», що відбулася на базі кафедри інформаційно-обчислювальних систем і управління факультету комп'ютерних інформаційних технологій Західноукраїнського національного університету.

Матеріали відображають результати досліджень за такими напрямками:

1. Інтелектуальні обчислення та програмні системи.
2. Проектно-орієнтоване управління та процеси прийняття рішень.
3. Штучний інтелект, аналітика даних і кібернетика.
4. Прикладні технології та інформаційно-комунікаційні системи.
5. Сучасні інформаційні технології в економіці, праві та освіті.

Рекомендовано до друку на засіданні кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету (протокол №12 від 26.05.2026 р.).

За зміст наукових праць, точність наведених цитувань, перекладу та достовірність фактологічних матеріалів відповідальність несуть автори публікацій. У збірнику збережено авторську стилістику, пунктуацію та орфографію.

© Кафедра інформаційно-обчислювальних систем і управління ЗУНУ, 2026
© Західноукраїнський національний університет, 2026

ЗМІСТ

*СЕКЦІЯ - ІНТЕЛЕКТУАЛЬНІ ОБЧИСЛЕННЯ ТА ПРОГРАМНІ СИСТЕМИ /
INTELLIGENT COMPUTING AND SOFTWARE SYSTEMS*

D. Hural, N. Dziubanovska, R. Skalii

OPERATING SYSTEM ARCHITECTURE FOR THE IMPLEMENTATION OF
INTELLIGENT CONTROL OF A SOLAR POWER INSTALLATION USING A
DIGITAL TWIN 12

Vitalii Leus, Oleksandr Osolinskyi

CONCEPT OF AN INTELLIGENT HARDWARE CONTROL SYSTEM BASED
ON GESTURE RECOGNITION 16

Вібла К.Т., Васильків Н.М.

СЕРВІС ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ХАРЧУВАННЯ І ФІЗИЧНИХ
НАВАНТАЖЕНЬ 20

Воевудський О.О., Турченко І.В.

БЕЗКОНТАКТНИЙ ІНТЕРФЕЙС ВІДЕО-КЕРУВАННЯ ЦИФРОВИМ
ДВІЙНИКОМ В ІМЕРСИВНОМУ ЦИФРОВОМУ СЕРЕДОВИЩІ 23

Ковтуненко А.О., Биковий П.Є.

ПРОГРАМНИЙ МОДУЛЬ ВИЯВЛЕННЯ ТА ВІДСЛІДКОВУВАННЯ ОБЛИЧ
ІЗ ВИКОРИСТАННЯМ ДРОНА RYZE TELLO 27

Матвійчук О.О., Биковий П.Є.

ПРОГРАМНИЙ МОДУЛЬ ОБРОБКИ ТЕКСТОВИХ ЗАПИТІВ ДЛЯ
РЕКОМЕНДАЦІЙ ФІЛЬМІВ У TELEGRAM-БОТІ 30

Новак Х.І., Турченко І.В.

СИСТЕМА УПРАВЛІННЯ НОТАТКАМИ З ФУНКЦІЯМИ МЕНЕДЖЕРА
ПАРОЛІВ ТА СПІЛЬНОГО ДОСТУПУ 34

Панасюк І.С., Лаштун М.М., Дубчак Л.О.

ПРОЄКТУВАННЯ МЕХАНІЗМУ ОБРОБКИ ТЕСТОВИХ ДАНИХ ТА
ІНТЕГРАЦІЇ У ФРЕЙМВОРК АВТОМАТИЗАЦІЇ 38

Росоловський Ю.М., Турченко І.В.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА АНАЛІЗУ ФІНАНСОВОЇ ПОВЕДІНКИ
КОРИСТУВАЧА НА ОСНОВІ МАШИННОГО НАВЧАННЯ 41

Савчук Д.В., Биковий П.Є.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ПРОХОДЖЕННЯ AR-
КВЕСТІВ ІЗ ВИКОРИСТАННЯМ ГЕОЛОКАЦІЇ 45

УДК 004.9 : 004.5

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ПРОХОДЖЕННЯ AR-КВЕСТІВ ІЗ ВИКОРИСТАННЯМ ГЕОЛОКАЦІЙ

Савчук Дмитро Володимирович, студент,
savcukd11@gmail.com

Науковий керівник: Биковий Павло Євгенович,
к.т.н., доцент, доцент кафедри інформаційно-
обчислювальних систем і управління
Західноукраїнський національний університет,
pb@wunu.edu.ua
ORCID: 0000-0002-5705-5702

Сучасний розвиток мобільних технологій та засобів доповненої реальності (Augmented Reality, AR) надає багато нових можливостей для гейміфікації таких сфер, як туризм і освіта. Поєднання AR-технологій з глобальним позиціонуванням дозволяє створювати імерсивні просторові сценарії, в яких віртуальні об'єкти інтегруються у фізичне середовище [1, 2]. Проте аналіз наявних рішень (наприклад, Pokémon GO або платформ для геокешингу) показав відсутність інструментів для простого створення власних AR-квестів з прив'язкою до координат місцевості. Більшість аналогів надають готові маршрути або не мають функціоналу доповненої реальності. Складніші системи розробки (Unity, Unreal Engine) потребують додаткових знань, що не є зручним для звичайного користувача.

З огляду на це, метою дослідження є проектування та реалізація мобільного застосунку, який дозволяє створювати, редагувати та проходити власні AR-квести з прив'язкою до координат місцевості.

Для реалізації застосунку обрано кросплатформне середовище Unity 3D (мова C#). Генерацію карти та отримання координат місцевості виконано за допомогою сервісу ArcGIS Maps. Для розпізнавання поверхонь та закріплення 3D-об'єктів у певній місцевості використовувався спеціалізований фреймворк Niantic Lightship ARDK [3-6].

Архітектуру розробленої системи поділено на три основні взаємопов'язані модулі: модуль управління даними, модуль створення сценаріїв та ігрову AR-сцену.

Модуль управління даними відповідає за доступ до бази даних та роботу з даними. Щоб уникнути використання централізованого серверу чи дорогих хмарних сервісів, для реалізації додатка було застосовано локальну СУБД SQLite. За допомогою цього усі дані про сценарії (опис, метадані, вибір 3D-моделі) зберігаються на пристрої. Цей модуль є основним для системи, адже сюди звертаються решта розділів при взаємодії з даними (CRUD операції, збереження та завантаження прогресу). Через відсутність централізованого

сервера обмін реалізовано за принципом Peer-to-Peer: система створює JSON-файл із даними квесту, який зберігається в пам'яті пристрою. Далі цей файл можна надіслати через будь-який месенджер та імпортувати в додатку іншого користувача.

Модуль створення сценаріїв є основним інструментом для авторів квесту. Цей розділ складається з двох підрозділів, в першому наданий функціонал створення та редагування самого квесту, а другий потрібен для позиціонування та налаштування точок інтересу.

Інтерфейс просторового конструктора базується на повноекранній інтерактивній карті, побудованій за допомогою ArcGIS. Автор може розміщувати точки інтересу на карті, налаштовувати для кожної: назву, опис, тип видимості (постійна, послідовна), 3D-модель. Усі допоміжні елементи реалізовані у вигляді панелей, які з'являються при натисканні відповідних кнопок, щоб не перекривати огляд карти.

Ігрова AR-сцена дозволяє проходити завантажені або створені квести. Для орієнтації та пошуку об'єктів гравець може використовувати міні-карту та динамічну 3D-стрілку. Ця сцена постійно обраховує відстані від користувача до об'єктів, контролюючи видимість об'єктів та зону взаємодії. Оскільки координати мають формат WGS 84 (широта та довгота), для визначення відстані використовується математична сферична модель (формула Гаверсінуса):

$$d = 2R \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right), \quad (1)$$

де φ – широта в радіанах;

λ – довгота в радіанах;

R – радіус Землі (~6371000 м).

Коли відстань d стає меншою за встановлений поріг (для видимості – 50 метрів), система починає рендерити віртуальний об'єкт за допомогою Lightship. Коли гравець наближається достатньо близько, він може взаємодіяти з точкою інтересу, після чого відкривається панель з інформацією та активується наступна точка.

Важливим елементом системи є сервіс перевірки підключення до мережі Інтернет та активності модуля GPS. При втраті підключення вмикається блокуюча панель з повідомленням для користувача. Цей сервіс працює у фоні весь життєвий цикл програми.

Екранні форми застосунку зображено на рисунку 1.

Таким чином, в результаті проведеного дослідження розроблено мобільний застосунок, що об'єднує інструменти картографії та доповненої реальності. Запропонований підхід до управління даними (через SQLite та JSON-файли) звільняє розробників від необхідності підтримки серверної інфраструктури, водночас надаючи користувачам потужний конструктор для створення унікальних авторських маршрутів.



Рисунок 1 – Екранні форми розробленої системи

Рішення має значний практичний потенціал для впровадження у туристичній галузі та сфері неформальної освіти.

Список використаних джерел

1. Теслюк В. М., Гадьо І. В., Гавриляк Т. В., Пазюк О. В. Модель та засоби системи навігації між корпусами закладу освіти з використанням технології доповненої реальності. *Український журнал інформаційних технологій*. 2024. Т. 6, № 1. С. 86–94. DOI: <https://doi.org/10.23939/ujit2024.01.086>.
2. Qiu Z., Mostafavi A., Kalantari S. Use of Augmented Reality in Human Wayfinding: A Systematic Review. *arXiv*. 2023. URL: <https://arxiv.org/abs/2311.11923> (дата звернення: 23.04.2026).
3. Niantic Spatial SDK Documentation. URL: https://www.nianticspatial.com/docs/nsdk/xr_settings/?platform=unity (дата звернення: 23.04.2026).
4. AR Foundation Documentation. URL: <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.4/manual/> (дата звернення: 23.04.2026).
5. Шевчук Р., Тихий Р. Аналіз технології та інструментів для реалізації систем доповненої реальності. *Оптико-електронні інформаційно-енергетичні технології*. 2024. № 47(1). С. 66–77. DOI: <https://doi.org/10.31649/1681-7893-2024-47-1-66-77>.
6. Методи та програмні засоби формування доповненої реальності. URL: <https://ela.kpi.ua/items/6616966e-49c0-4000-afc8-be952b57b217> (дата звернення: 23.04.2026).