МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЄКТ
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для гри у монополію «MonopolyUA» Frontend частина
(тема)

Виконав:
здобувач 3 курсу, групи ПЗПІ-22-6
Іван ДУХОТА
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Костянтин ОНИЩЕНКО
(посада, Власне ім'я, ПРІЗВИЩЕ)

Члени комісії (Власне ім'я, ПРІЗВИЩЕ, підпис)
Доц. Мазурова О. О.
Доц. Груздо І. В.
Ст. викл. Саманцов О. О.

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

Курс 3 Група ПЗПІ-22-6 Семестр 6

ЗАВДАННЯ
на курсовий проект(роботу) студента

здобувачеві Духоті Івану Євгенійовичі
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для гри у Монополію "MonopolyUA" Frontend частина

2. Термін здачі студентом закінченої роботи „20" червня 2025 р.

3. Вихідні дані до проєкту Розробка веб-застосунку для гри у монополію поділену на три частини: frontend, backend sync, backend async.

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз проблеми та її рішення, цільова аудиторія та UI/UX дизайн. Як розробити привабливий, зрозумілий та комфортний інтерфейс для користувачів?

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	07.06.2025	виконано
2	Розробка постановки задачі	07.06.2025	виконано
3	Проектування ПЗ	07.06.2025	виконано
4	Програмна реалізація	07.06.2025	виконано
5	Аналіз результатів	07.06.2025	виконано
6	Підготовка пояснювальної записки.	08.06.2025	виконано
7	Перевірка на наявність ознак академічного плагіату		виконано
8	Захист роботи		виконано

Дата видачі завдання “26” лютого 2025р.

Здобувач Духота Іван .
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Костянтин ОНИЩЕНКО
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 76 стор., 34 рис., 1 табл., 10 джерел.

ВЕБ-ЗАСТОСУНОК, МОНОПОЛІЯ, ФРОНТЕНД ЧАСТИНА, HTML, CSS, JS, FETCH API

Об’єкт розробки – веб-застосунок для гри у «Монополія» «MonopolyUA».

Мета розробки – створення frontend частини для веб-застосунку, яка матиме наступні сторінки: реєстрації/авторизації, головну, особистого профілю, маркету, відкриття кейсів, пошуку/створення гри та грального дошки.

Метод рішення – середовище розробки Visual Studio Code [9], мови програмування HTML, CSS, JavaScript.

У результаті розробки створено frontend частину веб-застосунку «MonopolyUA», котра складається з великої кількості сторінок з різними функціональними можливостями.

WEB APPLICATION, MONOPOLY, FRONTEND PART, HTML, CSS, JS, FETCH API

The object of development is a web application for the game «Monopoly» «MonopolyUA».

The purpose of development is to create a frontend part for the web application, which will have the following pages: registration/authorization, home, personal profile, market, opening cases, search/creation of the game and game board.

The solution method is the Visual Studio Code development environment, programming languages HTML, CSS, JavaScript.

As a result of the development, the frontend part of the web application «MonopolyUA» was created, which consists of a large number of pages with different functionalities.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS - JavaScript

MPA – Multiple Page Application

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

UML – Unified Modeling Language

UUID – Унікальний ідентифікатор об’єкта

JWT – JSON Web Token

DRF – Django REST Framework

ВСТУП

Темою комплексного курсового проєкту є створення frontend частини для веб-застосунку «MonopolyUA». Основне завдання полягає у надані кожному користувачеві зручного інтерфейсу для взаємодії з грою та основним функціоналом гри.

Метою роботи є розробка frontend частини для веб-застосунку, яка складається з великої кількості сторінок та інтерактивних елементів. Вона включає сторінку реєстрації та авторизації, де користувач може створити новий обліковий запис або війти у вже існуючий, головну сторінку з інформацію про застосунок, правила гри, та список найкращих гравців за весь час. Також реалізовано сторінку особистого профілю, де користувач може переглядати та змінювати інформацію про себе, переглядати список друзів, особисту статистику ігор та керувати інвентарем: відкривати, обирати або продавати предмети. Крім того користувач також має доступ до сторінки маркету, де у нього є можливість купувати предмети інших користувачів та контролювати продаж своїх. На сторінках створення або пошуку гри, а також на самій гральній дошці, можна починати матчі з друзями або брати участь у рейтингових поєдинках відповідно. Для розробки використовувалася Visual Studio Code та мови програмування HTML, CSS та JS [10].

Результати роботи можуть бути використані у сфері розробки веб-застосунків для онлайн-ігор, зокрема настільних стратегічних ігор, що мають елементи соціальної взаємодії та персоналізації. Створена frontend частина може стати основою для подальшої розробки повноцінного багатокористувацького ігрового сервісу, інтегрованого з іншими частинами великої системи. Також напрацювання можуть бути адаптовані для створення подібних продуктів у сфері освітніх або розважальних ігрових платформ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Настільна гра “Монополія” з’явилася ще у далекому 1935 році [1] та всього через рік досягла великих успіхів та отримала популярність. Але навіть цього дня, через 90 років, нажаль ми можемо спостерігати дуже малу кількість веб інтерпретацій цієї чудової гри, а ті що існують, в основному мають велику низку проблем. Саме це стало причиною для створення веб-застосунку для гри у “Монополія” онлайн.

Більшість таких рішень ігнорують сучасні принципи UX/UI [4], що робить їх нецікавими або незручними для нової аудиторії, яка звикла до інтуїтивного та інтерактивного дизайну, як у популярних веб-застосунках. Крім того, у багатьох з них відсутня система облікових записів, кастомізації, інвентарю, ринку предметів, що могли б значно покращити ігровий досвід.

Усе це створює незаповнену нішу у галузі веб-адаптацій настільних ігор. Саме тому актуальним є створення frontend інтерфейсу, який не просто відтворює класичну гру, але й адаптує її до цифрового середовища: зручна навігація, інтерактивні елементи, персоналізація, адаптивність до екранів і сучасний дизайн — усе це здатне вивести класичну «Монополію» на новий рівень.

1.2 Виявлення та вирішення проблем

Інтернет ніколи не стоїть на місці, тому майже всі веб-застосунки “старіють” та швидко забуваються користувачами всесвітньої павутини. Таке саме правило діє і на інтерфейс з дизайном, тому більшість сьогоднішніх сайтів з грою у “Монополія” виглядають дуже старо, незручно і незрозуміло для нових користувачів. Це стає причиною швидкої втрати інтересу до подібних сайтів.

Вони не враховують принципи сучасного UX-дизайну: відсутні логічна навігація, інтуїтивне розташування елементів, адаптивність до різних пристроїв і розширень екрана. Це створює значні труднощі при першому знайомстві користувача з грою, і, як наслідок, веде до швидкої втрати інтересу.

Ще однією поширеною проблемою є технічна реалізація: більшість сайтів не оптимізовані під нові браузери, використовують застарілі технології або не забезпечують належну інтерактивність. Інтерфейс часто не реагує на дії користувача, відсутні анімації, повідомлення про помилки або підтвердження дій, що значно знижує якість досвіду. Водночас у багатьох подібних проєктах немає таких сучасних функцій, як особистий інвентар, система друзів, ринок предметів або профіль із можливістю редагування — елементів, які є вже звичними для користувачів цифрових ігор.

1.2.1 Цільова аудиторія

Розроблений веб-застосунок «MonopolyUA» орієнтований на широку цільову аудиторію, що охоплює декілька ключових категорій користувачів, кожна з яких має свої потреби та очікування щодо інтерфейсу і функціональності. Ось основні групи аудиторії, які можуть бути користувачами веб-застосунку [8]:

- молодь та підлітки, які добре знайомі з онлайн іграми, активно користуються інтернетом та очікують від веб-застосунку сучасного вигляду, швидкого реагування, динамічної взаємодії та інтуїтивної навігації;
- казуальні гравці, які цінують класику та зручність. Їм важливо, щоб інтерфейс був зрозумілим, а сам процес – не перевантаженим складними налаштуваннями;
- родини та компанії друзів, які шукають безкоштовного та веселого відпочинку. Для них важлива можливість створення приватної гри, запрошення знайомих та мати внутрішній чат для зручної комунікації;
- фанати настільних ігор, які шукають зручну цифрову адаптацію. Для них ключовим є точність відтворення ігрових механік, наявність всіх правил, прозорість ігрових дій і відповідність класичній «Монополії».

Це означає що гра розрахована на велику кількість різних груп людей та задовольняє потреби кожного.

1.2.2 Монетизація

У межах розробленого веб-застосунку не передбачається жодної комерційної монетизації. Уся взаємодія в маркеті буде здійснюватися виключно за допомогою внутрішньої ігрової валюти, яку користувачі отримують за активність у грі – зокрема перемоги.

1.3 Аналіз аналогів програмного забезпечення

Перед розробкою frontend частини було оглянуто велику кількість конкурентів, виявлено плюси та мінуси кожного:

а) плюси кастомізації в грі «Rento Fortune»;

1) дозволяє обрати зовнішній вигляд ігрової дошки перед початком гри;

2) є декілька візуальних стилів, що підтримують відчуття «оновлення» гри кожного разу;

3) гравець може обирати аватар (пішака), що хоч і обмежено, але додає персоналізації.

б) мінуси кастомізації в грі «Rento Fortune»;

1) всі варіанти обмежені шаблонами – відсутність глибокої кастомізації;

2) неможливо створювати або додавати власні елементи (наприклад скіни або теми);

3) кастомізація доступна лише до старту гри – немає зміни у процесі.

в) плюси роботи з лобі в «Tabletop Simulator» [5];

1) користувач має повний контроль над створенням сесії, включаючи пароль, кількість гравців та налаштування правил;

2) можна клонувати сесії або створювати свої шаблони гри;

3) підтримка приватних і публічних лобі з гнучкими параметрами.

г) мінуси роботи з лобі в «Tabletop Simulator»;

1) складність у використані – новим користувачам важко розібратись із налаштуваннями;

2) відсутність автоматичного балансу гравців або підбору за рівнем;

3) неінтуїтивна навігація в UI при створенні гри.

д) плюси ринку в «Capitalista»;

1) є ідея обміну ресурсами/предметами між гравцями в ігровому середовищі;

2) деякі варіанти продажу відбуваються прямо під час гри, як частина механіки;

3) створено гнучку систему торгівлі між гравцями.

е) мінуси ринку в «Capitalista»;

1) ринок реалізовано умовно — без автономного магазину чи розділу;

2) відсутній інвентар або історія торгів;

3) жодної підтримки фільтрів, пошуку або користувацьких пропозицій.

Тепер розглянемо позитивні та негативні сторони frontend реалізації веб-застосунку «Monopoly Online»:

Розглянемо докладніше реалізацію механізму миттєвого старту гри, оскільки ця частина інтерфейсу визначає перше враження користувача і значною мірою впливає на бажання продовжити гру. Такий підхід до UX-дизайну дозволяє максимально швидко включатися в ігровий процес без зайвих натискань і затримок. Цей функціонал реалізується через:

- простий та інтуїтивний головний екран, де кнопка «Play» запускає гру миттєво;
- відсутність модальних вікон, реєстраційних форм або складних переходів;
- завантаження ігрової дошки без перезавантаження сторінки;
- мінімальний дизайн з акцентом на гру, а не на інтерфейсні елементи.

Було виявлено наступні переваги:

- надзвичайно короткий шлях користувача від відкриття сайту до початку гри;
- інтерфейс не перевантажений візуальними елементами, що робить його легким для сприйняття;
- повна відсутність зайвих кроків у навігації – користувач одразу в дії;
- швидке завантаження ігрової сцени завдяки оптимізації.

Було виявлено наступні недоліки:

- відсутність UI для кастомізації гри, налаштувань або вибору типу сесії;
- не передбачено жодної взаємодії з профілем;
- неможливо повернутись або переглянути попередні ігри, оскільки немає відповідних розділів інтерфейсу;
- слабка структуризація: окремі елементи інтерфейсу розташовані без чіткої логічної ієрархії.

Таким чином, «Monopoly Online» є прикладом frontend, орієнтованого на швидкий запуск, але такий підхід робить застосунок придатним лише для поверхневого використання.

Також розглянемо позитивні та негативні сторони веб інтерфейсу «Monopoly World Edition» на Pogo.com. Можемо спостерігати наступні переваги:

- різноманітність режимів гри: реалізовано кілька правил (Pogo Rules, Official, Speedy, Family, Custom) дозволяючи гравцю одразу обрати стиль гри і миттєво почати без налаштувань;
- соціальна взаємодія та чат: Ігри проводились у «кімнатах» з можливістю приєднання друзів або випадкових гравців, підтримується інтеграція чату — важливий елемент UX;
- проста завантажуваність через Flash/HTML5: гра працювала прямо в браузері: Flash або HTML5 із прогресивним переходом — старт в одне натискання, без завантажень;
- оптимізованість для фонового використання.

Тепер переглянемо недоліки:

- залежність від Flash та закриття серверів: оскільки архітектура базувалася на Flash, зупинка серверів у 2022 році виключила гру з доступу;
- застарілий дизайн та UX: попри функціонал, інтерфейс виглядав відстало: стандартні «кімнати» та статика чату, без динаміки або адаптивності;
- обмежена мобільність: Flash-версії не підтримували мобільні браузери, а переходу на сучасні клієнтські технології було недостатньо;

2 ПОСТАНОВКА ЗАДАЧІ

Головною задачею проєкту є розробка frontend частини для веб-застосунку «MonopolyUA», який повинен включати інтерфейс реєстрації та авторизації користувачів, головну сторінку з інформацією про веб-сайт, правилами гри, відеогайдом для новачків та списком найкращих гравців, профіль користувача з можливістю перегляду статистики, списку друзів та власного інвентарю з різноманітними ігровими предметами, сторінку маркету, на якому кожен користувач може знайти та придбати будь-який «скін» виставлений іншим гравцем. Розробити сторінку для створення лобі з різними параметрами та сторінку грального поля та чату відповідно.

Окрім реалізації базових сторінок, frontend має забезпечити динамічну взаємодію з сервером через API [7], асинхронне завантаження та оновлення даних, безперервну роботу інтерфейсу без перезавантаження сторінки. Усі компоненти повинні бути візуально логічними та зручними, надсилати повідомлення у разі виникнення помилок й успішних дій, підтримувати швидку реакцію на дії користувача. Основна мета полягає у створенні комфортного, сучасного та масштабованого інтерфейсу, що відтворює класичну гру «Монополія» в онлайн-середовищі.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Перед початком розробки frontend частини веб-застосунку «MonopolyUA» були визначені основні функціональні вимоги зі сторони гравця. Після аналізу сценаріїв використання інтерфейсу користувачем було побудовано UML діаграму, яка демонструє ключові взаємодії гравця із frontend частиною проєкту (див. рис. 3.1).

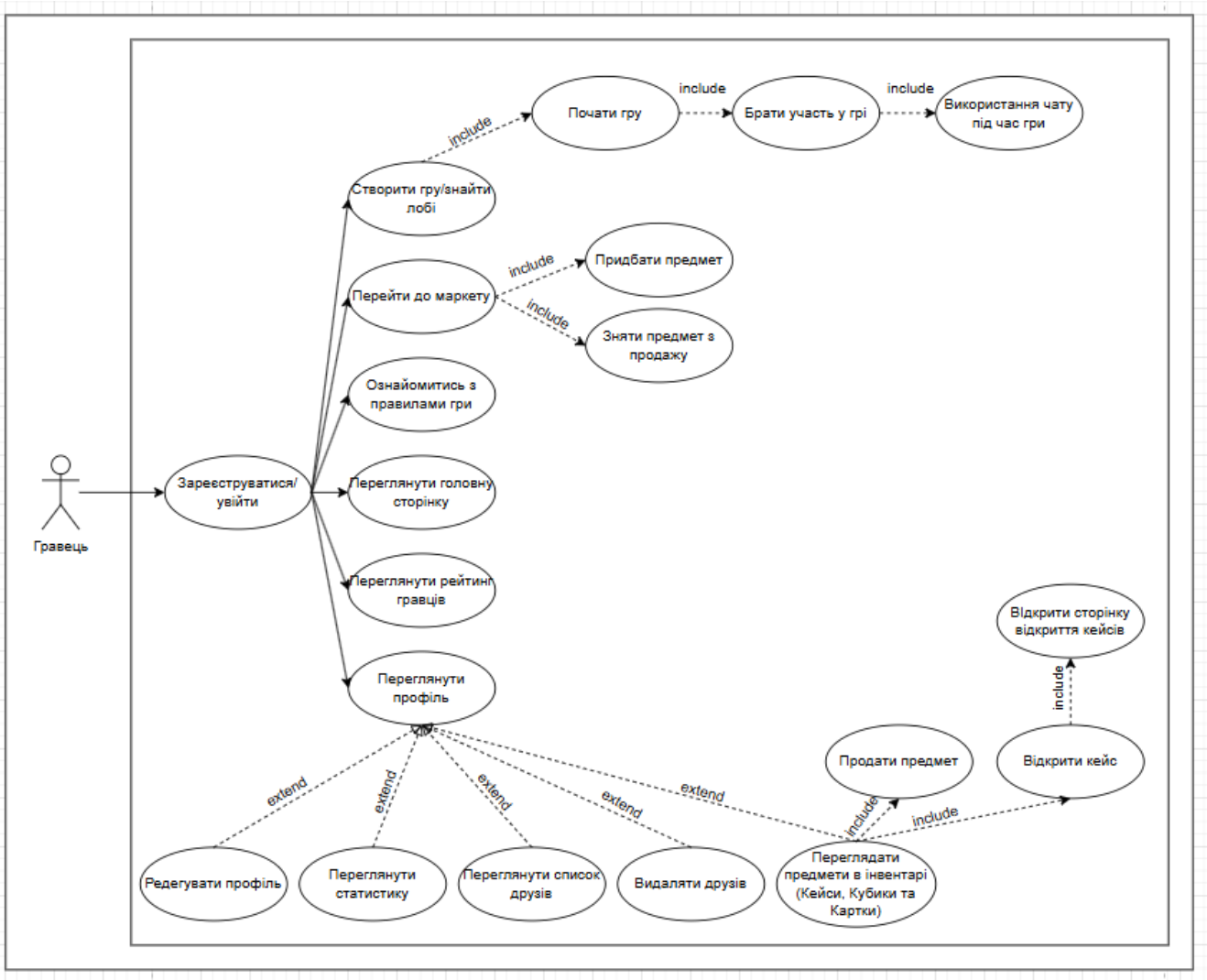


Рисунок 3.1 – UML діаграма взаємодії користувача з frontend частиною (рисунок виконаний самостійно)

Користувачем даного веб-застосунку є гравець. Користувач спочатку повинен створити або увійти в акаунт, після чого отримає доступ до головної сторінки, починаючи з якої має повний доступ до всіх функцій веб-застосунку.

3.2 Проєктування архітектури ПЗ

Метою даного проєкту є створення frontend частини веб-застосунку «MonopolyUA». Основне призначення frontend системи – забезпечити взаємодію користувача з грою через браузерний інтерфейс. Система повинна бути зручною, доступною та сумісною з популярними веб браузерами.

Основні функції інтерфейсу:

- реєстрація та авторизація;
- перегляд головної сторінки, правил гри, рейтингу;
- редагування профілю, інвентар, друзі, відкриття кейсів;
- торгівля предметами на маркеті;
- створення ігор, участь в них, чат.

Зацікавленими сторонами є кінцеві користувачі (гравці) та розробники.

Обрана клієнт-серверна архітектура, що означає що frontend функціонує як окремий клієнт, що працює у браузері та обмінюється даними з сервером за допомогою HTTP-запитів (fetch). Веб-застосунок є монолітним [2] MPA-подібним рішенням [3] на основі окремих HTML-сторінок, пов’язаних між собою.

Далі буде надано опис компонентів frontend частини веб-застосунку «MonopolyUA»:

а) інтерфейс реєстрації та авторизації;

1) відповідає за збирання даних;

2) передає їх на сервер через POST-запити;

3) валідація виконується як у браузері так і на сервері.

б) головна сторінка;

1) містить правила гри, інформацію про проект, відеогайд, список найкращих гравців;

2) завантажує динамічні дані з API (топ гравців).

в) профіль користувача;

1) дозволяє переглядати та редагувати особисті дані;

2) показує статистику, список друзів та інвентар;

3) надає можливість продавати, обирати або відкривати відповідні предмети.

г) маркет;

1) дозволяє переглядати предмети виставлені іншими гравцями, шукати їх за допомогою різних фільтрів;

2) надає можливість купувати предмети по одинці або декілька за один раз;

3) надає можливість контролювати особисті предмети, а саме знімати їх з продажу.

д) сторінка створення та пошуку гри;

1) відображати різні вікна для створення та пошуку лобі;

е) сторінка гри;

1) відображає ігрову дошку;

2) реалізовано модальні вікна для відображення відповідної інформації в залежності від ситуації.

Комунікація між компонентами здійснюється через REST API з використанням функції fetch, для якої використовується протокол HTTP.

Для того щоб показати всі основні компоненти та їх взаємозв’язки, відобразити інтерфейси та залежності між компонентами було спроектовано UML діаграму компонентів (див. рис. 3.2).

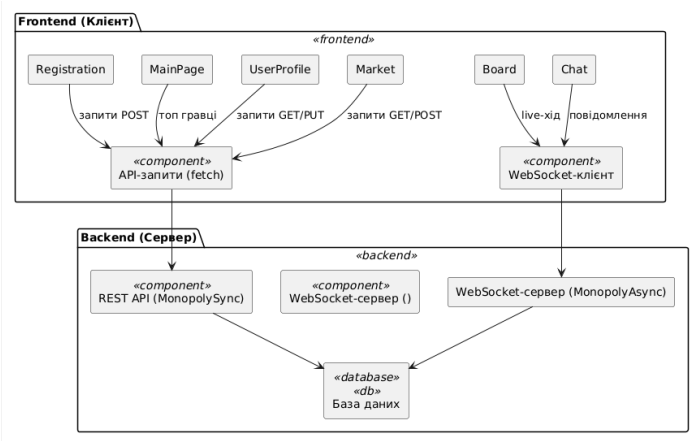


Рисунок 3.2 – UML діаграма компонентів (рисунок виконано самостійно)

Для візуалізації розміщення компонентів на вузлах та їхніх зв’язків та для розуміння того як система буде розгорнута на програмних та/або апаратних платформах було створено UML діаграму розгортування (див. рис. 3.3).

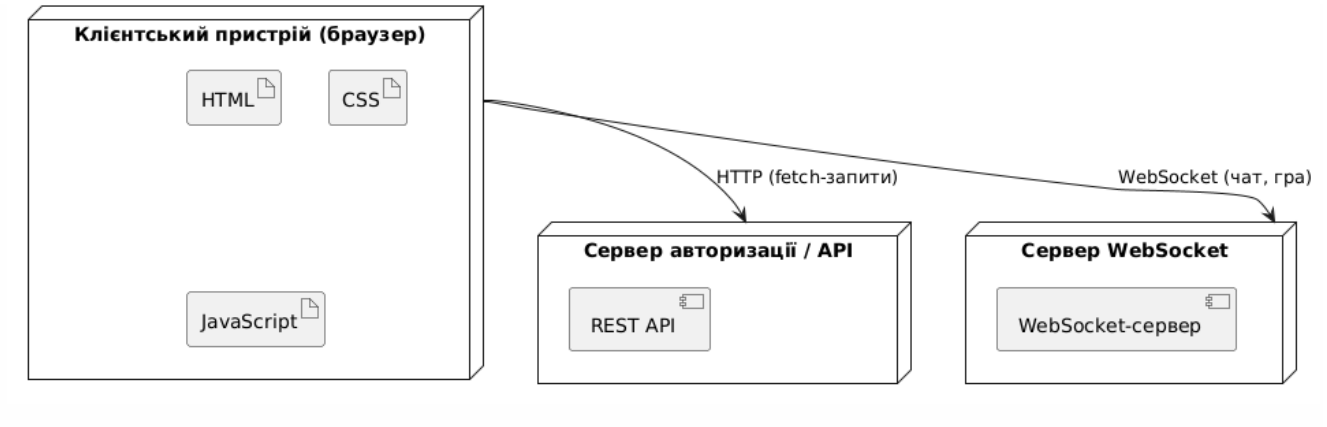


Рисунок 3.3 – UML діаграма розгортання (рисунок виконано самостійно)

Нефункціональні вимоги:

- продуктивність: сторінки повинні завантажуватися за короткий проміжок часу;
- масштабованість: можливість додавання нових сторінок і модулів без переробки всієї структури;
- надійність: перевірка даних на клієнті до надсилання;
- безпека: збереження токенів доступу в localStorage, валідація з боку backend.

Оцінка ризиків полягає у врахуванні можливих технічних проблем, пов’язаних із роботою клієнтської частини. Однією з них є потенційна некоректна робота в застарілих браузерах, що вирішується шляхом використання базових HTML та JavaScript конструкцій, без застосування сучасних нестабільних API. Іншим важливим ризиком є перехоплення даних під час передавання запитів – для цього потрібно впровадити шифрування HTTPS. Також можливі збої під час обміну даними з API, які компенсуються обробкою помилок на рівні fetch та виведенням відповідних повідомлень для користувача.

3.3 Проєктування структури зберігання даних

У рамках реалізації frontend частини веб-застосунку «MonopolyUA» було розглянуто структуру зберігання даних, необхідну для повноцінної взаємодії з сервером. Серверна частина проєкту реалізує реляційну базу даних, з якою frontend працює через API-запити. Клієнтська частина не зберігає дані постійно, але взаємодіє з основними таблицями, що визначають логіку гри, маркету та профілю користувачів.

3.4 Приклади використання алгоритмів та методів

У межах реалізації frontend частини веб-застосунку «MonopolyUA» було застосовано низку прикладних алгоритмів та методів, необхідних для забезпечення логіки інтерфейсу та інтерактивних елементів сайту. Приклади наведено нижче:

- метод перевірки валідності форми: для форм реєстрації та авторизації реалізовано перевірку полів (чи не порожні, чи правильний формат email). Перевірка відбувається в реальному часі перед відправкою запиту;
- алгоритм фільтрації масиву товарів: функція updateItems() виконує фільтрацію товарів за кількома критеріями: назва, ціна, рідкість, кількість, тип. Це реалізується за допомогою послідовного застосування логічних умов усередині методу. Такий підхід дозволяє динамічно відображати лише ті товари, які відповідають параметрам що ввів користувач;
- алгоритм збереження та відновлення стану полів профілю: при завантаженні профілю з API зберігаються оригінальні значення імені та регіону у відповідних змінних. У разі скасування редагування ці значення відновлюються, дозволяючи повернутись до попереднього стану без повторного запиту.

3.5 Проєктування UI/UX

Проєктування інтерфейсу користувача та досвіду користувача є важливим етапом створення frontend частини веб-застосунку «MonopolyUA», оскільки в

залежності від того, наскільки зручним та привабливим буде інтерфейс залежить залучення та утримання гравців.

У рамках UX-дизайну було реалізовано зручну та логічну структуру навігації між усіма сторінками веб-застосунку (за виключенням тих сторінок, що потребують додаткових дій для їх відкриття, таких як сторінка відкриття кейсів та сторінка дошки гри). Всі основні розділи згруповані у верхній панелі навігації, що постійно доступна користувачу незалежно від того, на якій сторінці він перебуває. Такий підхід дозволяє швидко та інтуїтивно зрозуміти, де саме знаходиться необхідна функція. Для спрощення взаємодії були також використані іконки, які супроводжують текстові назви пунктів меню, підвищуючи візуальне сприйняття.

Користувач має змогу швидко переміщатися між всіма сторінками, при цьому кожен перехід супроводжується швидким відображенням, що робить взаємодію більш комфортною для всіх користувачів. Всі маршрути між розділами продумані таким чином, щоб не створювати відчуття "лабіринту": з будь-якого розділу можна повернутися назад або перейти в інший з мінімальною кількістю кліків.

Більшість дій, які користувач має змогу виконувати, реалізовані так, щоб уникати зайвих кроків або непотрібних переходів між сторінками. Наприклад, деякі дії реалізовано через модальні вікна, які дозволяють завершити операцію, не залишаючи поточної сторінки. Це суттєво зменшує час виконання типових сценаріїв і покращує загальне враження від взаємодії із застосунком.

UI-частина була розроблена із застосуванням єдиного стилістичного підходу. Візуальні елементи, такі як кнопки, поля вводу, таблиці та всі інші елементи оформлені з використанням CSS-стилів, які забезпечують єдність всього наповнення. Колірна палітра базується на поєднанні відтінків сірого, задні фони виконані у більш темних тонах, а елементи що знаходяться попереду – у більш світлих відповідно. Основні кольори мають такі hex-коди: #333, #444, #555, #1c1c1c, #2c2c2c. Весь текст на сторінках поділено на два кольори – білий та помаранчевий. Білий використовується для текстів та наповнення, а помаранчевий для заголовків та важливих елементів.

Для підвищення інтерактивності було використано hover-ефекти, модальні вікна та прості анімації, які дозволяють взаємодіяти з елементами без переходів.

Завдяки мінімальному дизайну, інтерфейс є легким для розуміння навіть новим користувачам. Це дозволяє зосередитися на ігровому процесі, не відволікаючись на пошук функцій чи складу навігацію.

Для проєктування інтерфейсу був використаний сервіс Figma, що дало змогу створити інтерактивні прототипи майбутніх сторінок. На етапі макетування були ретельно опрацьовані логіка взаємодії, послідовність дій користувача та взаємне розташування елементів, щоб забезпечити максимальну зручність під час гри. Кожна сторінка містить лише необхідні функції та не перевантажена другорядним вмістом, що дозволяє гравцеві швидко орієнтуватися в інтерфейсі. Усі кнопки, повідомлення та вікна мають зрозуміле візуальне оформлення та інтуїтивні підказки, що зменшує потребу в поясненнях. Особливу увагу було приділено зворотному зв’язку — після натискання кнопок або виконання дій користувач отримує візуальні підтвердження, що сприяє відчуттю контролю над процесом.

3.6 Опис підготовки даних

У рамках frontend частини веб-застосунку «MonopolyUA» обробка та підготовка даних відбувається перед їх надсиланням на сервер або після отримання відповіді. Основним форматом є JSON, що забезпечує легку інтеграцію з API.

У випадках коли користувач заповнює форму, наприклад реєстрація, авторизація або редагування профілю, введені дані перевіряються у браузері на відповідність формату. В результаті дані перетворюються у структуру JavaScript-об’єкта і серелізуються за допомогою JSON.stringify().

У разі завантаження даних у форматі наприклад аватарки, використовується структура FormData, яка дозволяє надсилати бінарні дані.

При отримані даних з серверної частини, наприклад список даних, спочатку проводиться розбір відповіді через response.json(), а потім перевірка структури на наявність необхідних полів та чи є відповідність формату.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Реалізація ринку предметів

Для реалізації маркету у frontend частині веб-застосунку «MonopolyUA» було обрано підхід динамічного оновлення DOM [6] без перезавантаження сторінки. Алгоритм фільтрації реалізовано з використанням методу Array.filter() на стороні клієнта, що надає можливість швидко реагувати на зміни параметрів пошуку (назва, рідкість, ціна, кількість). Вся інформація про товари, які знаходяться на маркеті завантажується з API. Для купівлі передбачено два сценарії, перший це купівля одиночних товарів, другий це придбання декілька предметів за один раз вказуючи кількість та ціну за один предмет. Кожна дія на цій сторінці супроводжується перевіркою авторизації та обробкою помилок.

Програмний код фільтрації товарів (Array.filter):

```
const filteredItems = items.filter(item => {
const matchesName = item.name.toLowerCase().includes(nameFilter);
const matchesPrice = item.price >= priceMin && item.price <= priceMax;
const matchesRarity = rarityFilter === 'all' || item.rarity === 
rarityFilter;
const matchesAmount = (item.amount ?? Infinity) >= amountFilter;
const matchesType = !activeType || item.category === activeType || 
item.type === activeType;
return matchesName && matchesPrice && matchesRarity && matchesAmount 
&& matchesType;
});
```

Програмний код оновлення DOM без перезавантаження:

```
itemDisplayMid.innerHTML = '';
filteredItems.forEach(item => {
const itemElement = createItemElement(item);
itemDisplayMid.appendChild(itemElement);
});
```

Програмний код завантаження товарів з API:

```
async function loadItems() {
try {
const response = await fetch('http://localhost:8000/market/items/', {
headers: {
```

```
'Content-Type': 'application/json',
'Authorization': `Bearer ${localStorage.getItem('authToken')}`,
}
});
if (!response.ok) {
throw new Error(`HTTP error! status: ${response.status}`);
}
items = await response.json();
updateItems();
} catch (error) {
console.error('Failed to load items:', error);
itemDisplayMid.innerHTML = '<p>Помилка завантаження товарів</p>';
}
}
```

Програмний код для купівлі товарів:

```
modal.querySelector('.modal-content-bot').addEventListener('click',
async
(event) => {
if (!event.target.classList.contains('buy-listing-btn')) return;
const listingId = event.target.dataset.listingId;
if (!listingId) return;
try {
const response = await
fetch(`http://localhost:8000/market/item/buy/${listingId}`, {
method: 'POST',
headers: {
'Content-Type': 'application/json',
'Authorization': `Bearer ${localStorage.getItem('authToken')}`,
},
});
if (!response.ok) {
const errorData = await response.json();
throw new Error(errorData.error || 'Purchase failed');
}
const result = await response.json();
alert(result.message);
modal.classList.add('hidden');
} catch (error) {
console.error('Error buying item:', error);
alert('Помилка при купівлі товару: ' + error.message);
}
});
```

4.2 Редагування профілю користувача

Інтерфейс редагування профілю було побудовано за принципом перемикання між режимами перегляду й редагування. Логіка полягає у тому, що при завантажені сторінки користувач бачить поля для введення даних але вони вже заповнені його

даними. Для того щоб почати редагувати користувач повинен натиснути відповідну кнопку та після цього ці поля стануть активними, а також з’являться нові. У разі якщо користувач вирішить не завершувати редагування та просто відмінить його, дані не буде змінено і вони залишаться такими ж якими були до початку редагування. Збереження реалізовано через PATCH-запит з перевіркою токена.

Програмний код перемикання між режимами перегляду та редагування:

```
editBtn.addEventListener("click", () => {
if (!editing) {
editing = true;
nameInput.disabled = false;
regionInput.disabled = false;
removeDisabledStyles();
passInput.style.opacity = "1";
repeatPassInput.style.opacity = "1";
passInput.style.pointerEvents = "auto";
repeatPassInput.style.pointerEvents = "auto";
cancelBtn.style.opacity = "1";
cancelBtn.style.pointerEvents = "auto";
editBtn.textContent = "Зберегти";
} else {
if (passInput.value !== repeatPassInput.value) {
alert("Паролі не співпадають!");
return;
}
fetch("http://localhost:8000/profile/update/", {
method: "PATCH",
headers: {
"Content-Type": "application/json",
"Authorization": `Bearer ${token}`
},
body: JSON.stringify({
username: nameInput.value,
region: regionInput.value,
password: passInput.value || undefined
})
})
.then(response => {
if (!response.ok) throw new Error("Заповніть всі необхідні поля");
return response.json();
})
.then(data => {
nameInput.disabled = true;
regionInput.disabled = true;
applyDisabledStyles();
passInput.style.opacity = "0";
repeatPassInput.style.opacity = "0";
passInput.style.pointerEvents = "none";
repeatPassInput.style.pointerEvents = "none";
cancelBtn.style.opacity = "0";
cancelBtn.style.pointerEvents = "none";
```

```
editBtn.textContent = "Редагувати";
editing = false;
originalName = data.username;
originalRegion = data.region || "";
alert("Профіль оновлено");
})
.catch(error => {
alert(error.message);
});
}
});
```

Програмний код скасування редагування та відкат даних:

```
cancelBtn.addEventListener("click", () => {
nameInput.value = originalName;
regionInput.value = originalRegion;
nameInput.disabled = true;
regionInput.disabled = true;
applyDisabledStyles();
passInput.value = "";
repeatPassInput.value = "";
passInput.style.opacity = "0";
repeatPassInput.style.opacity = "0";
passInput.style.pointerEvents = "none";
repeatPassInput.style.pointerEvents = "none";
cancelBtn.style.opacity = "0";
cancelBtn.style.pointerEvents = "none";
editBtn.textContent = "Редагувати";
editing = false;
});
```

Програмний код завантаження поточних даних профілю при відкритті сторінки:

```
fetch("http://localhost:8000/profile/info/", {
method: "GET",
headers: {
"Content-Type": "application/json",
"Authorization": `Bearer ${token}`
}
})
.then(response => response.json())
.then(data => {
avatar.src = data.avatar;
nameInput.value = data.username;
if (data.region) {
const optionExists = Array.from(regionInput.options).some(
option => option.value === data.region
);
if (optionExists) {
```

```
regionInput.value = data.region;
} else {
regionInput.value = "";
}
} else {
regionInput.value = "";
}
originalName = data.username;
originalRegion = regionInput.value;
nameInput.disabled = true;
regionInput.disabled = true;
applyDisabledStyles();
})
```

4.3 Інтерактивна робота з інвентарем

Інвентар у frontend частині веб-застосунку «MonopolyUA» реалізовано через асинхронне завантаження даних за категоріями. Контейнер інвентарю поділено на 3 різні частини, які перемикаються натисканням відповідних кнопок. Завантаження даних про відповідні предмети відбувається тільки при обиранні відповідної категорії, що не навантажує сервер постійними запитами. Кожен елемент, тобто предмет має в собі контекстне меню з діями: «Продати», «Обрати» та кейси мають унікальну кнопку «Відкрити». В залежності від вибору дії відбуваються різні процеси, наприклад при спробі продати предмет користувач відкриє модальне вікно у якому потрібно буде ввести необхідні дані для продажу, а саме ціну за кількість. У разі натискання кнопки «Відкрити» у контекстному меню кейсів, користувача буде направлено на сторінку відкриття кейсів.

Програмний код асинхронного завантаження даних при перемиканні:

```
async function renderItems(type) {
inventorySection.innerHTML = '';
try {
const response = await fetch(`http://localhost:8000/market/user-
inventory/?category=${type}`, {
headers: {
'Authorization': 'Bearer ' + localStorage.getItem('authToken')
}
});
if (!response.ok) throw new Error('Не вдалося завантажити інвентар.');
const items = await response.json();
let openMenu = null;
items.forEach(itemData => {
const item = document.createElement('div');
```

```
item.className = 'item';
item.dataset.id = itemData.item.id;
item.dataset.type = itemData.item.category;
```

Програмний код модального вікна продажу:

```
async function openSellModal(itemId) {
modal.classList.remove('hidden');
try {
const response = await
fetch(`http://localhost:8000/market/item/${itemId}`,
{
headers: {
'Authorization': 'Bearer ' + localStorage.getItem('authToken')
}
});
const data = await response.json();
modalTitle.textContent = `Продати ${data.name}`;
} catch (err) {
console.error(err);
modalTitle.textContent = `Продати товар`;
}
}
```

4.4 Відображення статистики та друзів

Статистика гравця відображається у вигляді таблиці з ключовими метриками: кількість ігор, перемог, поразок, відсоток перемог та кількість рейтингових очків. Дані отримуються за допомогою API-запиту та оновлюються при кожному вході на сторінку. Список друзів було реалізовано з пагінацією. Видалення друга виконується з підтвердженням та оновленням інтерфейсу без перезавантаження.

Програмний код відображення статистики користувача:

```
fetch("http://localhost:8000/profile/stat/", {
method: "GET",
headers: {
"Content-Type": "application/json",
"Authorization": "Bearer " + localStorage.getItem("authToken"),
}
})
.then(response => {
if (!response.ok) {
throw new Error("Не вдалося отримати статистику");
}
return response.json();
```

```
})
.then(data => {
document.getElementById("stat-games").textContent = data.games;
document.getElementById("stat-wins").textContent = data.wins;
document.getElementById("stat-loses").textContent = data.lose;
document.getElementById("stat-percentage").textContent =
data.percentage + "%";
document.getElementById("stat-points").textContent = data.points;
})
.catch(error => {
console.error("Помилка при завантаженні статистики:", error);
});
```

Програмний код для отримання списку друзів з API:

```
fetch('http://localhost:8000/friends/user-friends/', {
method: 'GET',
headers: {
'Content-Type': 'application/json',
'Authorization': `Bearer ${token}`
}
})
.then(response => {
if (!response.ok) {
throw new Error('Network response was not ok ' + response.statusText);
}
return response.json();
})
.then(data => {
console.log("Friends:", data);
data.forEach(friend => {
friends.push(friend);
});
renderFriends(currentPage);
console.log("Friends array:", friends);
})
```

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Усі ключові функціональні компоненти, зазначені у технічному завданні, були повністю реалізовані в рамках frontend частини веб-застосунку «MonopolyUA». Інтерфейс містить всі заявлені сторінки: головну сторінку з інформаційним блоком, правилами гри, відеогайдом та списком найкращих гравців, сторінку профілю користувача з можливістю редагування даних, перегляду статистики та списку друзів, видалення друзів, перегляд предметів в інвентарі та виконання дій з ними, наприклад: продати, обрати, відкрити. Маркет з підтримкою купівлі предметів різними способами та контролюванням власних продажів. Сторінку створення лобі та гральну дошку з чатом. Комунікація з backend частиною здійснюється за допомогою fetch-запитів, що забезпечує відповідність функціоналу первинним очікуванням.

Інтерфейс було розроблено без використання сторонніх фреймворків, що дало змогу досягти повного контролю над логікою поведінки елементів. Завдяки використанню чистого JavaScript, реалізовано асинхронну обробку подій та динамічне оновлення DOM. Усі інтерактивні елементи відповідають сучасним вимогам до UX – користувач може виконати більшість дій у межах декількох натискань. Також підтверджено стабільність інтерфейсу під час навантажень, зокрема при рендері великої кількості товарів або при зміні станів елементів у режими реального часу.

Відмова від фреймворків на користь чистого JavaScript дозволила гнучко адаптувати рішенні під конкретні потреби без надмірного навантаження на сторінку. Вибір окремого завантаження даних для кожної категорії інвентарю й маркету зменшив кількість одночасних запитів, що позитивно вплинуло на продуктивність. Також завдяки мінімізації DOM-операцій вдалося знизити час рендерингу навіть у динамічних секціях, таких як контекстні меню або списки користувачів.

Серед обмежень можна відзначити відсутність адаптації до дуже малих екранів, що ускладнює використання інтерфейсу на мобільних пристроях. Також

реалізація на чистому JavaScript потребує ретельного контролю стану вручну, що ускладнює масштабування функціональності. Ці недоліки можуть бути вирішені на наступних етапах оптимізації.

Крім основних функцій, було реалізовано низку додаткових можливостей, які підвищують зручність взаємодії з інтерфейсом. Наприклад, у маркеті впроваджено фільтри за категоріями, станом та ціною предметів, що дозволяє користувачам швидше знаходити потрібні об'єкти.

Особливу увагу було приділено сценаріям взаємодії в межах грального процесу. Наприклад, дошка гри підтримує оновлення в реальному часі завдяки періодичним запитам та внутрішньому механізму відстеження змін. Це дозволяє користувачам бачити дії інших гравців, зміну позицій фішок та повідомлення чату без потреби оновлювати сторінку.

З метою підвищення підтримуваності коду та уникнення дублювання логіки, структура frontend була розділена на модулі — окремі скрипти відповідають за завантаження даних, відображення, події користувача та валідацію. Це спрощує майбутню підтримку, внесення змін або розширення функціональності. Крім того, код був частково задокументований коментарями для полегшення навігації розробників, що будуть працювати з проєктом у подальшому.

У результаті реалізації було створено повноцінний, зручний та інтуїтивно зрозумілий інтерфейс, що відповідає всім функціональним вимогам. Frontend частина «MonopolyUA» успішно демонструє стабільну роботу, логічну структуру та готовність до подальшого масштабування, що свідчить про ефективність обраних рішень та досягнення поставлених цілей.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Визначення плану впровадження

Після завершення етапів проєктування, реалізації та тестування frontend частини веб-застосунку «MonopolyUA», було визначено план впровадження розробленого програмного забезпечення. Frontend є частиною, яка є незалежною від серверної реалізації і може бути розгорнута на будь-якому сучасному веб сервері або інший платформі для хостингу. Основні етапи зображено в таблиці 6.1.

Таблиця 6.1 – План впровадження frontend частини веб-застосунку «MonopolyUA»

Назва етапу	Дата початку	Дата завершення
Повне тестування роботи інтерфейсу перед деплоєм, включно з доступністю API	11.06.2025	11.06.2025
Підготовка фінальної збірки клієнтських файлів (HTML, CSS, JavaScript)	11.06.2025	11.06.2025
Деплой frontend частини на Docker	11.06.2025	11.06.2025
Тестування роботи інтерфейсу після деплою	11.06.2025	11.06.2025

ВИСНОВКИ

У ході виконання комплексного курсового проєкту було реалізовано frontend частину для веб-застосунку «MonopolyUA». Основною метою було створення функціонального, зручного та інтуїтивно зрозумілого інтерфейсу користувача з використанням стандартних веб технологій – HTML, CSS та JavaScript, без застосування сторонніх фреймворків.

У процесі роботи було:

- розроблено повноцінний набір інтерфейсних сторінок, зокрема: авторизації та входу, профілю, маркету, відкриття кейсів, головну, пошуку та створення гри та дошки для гри відповідно;
- реалізовано механізми взаємодії з серверною частиною через REST API;
- впроваджено алгоритми фільтрації, валідації, обробки асинхронних запитів, динамічного оновлення вмісту та взаємодії з DOM;
- забезпечено єдиний стиль UI з урахуванням UX-принципів для зручності навігації та взаємодії з елементами інтерфейсу.

Проведено ручне тестування ключових функціональних модулів, за результатами якого було виявлено низку помилок, пов’язаних з відображенням елементів на різних сторінках. Усі виявлені недоліки були своєчасно усунені шляхом уточнення стилів, покращення логіки рендерингу та доопрацювання перевірок введених даних. Це дозволило забезпечити коректну роботу інтерфейсу на різних етапах використання веб-застосунку.

У підсумку, розроблена frontend частина повністю відповідає поставленим вимогам. Вона є стабільною у роботі, ефективною з погляду взаємодії з сервером, зручною для користувачів і готовою до подальшої інтеграції з backend логікою та іншими компонентами системи. Досягнуто високого рівня відповідності між технічним завданням, дизайном та реалізацією. Веб-застосунок «MonopolyUA» у результаті став платформою, здатною забезпечити повноцінний ігровий досвід у класичну настільну гру в онлайн-форматі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Monopoly Game Invented [Електронний ресурс] – URL: https://keepincalendar.com/March-7/Monopoly-Game Invented/66#:~:text=March%207%2C%201933%20in%20Atlantic,the%20famous%20board%20game%20Monopoly (дата звернення: 07.06.2025)

2. What is monolithic architecture? [Електронний ресурс] – URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith#:~:text=A%20monolithic%20architecture%20is%20a,monolith%20architecture%20for%20software%20design (дата звернення: 07.06.2025)

3. MPA [Електронний ресурс] – URL: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58 (дата звернення: 07.06.2025)

4. UI дизайн [Електронний ресурс] – URL: https://turumburum.ua/blog/klyuchovi-ui-principi-dlya-stvorennya-yakisnogo-dizaynu-vashogo-veb-produktu (дата звернення: 07.06.2025)

5. «TableTop Simulator» [Електронний ресурс] – URL: https://ru.wikipedia.org/wiki/Tabletop_Simulator (дата звернення: 07.06.2025)

6. «Introduction to the DOM» [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (дата звернення: 07.06.2025)

7. What is an API? [Електронний ресурс] – URL: https://www.ibm.com/think/topics/api (дата звернення: 07.06.2025)

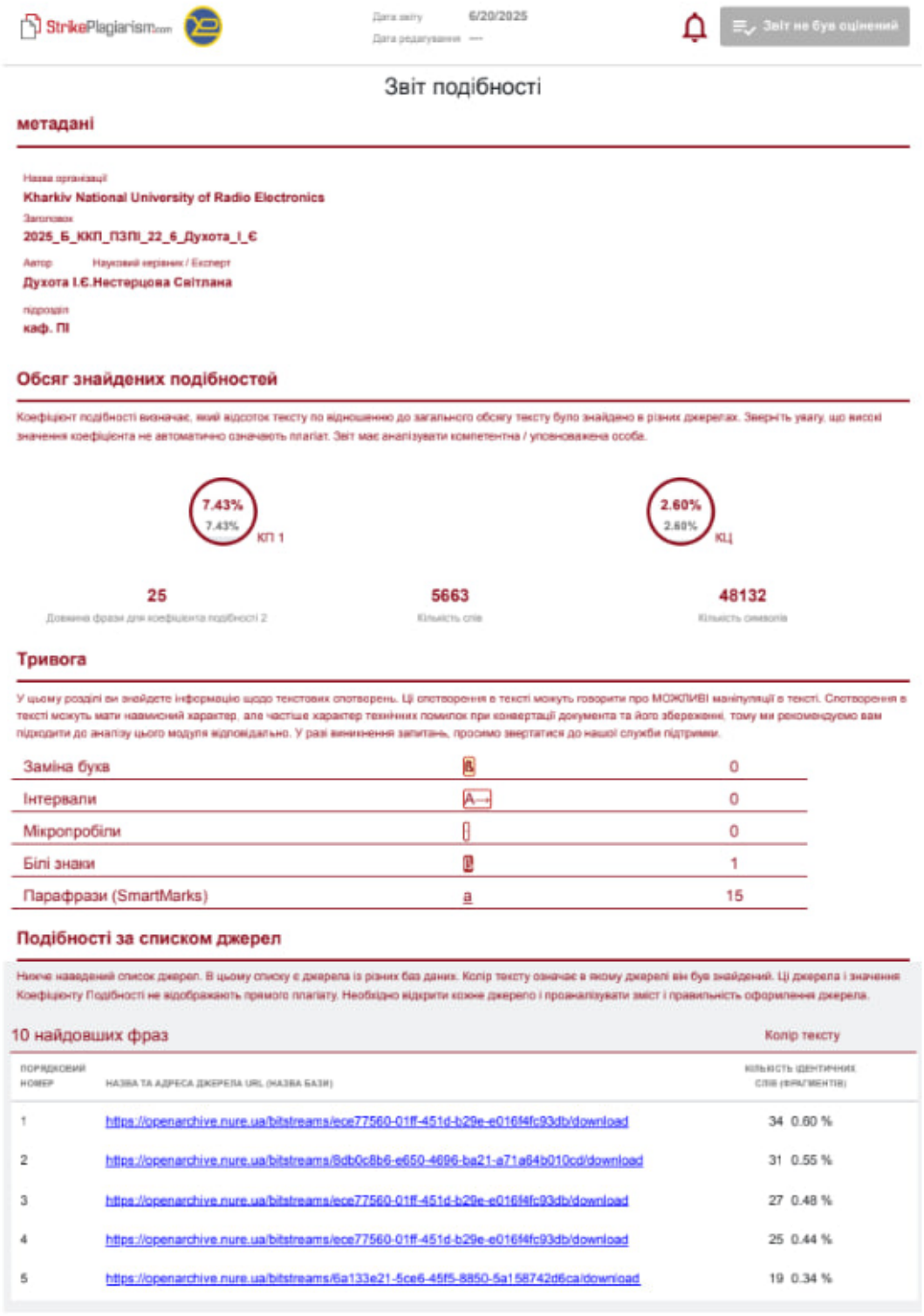
8. Як визначити ЦА за три кроки [Електронний ресурс] – URL: https://remonline.app/uk/blog/target-audience/ (дата звернення: 07.06.2025)

9. Visual Studio Code [Електронний ресурс] – URL: https://code.visualstudio.com/ (дата зверення: 07.06.2025)

10. CSS and JavaScript accessibility best practices [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Accessibility/CSS_and_JavaScript (дата звернення: 07.06.2025)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базу ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/20/2025
Дата редагування —

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ККП_ПЗПІ_22_6_Духота_І_Є
Автор Науковий керівник / Експерт
Духота І.Є.Нестерцова Світлана
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

7.43% 7.43% КП 1

2.60% 2.60% КЦ

25	5663	48132
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції з текстом. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

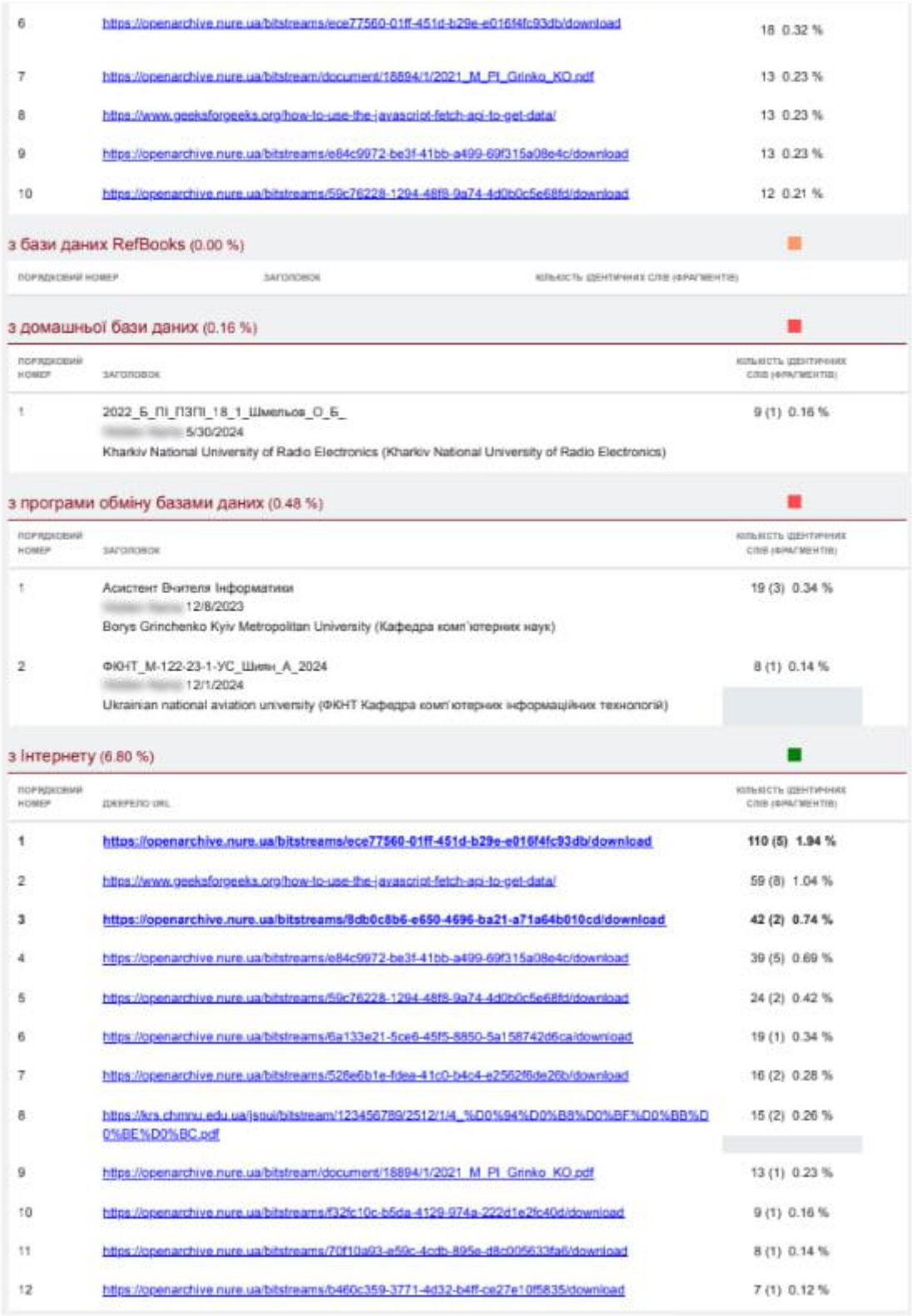
Заміна букв	а	0
Інтервали	A→	0
Мікропробіли		0
Білі знаки		1
Парафрази (SmartMarks)	a	15

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	34 0.60 %
2	https://openarchive.nure.ua/bitstreams/8db0c8b6-e650-4696-ba21-a71a64b010cd/download	31 0.55 %
3	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	27 0.48 %
4	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	25 0.44 %
5	https://openarchive.nure.ua/bitstreams/6a133e21-5ce6-45f5-8850-5a158742d6ca/download	19 0.34 %

Рисунок А.1 – Перша сторінка короткого звіту перевірки на плагіат



6	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	18 0.32 %
7	https://openarchive.nure.ua/bitstream/document/18894/1/2021_M_PI_Grinko_KO.pdf	13 0.23 %
8	https://www.geeksforgeeks.org/how-to-use-the-javascript-fetch-api-to-get-data/	13 0.23 %
9	https://openarchive.nure.ua/bitstreams/e84c9972-be3f-41bb-a499-69f315a08e4c/download	13 0.23 %
10	https://openarchive.nure.ua/bitstreams/59c76228-1294-48f8-9a74-4d0b0c5e68fd/download	12 0.21 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.16 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2022_Б_ПІ_ПЗПІ_18_1_Шмельов_О_Б_ [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	9 (1) 0.16 %

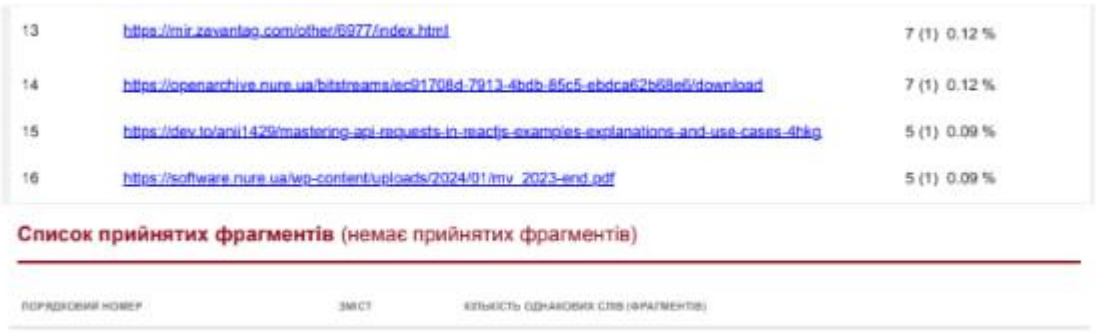
з програми обміну базами даних (0.48 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Асистент Вчителя Інформатики [redacted] 12/8/2023 Borys Grinchenko Kyiv Metropolitan University (Кафедра комп`ютерних наук)	19 (3) 0.34 %
2	ФІКНТ_М-122-23-1-УС_Шиян_А_2024 [redacted] 12/1/2024 Ukrainian national aviation university (ФІКНТ Кафедра комп'ютерних інформаційних технологій)	8 (1) 0.14 %

з Інтернету (6.80 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	110 (5) 1.94 %
2	https://www.geeksforgeeks.org/how-to-use-the-javascript-fetch-api-to-get-data/	59 (8) 1.04 %
3	https://openarchive.nure.ua/bitstreams/8db0c8b6-e650-4696-ba21-a71a64b010cd/download	42 (2) 0.74 %
4	https://openarchive.nure.ua/bitstreams/e84c9972-be3f-41bb-a499-69f315a08e4c/download	39 (5) 0.69 %
5	https://openarchive.nure.ua/bitstreams/59c76228-1294-48f8-9a74-4d0b0c5e68fd/download	24 (2) 0.42 %
6	https://openarchive.nure.ua/bitstreams/6a133e21-5ce6-45f5-8850-5a158742d6ca/download	19 (1) 0.34 %
7	https://openarchive.nure.ua/bitstreams/526e6b1e-fdea-41c0-b4c4-e2562f6de26b/download	16 (2) 0.28 %
8	https://krs.chmnu.edu.ua/jspui/bitstream/123456789/2512/1/4_%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC.pdf	15 (2) 0.26 %
9	https://openarchive.nure.ua/bitstream/document/18894/1/2021_M_PI_Grinko_KO.pdf	13 (1) 0.23 %
10	https://openarchive.nure.ua/bitstreams/f32fc10c-b5da-4129-974a-222d1e2fc40d/download	9 (1) 0.16 %
11	https://openarchive.nure.ua/bitstreams/70f10a93-a59c-4cdb-895e-d8c005633fa8/download	8 (1) 0.14 %
12	https://openarchive.nure.ua/bitstreams/b460c359-3771-4d32-b4ff-ce27e10f5835/download	7 (1) 0.12 %

Рисунок А.2 – Друга сторінка короткого звіту перевірки на плігат



13	https://mir.zavantag.com/other/6977/index.html	7 (1) 0.12 %
14	https://openarchive.nure.ua/bitstreams/ec91708d-7913-4bdb-85c5-ebdca62b68e6/download	7 (1) 0.12 %
15	https://dev.to/anil1429/mastering-api-requests-in-reactjs-examples-explanations-and-use-cases-4hkg	5 (1) 0.09 %
16	https://software.nure.ua/wp-content/uploads/2024/01/mv_2023-end.pdf	5 (1) 0.09 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

Рисунок А.3 – Третя сторінка короткого звіту перевірки на плагіат

ДОДАТОК Б

Слайди презентації

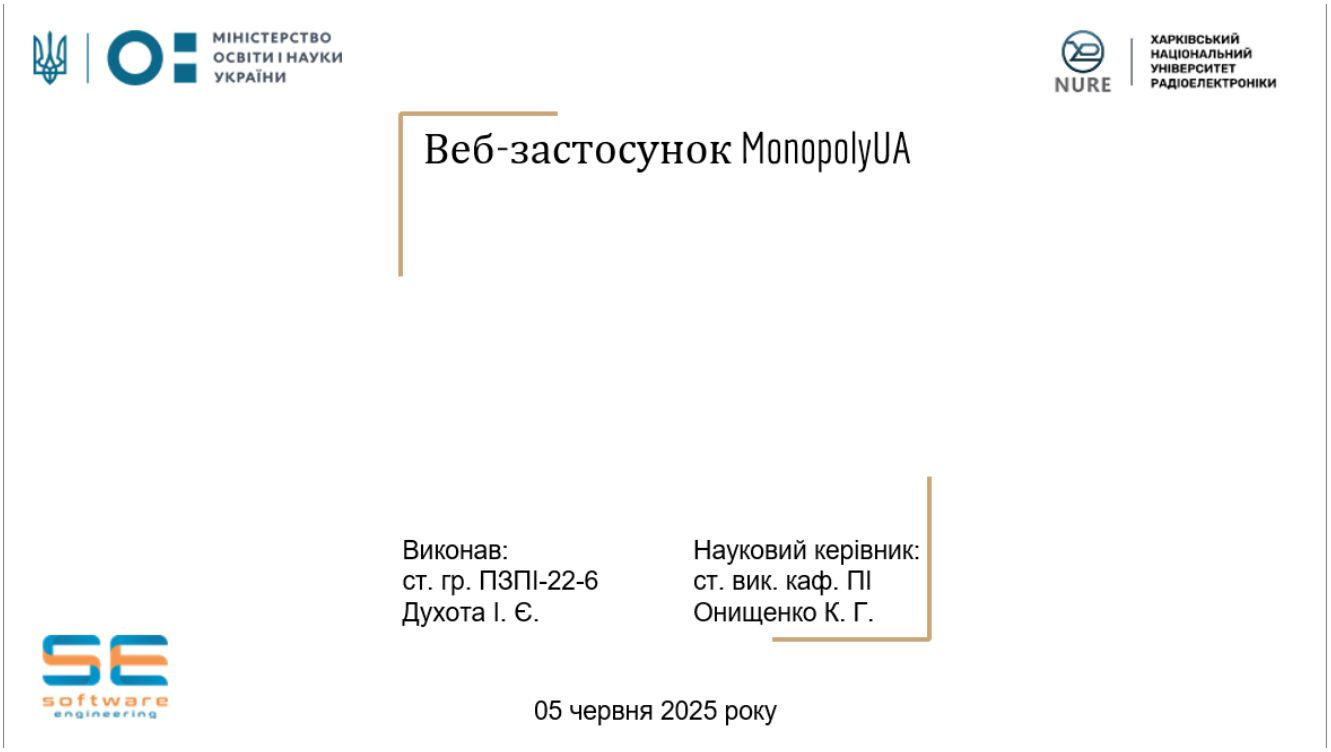


Рисунок Б.1 – Перший слайд презентації

Рисунок Б.2 – Мета роботи (чітке визначення мети роботи)

Мета роботи

- Актуальність роботи:

Настільна гра “Монополія” з’явилася ще у далекому 1935 році та всього через рік досягла великих успіхів та отримала популярність. Але навіть цього дня, через 90 років, нажаль ми можемо спостерігати дуже малу кількість веб інтерпретацій цієї чудової гри, а ті що існують, в основному мають велику низку проблем. Більшість таких рішень ігнорують сучасні принципи UX/UI, що робить їх нецікавими або незручними для нової аудиторії, яка звикла до інтуїтивного та інтерактивного дизайну, як у популярних веб-застосунках. Усе це створює незаповнену нішу у галузі веб-адаптацій настільних ігор. Саме тому актуальним є створення frontend інтерфейсу, який не просто відтворює класичну гру, але й адаптує її до цифрового середовища: зручна навігація, інтерактивні елементи, персоналізація, адаптивність до екранів і сучасний дизайн — усе це здатне вивести класичну «Монополію» на новий рівень.

software engineering

3

Рисунок Б.3 – Мета роботи (актуальність роботи)

Аналіз проблеми (аналіз існуючих рішень)

- Перелік досліджуваних конкурентів:

На ринку існує велика кількість конкурентів, але у всіх є достатньо недоліків та обмеженостей у питанні функціональних можливостей:

- Pogo Monopoly (Hasbro/Pogo Games): вебверсія класичної монополії з простим інтерфейсом, обмеженим функціоналом і застарілим дизайном.
- Rento Fortune (RentoMonopoly.com): Браузерна гра з базовою графікою, підтримує гру з друзями, але з незручним UI та мінімальною кастомізацією.
- Monopoly Online (CrazyGames): Простий клієнт без реєстрації, але не підтримує профілі, ринок або взаємодію користувачів.
- Tabletop Simulator (Steam): Потужний емулятор настільних ігор із можливістю грати в «Монополію», але вимагає встановлення та має складний інтерфейс.
- Capitalista (альтернативна версія гри онлайн): Має непогану графіку, але обмежений функціонал: відсутній повноцінний профіль, ринок та геймфіксація.

software engineering

4

Рисунок Б.4 – Аналіз проблеми (аналіз існуючих рішень)

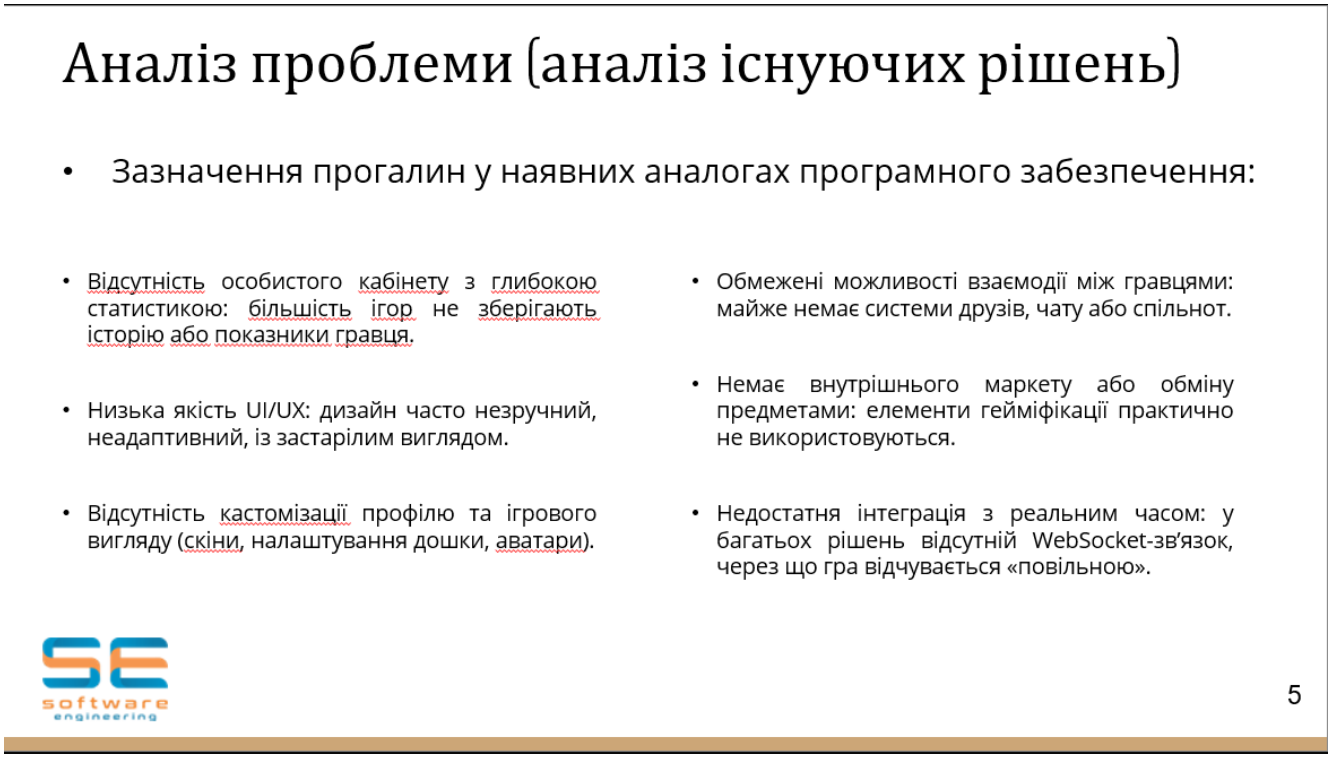


Рисунок Б.5 – Аналіз проблеми (аналіз існуючих рішень)

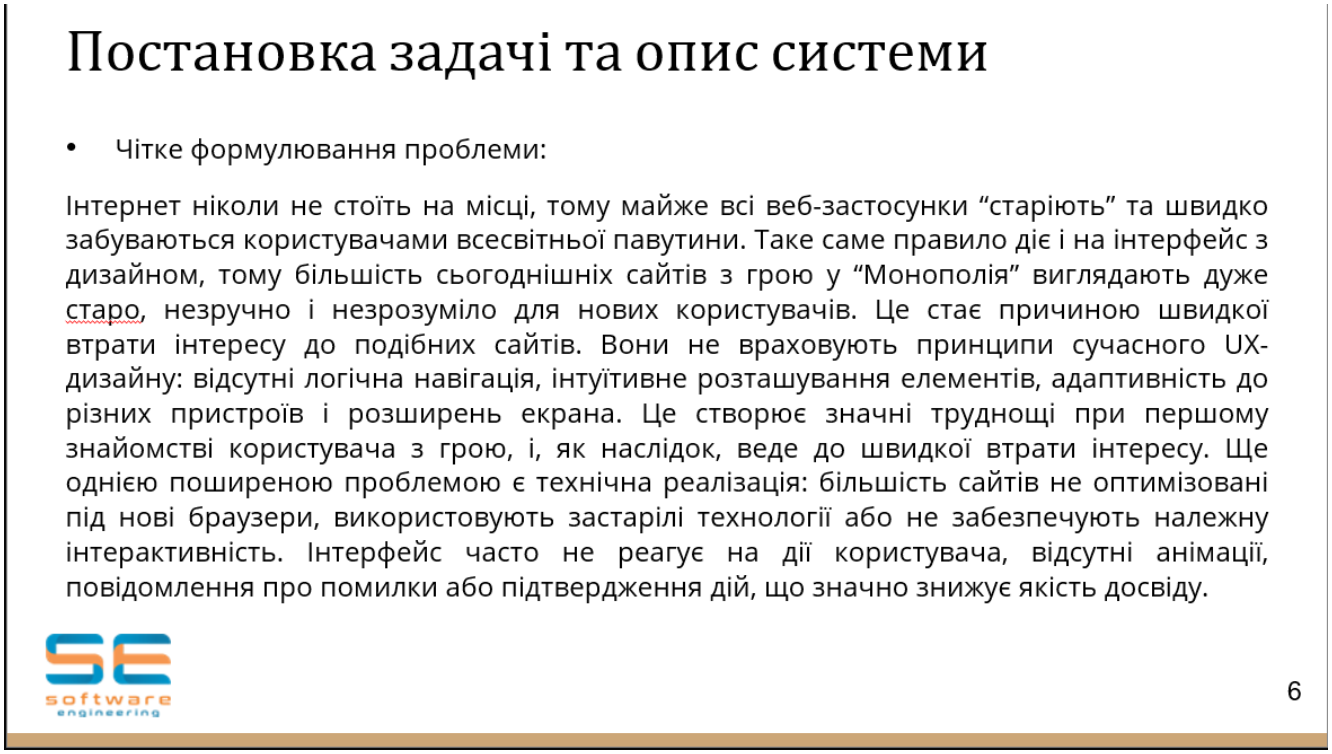


Рисунок Б.6 – Постановка задачі та опис системи (чітке формулювання проблеми)

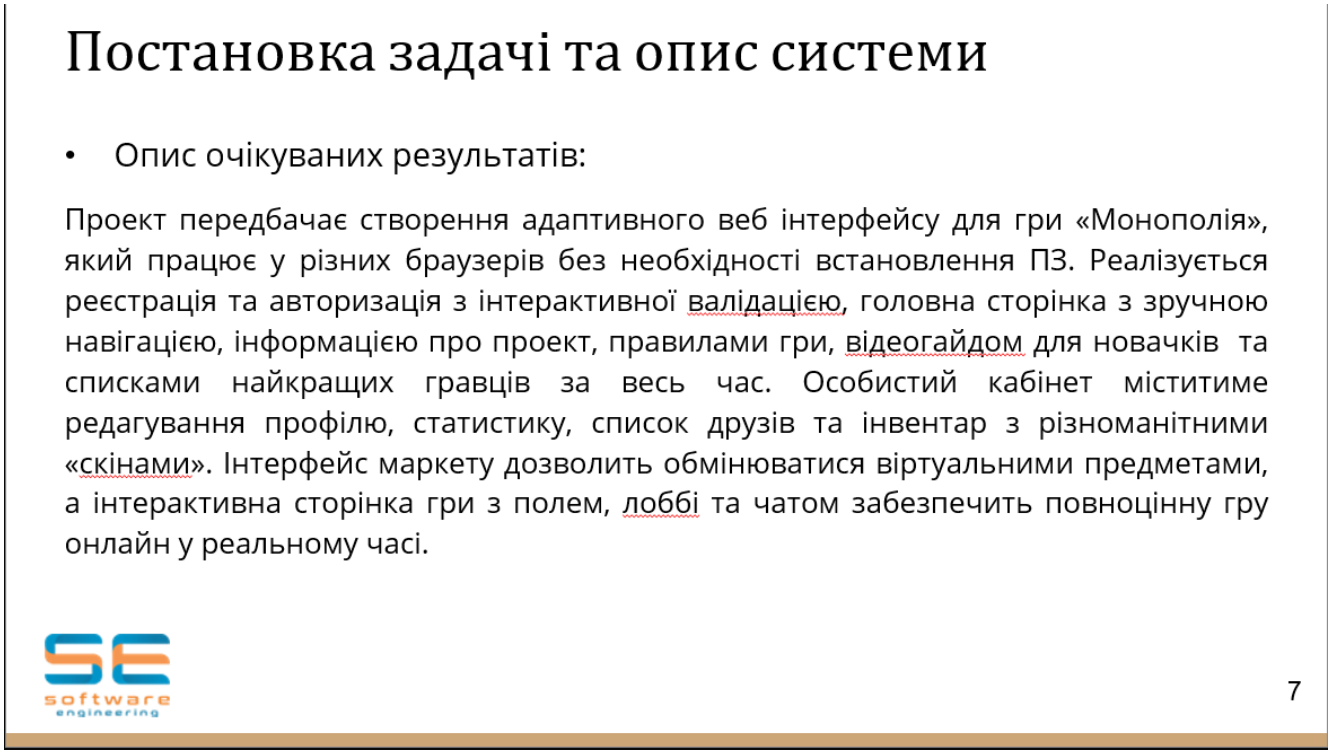


Рисунок Б.7 – Постановка задачі та опис системи (опис очікуваних результатів)

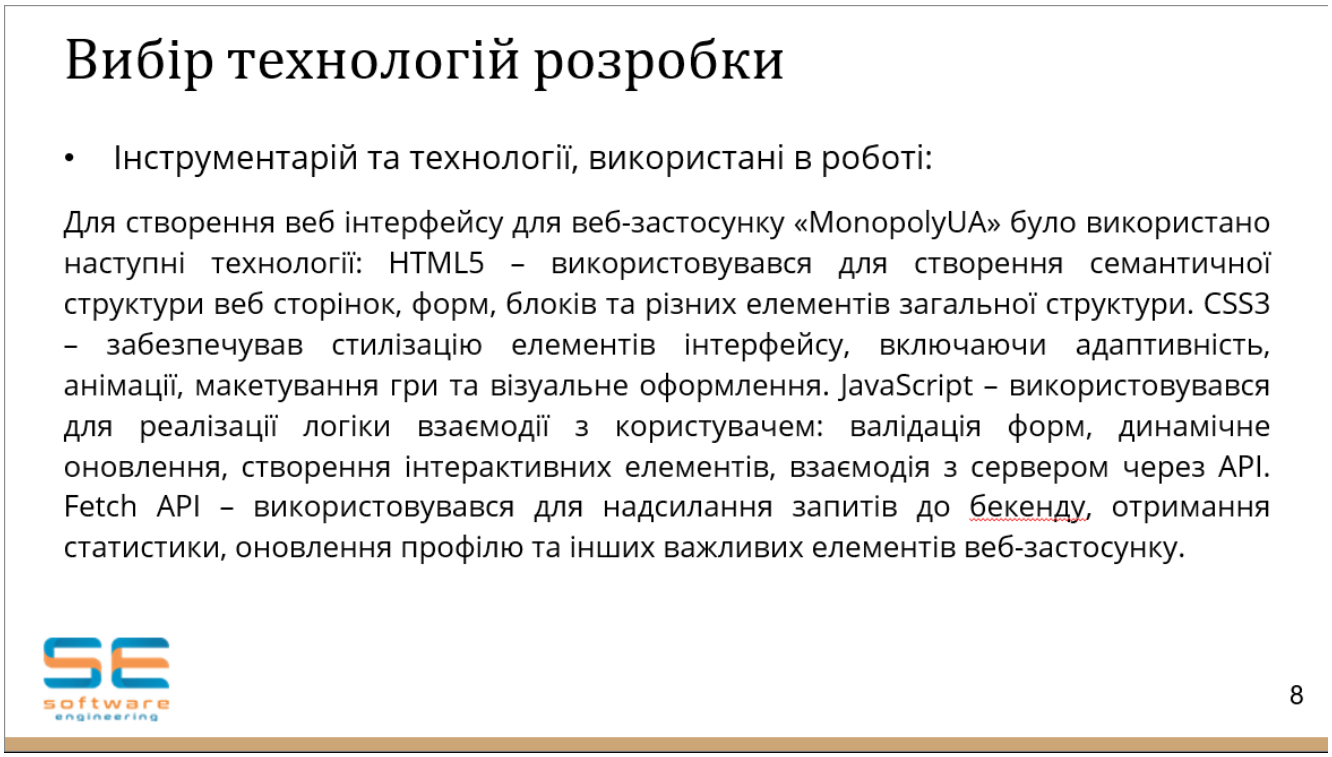


Рисунок Б.8 – Вибір технологій розробки (інструментарій та технології, використані в роботі)

Рисунок Б.9 – Архітектура створеного програмного забезпечення (схема архітектури розробленої системи)

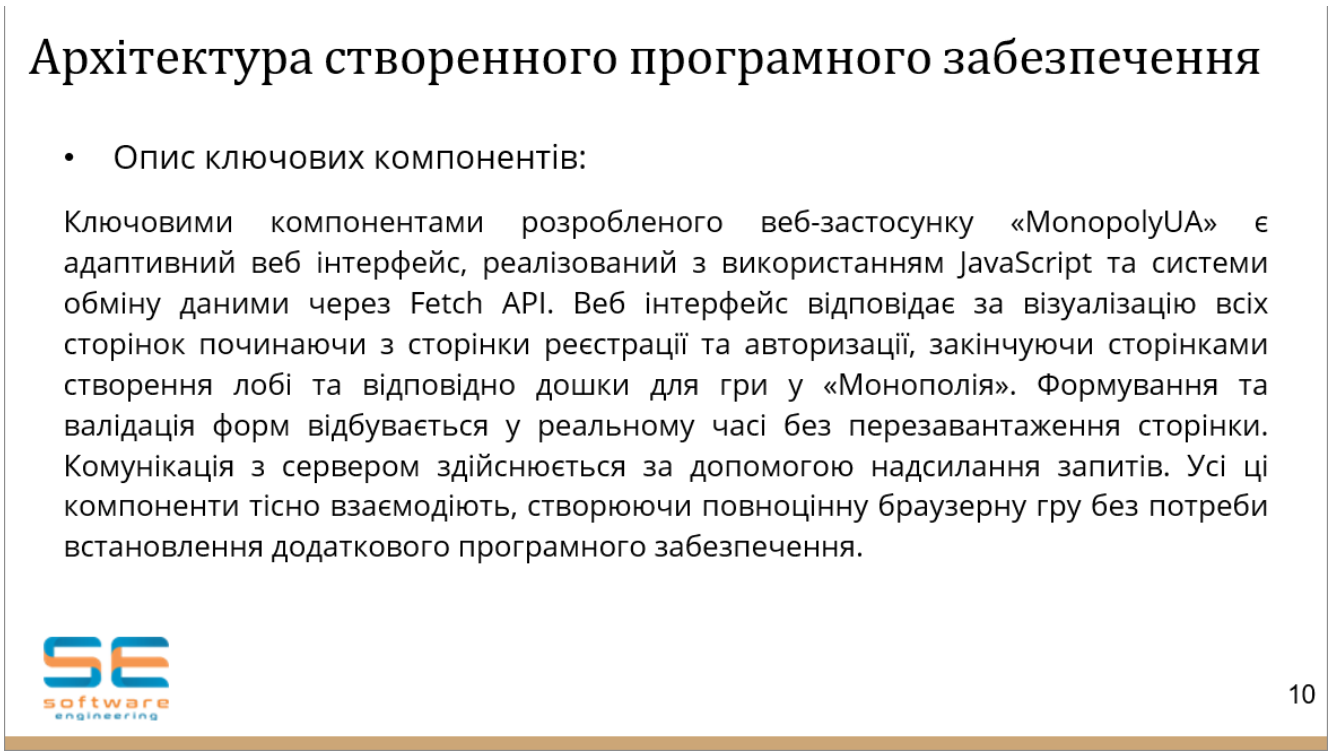


Рисунок Б.10 – Архітектура створеного програмного забезпечення (опис ключових компонентів)

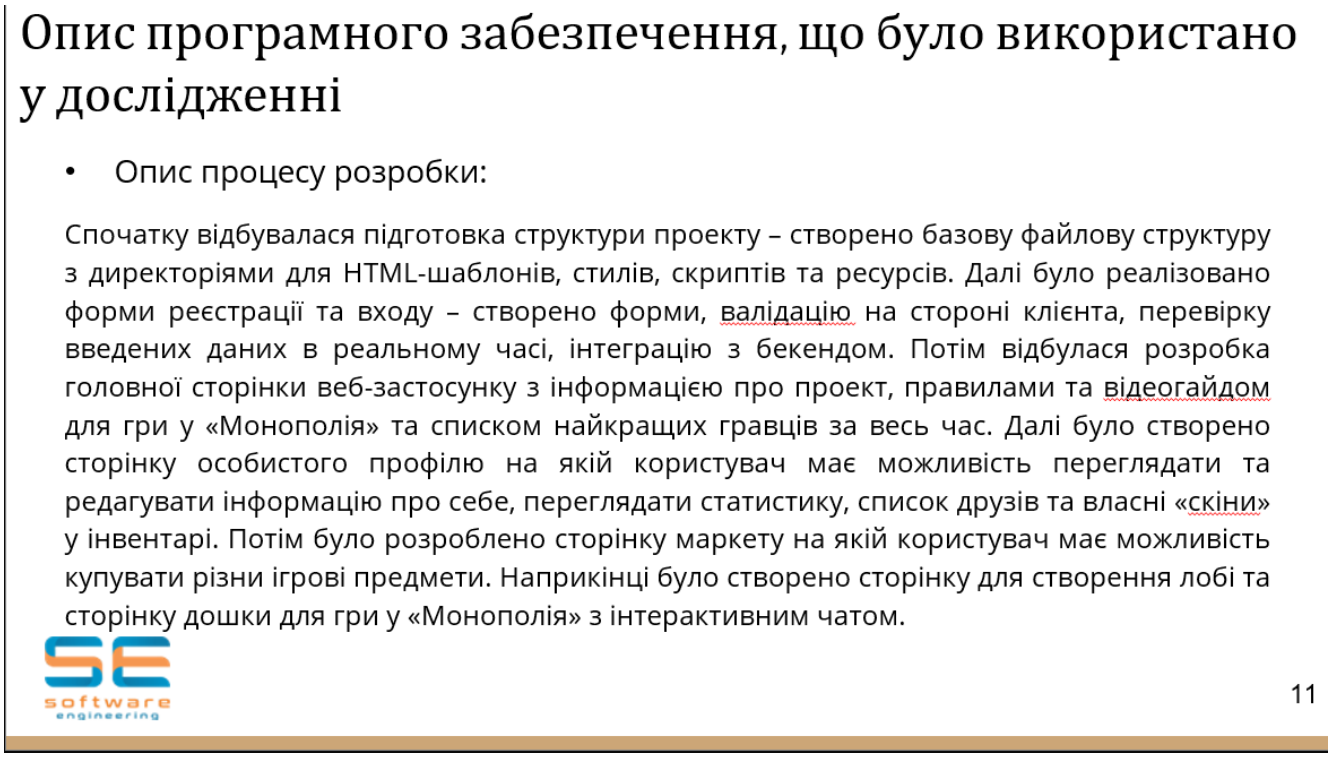


Рисунок Б.11 – Опис програмного забезпечення, що було використано у дослідженні (опис процесу розробки)

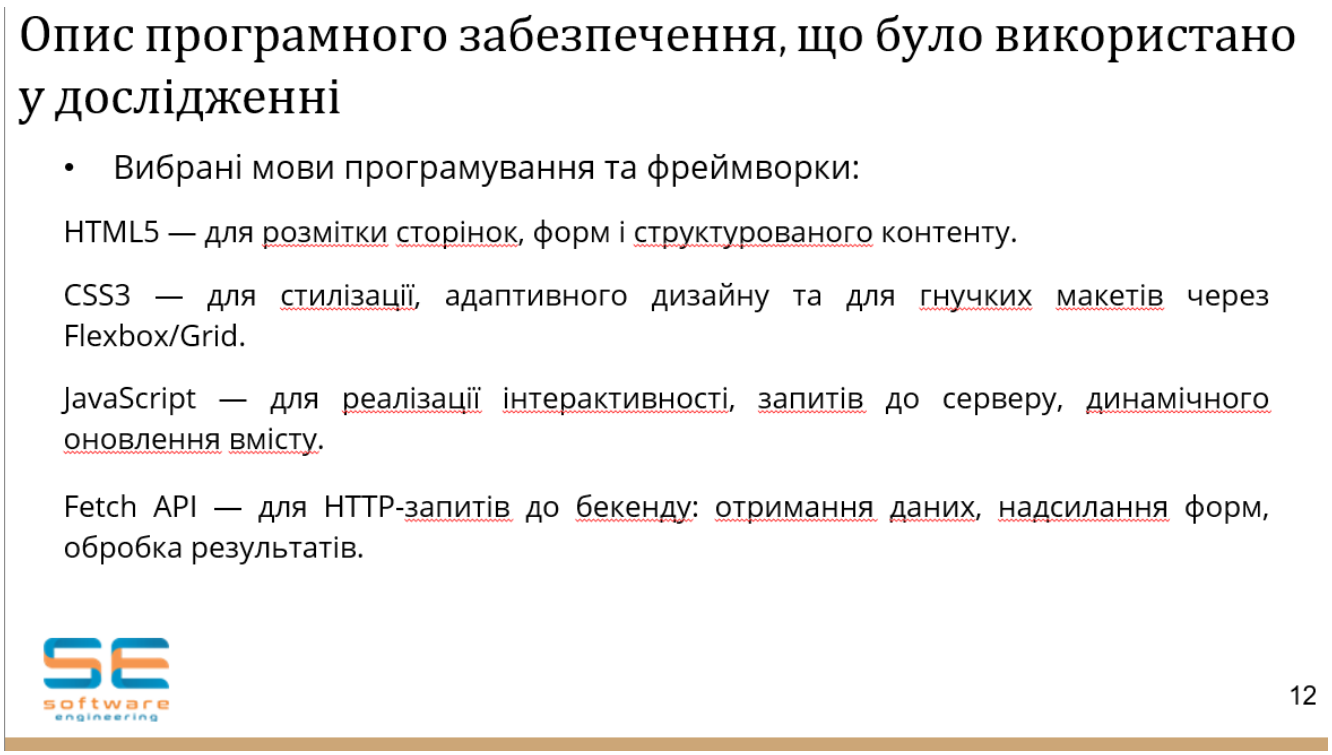


Рисунок Б.12 – Опис програмного забезпечення, що було використано у дослідженні (вибрані мови програмування та фреймворки)

Дизайн системи

- Методи, послідовність та технології:

Для розробки веб застосунку «MonopolyUA» було використано метод прототипування, за допомогою якого було створено макети сторінок та окремих елементів структури.

Послідовність: спочатку було проведено аналіз функціональних вимог до застосунку (сторінки, елементи, необхідні блоки), потім було створено прототипи ключових сторінок та елементів. Далі було затверджено основну кольорову гамму для веб-застосунку. Наприкінці було додано динаміку через JavaScript та інтегровано веб частину з бекендом за допомогою Fetch API.

Для виконання всіх необхідних дій використовувався стандартний стек технологій, а саме HTML та CSS для основної верстки інтерфейсу, JavaScript та Fetch API для взаємодії з серверною частиною та веб-сайти ColorHunt та Flaticon для визначення з кольоровою схемою та завантаження необхідних «іконок» для веб застосунку відповідно.

software engineering

13

Рисунок Б.13 – Дизайн системи (методи, послідовність та технології)

Приклад реалізації

- Цікаві фрагменти коду (анімація відкриття «кейсів»):

```
function startAnimation() {
  title.textContent = items.name + " x " + case_info.quantity;
  const moveTime = 50;
  const duration = 10000;
  const itemWidth = dropLine.firstElementChild.offsetWidth + 20;
  button.disabled = true;
  dropLine.style.transition = `transform ${moveTime}ms ease-in-out`;

  const shuffleInsert = (element) => {
    const children = Array.from(dropLine.children);
    const randomIndex = Math.floor(Math.random() * children.length);
    dropLine.insertBefore(element, dropLine.children[randomIndex]);
  };

  let animationFrameId;
  let stopped = false;

  const animate = () => {
    if (stopped) return;

    dropLine.style.transform = `translateX(-${itemWidth}px)`;
```

```
  setTimeout(() => {
    if (stopped) return;

    dropLine.style.transition = 'none';
    dropLine.style.transform = 'translateX(0)';

    const first = dropLine.firstElementChild;
    shuffleInsert(first);

    void dropLine.offsetWidth;
    dropLine.style.transition = `transform ${moveTime}ms ease-in-out`;

    animationFrameId = requestAnimationFrame(animate);
  }, moveTime);
};

animate();

setTimeout(() => {
  stopped = true;
  cancelAnimationFrame(animationFrameId);
  isAnimating = false;
  button.disabled = false;

  dropLine.style.transition = 'none';
  dropLine.style.transform = 'none';

  dropLine.innerHTML = '';
  const resultItem = createItemElement(selectedItemFromServer);
  resultItem.classList.add('highlighted-item');
  dropLine.appendChild(resultItem);

  button.textContent = "Забрати";
}, duration);
}
```

software engineering

14

Рисунок Б.14 – Приклад реалізації (цікаві фрагменти коду)

Приклад реалізації

- Цікаві фрагменти коду (анімація відтворення відеогайду для новачків та слайдер з інформацією про проект):

```
// ======= Відео-блок: відкриття/закриття =======

const watchBtn = document.querySelector('.watch-video-btn');
const closeBtn = document.querySelector('.close-video-btn');
const containerVideo = document.querySelector('.container-video');

// Обробник відкриття відео
watchBtn.addEventListener('click', () => {
  containerVideo.classList.add('active');

  setTimeout(() => {
    containerVideo.classList.add('show-video');
  }, 1000);
});

// Обробник закриття відео
closeBtn.addEventListener('click', () => {
  containerVideo.classList.remove('show-video');

  setTimeout(() => {
    containerVideo.classList.remove('active');
  }, 300);
});
```

```
// ======= Автоматичний слайдер з керуванням вручну =======

let currentSlide = 0;
const slides = document.querySelector('.slides');
const totalSlides = 6;
let slideInterval = startSlideInterval();

// Функція для запуску інтервалу автозміни слайдів
function startSlideInterval() {
  return setInterval(() => {
    moveSlide(1);
  }, 10000);
}

// Функція для зміщення слайду вперед або назад
function moveSlide(step) {
  currentSlide = (currentSlide + step + totalSlides) % totalSlides;
  updateSlide();
}

// Функція для оновлення відображення слайду
function updateSlide() {
  slides.style.transform = `translateX(-${currentSlide * 100}%)`;
}

// Функція для переходу до наступного слайду вручну
function nextSlide() {
  clearInterval(slideInterval);
  moveSlide(1);
  slideInterval = startSlideInterval();
}

// Функція для переходу до попереднього слайду вручну
function prevSlide() {
  clearInterval(slideInterval);
  moveSlide(-1);
  slideInterval = startSlideInterval();
}

// Обробники подій для кнопок перемикання слайдів
document.querySelector('.next').addEventListener('click', nextSlide);
document.querySelector('.prev').addEventListener('click', prevSlide);
```

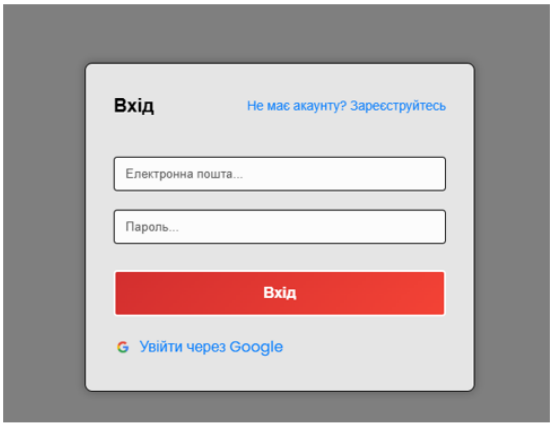
software engineering

15

Рисунок Б.15 – Приклад реалізації (цікаві фрагменти коду)

Інтерфейс користувача

Скріншоти вікна реєстрації та входу:



software engineering

16

Рисунок Б.16 – Скріншот вікна реєстрації та входу

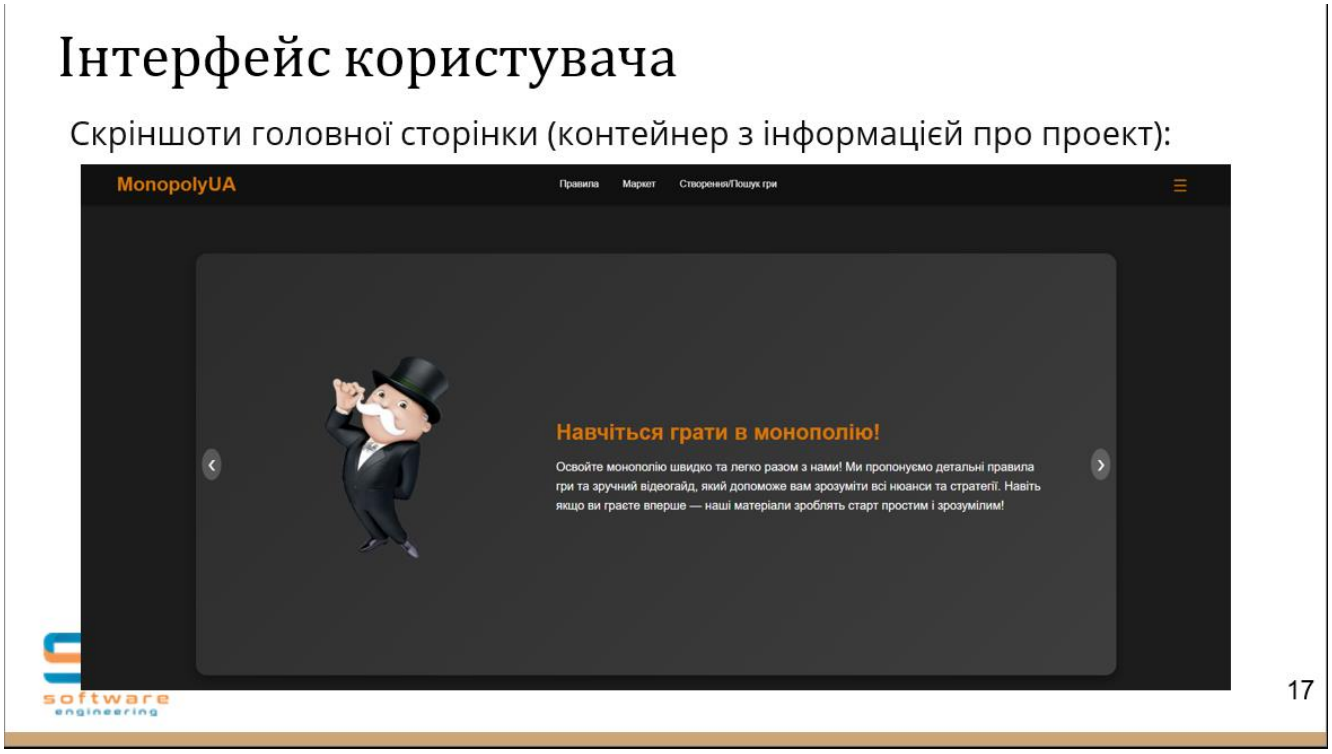


Рисунок Б.17 – Скріншот головної сторінки

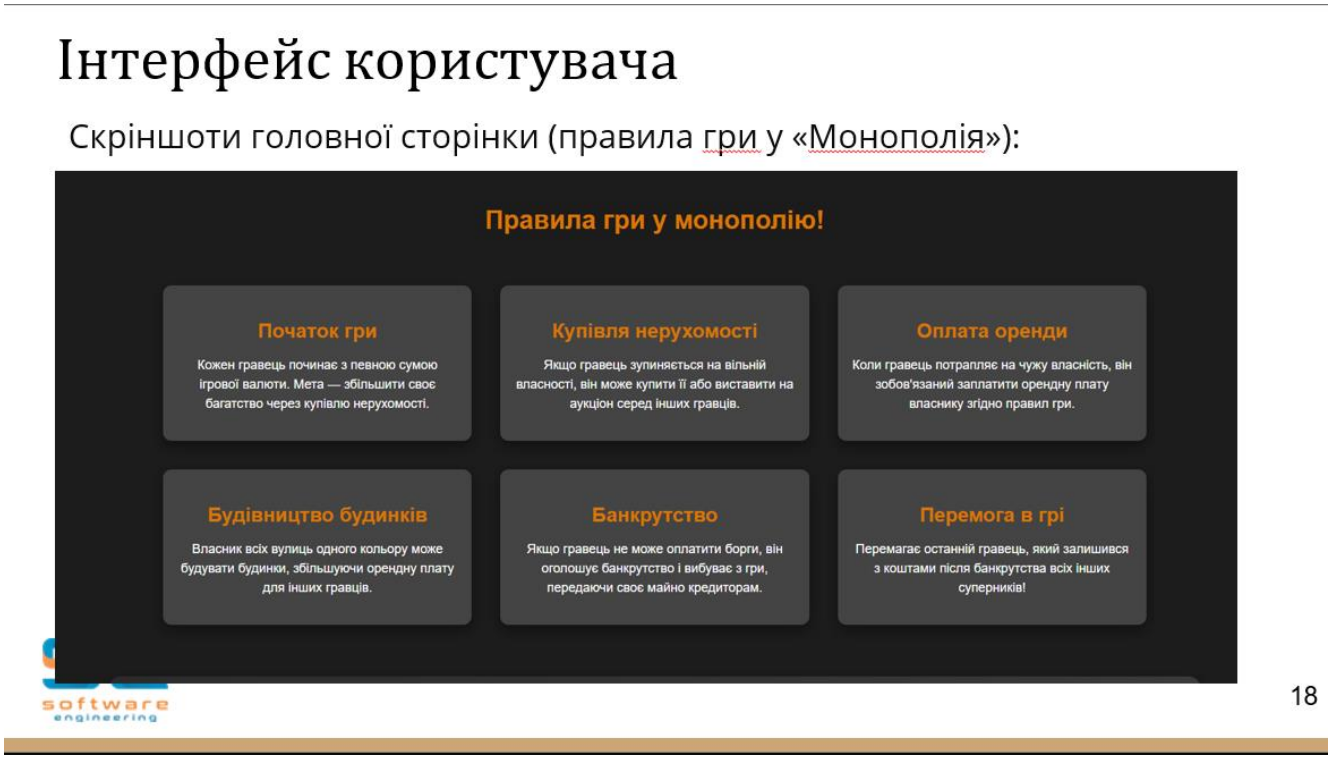


Рисунок Б.18 – Скріншот головної сторінки

Інтерфейс користувача

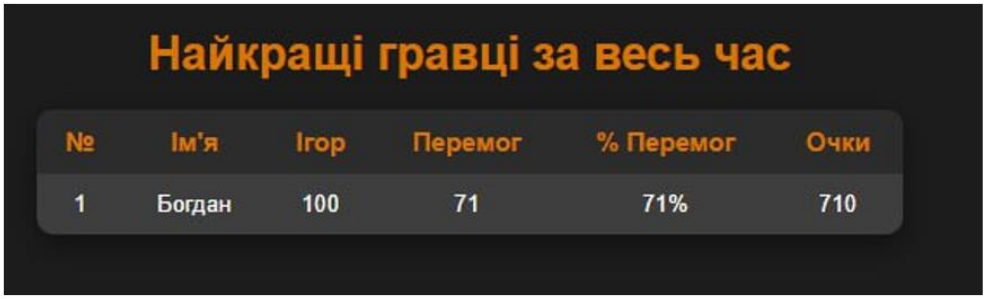
Скріншоти головної сторінки (відеогайд для новачків):

19

Рисунок Б.19 – Скріншот головної сторінки

Інтерфейс користувача

Скріншоти головної сторінки (таблиця найкращих гравців):



20

Рисунок Б.20 – Скріншот головної сторінки

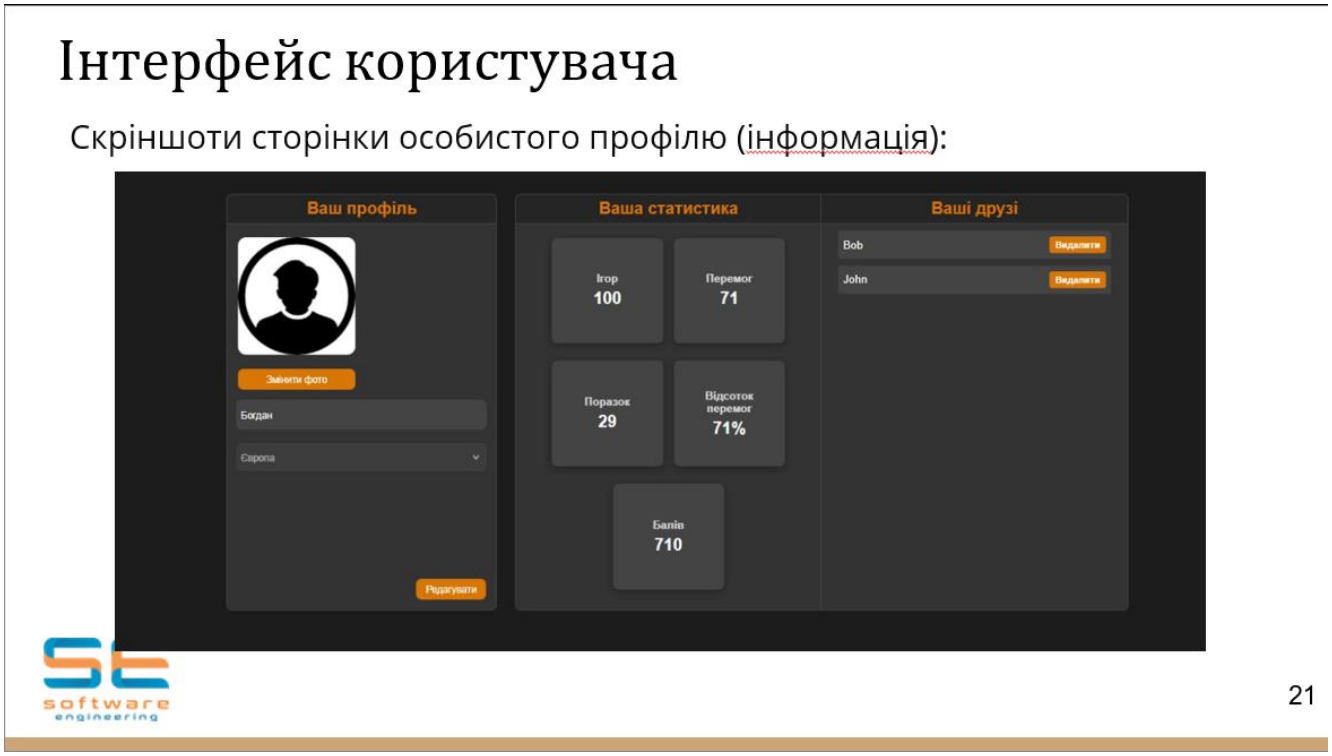


Рисунок Б.21 – Скріншот сторінки особистого профілю

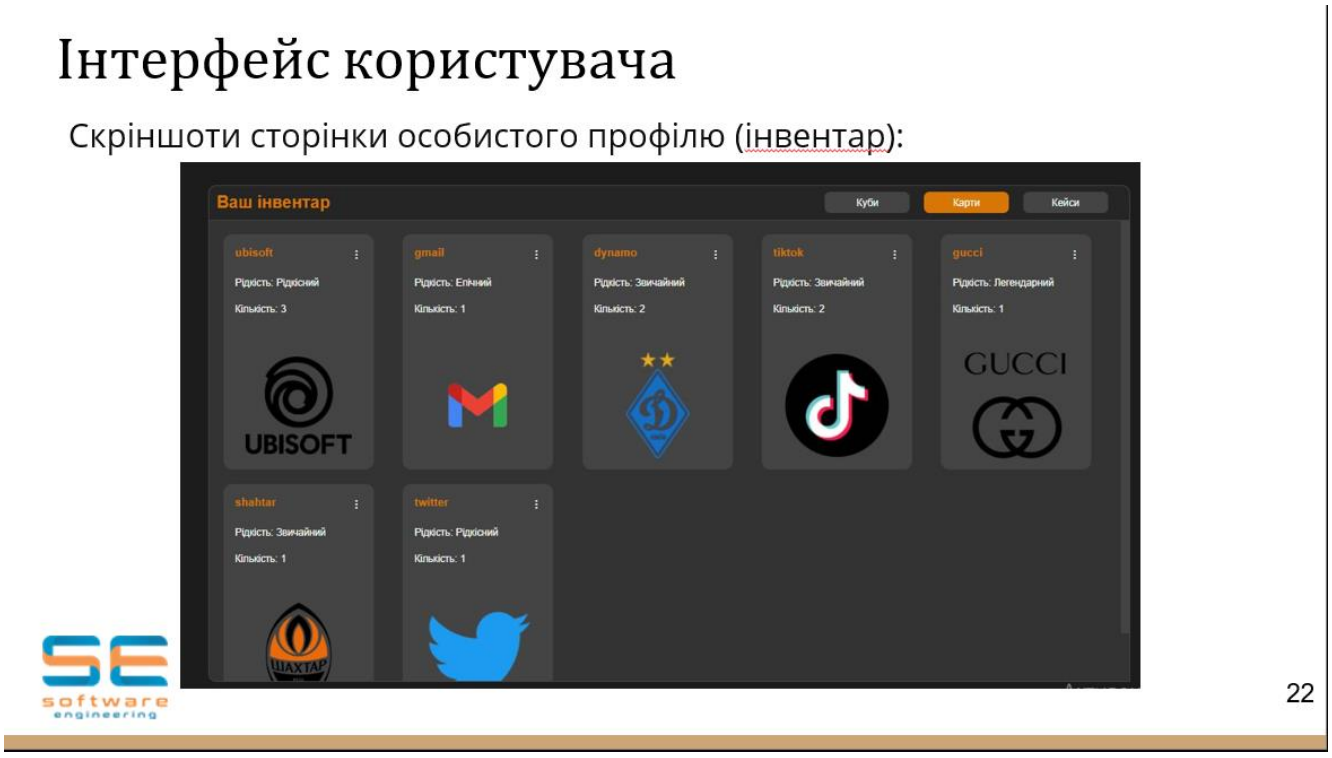


Рисунок Б.22 – Скріншот сторінки особистого профілю

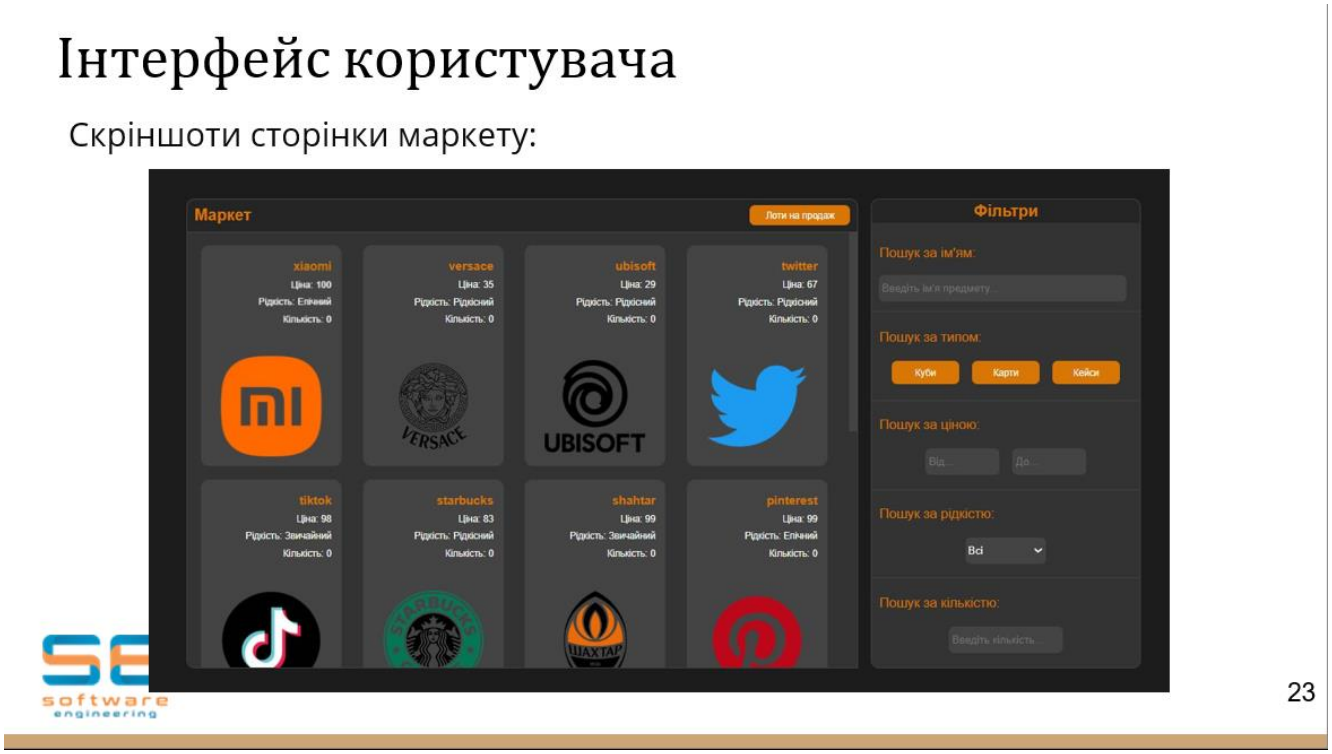


Рисунок Б.23 – Сторінка маркету

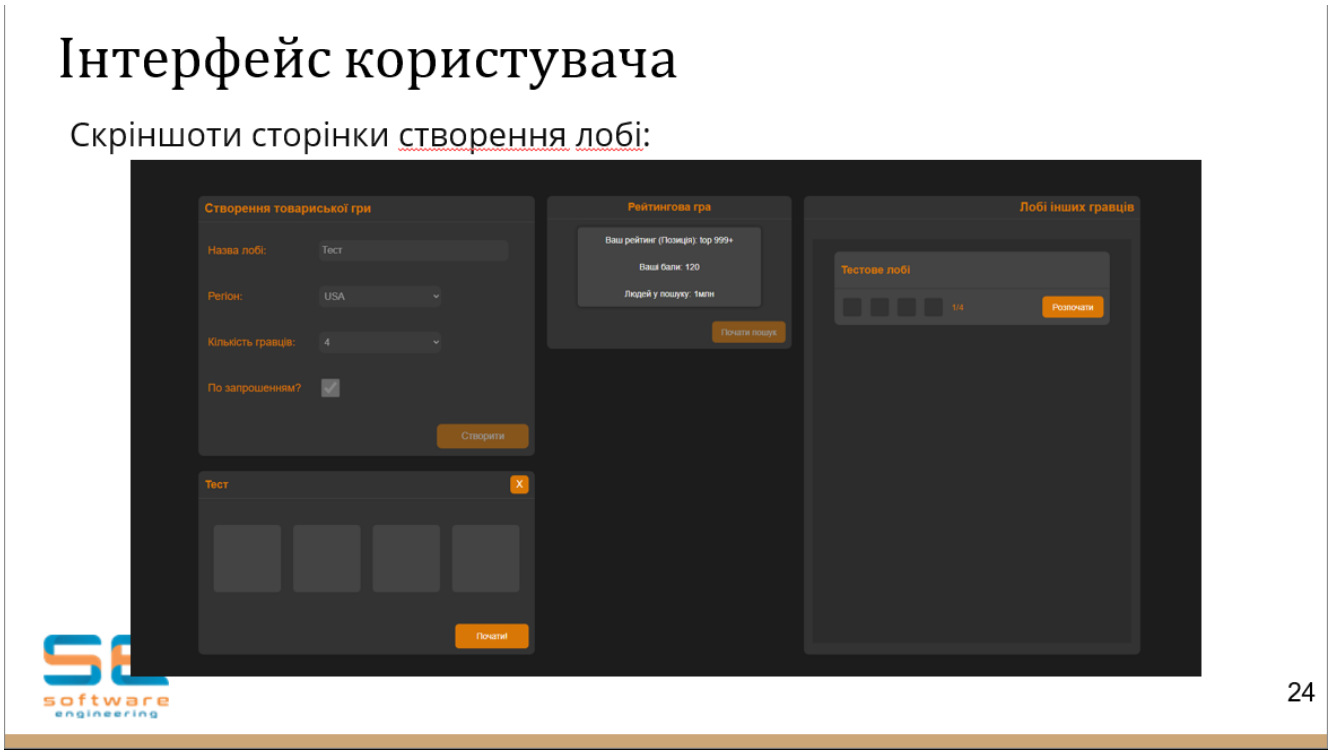


Рисунок Б.24 – Сторінка створення лобі

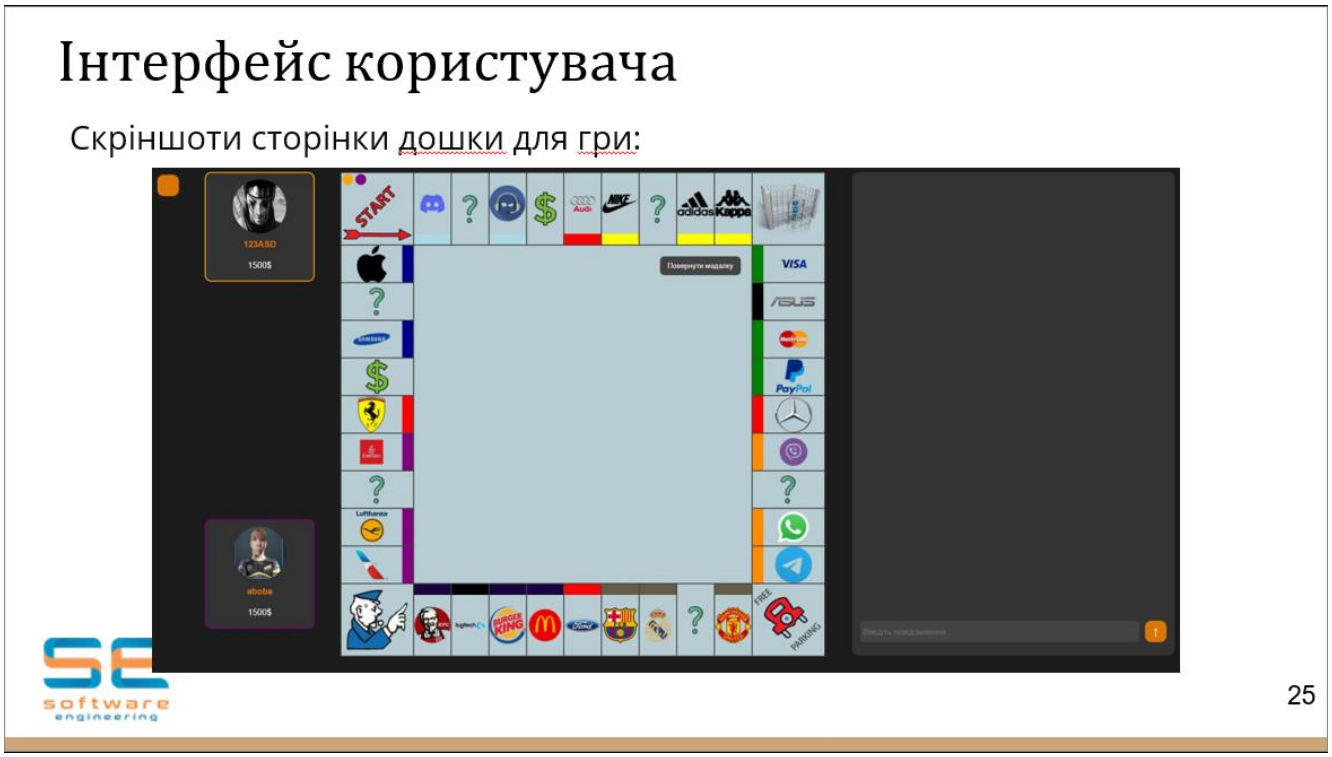


Рисунок Б.25 – Сторінка гри у Монополію

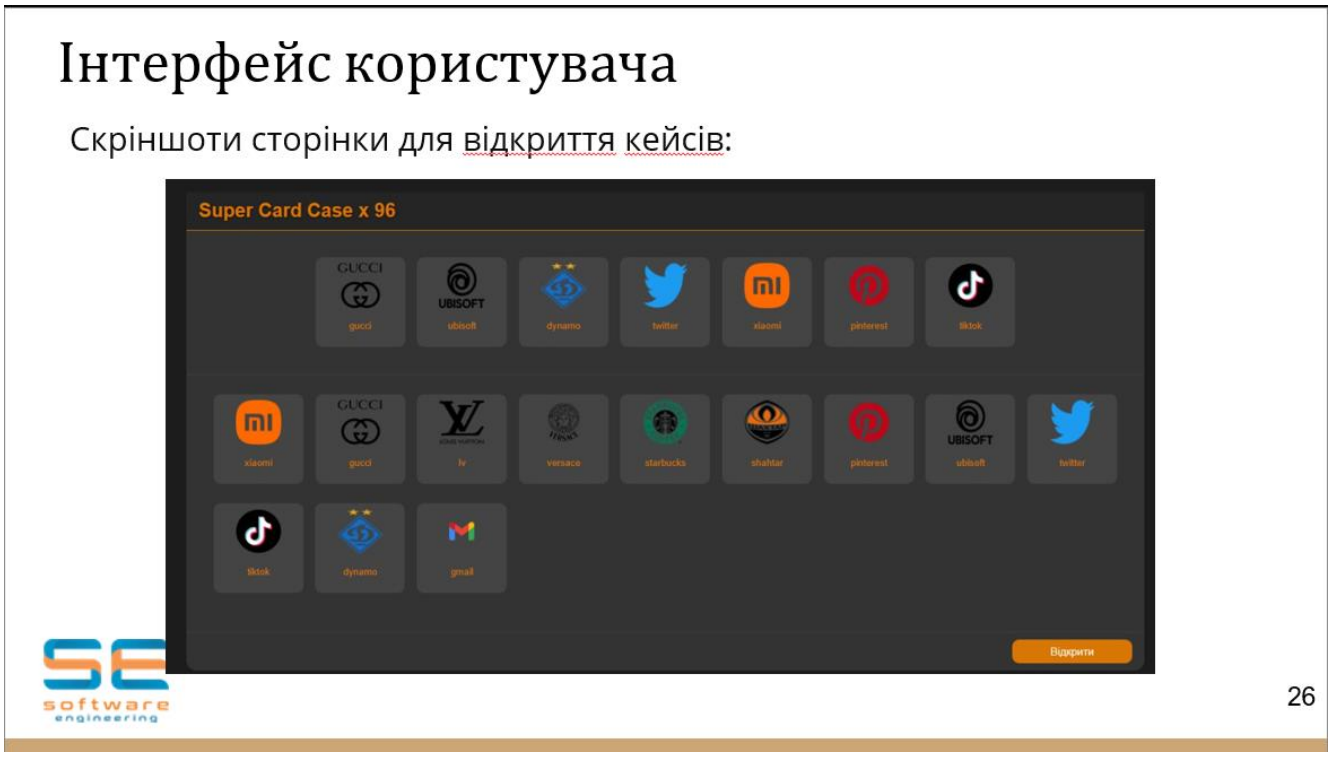


Рисунок Б.26 – Сторінка для відкриття кейсів

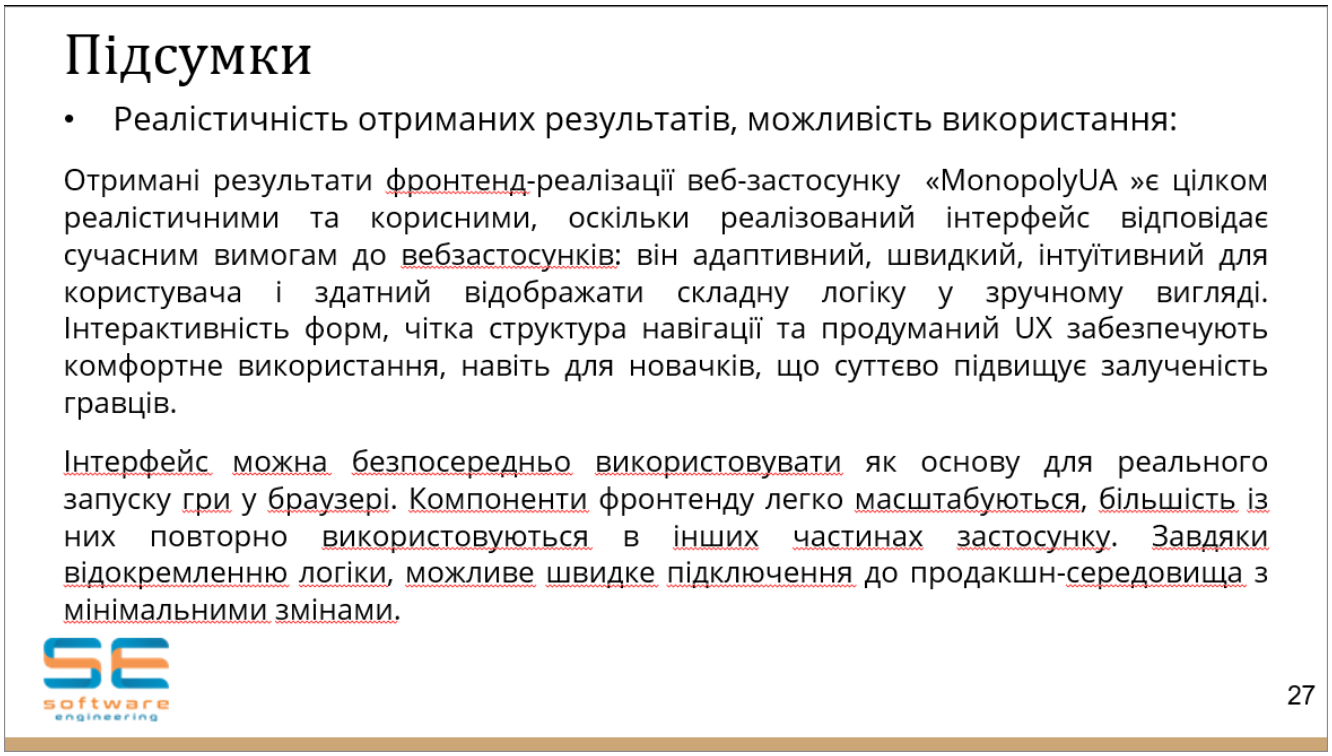


Рисунок Б.27 – Підсумки (реалістичність отриманих результатів, можливість використання)

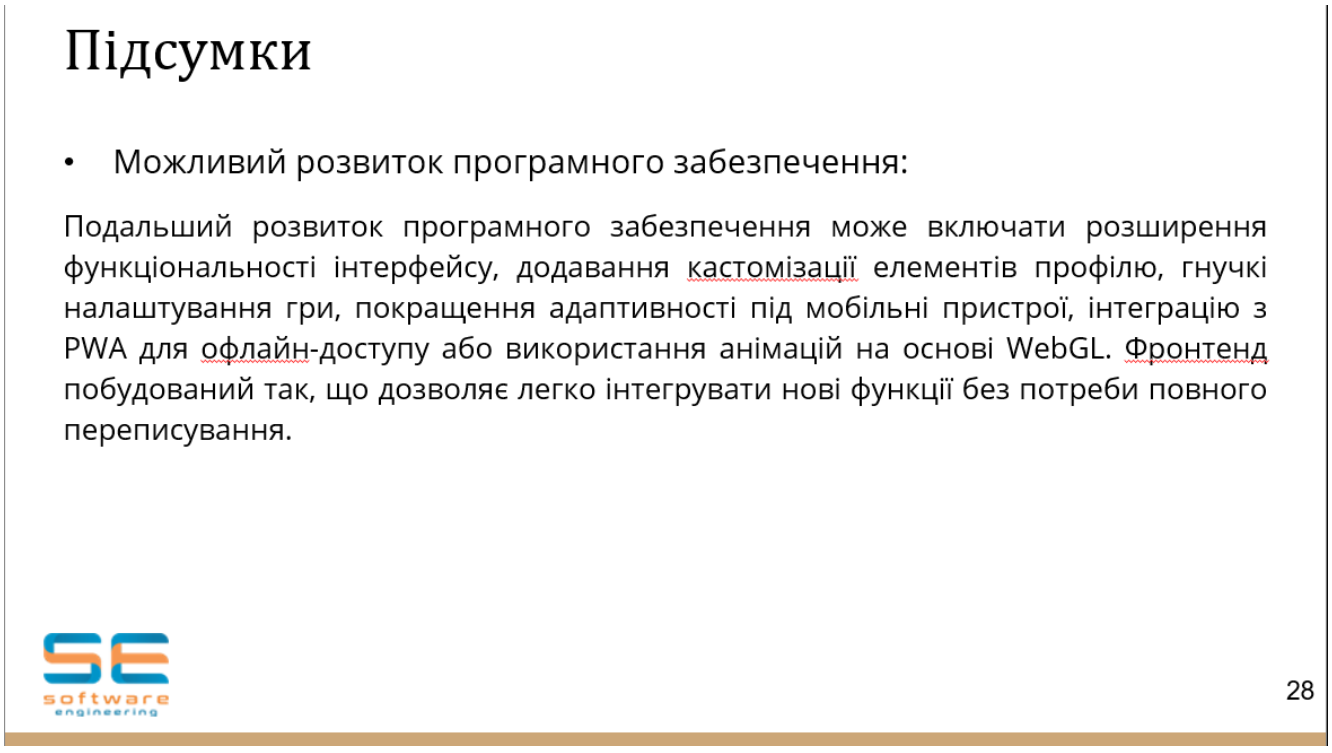


Рисунок Б.28 – Підсумки (можливий розвиток програмного забезпечення)

ДОДАТОК В

Специфікація програмного забезпечення (SRS)

1. Introduction

1.1 Purpose

Цей документ є специфікацією вимог до програми (SRS) для веб-застосунку «MonopolyUA». Головна ціль документу це визначення, що повинна виконувати система, щоб розробники могли перейти до проєктування та реалізації.

Аудиторія:

- розробники backend частини;
- розробник frontend частини.

1.2 Scope

Продукт: веб-застосунок «MonopolyUA».

Що робить:

- реєстрація/авторизація користувачів (JWT, Google Auth);
- головна сторінка, профілю, маркету, створення гри, дошки для гри;
- рейтингові таблиці та топ-гравці;
- створення/приєднання до ігрового лобі;
- відображення динамічної дошки гри з ходами, оплатами та купівлею.
- чат у реальному часі;
- обробка онлайн-статусів та AFK.

Що не робить:

- не націлена на мобільні застосунки;
- не підтримує інтеграцію реальних грошей.

Бізнес-мета: створити інтерактивну онлайн-версію «Монополії» з соціальними компонентами та транзакціями між гравцями.

1.3 Definitions, Acronyms and Abbreviations

- HTML / CSS / JS - Технології фронтенду;
- Django - Python фреймворк;
- DRF - Django REST Framework;
- Channels - Django Channels для WebSocket;
- Daphne - ASGI-сервер;
- Redis - in-memory datastore;
- PostgreSQL - Реляційна СУБД;
- Asyncio - Асинхронна бібліотека Python;
- UUID - Унікальний ідентифікатор об’єкта;
- JWT - JSON Web Token;
- CORS - Cross-Origin Resource Sharing;
- Simple JWT - Django-бібліотека для JWT;
- django-redis - Бекенд кешування через Redis.

1.4 References

- RFC 7519 — JSON Web Token;
- Django 5.1 documentation;
- Django REST Framework docs;
- Django Channels docs + ASGI/Redis integration;
- PostgreSQL 15 docs;
- Relevant articles on WebSocket-based real-time games.

1.5 Overview

- Розділ 2: загальний опис системи та її контекст;
- Розділ 3: функціональні та нефункціональні вимоги;
- Розділ 4: сценарії використання;
- Розділ 5-6: зовнішні інтерфейси, інші вимоги.

2 Overall Description

2.1 Product Perspective

Веб-застосунок «MonopolyUA» є незалежною програмною системою, що працює автономно у вигляді повнофункціонального веб-клієнта та серверної частини з підтримкою реального часу. Його основне призначення — забезпечити користувачам інтерактивне середовище для гри у цифрову версію настільної гри «Монополія», орієнтовану на український культурний контекст. Програма не є компонентом більшого корпоративного рішення, однак вона побудована з використанням стандартних веб-протоколів і може масштабуватись або інтегруватись у ширші системи (наприклад, з платформами рейтингів чи сторонніми сервісами для зберігання профілів).

Застосунок включає фронтенд-частину, реалізовану засобами HTML, CSS та JavaScript без використання фреймворків, та бекенд-частину, розроблену на базі Django 5.1 із поділом на синхронні REST API та асинхронну логіку на WebSocket через Django Channels. Основним джерелом збереження даних виступає PostgreSQL, а Redis відіграє роль як кеш-сервера, так і каналу повідомлень. Завдяки цьому архітектура дозволяє реалізувати повноцінну багатокористувацьку гру в реальному часі з інтегрованим чатом та постійною передачею станів гри.

2.1.1 System Interfaces

Система взаємодіє з кількома зовнішніми службами та протоколами, необхідними для її роботи. Однією з ключових інтеграцій є Google OAuth 2.0, що використовується для авторизації користувачів за допомогою облікових записів Google. Це забезпечує безпечний і зручний доступ без потреби вводити пароль. Додатково, для відновлення пароля через email використовується SMTP-сервер, налаштований для надсилання листів із посиланнями на зміну пароля.

Ще одним важливим елементом є Redis, який, окрім кешування, використовується для організації обміну повідомленнями між клієнтом і сервером за допомогою Pub/Sub моделі. Цей механізм критично важливий для роботи чату,

ігрових ходів і системи статусів онлайн/офлайн. Сама система також може бути інтегрована з моніторинговими сервісами або CI/CD пайплайнами для автоматизації розгортання.

2.1.2 Interfaces

Інтерфейс користувача системи представлений у вигляді традиційного багатосторінкового веб-сайту, де всі елементи візуального відображення реалізовані засобами HTML, CSS та JavaScript. Головна мета інтерфейсу — забезпечити простоту навігації, інтуїтивне керування та візуальну ідентичність, притаманну грі «Монополія». Усі основні розділи — профіль, лобі, ігрове поле, магазин — доступні з головного меню, а реакції на дії користувача реалізовані без перевантаження сторінки, з допомогою динамічної взаємодії з API. Інтерфейс також враховує основи доступності: використання контрастних кольорів, альтернативного тексту для зображень.

2.1.3 Hardware Interfaces

Веб-застосунок «MonopolyUA» не має прямої взаємодії з апаратним забезпеченням. Усі процеси виконуються у віртуальному або хмарному середовищі, тому вимоги до апаратної частини мінімальні та стандартні для типових серверів. Система не контролює зовнішні пристрої, як-от сканери, сенсори чи інші периферійні модулі. Усі компоненти взаємодіють через програмні інтерфейси, і будь-яке апаратне забезпечення розглядається лише як середовище виконання.

2.1.4 Software Interfaces

Програмна система «MonopolyUA» побудована на низці зовнішніх бібліотек та платформ. Бекенд реалізовано з використанням фреймворку Django (версія 5.1), що забезпечує базову структуру, маршрутизацію та взаємодію з базою даних PostgreSQL. Для створення REST API використовується Django REST Framework.

Авторизація реалізована з допомогою бібліотеки Simple JWT, яка надає функціональність створення, оновлення та перевірки токенів доступу.

Асинхронна частина системи базується на Django Channels, що працює поверх ASGI-сервера Daphne, а Redis служить як кеш і транспортний канал. Частина клієнтської частини системи взаємодіє з сервером через WebSocket-протокол, що забезпечує двосторонню комунікацію в реальному часі, необхідну для гри, чату та статусів гравців.

2.1.5 Communications Interfaces

Застосунок використовує кілька типів комунікаційних інтерфейсів. Основна взаємодія між клієнтом і сервером здійснюється через HTTP, що забезпечує захищене з'єднання з REST API. Для обміну повідомленнями в режимі реального часу система використовує WebSocket-з’єднання, яке працює через ASGI та Redis. Крім цього, система підтримує CORS, що дозволяє безпечний доступ до API з фронтенду, розгорнутого на окремому домені.

2.1.6 Memory Constraints

У системи немає жорстко встановлених обмежень на об’єм оперативної пам’яті, проте з огляду на використання Redis та розподіленого оброблення даних, мінімальна рекомендована конфігурація для сервера передбачає не менше 2 GB RAM. Це дозволяє ефективно підтримувати декілька одночасних ігрових сесій, чат та кешування без значного навантаження. Під час масштабування рекомендується дотримуватись балансу між доступною пам’яттю Redis і кількістю активних WebSocket-підключень.

2.1.7 Operations

Система працює у постійному інтерактивному режимі. Більшість дій користувачів відбуваються у реальному часі, з мінімальним затримками та без необхідності оновлення сторінки. Адміністративна панель передбачає базові функції моніторингу, а також можливість автоматичного перезапуску сесій гри у

разі збоїв. Для збереження стабільної роботи системи рекомендовано збереження даних про завершені ігри.

2.1.8 Site Adaptation Requirements

Перед початком роботи з системою необхідно виконати кілька підготовчих кроків, зокрема розгортання інфраструктури з встановленням Redis, PostgreSQL, Django-серверу та необхідних залежностей. У випадку хостингу в хмарному середовищі (наприклад, AWS або DigitalOcean), також слід враховувати налаштування SSL-сертифікатів і проксісерверів. Система може бути адаптована до конкретних умов інсталяції шляхом налаштування конфігураційних файлів середовища, а також вибору відповідного плану масштабування залежно від кількості користувачів.

2.2 Product Functions

Основне функціональне призначення веб-застосунку «MonopolyUA» полягає у забезпеченні гравцям повноцінного онлайн-досвіду, наближеного до класичної настільної гри «Монополія», із розширеннями, адаптованими до цифрового формату. Програма повинна надавати користувачу повний цикл взаємодії — від реєстрації та входу до участі в динамічному ігровому процесі з іншими гравцями, управління власним профілем, обміну предметами та спілкування у реальному часі.

Першочерговою функцією системи є створення безпечного механізму автентифікації, що реалізується за допомогою власного інтерфейсу логіна з JWT-токенами та авторизації через Google. Сюди ж входить функція відновлення паролю, яка здійснюється через email. Після входу користувач має змогу керувати власним профілем: змінювати особисті дані, нікнейм, аватарку, переглядати ігрову статистику та керувати друзями.

Другою ключовою підсистемою є особистий кабінет гравця. Він включає інвентар з предметами (скінами), які можна застосовувати в грі для персоналізації, продавати та у деяких випадках відкривати, якщо тип предмету – кейс. Також існує

система друзів, яка дозволяє додавати, підтверджувати або видаляти. Це посилює соціальний аспект проєкту.

Третій функціональний блок — це ігровий маркет, що дозволяє купівлю предметів інших користувачів. Усі транзакції супроводжуються перевіркою балансу, зняттям коштів та оновленням інвентаря. Додатково реалізована система кейсів із випадковими винагородами, яка стимулює активність у грі.

Найважливішим ядром застосунку є сам ігровий процес. Користувачі можуть створювати або приєднуватись до ігрових лобі, в яких вони налаштовують параметри майбутньої гри. Кожна сесія містить повноцінне ігрове поле з елементами «Монополії»: гральними кубиками, об’єктами, грошима, штрафами та ін. Ігрова логіка реалізована через WebSocket, що дозволяє всім гравцям бачити зміни в реальному часі, включаючи чат, таймери та зміни стану гри.

Крім того, система виконує функції збору статистики, підрахунку перемог і формування рейтингових списків гравців. Ці рейтинги впливають на підбір суперників у майбутніх матчах, а також дозволяють реалізовувати змагання та нагороди для найактивніших учасників.

Усі ці функції пов’язані між собою логічно та реалізовані через взаємодію фронтенд-компонентів і бекенд-служб у режимі реального часу або через API-запити, що забезпечує динамічну і плавну роботу застосунку.

2.3 User Characteristics

Цільова аудиторія застосунку «MonopolyUA» складається з широкого кола користувачів, основну частину яких формують підлітки, молодь та дорослі віком від 14 до 35 років. Більшість користувачів мають базовий або середній рівень комп’ютерної грамотності, тому система повинна бути простою, інтуїтивно зрозумілою та доступною без потреби проходження навчання або читання інструкцій.

Користувачі знайомі з класичними браузерними іграми, соціальними мережами та мобільними додатками, що формує очікування до швидкого відгуку, інтерактивності інтерфейсу та соціальної взаємодії. З огляду на це, інтерфейс

застосунку повинен мати мінімалістичний, адаптивний дизайн, бути зрозумілим навіть без текстових підказок і працювати стабільно на більшості сучасних браузерів.

З огляду на наявність функцій для гри в реальному часі, чатів і рейтингових систем, очікується, що користувачі мають досвід участі в багатокористувацьких онлайн-іграх або соціальних платформах. Це дозволяє припустити достатній рівень комфортності з механіками типу «додати у друзі», «запросити в лобі», «вийти з гри», «перевірити інвентар» тощо.

Водночас розробка враховує й менш досвідчених користувачів, тому реалізована система підказок, автоматичних повідомлень про хід гри, а також механізм обробки AFK-ситуацій, щоб уникнути збоїв у ігровому процесі. Усі ці аспекти враховуються в дизайні інтерфейсів і логіці бекенду з метою створення комфортного та збалансованого досвіду для всіх користувачів.

2.4 Constraints

В процесі розробки веб-гри «MonopolyUA» необхідно врахувати низку обмежень, що визначають технічні, організаційні та регуляторні рамки проєкту:

а) регуляторні політики та захист даних – система обробляє персональні дані користувачів (електронна пошта, аватар, історія ігор), тому розробка та експлуатація повинні відповідати вимогам GDPR (ЄС), Закону «Про захист персональних даних» (Україна) та аналогічних регуляцій. Надсилання паролів і повідомлень через електронну пошту реалізується з використанням TLS-з’єднання та захищеного SMTP-серверу;

б) інтеграція з зовнішніми інтерфейсами:

1) OAuth 2.0 для Google Sign-In;

2) SMTP-сервер для відновлення паролю та системних сповіщень.

Всі API-виклики до сторонніх сервісів повинні відбуватися через захищені HTTP-канали з перевіркою сертифікату.

в) паралельна робота та висока доступність – система підтримує одночасні ігрові сесії та кілька WebSocket-з’єднань на одного користувача. Використання

Redis для розподіленого зберігання станів лобі та ігор забезпечує горизонтальне масштабування й синхронну роботу кількох інстансів сервера.

г) аудит і логування - усі ключові події (створення лобі, приєднання/вихід гравця, хід гравця, фінансові транзакції, банкрутство тощо) фіксуються в хронологічних лістах Redis та можуть бути експортовані для подальшого аналізу. Журнали також відіграють роль розробницького та експлуатаційного аудиту.

д) контрольні та безпекові функції:

1) CSRF-захист для REST-інтерфейсу;

2) валідація та очищення всіх вхідних даних, включно з повідомленнями чату;

3) захист WebSocket-каналів через JWT-мідлвера із обмеженим часом життя токена.

е) критичність додатку – хоча «MonopolyUA» не є системою із життєво критичними операціями, користувачі очікують безперебійної та чуйної роботи гри. Будь-яка втрата стану партії може призвести до розчарування та відтоку гравців;

ж) мовні та платформові вимоги – розробка виконана на Python 3.8+ з використанням Django 5.1 і Django Channels. Для клієнтської частини застосовано сучасні версії JavaScript ES6+. Будь-які сторонні бібліотеки повинні бути з відкритим кодом та підтримуватися спільнотою.

Ці обмеження визначають рамки вибору інструментів, архітектурних рішень та процедур розробки, усуваючи ризики невідповідності регламентам, гарантуючи безпеку й стабільність системи.

2.5 Assumption and Dependencies

Реалізація функціоналу веб-застосунку «MonopolyUA» ґрунтується на низці припущень і зовнішніх залежностей, які безпосередньо впливають на виконання вимог, описаних у цьому документі. У разі зміни цих припущень — вимоги можуть потребувати перегляду.

- вся логіка зберігання стану гри, таймерів, хронології подій та менеджера ходів залежить від стабільної роботи Redis-сервера. У разі зміни

архітектури з Redis на інше in-memory сховище (наприклад, Memcached або PostgreSQL Pub/Sub), вимоги щодо продуктивності та синхронізації повинні бути оновлені;

- передбачається, що серверна частина продовжуватиме функціонувати в середовищі Django + Channels. Заміна цих технологій (наприклад, на FastAPI або Node.js) потребуватиме переосмислення архітектури WebSocket-комунікацій, мідлверів, а також механізмів автентифікації;
- уся ідентифікація користувачів у WebSocket-з'єднаннях базується на JWT-токенах. Якщо зміниться метод автентифікації (наприклад, перехід на session-based або OAuth-only), буде потрібно оновити логіку обробки scope у WS-підключеннях;
- механізм відновлення паролю залежить від можливості відправити лист користувачу з новим згенерованим паролем. У разі недоступності поштового сервісу функціональність скидання пароля буде порушено, що вимагає зміни SRS (наприклад, через перехід до верифікаційних токенів);
- реалізація сторонньої автентифікації залежить від працездатності Google API. У разі зміни політики OAuth або недоступності Google Identity Platform — необхідна адаптація системи (наприклад, через підтримку інших провайдерів або fallback на внутрішню автентифікацію);
- система продажу/купівлі скінів ґрунтується на внутрішньому ігровому балансі користувача. Припускається, що не планується підключення реальних платіжних систем. Якщо таке припущення зміниться, SRS доведеться переглянути з урахуванням фінансових і юридичних вимог (зокрема, підтримки реальних грошей, податків і безпеки транзакцій);
- передбачається, що користувачі взаємодіють із грою через сучасні браузери, які підтримують WebSocket, localStorage та інші сучасні веб-технології. У разі зміни цільової платформи (наприклад, підтримка мобільного додатку або застарілих браузерів), інтерфейс та архітектура клієнта мають бути адаптовані.

Ці припущення є фундаментом для визначення поточних вимог. У разі їх зміни функціональні або нефункціональні вимоги до системи можуть потребувати перегляду, адаптації або переосмислення.

2.6 Apportioning of Requirements

У процесі планування розробки системи «MonopolyUA» було виявлено, що деякі вимоги, хоча й важливі, можуть бути реалізовані на подальших етапах через обмеження часу, ресурсів або складність реалізації. Враховуючи обсяг проекту, розробка розбивається на кілька ітерацій, де пріоритетними є функції, критичні для базової роботи системи.

До першої ітерації включено:

- реалізація основної гри з класичними правилами монополії;
- підтримка WebSocket-з’єднання для гри в реальному часі;
- механізм черги ходів із таймерами;
- базова система логів і чату;
- авторизація (звичайна + Google OAuth);
- профіль користувача з переглядом статистики та редагуванням даних;
- базова система друзів і повідомлень;
- система інвентарю та торгівлі скінів за внутрішню валюту;
- застосування скінів до придбаних власностей під час гри.

До наступних ітерацій (можуть бути реалізовані в майбутньому):

- розширена система анімацій на полі гри;
- візуальна мапа всіх власностей і скінів гравців у поточній сесії;
- повноцінна мобільна версія інтерфейсу (адаптація для touch-інтерфейсів);
- доопрацювання торгової системи з можливістю створення аукціонів або обміну між гравцями;
- введення нагород, досягнень та бойових пропусків;
- створення системи репортів/скарг на гравців;
- статистика по іграх друзів;

- публічні/приватні лобі з паролем і фільтрами;
- можливість створювати кастомні правила гри (варіативність правил).

Пріоритетність реалізації функцій у майбутніх ітераціях визначатиметься відповідно до відгуків користувачів, технічних можливостей та стратегічних цілей проєкту. Рішення про відкладення реалізації певних вимог приймалося у співпраці з замовником і командою розробки.

3 Specific Requirements

3.1 External Interfaces

3.1.1 Інтерфейс автентифікації користувача:

а) опис: Забезпечує реєстрацію, вхід, авторизацію через Google (OAuth) та відновлення паролю;

б) джерело введення: Користувач через форму входу/реєстрації;

в) призначення виводу: Сервер автентифікації;

г) допустимі значення:

1) електронна пошта: у форматі, що відповідає стандарту RFC 5322;

2) пароль: від 6 до 128 символів, щонайменше одна літера та одна цифра.

д) одиниці виміру: Текстові поля (рядки);

е) часова характеристика: Під час взаємодії з формою входу або реєстрації;

ж) зв’язки з іншими інтерфейсами: Профіль користувача, сесії гри.

з) формат вікна: Окремі форми входу та реєстрації;

и) формат даних: JSON;

к) фінальні повідомлення: “Вхід успішний”, “Невірний пароль”, “Користувача не знайдено”.

3.1.2 Інтерфейс профілю користувача:

а) опис: Дозволяє переглядати та редагувати нікнейм, аватар, пароль і переглядати статистику ігор;

б) джерело введення: Користувач на сторінці профілю;

в) призначення виводу: База даних користувачів;

г) допустимі значення:

1) нікнейм: 3–20 символів, тільки латинські літери, цифри, символи "_", "-";

2) Зображення: .png, .jpg, розмір до 5MB.

д) одиниці виміру: Рядки, зображення;

е) часова характеристика: Після входу до профілю;

ж) зв’язки з іншими інтерфейсами: Аутентифікація;

з) формат вікна: Вкладка "Профіль";

и) формат даних: JSON або multipart/form-data;

к) фінальні повідомлення: “Профіль оновлено”, “Файл занадто великий”.

3.1.3 Інтерфейс лобі гри:

а) опис: Дозволяє створювати ігрові кімнати або приєднуватися до них;

б) джерело введення: Користувач із головного меню або за посиланням-запрошенням;

в) призначення виводу: Менеджер сесій гри;

г) допустимі значення:

1) назва кімнати: 3–30 символів;

2) кількість гравців: від 2 до 4.

д) одиниці виміру: Рядки, числа;

е) часова характеристика: До початку гри;

ж) зв’язки з іншими інтерфейсами: Список друзів, повідомлення;

з) формат вікна: Список доступних кімнат або вікно створення;

и) формат даних: JSON;

к) фінальні повідомлення: “Кімната створена”, “Гравець приєднався”, “Гру розпочато”.

3.1.4 Інтерфейс самої гри:

а) опис: Головне ігрове поле, чат, лог ходів, панель гравця, менеджер ходу;

б) джерело введення: Дії користувача під час гри;

в) призначення виводу: Сервер гри через WebSocket;

г) допустимі значення: Команди: "кинути кубики", "купити", "пропустити", "передати", "взяти заставу" тощо.

д) одиниці виміру: Команди, повідомлення;

е) часова характеристика: У режимі реального часу;

ж) зв’язки з іншими інтерфейсами: Менеджер черги, чат, профіль;

з) формат вікна: Ігрове поле з боковою панеллю дій;

и) формат даних: WebSocket (JSON);

к) фінальні повідомлення: “Хід завершено”, “Карточку куплено”, “Гравець банкрут”.

3.1.5 Інтерфейс торгової платформи:

а) опис: Купівля та продаж скінів за внутрішню валюту;

б) джерело введення: Користувач через сторінку ринку;

в) призначення виводу: Сервер торгівлі;

г) допустимі значення:

1) ціна: не менше 1 одиниці внутрішньої валюти;

2) кількість: не менше 1.

д) одиниці виміру: Числові та текстові фільтри;

е) часова характеристика: Доступна будь-коли;

ж) зв’язки з іншими інтерфейсами: Інвентар, база предметів;

з) формат вікна: Таблиця з фільтрами, кнопки дій;

и) формат даних: JSON;

к) фінальні повідомлення: “Предмет куплено”, “Товар знято з продажу”.

3.1.6 Інтерфейс інвентарю:

а) опис: Перегляд наявних скінів, вибір для гри, виставлення на продаж;

б) джерело введення: Користувач із вкладки інвентарю;

в) призначення виводу: Сервер інвентарю / торгівлі;

г) допустимі значення: Тільки ті скіни, що є у власності;

д) одиниці виміру: Назва, тип, кількість;

е) часова характеристика: У будь-який момент;

ж) зв’язки з іншими інтерфейсами: Гра, торгова платформа;

з) формат вікна: Плиткове відображення предметів;

и) формат даних: JSON;

к) фінальні повідомлення: “Предмет виставлено на продажу”.

3.1.7 Інтерфейс повідомлень:

а) опис: Обмін системними повідомленнями, запрошення в гру, запити в друзі;

б) джерело введення: Користувач або система;

в) призначення виводу: Інший користувач або сервер повідомлень;

г) допустимі значення: Повідомлення до 256 символів;

д) одиниці виміру: Текст;

е) часова характеристика: У режимі реального часу;

ж) зв’язки з іншими інтерфейсами: Друзі, лобі гри, профіль;

з) формат вікна: Панель сповіщень або вкладка повідомлень;

и) формат даних: JSON, WebSocket;

к) фінальні повідомлення: “Запрошення надіслано”, “Запит в друзі прийнято”.

3.2 Functions

3.2.1 Реєстрація та авторизація

- система повинна перевіряти коректність введеної електронної пошти за встановленим шаблоном;
- система повинна перевіряти, щоб пароль містив щонайменше одну цифру та одну літеру;

- система повинна надавати користувачу можливість входу через Google OAuth 2.0;
- система повинна надсилати лист із новим згенерованим паролем у разі запиту на відновлення доступу.

3.2.2 Керування профілем користувача

- система повинна дозволяти користувачу змінювати нікнейм, аватар та пароль;
- система повинна перевіряти унікальність нікнейму під час збереження.
- система повинна відображати статистику завершених ігор користувача (кількість ігор, перемоги, поразки тощо);
- система повинна перевіряти розмір та формат файлу аватару.

3.2.3 Ігрове лобі та менеджер сесій

- система повинна дозволяти створення нової ігрової кімнати з унікальним ідентифікатором;
- система повинна надавати можливість приєднання до існуючої гри за посиланням або зі списку доступних;
- система повинна перевіряти, що кількість гравців не перевищує 6 і не менше 2;
- система повинна автоматично запускати гру після досягнення мінімального числа гравців і підтвердження усіх учасників.

3.2.4 Основна ігрова логіка

- система повинна дозволяти користувачу кидати кубики та рухати фішку відповідно до результату;
- система повинна обробляти події при потраплянні на певну клітинку (купівля, оренда, податки, тюрма тощо);

- система повинна автоматично змінювати вигляд картки на полі, якщо у гравця є відповідний скін;
- система повинна виводити лог подій гри в режимі реального часу;
- система повинна обробляти перемогу, банкрутство та завершення гри згідно з правилами.

3.2.5 Інвентар і скіни

- система повинна дозволяти гравцю переглядати список наявних скінів;
- система повинна дозволяти вибирати активні скіни для гри;
- система повинна перевіряти наявність обраного скіна у гравця до його застосування в грі.

3.2.6 Торгова платформа

- система повинна дозволяти виставити предмет на продаж, вказавши кількість і ціну у внутрішній валюті;
- система повинна дозволяти переглядати доступні товари з фільтрацією за параметрами (тип, рідкість, ціна);
- система повинна перевіряти, чи дійсно користувач володіє відповідною кількістю товару перед виставленням;
- система повинна знімати предмет з торгівлі на запит користувача;
- система повинна зменшувати кількість валюти користувача при покупці та збільшувати її у продавця;

3.2.7 Повідомлення та запрошення

- система повинна дозволяти надсилання запрошення в гру іншим користувачам;
- система повинна дозволяти надсилання запитів у друзі;
- система повинна повідомляти користувача про отримання нових повідомлень у режимі реального часу.

3.2.8 Обробка помилок та відновлення

- система повинна інформувати користувача про неправильні або відсутні поля введення;
- система повинна зберігати стан гри при розриві з’єднання та дозволяти повторне приєднання;
- система повинна автоматично завершувати хід, якщо користувач не діє протягом визначеного часу.

3.2.9 Канали зв’язку

- система повинна підтримувати постійне WebSocket-з'єднання для обміну ігровими подіями в реальному часі;
- система повинна падати у fallback-режим при втраті WebSocket-з’єднання.

3.3 Performance Requirements

3.3.1 Статичні вимоги

- програмний засіб повинен забезпечувати підтримку щонайменше 500 одночасно авторизованих користувачів без деградації продуктивності;
- програмний засіб повинен підтримувати до 50 активних ігрових сесій одночасно на сервері з рекомендованими характеристиками;
- програмний засіб повинен обробляти до 1000 внутрішньоігрових транзакцій на хвилину;
- кількість одночасно відкритих з’єднань WebSocket не повинна перевищувати 1000 для одного інстансу сервера;
- кожен користувач може мати до 500 об’єктів (скінів) у власному інвентарі.

3.3.2 Динамічні вимоги

- 95% усіх ігрових дій (купівля, продаж, хід, обмін) повинні оброблятися протягом менше ніж 1 секунди після підтвердження користувачем;

- повідомлення через WebSocket повинні доставлятися з затримкою не більше 300 мс у межах однієї ігрової сесії;
- час від моменту запуску гри до повного завантаження ігрового поля на клієнтському інтерфейсі не повинен перевищувати 2 секунд;
- відкриття торгової платформи з фільтрами та переліком товарів має відбуватися за менше ніж 1.5 секунд для користувача із середнім інтернет-з'єднанням (10 Мбіт/с);
- пошук по торговій платформі повинен повертати результати за менше ніж 500 мс при вибірці до 1000 елементів.

3.3.3 Продуктивність при пікових навантаженнях

- під час пікових навантажень (до 500 користувачів онлайн), програмний засіб повинен зберігати не нижче 90% від заявленої базової швидкодії для обробки ігрових подій;
- програмний засіб повинен виконувати балансування навантаження при розгортанні на кількох інстансах для підтримки горизонтального масштабування.

3.4 Logical Database Requirements

3.4.1 Типи інформації

До основних типів даних, що використовуються в системі, належать:

- облікові записи користувачів (e-mail, хеш пароля, нікнейм, аватар, ID, Google OAuth-токени);
- друзі та запити в друзі;
- повідомлення (тип, відправник, одержувач, час надсилання, статус);
- ігрові сесії (учасники, стан гри, ігрове поле, лог ходів, чат);
- статистика користувача (кількість перемог, поразок, зіграних ігор, середня тривалість гри);
- інвентар (список скінів, кількість, ID користувача);

- торгівельна платформа (активні лоти, ціна, кількість, продавець);
- скіни (ID, тип, належність до типу властивості на полі);
- ігрова валюта (баланс користувача, історія транзакцій).

3.4.2 Частота використання

- дані профілю користувача: кожен сеанс входу / перегляду профілю;
- ігрові сесії: активне читання/запис під час гри;
- повідомлення: в режимі реального часу (висока частота доступу);
- торгові операції: помірна частота доступу;
- інвентар: висока частота у гравців, що часто змінюють скіни або торгують;
- статистика: обробка після завершення кожної гри, доступ для перегляду в профілі.

3.4.3 Можливості доступу

- дані користувача: доступ тільки авторизованого користувача або адміністратора;
- ігрові сесії: доступ лише учасників гри;
- повідомлення: доступ для відправника/одержувача;
- торгова платформа: публічний перегляд, редагування — лише власник лоту;
- інвентар: доступ користувача до власного інвентаря.

3.4.4 Сутності та зв’язки

Основні сутності:

- user — зберігає інформацію про користувача, має інвентар, скіни, статистику, список друзів, ігрову валюту;
- item — ігровий предмет, який користувачі можуть купити, продати або обмінювати;

- inventoryItem — сутність, що належить користувачу, зберігає item та кількість цього предмета;
- marketListing — оголошення про продаж, створене користувачем, містить item, кількість, ціну та дату створення;
- notifications — сповіщення про події у системі, містить відправника, одержувача, текст повідомлення, дату та статус;
- friends — зв'язок між двома користувачами, ініційований одним користувачем та спрямований іншому;
- gamestat — таблиця, що зберігає статистику користувача: кількість зіграних ігор, перемог, поразок та набраних очок;
- gamehistory — таблиця, що зберігає історію ігор користувача: дата, результат, тривалість, набрані очки;
- lobby — тимчасова кімната, що зберігається як Redis-хеш lobby:{lobby_id}:meta, містить lobby_id, name, region, max_players, invite_only, creator_id, created_at, started;
- player — гравець, представлений як Redis-хеш user:{player_id}, містить player_id, username, avatar_url, color;
- lobbyPlayer — проміжна сутність між lobby та player, зберігається як множина SADD lobby:{lobby_id}:players {player_id}, гарантує унікальність гравців у лобі;
- gameSession — ігрова сесія, створюється при старті лобі, зберігається як Redis-хеш game:{session_id}:meta з полями session_id, lobby_id, created_at, current_turn, turn_order та список гравців RPUSH game:{session_id}:players {player_id};
- gamePlayer — гравець у сесії, представлений хешем game:{session_id}:player:{player_id}, містить gp_id, balance, position, in_jail, jail_turns_left, immune_to_jail, last_roll, acted_props;
- property — клітинка ігрового поля, зберігається як Redis-хеш game:{session_id}:property:{property_id}, містить property_id, type, name,

buy_price, house_price, hotel_price, rent, group, owner_id, houses, mortgaged, mortgage_turns_left;

- turnState — стан ходу гри в сесії, зберігається як Redis-хеш game:{session_id}:turn_state з полями ts_id, phase, current_player, expires_at, action_payload;
- gameLog — лог дій гравців, зберігається як список RPUSH game:{session_id}:logs {log_entry}, кожен запис — JSON з log_id, session_id, timestamp, message;
- chatMessage — повідомлення чату в грі, зберігається у списку RPUSH game:{session_id}:chat {chat_obj}, кожен об'єкт містить chat_id, session_id, player_id, timestamp, text, а при поверненні клієнту додається username, color;

3.4.5 Обмеження цілісності

- унікальність email та нікнейму користувача;
- неможливість продажу більшої кількості скінів, ніж є в інвентарі;
- неможливість участі користувача в декількох активних іграх одночасно;
- валідація цінових значень на торговій платформі (додатні числа);
- захист від SQL-ін’єкцій та некоректного вводу даних.

3.4.6 Вимоги до збереження даних

- дані користувача та інвентарю мають зберігатися постійно, навіть після тривалого періоду неактивності;
- історія ігор та статистика зберігається не менше 1 року;
- повідомлення видаляються після 6 місяців, якщо не прочитані;
- ігрові сесії очищуються через 24 години після завершення гри;
- лоти з торгової платформи видаляються через 30 днів, якщо не реалізовані.

3.5 Design Constraints

3.5.1 Standards Compliance

У процесі розробки веб-застосунку "MonopolyUA" враховуються обмеження, пов'язані з дотриманням сучасних веб-стандартів, особливостями використовуваного середовища розгортання та регламентами обліку даних. Всі звіти та повідомлення, що створюються системою, повинні мати зрозумілий формат, відповідний сучасним вимогам до веб-інтерфейсів. Назви полів і змінних даних повинні відповідати зрозумілим іменам, прийнятим у галузі веб-розробки, із використанням camelCase або snake_case відповідно до призначення. У системі буде реалізовано механізм журналювання критичних змін, зокрема купівлі, продажу або відкриття кейсів, що дозволить створити повну трасу змін стану гравця та інвентарю для перевірки цілісності транзакцій. Дані мають зберігатися відповідно до політик безпечного обміну та захисту персональних даних, що особливо важливо при авторизації через сторонні сервіси, як-от Google API.

3.6 Software System Attributes

3.6.1 Reliability

Система повинна бути достатньо надійною для стабільної роботи під навантаженням до 500 одночасних користувачів. Усі критичні компоненти, як-от обробка авторизації, ігрового процесу та транзакцій у маркеті, повинні бути протестовані на стійкість до збоїв. Надійність перевірятиметься шляхом багатократного відтворення основних сценаріїв взаємодії з користувачем із вимірюванням кількості помилок на 1000 транзакцій. Очікувана середня безвідмовна тривалість роботи має становити не менше 200 годин.

3.6.2 Availability

Програмна система має бути доступною цілодобово, оскільки передбачає багатокористувацький онлайн-доступ до сервісу. У разі збоїв користувач повинен мати можливість поновити сесію або гру із втратою не більше 5 секунд ігрових

даних. Для забезпечення цього буде використовуватись збереження проміжного стану гри та асинхронна обробка запитів.

3.6.3 Security

Система повинна захищати користувацькі дані від несанкціонованого доступу, використовуючи JWT для автентифікації. Критичні дані (наприклад, паролі та токени) мають зберігатися у зашифрованому вигляді. Має бути реалізовано контроль за правильністю передачі повідомлень у WebSocket-каналах.

3.6.4 Maintainability

Проєкт побудований за принципом клієнт-серверної архітектури, де фронтенд і бекенд реалізовані як окремі незалежні додатки. Такий підхід дозволяє розвивати інтерфейс користувача та серверну логіку автономно, що спрощує підтримку й модернізацію системи.

Бекенд реалізовано з використанням Django REST Framework, що забезпечує модульну організацію коду. Основна логіка поділена на функціональні сегменти: автентифікація, управління профілем, маркет, ігрові механіки та чат. Це дає змогу легко змінювати або розширювати окремі частини системи без впливу на інші компоненти.

3.6.5 Portability

Портативність веб-застосунку «MonopolyUA» забезпечується використанням платформонезалежних технологій: Django для бекенду та стандартних технологі HTML/CSS/JS для фронтенду. Усі системозалежні налаштування (наприклад, шляхи до файлів, змінні середовища) винесені у конфігураційні файли, що дозволяє легко адаптувати систему під інші хост-машини або ОС. Використання перевірених інструментів, таких як Docker, забезпечує спільне середовище виконання на будь-якій платформі. Код не містить значної кількості хост-залежних фрагментів і протестований на Windows.

Вимірювання портативності відбуватиметься шляхом запуску застосунку в різних середовищах та оцінки змін, потрібних для успішної роботи. Таким чином, переносимість оцінюється як висока.

3.7 Organizing the Specific Requirements

3.7.1 System Mode

Веб-застосунок «MonopolyUA» має різні режими роботи, зокрема режим реєстрації та авторизації, режим роботи з профілем користувача, ігровий режим, режим роботи магазину та лідерборду. Кожен режим має свої особливості інтерфейсу та логіки, тому вимоги до системи формулюються окремо для кожного режиму, враховуючи специфіку відображення даних і поведінки системи, що дозволяє підвищити зручність та продуктивність.

3.7.2 User Class

Система розподіляє функціонал залежно від класу користувача. Зареєстровані гравці мають доступ до повного набору можливостей — участь в іграх, покупка скінів, додавання друзів. Адміністратори отримують розширені права для керування контентом і користувачами, а незареєстровані користувачі можуть лише ознайомитися з правилами та зареєструватися.

3.7.3 Objects

Основними об’єктами системи є користувач, ігрова сесія, ігрове поле, інвентар скінів, товари магазину, чат, рейтингові таблиці та списки друзів. Кожен об’єкт має свої атрибути (наприклад, профіль користувача містить ім’я, аватар, статистику), а також функції (додавання друга, створення лоббі, відправка повідомлення), що забезпечують реалізацію бізнес-логіки.

3.7.4 Feature

Функції системи визначаються як конкретні послуги, які вона надає користувачам, наприклад, реєстрація, авторизація, створення та пошук ігрових сесій, купівля скінів, обмін повідомленнями в чаті, оновлення профілю. Для кожної функції описується послідовність дій, необхідних для її виконання, включно з вхідними даними та очікуваною відповіддю системи.

3.8 Additional Comments

Для веб-застосунку «MonopolyUA» доцільно застосовувати одночасно кілька способів організації специфічних вимог, описаних у пункті 3.7, оскільки це допоможе краще структурувати вимоги та врахувати різні аспекти роботи системи.

Зокрема, організація вимог за режимами роботи системи (реєстрація, ігровий режим, магазин тощо) допоможе чітко виділити логіку для кожного сценарію використання. Водночас розподіл за класами користувачів (гравці, адміністратори, гості) дозволить чітко визначити права доступу та функціональні можливості.

Використання об’єктного підходу сприяє кращому опису структури системи та взаємодії між основними компонентами, такими як ігрова сесія, користувачі та інші. Організація вимог за функціями дозволяє деталізувати послідовність дій користувача та реакцію системи, що важливо для реалізації бізнес-логіки.

4. Change Management Process

Зміни в вимогах до веб-застосунку «MonopolyUA» будуть контролюватися через чітко встановлений процес управління змінами. Клієнт не може просто зателефонувати або висловити нову ідею усно — всі пропозиції щодо змін мають подаватися офіційно у письмовій формі електронною поштою. Після отримання запиту команда проекту проводить його оцінку з урахуванням технічних можливостей, впливу на поточний план робіт і бюджету. Всі рішення щодо впровадження змін ухвалюються колективно командою.

5. Document Approvals

TBD.

6. Supporting Information

Цей розділ містить допоміжні матеріали, що полегшують читання, розуміння та використання документа Специфікації вимог до програмного забезпечення (SRS). Хоча вони не є безпосередньою частиною формальних вимог, ці матеріали служать важливим доповненням для розробників, тестувальників, замовників та інших зацікавлених осіб.

Зміст документа представлено на початку файлу й охоплює всі розділи: від вступу до функціональних і нефункціональних вимог, включно з додатками.

Інтерактивний або текстовий покажчик термінів, абревіатур і ключових понять, які використовуються у документі. У разі публікації в цифровому вигляді, передбачена можливість пошуку за ключовими словами (наприклад: "WebSocket", "JWT", "Redis").