Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Гришакін Денис Дмитрович
студент групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Застосування для контролю кількості часу, що проведено за іграми

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Писаренко Катерина Олександрівна
канд. техн. наук, доцент

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Любченко В.В.
«___»_____20__р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гришакіна Дениса Дмитровича, група АС-161

1. Тема роботи: *Застосування для контролю часу, що проведено за іграми*

Керівник роботи: *Писаренко Катерина Олександрівна, канд. техн. наук, доцент*

Затверджені наказом ректора від «25» жовтня 2021р. № 374-в

2. Зміст роботи: підстави для розробки, опис предметної області, формалізація вимог, проектування системи, програмна реалізація, тестування

3. Перелік ілюстративного матеріалу:

Згідно зі слайдами презентації

4. Консультанти розділів роботи:

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

5. Дата видачі завдання: «___»_____20__р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Підстави для розробки	10.09.2021 – 15.09.2021	вик.
2	Опис предметної області	16.09.2021 – 23.09.2021	вик.
3	Формалізація вимог	25.09.2021 – 12.10.2021	вик.
4	Проектування програмної системи	21.10.2021 – 01.11.2021	вик.
5	Програмна реалізація та тестування	02.11.2021 – 21.11.2021	вик.
6	Оформлення пояснювальної записки та графічного матеріалу	22.11.2021 – 29.11.2021	вик.

Здобувач вищої освіти _____ *Д.Д. Гришакін*

Керівник роботи _____ *К.О. Писаренко*

АНОТАЦІЯ

Метою роботи є загальне зниження проведеного часу за іграми до нормальних показників минулих років та підвищення продуктивності в інших діяльностях, використовуючи програмну реалізацію веб-застосування, який допомогає шукати людей у радіусі гео-мітки та аналізуючи вподобання, визначає потенційно цікаві діяльності вдома та на вулиці. Технологіями розробки є мова програмування JavaScript, React як бібліотека для побудови SPA (Single Page Application), Node.js як сервер для обміну запитами між клієнтом та базою даних, Google Maps Geolocation API для визначення місцезнаходження та документо-орієнтована система керування базами даних MongoDB. Результатом роботи є виконана програмна реалізація веб-сервісу для контролю та асоціалізації користувачів.

Ключові слова: пошук новин, визначення місцезнаходження, аналіз, сортування новин, JavaScript, React, MongoDB.

ABSTRACT

The aim of the work is to reduce the overall time spent on games to normal levels of previous years and increase productivity in other activities, using software implementation of a web application that helps search for people within a geo-tag and analyze preferences, identify potentially interesting activities at home and outdoors. Development technologies include JavaScript programming language, React as a library for building SPA (Single Page Application), Node.js as a server for exchanging requests between client and database, Google Maps Geolocation API for location and document-oriented database management system MongoDB. The result is a software implementation of a web service for control and socialization of users.

Keywords: news search, location determination, analysis, news sorting, JavaScript, React, MongoDB.

ВСТУП

На сьогоднішній день, онлайн-ігри грають велику роль у житті багатьох людей. Станом на 2020 рік, помічен додатковий скачок популярності відео-ігор. Так у цей період пандемії продажі та час за іграми збільшився на 30% порівнюючи із минулими роками.

Тим часом, експерти вважають, що зростання популярності комп'ютерних ігор був помітний ще до пандемії і буде зберігатися досить довгий час. Більш того, на їхню думку, споживачі поступово відмовляться від звичних розваг на користь онлайн-інтертейменту.

В свою чергу, через надмірне використання онлайн-ігор, люди починають деградувати, відмовлятися від своїх звичних хобі, знижується їх загальна продуктивність та комунікативні здібності. Вони потрапляють у сильну зону комфорту, тому що, у відео-іграх все дуже просто, не потрібно виходити з дому, контактувати із іншими людьми.

Метою роботи є загальне зниження проведеного часу за іграми до нормальних показників минулих років та підвищення продуктивності в інших діяльностях, використовуючи програмну реалізацію веб-застосування, який допомогає шукати людей у радіусі гео-мітки та аналізуючи вподобання, визначає потенційно цікаві діяльності вдома та на вулиці. А також мотивує користувачів щоденними завданнями, за успішне проходження яких користувач отримує бали, які може витрати у онлайн-магазині на книжки, спортивну атрибутику та інше.

Основною задачею роботи є розробка веб-застосування у форматі клієнт-сервер, із використанням пошуку гео-позиції, за допомогою Google API, а також Steam API для формування статистики користувачів та декілька інших API, які наддадуть дані про книжки, фільми та спортивні заняття.

Таким чином, робота може поділитися на дві частини, які безпосередньо важливі один для одного, і не можуть працювати досконало поодинці. Веб-інтерфейс, він же клієнт, буде розроблений за допомогою бібліотеки React, що наддасть веб-сервісу надійності та швидкості. В свою чергу, веб-сервер буде

розроблений на базі Node js, який славиться своєю захищеністю та надійною роботою із будь-яким об’ємом даних.

Основний функціонал розроблюваного веб-застосування з урахуванням аналізу аналогів та сучасних засобів містить наступні можливості: формування пропозицій щодо діяльностей за вподобаннями користувача, аналіз вподобань користувачів та пошук за геопозицією користувача, визначення часу, проведеного за комп’ютером, його аналіз та визначення діяльності для відпочинку; проходження щоденних рівнів; отримання балів та покупки у магазині; формування формування списку користувачів; створення коментарів; редагування профілю; перегляд профілю інших користувачів; перегляд публікацій інших користувачів. Кожний користувач вказує інформацію щодо персональних даних.

Продукт стане важливим помічником для людей під час довгого проведення часу за іграми, швидко аналізуючи та шукаючи релевантні діяльності для відволікання.

Основним завданням перших трьох розділів є формування вимог на розробку веб-сервісу. Сюди входять: опис предметної області, обґрунтування розробки та виявлення актуальності, що дозволяє знайти аналоги системи та провести порівняльний аналіз та виявити особливості, недоліки та переваги. а також опис вимог та обмежень системи.

Четвертий розділ присвячений проектуванню та містить повний опис всіх модулів веб-сервісу, бази даних та всіх її сутностей, а також опис інтерфейсу користувача.

У п’ятому розділі зазначена програмна реалізація проекту, що включає в себе опис алгоритмів та структур даних, технологій розробки та особливості розроблюваної системи.

1 ПРОБЛЕМИ ІСНУЮЧИХ РІШЕНЬ

Сьогодні у світі існує досить багато сервісів для контролю свого часу за будь-якою справою. У комп’ютерів та смартфонів компанії Apple є вбудована функція контролю часу. Але вони в повній мірі не контролюють конкретно ігрову активність і не можуть радити змінити поточну діяльність та мають свої особливості, котрі необхідно розглянути перед розробкою програмного продукту.

Більшість схожих за тематикою веб-застосувань не мають функціоналу для мотивування користувача відволікатися від відео-ігор для інших активно-соціальних діяльностей, прочитання книг або статей, проведення часу з друзями та пошуку нових знайомств. Найбільш схожими за тематикою та функціоналом аналогами вважаються:

1) Qustodio – сервіс для батьківського контролю. У цієї програми є безкоштовна версія, якою можна користуватися без будь-яких обмежень за часом. З її допомогою можна встановлювати розклад і правила запуску для окремих додатків на ПК. Список функцій утиліти дозволяє вважати її однією з найбільш ефективних безкоштовних програм для контролю дій і блокування доступу в Мережу. Тим більше, що вона підтримує не тільки Windows і Mac OS, але ще iOS, Android і навіть Fire OS на електронних книгах марки Kindle.

Переваги:

- кросплатформеність;
- обширний функціонал;
- простий та зрозумілий інтерфейс;
- публічний профіль;

Недоліки:

- відсутня можливість підключення Steam, для контролю ігрового часу;
- відсутня можливість обміну повідомленнями між користувачами;
- безплатна версія має невеликий обсяг можливостей;
- відсутність коментарів;

2) Witigo Parental Filter – батьківський фільтр Witigo Parental Filter призначений для роботи практично на будь-якому пристрої на базі Windows, Mac OS, iOS, Linux і Android. Утиліта забезпечує контроль і блокування контенту в режимі реального часу.

Переваги:

- універсальність та адаптованість;
- кросплатформеність;
- простий інтерфейс;
- висока ефективність пошуку категорії контенту;

Недоліки:

- безплатна версія має невеликий обсяг можливостей;
- відсутня можливість підключення Steam, для контролю ігрового часу;
- відсутність комунікації між користувачами;
- відсутність пошуку людей за інтересами та за геолокацією;

3) Gameplay Time Tracker – безкоштовний інформаційний портал, який дозволяє слідкувати за ігровою статистикою.

Переваги:

- повна статистика про ігрові сеанси;
- особиста сторінка;
- весь фунціонал доступний у безкоштовній версії;
- автоматизований пошук ігор на комп’ютері;

Недоліки:

- для роботи потрібно витратити багато часу на встановлення та налаштування;
- незручний інтерфейс;
- відсутня кросплатформеність;
- відсутність геолокації;

4) Toggl – спрощена версія середньостатистичного трекера часу. Toggl забезпечує ключові функції аналогічних програм.

Переваги:

- зручний контроль часу;
- простий та зрозумілий інтерфейс;
- можливість створювати будь-які проекти для контролю часу;

Недоліки:

- платний функціонал;
- відсутня можливість додавати ігровий профіль;
- відсутня можливість аналізу уподобань користувача для пошуку діяльностей;
- відсутність коментарів;
- відсутність геолокації;

Отже, розроблюваний веб-сервіс матиме покращені функції сайтів-аналогів, а також додатковий функціонал, що в значній мірі допоможе знизити ігровий час та знайти нові хобі та знайомих. Результати аналізу аналогів представлені у табл. 1.1.

Таблиця 1.1 – Аналіз аналогів веб-сервісу

Критерій/ПЗ	Unforgotten	Qustodio	Witigo	Gameplay	Toggl
Веб-доступ	+	+	+	-	+
Desktop	-	-	+	+	+
Гео-локація	+	-	-	-	-
Редагування профілю	+	-	+	-	+
Створення постів та коментарів	+	-	-	+	-

Продовження таблиці 1.1.

Пошук за інтересами	+	+	-	-	-
Ігровий профіль	+	-	-	+	-
Контроль часу	+	+	+	+	+
Щоденні квести	+	-	-	-	-
Онлайн-магазин	+	-	-	-	+

Після вивчення існуючих систем-аналогів, було прийнято рішення створити програмний продукт, що дозволяє:

1) обмінюватися коментарями з іншими користувачами;

2) створювати ігровий профіль для контролю часу;

3) шукати заняття за геопозицією та інтересами;

4) шукати схожих людей на мапі за інтересами;

5) Пошук книг та фільмів за особистими категоріями користувача

6) Створювати пости в особистій стрічці профілю користувача

7) Перегляд профілів користувачів

8) Проходження квестів та цікавих завдань

9) Система балів, які можна витрачати у магазині на ексклюзивні товари

10) аналіз ігрової статистики;

11) можливістю формувати пропозиції щодо діяльностей відповідно до часу проведеного за грою;

2 РОЗРОБКА МОДЕЛІ КВЕСТІВ ТА БАЛІВ

2.1 Виявлення проблеми

Станом на сьогоднішній день, значно зросла кількість часу, проведеного за іграми, що призводить до проблем із емоційною складовою, а також із фізичним станом людей. При постійній грі зростає ризик підвищення залежності, що може заважати потенційному росту продуктивності людини у інших справах.

Особи, які часто грають у комп'ютерні ігри сильніше піддаються фізичним та психічним розладам. Ті, хто мають симптоми комп'ютерної залежності, дуже швидко перевтомлюються, часто мають болі в ногах та суглобах пальців рук. Також можна відзначити погіршення зору, підвищену збудливість та безсоння.

Найголовніша небезпека, представлена комп'ютерними іграми - виникнення ігрової залежності. Це справжнє відхилення психіки, що потребує допомоги лікаря. Постійний контакт із комп'ютером змушує людини захиститися від навколишнього світу та людей. Особи, які часто грають у комп'ютерні ігри сильніше піддаються фізичним та психічним розладів. Ті, хто мають симптоми комп'ютерної залежності, дуже швидко перевтомлюються, часто мають болі в ногах та суглоби пальців рук. Також можна відзначити погіршення зору, підвищену збудливість та безсоння. Дослідження показують, що у підлітків, що захоплюються комп'ютером, отримані нижчі показники інтересів у гуманітарній галузі (Мистецтво, література). Крім того, вони більше дивляться відео та телевізор і, як це не парадоксально, більше займаються спортом.

За даними проведеного опитування, можна зробити висновок про те, що школярі витрачають набагато більше кількість часу на комп'ютерні ігри, ніж студенти. За отриманими даними, можна дійти висновку про те, що студенти здебільшого обирають живе спілкування, також майже половина опитаних воліє суміщене віртуальне та живе спілкування. Школярі, як молодше покоління, вже зробили вибір і на користь лише віртуального спілкування. Поєднане віртуальне та живе спілкування та суто живе спілкування знаходяться на рівних рівнях у школярів.

2.2 Основи квестів

Для мотивації гравців у застосуванні буде присутня можливість проходження квестів різни рівнів та категорій. Це допоможе підвищити зацікавленість користувачів у кінцевій меті.

Квест - це пригодницька гра, яка дозволяє гравцю поринути у світ цікавих завдань та головоломок. ключову роль в ігровому процесі грає вирішення головоломок та завдань, які вимагають від гравця розумових зусиль. Такі характерні для інших жанрів комп'ютерних ігор елементи, як бої, економічне планування та завдання, що вимагають від гравця швидкості реакції та швидких дій у відповідь, у квестах зведені до мінімуму або зовсім відсутні. Ігри, що поєднують у собі характерні ознаки квестів та жанру action, виділяють в окремий жанр – action-adventure.

У квестах-головоломках на чільне місце ставиться вирішення будь-яких логічних завдань, загадок, наприклад, у вигляді різних механізмів, доступних для обстеження гравцем; при цьому число загадок дуже велике, а оповідання може бути схематичним або зовсім відсутнім.

2.3 Створення квестів у застосуванні

Користувач на сторінці налаштувань свого профілю має можливість задати точні категорії квестів, які він бажає проходити. На початку використання застосування, у користувача буде декілька основних категорій: кіно або серіали та книжки. Він зможе змінити категорії, коли підвищить свій ігровий рівень. Користувача може підняти ігровий рівень проходженням квестів, написанням постів та коментарів. Кількість балів які можна отримати, зростає із кожним рівнем. Наприклад, за перший рівень гравець отримує 100 балів, а вже за 10 пройдений рівень 1000.

Завдання поділяються на категорії: кіно та серіали, книги, спортивні заняття, соціальна діяльність та інші. Таким чином, кожен користувач знайде свою

улюблену категорію, що може підвищити рівень його зацікавленості та бажання пройти квест.

Кожне завдання буде відповідати своїй тематиці. Наприклад для категорії книжок завдання може звучати таким чином: “Прочитати 230 сторінок “Гаррі Поттера”” або “Знайти у романі “Природа всіх речей” 20 поетичних обертів”.

Під час виконання завдань система буде зчитувати прогрес користувача та відображати його. Завдання вважається виконаним, якщо прогрес дійшов до ста відсотків. Тоді активується кнопка “Виконано”, яку гравець натискає та отримує бали.

Для отримання результатів ефективності даного методу, було проведено соціальне опитування, а також зібрана статистика переходів під час використання застосуванням та його аналогами (табл. 2.1).

Таблиця 2.1 – Соціальне опитування

Критерій	Поточне застосування	Аналоги
Бажання закінчити гру	10	7
Мотивація на отримання балів	10	5
Нестандартність	7	8
Зацікавленість	9	5
Важкість рівніх квестів	8	7
Вибір категорій	10	6
Бали для друзів	8	3
Онлайн-покупки за бали	7	4
Сумісні квести	10	6
Тематичні квести	9	7

Таким чином, можна зробити висновок, що система квестів та балів більше спонукає гравців переключатися з гри на інші сфери діяльності, а також підвищувати кількість знайомих та зустрітися із ними не тільки онлайн. Квести надають змогу для сумісного проходження, для покращення стосунків між людьми.

Бали можна використовувати в якості валюти під час використання застосування. Їх можна передавати іншим користувачам, в якості подарунків або під постами, таким чином виказуючи свою враженність.

Сумуючи отримані результати, можна виділити особливу активність людей під час проходження квестів. Елемент змагань між користувачами ще більше підігрів їх інтерес. Також статистика використання застосування підтверджує зниження кількості часу, проведеного за іграми, під час введення квестів. Також користувачі мали підвищення соціальної активності, такої як створення постів та їх коментування. Багато користувачів використовували гео-локацію для пошуку знайомих для сумісного проходження спортивних квестів на вулиці.

Квести розвивають уважність, тому що в іграх даного жанру гравець повинен шукати рішення до завдань. Стратегії допомагають людині грамотно використовувати свій бюджет, що розвивають навички підприємництва. Ще є складні командні ігри-головоломки, де групі людей доводиться вирішувати дуже складні завдання, використовуючи всі їх здібності та синхронізуючи дії.

Таким чином, було виявлено основний варіант використання веб-застосування для зниження середньої кількості годин, що проведено у іграх за окремий період, що надає змогу виділити основний вектор розвитку програмного забезпечення та правильного розробку веб-інтерфейсу користувача.

3 ФОРМУВАННЯ ВИМОГ НА РОЗРОБКУ

3.1 Опис предметної області

Виходячи із предметної області система розробляється з урахуванням того, що користувач реєструється у системі та потім проходить авторизацію, для отримання доступу до всього функціоналу веб-сервісу. Авторизований користувач отримує можливість завантажити ігровий профіль Стім та переглядати свою ігрову статистику, редагувати свій профіль, вказуючи вподобані категорії для формування списку цікавих книжок або фільмів, визначення геолокації та пошуку діяльностей поблизу, а також перегляду постів інших користувачів, коментування та спілкування із іншими користувачами, проходження квестів та рівнів для отримання балів, які можна витратити у магазині веб-застосування на покупку книжок або фільмів або спортивних атрибутів. Кількість публікацій, що може створити користувач – необмежено. Користувачі мають змогу скаржитися на публікації, що призводить к перевіркам адміністраторами системи.

Користувач вказує персональні дані (ПІБ, дата народження, фото), своє місцезнаходження та пароль, який проходить декілька етапів шифрування. Завдяки цьому користувач може слідкувати за новинами тільки свого регіону.

Кожна публікація, яку створив користувач, включає у себе: тему, основний текст, декілька посилань на інші джерела або сторінки, дату створення, розмір не має переважати 1500 символів.

3.2 Актуальність розробки

Експерти та учасники галузі відзначають розвиток низки сильних тенденцій, які визначають розвиток глобальної ігрової промисловості. Так, велику роль продовжують відігравати нові технології (віртуальні, мобільні, хмарні та ін.), спостерігається злиття віртуальних та фізичних середовищ, соціалізація ігор тощо.

На рисунку 3.1 зображена шкала відносного середнього часу проведеного за іграми.

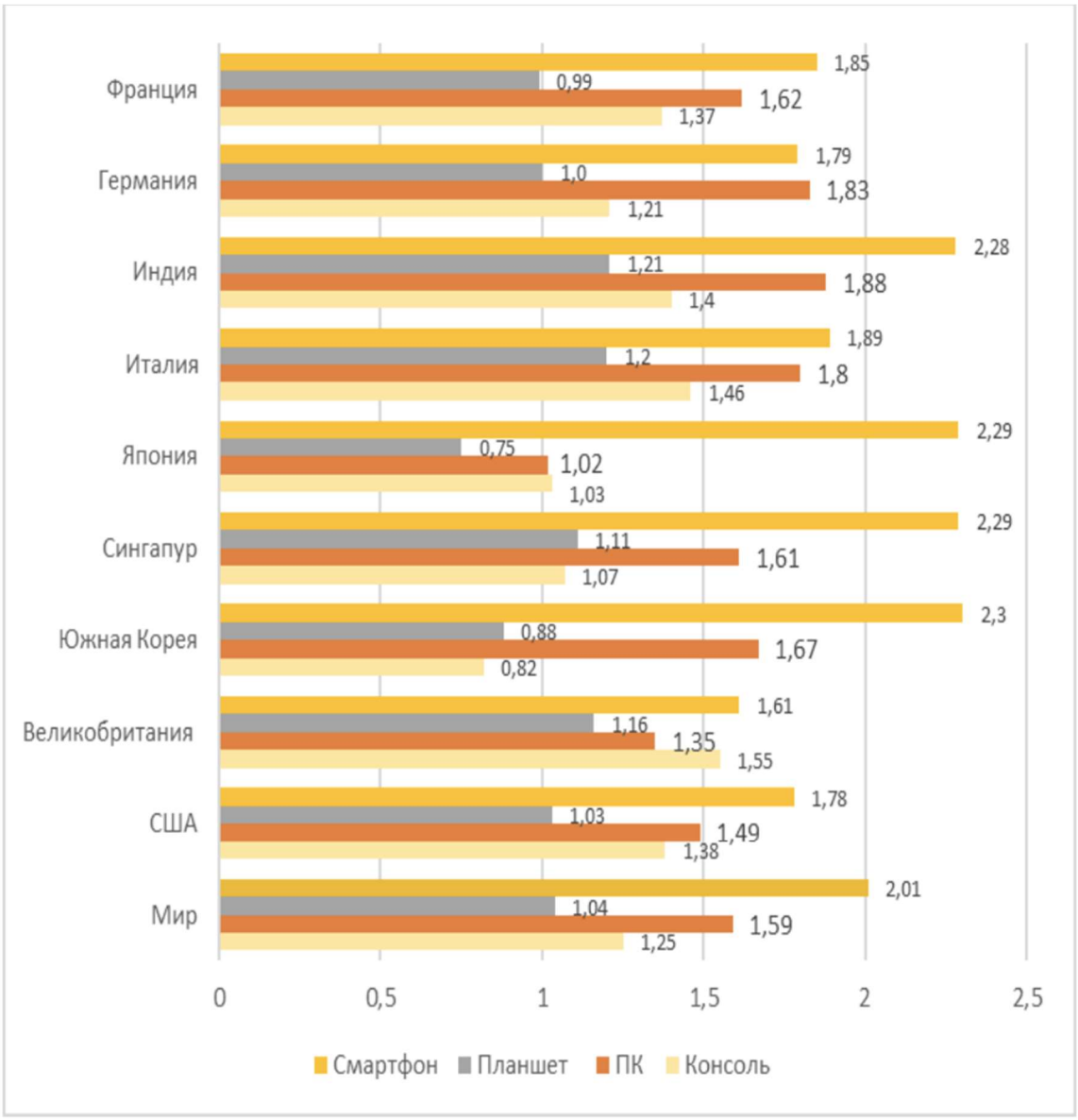


Рисунок 3.1 - Шкала відносного середнього часу проведеного за іграми

На даний момент мобільні ігри - ігрові програми для смартфонів і планшетів – найшвидше зростаючий сегмент ігрового ринку. Це пояснюється доступністю мобільних пристроїв та мобільного трафіку. Дані Limelight підтверджують це: смартфони найпопулярніші ігровим пристроєм у всьому світі.

На даний момент існує невелика кількість веб-застосувань, які дозволяють завантажувати ігрову статистику та мониторити її, а весь загальний функціонал таких застосувань досить невеличкий та потребує значного покращення.

Кожен такий веб-сервіс має недоліки і переваги. Тому перед розробкою потрібно провести аналіз існуючих рішень програм-аналогів.

Існуючі веб-застосування мають обмеження в спілкуванні та соціалізації з іншими користувачами. Більша кількість взагалі не мають функції місцезнаходження, а ті, що мають, використовують не всі її переваги. Основною перевагою є те, що користувач має змогу за декілька секунд отримати данні про своє місцезнаходження та сформувати список активних діяльностей в радіусі місцезнаходження. Користувача мотивують замість гри зайнятися іншими діяльностями та отримати за це винагороду у вигляді балів, а найкращий користувач за місяць має змогу отримати великий подарунок, який залежить від партнерів. Користувач отримує велику кількість книг та фільмів в безкоштовному доступі, які унікально доступні для даного веб-застосування.

Веб-сервіс буде створений с урахуванням інших новинних сайтів. На базі їх інформації буде формуватися основний контент веб-сервісу. Користувачі зможуть швидко аналізувати новини та формувати свою думку про джерела та їх бачення новин.

Такий різновид веб-застосування відмінно підійде для людей, які під час карантину звикли проводити багато часу за комп’ютером або смартфоном, граючи у відео-ігри. Веб-застосування надає змогу об’єднати людей із схожими інтересами по іграм та надає їм можливість для сумісного пошуку нових можливостей, діяльностей або хобі. Вони зможуть створювати персональні пости, щоб зацікавлювати один одного та коментувати їх і відмічати місцезнаходження у постах для пошуку нових знайомих поблизу.

Користувачі матимуть змогу обмінюватися повідомленнями, відмічатися на мапі для зустрічей та формувати спільні теми для перегляду фільмів або книжок.

3.3 Глосарій

Глосарій – це словник значень предметної області для налагодження спілкування між замовником та розробником сервісу.

Комп'ютерна гра — це гра, в яку користувач грає за допомогою комп'ютерного інтерфейсу, дані на який поступають з відео-пристрою.

Соціалізація - це процес активних діяльностей користувача, під час використання веб-застосування.

Щоденний квест - це змагання, під час якого користувач послідовно виконує завдання, щоб отримати бали, які можна витратити для придбання продуктів у локальному магазині.

Фільтр — швидкий і простий спосіб знайти підмножину даних.

Категорія (класифікація) – система розподілення об'єктів по групах відповідно до наперед визначених ознак.

Геолокація – визначення реального географічного розташування електронного пристрою підключеного до Інтернету.

Одно-сторінковий застосунок (інтерфейс) – це веб-сервіс, який взаємодіє з веб-браузером, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера, замість методу браузера, який завантажує сторінку знову.

3.4 Специфікація вимог до системи

Функціональні вимоги описують внутрішню функціональність системи, її поведінку: обчислення даних, обмін та керування даними, вивчення даних та інших конкретних функцій, які система повинна робити. Як правило, функціональні вимоги визначають, що система повинна робити [1].

Функціональні вимоги представляються у якості варіантів використання (use-case). Варіант використання, прецедент – у розробці програмного забезпечення та системному проектуванні це опис поведінки системи та як вона відповідає на зовнішні запити [2].

Use-case діаграма, що описує функціональні вимоги до веб-сервісу зображена на рис. 3.2.

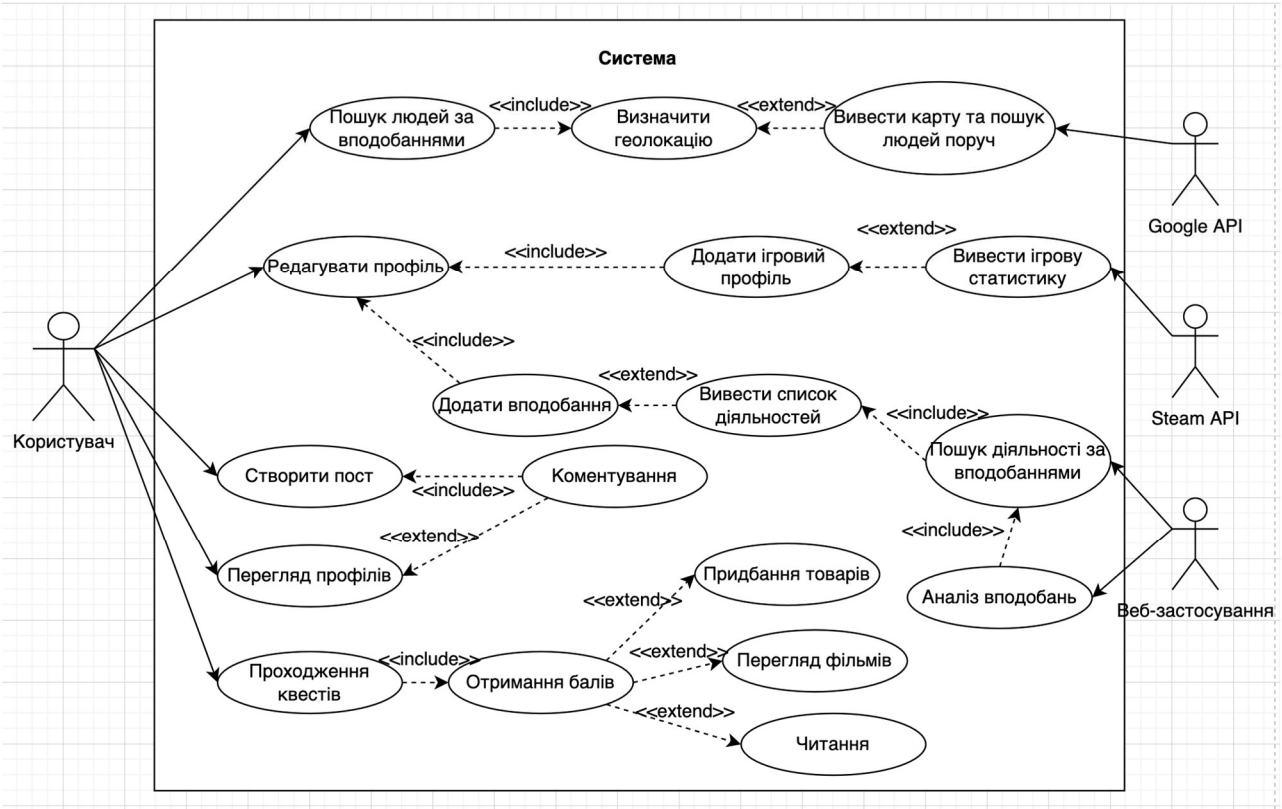


Рисунок 3.2 – Use-case діаграма веб-сервісу

Сценарії використання:

1) Редагування профілю

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач зареєстрований у системі та знаходиться на сторінці профілю.

Тригер: Користувач натискає кнопку «Редагувати профіль».

Основний сценарій:

1 Користувач відкриває форму для заповнення даних.

2 Система перевіряє запит та перенаправляє користувача.

3 Система запитує дані. Користувач вводить нові дані та зберігає.

4 Система підтверджує дії та зберігає.

Альтернативний сценарій:

1. Користувач не ввів дані.

    1. Система виводить повідомлення про те, що користувач не ввів дані. Перехід до П.3.

2) Створення ігрового профілю

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач зареєстрований у системі.

Тригер: Користувач переходить у свій профіль та натискає «Підключити ігрову статистику».

Основний сценарій:

1. Система переводить користувача на сторінку API.
2. Користувач підключає аккаунт.
3. Система обробляє та виводить статистику.
4. Користувач переглядає.

3) Визначити місце розташування

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач зареєстрований у системі.

Тригер: Користувач переходить на сторінку «Геолокація».

Основний сценарій:

1. Система виводить сторінку геолокації.
2. Користувач натискає кнопку «Визначити місцезнаходження»
3. Система запитує у користувача право на сканування його місцезнаходження. Користувач підтверджує.
4. Система починає сканування.
5. Система виводить повідомлення про отримане місце користувача та карту.
6. Користувач підтверджує отримані дані.
7. Система зберігає.

Альтернативний сценарій:

3. Користувач не підтверджує сканування.

1. Система повідомляє, що без дозволу користувача не може сканувати місцезнаходження.

2. Кінець сценарію.

4. Система не може провести сканування.

1. Система повідомляє користувача про помилку.

2. Кінець сценарію.

5.1. Користувач не підтверджує дані.

3. Система виводить карту та пошуковий рядок.

4. Користувач обирає місцезнаходження.

5. Система зберігає.

6. Кінець сценарію.

4) Вподобання

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач переходить у налаштування профілю.

Основний сценарій:

1. Система виводить налаштування вподобань користувача.
2. Користувач знаходить потрібні інтереси та додає їх.
3. Система перевіряє правильність.
4. Користувач схвалює.
5. Система зберігає та виводить можливі діяльності за вподобаннями.
6. Користувач обирає.

5) Фільтрація вподобань за категоріями

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач переходить на сторінку «Інтереси»

Основний сценарій:

1. Система виводить усі доступні категорії
2. Користувач обирає. Система оброблює та зберігає.
3. Система виводить діяльності згідно з обраною категорією.
4. Користувач переглядає.

6) Люди поруч

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі та визначив своє місцезнаходження [3].

Тригер: Користувач переходить на сторінку «Люди поруч»

Основний сценарій:

5. Система виводить людей згідно із даними геолокації.
6. Користувач переглядає.

7) Створити пост

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач натискає кнопку «Створити пост» у профілі.

Основний сценарій:

1. Система надає форму для створення посту.
2. Користувач заповнює.
3. Система зберігає та виводить.

Альтернативний сценарій:

1. Користувач не заповнив форму.
2. Система повідомляє, що потрібно заповнити форму. Перехід до П.1.

8) Створити коментар

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач натискає кнопку «Додати коментар» у профілі.

Основний сценарій:

1. Система надає форму для коментаря.
2. Користувач заповнює.
3. Система зберігає та виводить.

Альтернативний сценарій:

1. Користувач не заповнив форму.
2. Система повідомляє, що потрібно заповнити форму. Перехід до П.1.

9) Перегляд профілів

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач натискає кнопку «Перейти у профіль» на сторінці зі списком користувачів.

Основний сценарій:

1. Система надає профіль обраного користувача.
2. Користувач переглядає.

10) Проходження квестів

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач авторизований у системі.

Тригер: Користувач натискає на картку квесту у своєму профілі.

Основний сценарій:

1. Система надає повну інформацію про квест.
2. Користувач ознайомлюється та виконує завдання.
3. Система перевіряє прогрес користувача та виводить інформацію про перемогу.
4. Система нараховує бали.

Альтернативний сценарій:

1. Користувач не виконує завдання.

2. Система повідомляє, про поразку у щоденному квесті. Перехід до профілю користувача.

11) Реєстрація

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач ще не зареєстрований у системі.

Тригер: Користувач перейшов на сторінку реєстрації.

Основний сценарій:

1. Користувач починає реєстрацію. Система надає форму для заповнення даних.

2. Система запитує дані для реєстрації.

3. Користувач заповнює дані реєстрації. Система оброблює та підтверджує.

4. Перехід до сценарію Authorization.

Альтернативний сценарій:

3 Користувач ввів неправильні дані.

1. Система повідомляє про неправильність даних. Перехід до П.2.

3 Почтова скринька або логін вже існують.

2. Система повідомляє, що такі дані вже існують. Перехід до П.2.

10) Авторизація

Область дії: Система.

Учасники: Користувач.

Передумова: Користувач успішно авторизований у системі.

Тригер: Користувач перейшов на сторінку авторизації.

Основний сценарій:

1. Користувач починає авторизацію. Система надає форму для заповнення.

2. Система запитує дані для заповнення.

3. Користувач вводить.

4. Система оброблює та перевіряє.

5. Система підтверджує та переадресує користувача на сторінку його профілю.

Альтернативний сценарій:

4.1 Користувач ввів неправильну поштову скриньку.

1. Система повідомляє. Перехід до П.2.

4.2 Користувач ввів неправильний пароль.

1. Система повідомляє. Перехід до П.2.

Нефункціональні вимоги визначають критерії оцінки якості роботи програмного забезпечення, визначають якою повинна бути система. Вимоги характеризують принципи взаємодії із середовищами або іншими системами, а також визначають показники якості роботи і захисту даних.

Операційні вимоги:

1) Зручність використання.

1) Користувач може створити додати нову гру менше, ніж за хвилину.
2) Користувач може відредагувати профіль менше, за 30 секунд.
3) Користувач може визначити своє місцезнаходження менше, ніж за 30 секунд.

2) Керування помилками.

1) Веб-сервіс повинен реагувати на усі помилки та контролювати їх для стабільної роботи.
2) Система визначає помилки у 90% випадків.
3) При помилках під час вводу даних, система повідомляє користувача менш, ніж за 2 секунди.
4) Сервер обробляє помилки менш, ніж за 10 секунд.

3) Надійність.

1) У разі помилки система продовжує коректну роботу у 80% випадків.
2) Ймовірність безвідмовної роботи – 0.85.
3) У разі помилки при відправці даних на сервер, дані будуть не пошкодяться у 90% випадків.

4) Супровід.

1) Виправлення дефектів програмного забезпечення менше, ніж за тиждень.

2) Оптимізація нового функціоналу менше, ніж за 2 тижні.

3) Модифікація програмного забезпечення менше, ніж за 2 тижні.

Обмеження проектування:

1) Вимоги до програмного забезпечення back-end частини:

1) Підтримка Node.js 3.0 або вище.

2) Підтримка бази даних MongoDB.

2) Вимоги до програмного забезпечення клієнтської частини:

1) Браузери: Google Chrome, Mozilla із ввімкненою опцією JavaScript та Геолокації;

3) Вимоги до швидкості інтернету:

1) Мінімальна швидкість 2 Мбіт/сек.

2) Рекомендована швидкість 15 Мбіт/сек.

4) Апаратні обмеження (мінімальні):

1) Розширення екрану 800*600.

2) Процесор з частотою >= 1,3 Ггц.

3) Оперативна пам’ять >= 2 Гбайт.

5) Юридичні обмеження:

1) Узгодження на збір та обробку персональних даних. Закон України «Про захист персональних даних» регулює правові відносини, пов'язані із захистом і обробкою персональних даних, і спрямований на захист основних прав і свобод людини і громадянина. Відповідно до частини 5 та частини 6 статті 6 Закону обробка персональних даних здійснюється для конкретних і законних цілей, визначених за згодою суб'єкта персональних даних, або у випадках, передбачених законами України, в порядку, встановленому законодавством.

4 ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ

4.1 Розробка моделей

Перщ за все потрібно виконати виділення сутностей та організація зв’язків між таблицями БД.

На рисунку показано модулі та описано зв’язок між ними для кожної з моделей визначено атрибути, що забезпечують взаємодії між сутностями [4].

На етапі проектування були виділені шість моделей веб-сервісу: User, Game, GameProfile, Comment, Image, Category. Структура моделей представлена на рис. 4.1.

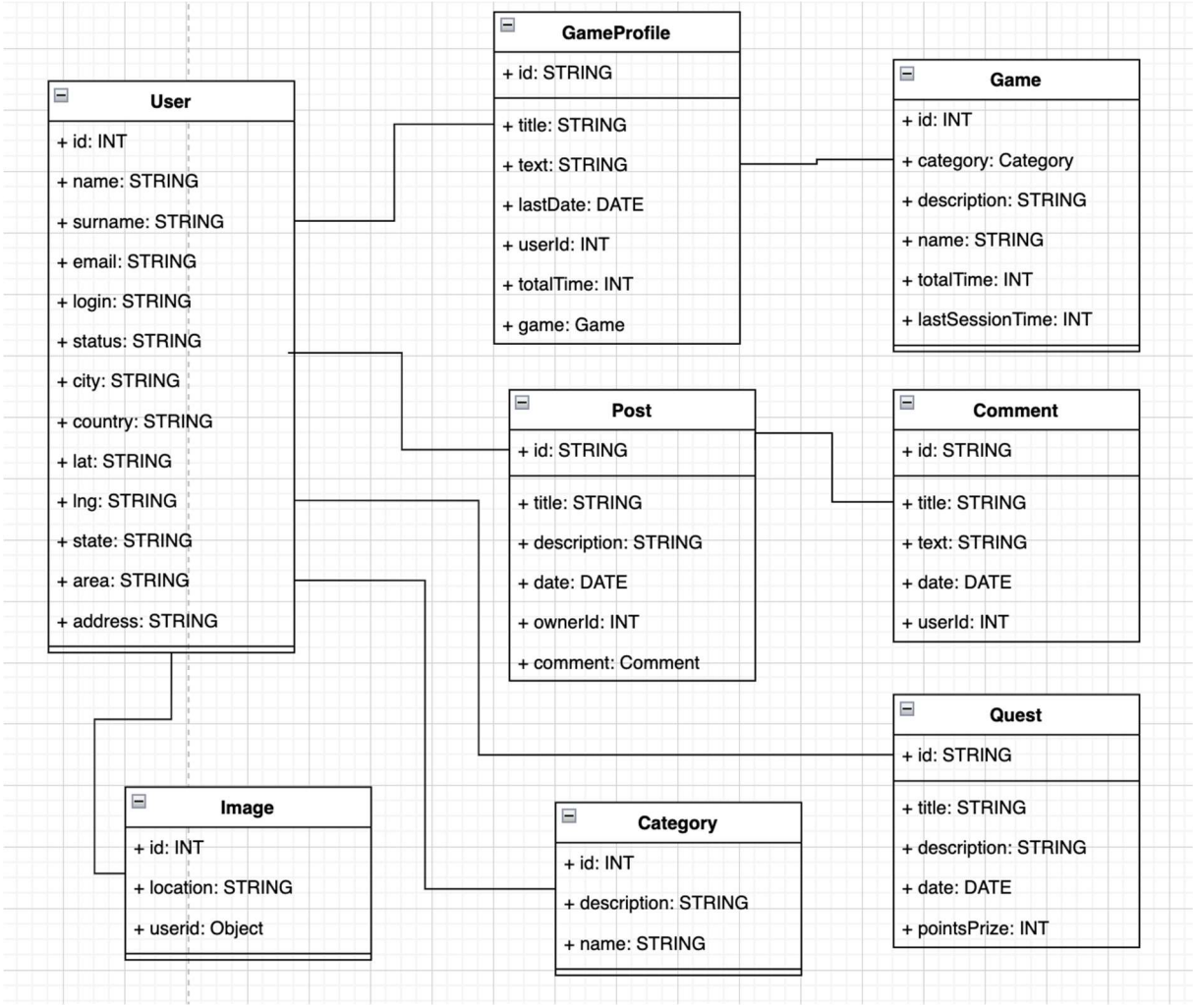


Рисунок 4.1 – Моделі веб-сервісу та їх взаємозв’язки

Модель User містить всю потрібну інформацію про користувача, для використання всіх можливих функцій.

Модель Comment представляє собою набір даних про створені користувачами коментарі.

Модель Game – це набір даних про особисті публікації користувачів, які можуть бачити усі інші користувачі системи, які відвідують профіль.

Модель GameProfile – це набір даних про ігровий профіль користувача , які можуть бачити усі інші користувачі системи, які відвідують профіль та ігровий профіль.

Модель Category містить повний опис категорій новин, що дозволяє швидко ознайомити користувача із всіма представленими категоріями.

Модель Image містить в собі дані про рисунки веб-сервісу.

Модель Quest містить всю інформацію про можливі квести користувачів.

Модель Post містить інформацію про створені коритсувачами пости.

Виконуючи побудову ER-моделі системи були виділені наступні сутності та зв’язки:

1) User (id, name, surname, password, email, phone, login, avatar, status, lat, leg, state, country, city, area, address);

2) GameProfile (id, title, text, lastDate, userId, totalTime, game);

3) Comment(id, , userId, title, text, date);

4) Game (id, category, description, name, totalTime, lastSessionTime);

5) Image (id, location, userId);

6) Category (id, description, title, name);

7) Post (id, title, description, date, ownerId, Comment_id);

8) Quest (id, title, description, date, pointsPrize);

4.2 Визначення модулів системи

Загальна архітектура веб-сервісу із виділенням модулів представлена на рис. 4.2. Для представлення, вона поділена на три частини.

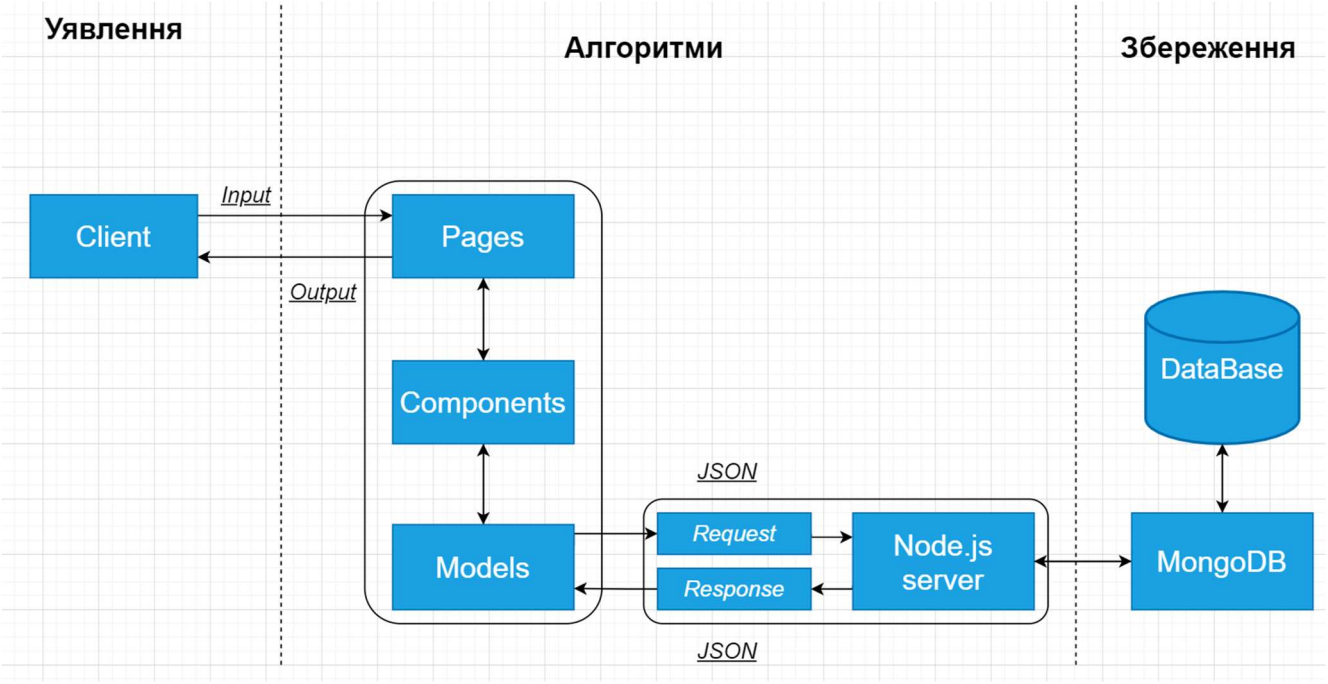


Рисунок 4.2 – Архітектура веб-сервісу

Клієнтська частина розроблюваного веб-сервісу складається з наступних модулів:

1) Pages – це модуль-фасад веб-сервісу, який відповідає за показ інформації для користувача. Тобто, саме це бачить користувач, коли відвідує сайт.

2) Components – це модуль-будівник, який відповідає за побудову сторінок (Pages). Кожна сторінка складається із декількох компонентів, це дозволяє повторно використовувати деякі з них. Кожний компонент відповідає за окремий функціонал веб-сервісу. Повторне використання компонентів дозволяє підвищити продуктивність та знизити витрати інтернет-трафіку.

3) Models – це модуль-сховище, який відповідає за зберігання моделей даних для сутностей системи. А також цей модуль відповідає за передачу та прийом даних.

Серверна частина веб-сервісу складається з модулів:

1) Request/Response – це модуль, який відповідає за прийом та передачу між частинами веб-сервісу ajax запитів, таких як: GET, POST, PUT або DELETE [5].

2) Node.js Server – веб-сервер, який приймає HTTP запити та відповідає клієнту, шифруючи їх для захисту персональних даних. Також сервер виступає сполучною ланкою між клієнтом та базою даних.

3) MongoDB – це документо-орієнтована система керування базами даних, яка дозволяє виконувати запити до БД, та отримувати файли у форматі JSON.

4) Database – цей модуль відповідає за сховище даних веб-сервісу.

4.3 Вимоги до модулів системи

Для стабільної роботи веб-сервісу, потрібне постійне підключення до сервісу керування базами даних, тому інтернет з’єднання повинно бути швидким та стабільним.

Веб-сервер повинен підтримувати підключення до бази даних, та контролювати помилки у системі, щоб зменшити ризик критичної помилки та втрати даних користувача.

Основні три модулі клієнтської частини тісно взаємозв’язані. Якщо виникне помилка в одному з них, перестануть працювати два інші. Саме тому всі модулі повинні вміти реагувати на помилки будь-якого характеру, відповідно до функціоналу, та протестовані на працездатність.

4.4 Структура бази даних

На рис. 4.3 наведено використовувані таблиці даних веб-сервісу [6].

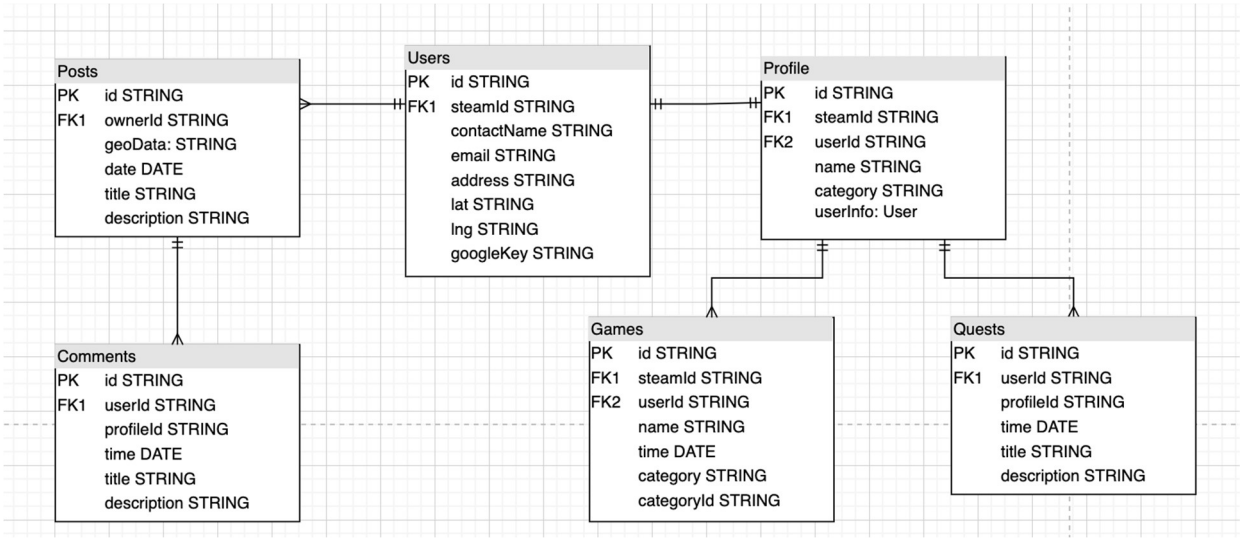


Рисунок 4.3 – Структура даних веб-сервісу

Таблиця Posts призначена для опису персональних постів, які створюють користувачі і складається з:

1) id – унікальний ідентифікатор.

2) ownerId - унікальний ідентифікатор користувача, який створив пос

3) geoData - об’єкт даних про гео-мітку, прикріплену до поста.

4) date - дата створення поста.

5) title - назва поста.

6) description - основний текст поста.

Таблиця Profile призначена для опису персональних даний користувача, і складається з:

1) id – унікальний ідентифікатор.

2) steamId - унікальний ідентифікатор акаунта Стім.

3) userId - унікальний ідентифікатор користувача.

4) name - ігрове ім’я користувача.

5) category - обрані категорії для фільтрації

6) userInfo - повна інформація про користувача

Таблиця Quest призначена для опису персональних даний користувача, і складається з:

1) id – унікальний ідентифікатор.

2) userId - унікальний ідентифікатор користувача.

3) profileId - унікальний ідентифікатор ігрового профілю.

4) time - час на проходження квесту.

5) pointsPrize - кількість балів за проходження квесту.

6) title - назва квесту.

7) description - опис квесту

Таблиця Games призначена для опису інформації про ігри користувача і складається з:

1) id – унікальний ідентифікатор.

2) steamId - ідентифікатор Steam акаунта.

3) userId - ідентифікатор користувача.

4) name – назва.

5) textarea – основний текст.

6) geolocation – місцезнаходження.

7) time – дата створення.

8) categoryId - ідентифікатор категорії.

9) category - назва категорії.

Таблиця Users призначена для повного опису користувачів веб-сервісу і містить:

1) id – унікальний ідентифікатор.

2) contactName – ФІО користувача.

3) email – почтова скринька користувача.

4) lat – широта.

5) lng – довгота.

6) adress - адреса користувача

7) googleKey – унікальний ключ Гугл.

8) avatar – персональна фотографія користувача.

Таблиця Comments призначена для зберігання даних про коментарі користувачів і містить:

1) id – унікальний ідентифікатор.

2) login – логін користувача.

3) userId – ідентифікатор користувача.

4) title - заголовок коментаря.

5) description – текст коментаря.

6) date – дата створення коментаря.

4.5 Проектування програмних класів

Діаграма класів – це статична діаграма для представлення структури моделі, що відображає декларативні елементи. Ця діаграма показує класи, інтерфейси та об’єкти, а також їх відносини [7].

Клас – це абстрактна конструкція, для групування пов’язаних змінних та функцій, що мають атрибути та операції.

У ході проектування було визначено основні програмні класи веб-сервісу та їх взаємозв’язки. Було детально розібрані можливості систем та аналогів для більш повного представлення основного функціоналу системи та побудови її структури. Для реалізації квестів використовуються класи Quest та Profile, для збору даних про проходження квестів та прогрес користувача. Для роботи із користувачами створено класи Profile та Users, які містять в собі методи для обробки запитів користувача. На діаграмі класів відображені всі компоненти, та підкомпоненти, що використовуються у системі (рис. 4.4).

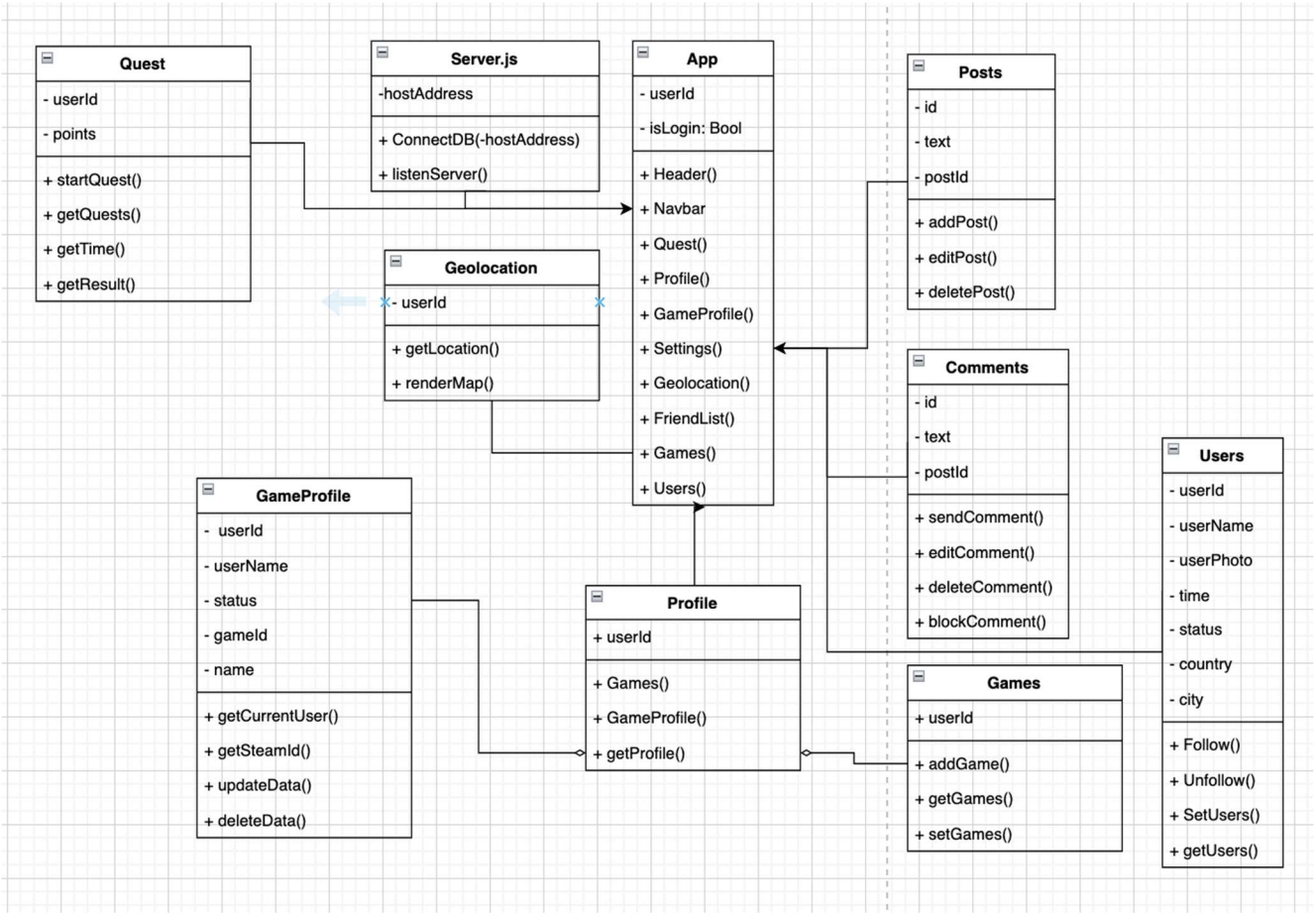


Рисунок 4.4 – Діаграма програмних класів

Клас App представляє собою основний клас, який об'єднує всі інші, та будує сторінки, які потім бачить користувач. Класи Dialogs, MyPosts, Profile, Login, News, Portals, Users йдуть наступними та відповідають за сторінки відповідно з назв. Опис класів наведений у табл. 4.1.

Таблиця 4.1 – Опис програмних класів

Назва класу	Опис класу
Users	Відповідає за отримання та зберігання даних про користувачів. Всі методи та функції направлені на роботу із користувальницькими даними.
Header	Відповідає за дії у шапці сайту, зокрема вихід із системи та перехід до профілю

Продовження таблиці 4.1.

Navbar	Контролює переходи між сторінками
Profile	Відповідає за формування профілю користувача, поєднуючи персональні дані та особисті коментарі
GameProfile	Відповідає за показ детальної інформації про ігрову статистику користувача
Games	Формує стрічку особистих ігор у сервісі, відповідає за методи створення, оновлення та видалення ігор
GameInfo	Містить детальну інформацію про кожну гру
Comment	Відповідає за методи контролю коментарів, та містить повну інформацію про кожний коментар
Login	Виконує авторизацію користувача, відповідає за методи для авторизації та перевірки даних
Quest	Відповідає за завантаження інформації про квести та методи для прохлдження квестів
Posts	Містить детальну інформацію про пости користувачів, та відповідає за обробку запитіва користувача по роботі із постами.
Geolocation	Клас, який відповідає за завантаження мапи та отримання гео-позиції користувача.

Кожний клас контролює методи та функції веб-сервісу для стабільної роботи, а також отримують, зберігають та використовують дані із БД. Для передачі даних між класами використовуються props (параметри), а також безпосереднє

підключення до редукторів бібліотеки Redux, які дозволяють отримувати доступ до даних напряму.

.

4.6 Діаграми активності

Для візуального представлення діяльності використовується діаграма активності. Дія є фундаментальною одиницею визначення поведінки. Вона перетворює множину вхідних сигналів у множину вихідних сигналів.

Блок-схема - це схематичне уявлення процесу, системи чи комп'ютерного алгоритму. Блок-схеми часто застосовуються у різних сферах діяльності, щоб документувати, вивчати, планувати, удосконалювати та пояснювати складні процеси за допомогою простих логічних діаграм. Для побудови блок-схем застосовуються прямокутники, овали, ромби та деякі інші фігури (для позначення конкретних операцій), а також сполучні стрілки, які вказують послідовність кроків або напрямок процесу. Блок-схеми варіюються від нехитрих, намальованих вручну до докладних, складених на комп'ютері діаграм з безліччю кроків та процесів.

Якщо врахувати всі можливі варіації, блок-схеми можна визнати одним із найпоширеніших видів схем у всьому світі. Вони широко використовуються в різних сферах як технічного, так і нетехнічного спрямування. Іноді блок-схеми отримують більш вузькоспеціальні назви, наприклад схема процесу, схема робочого процесу, функціональна блок-схема, моделювання бізнес-процесів, модель і нотація бізнес-процесів (BPMN) або схема технологічного процесу (PFD). Вони тісно пов'язані з іншими поширеними видами схем, такими як діаграми DFD та діаграми активності уніфікованою мовою моделювання (UML).

На рисунку 4.5 представлена діаграма діяльності «Find users» із використанням бібліотек Redux та Axios:

Рисунок 4.5 – Блок-схема для функції «Отримати користувачів»

Весь процес пошуку та виводу даних займає декілька секунд і більше не потребується, поки користувач не перезапустить сторінку. Цей принцип дозволяє користувачу швидко переміщатися між сторінками без перезапуску.

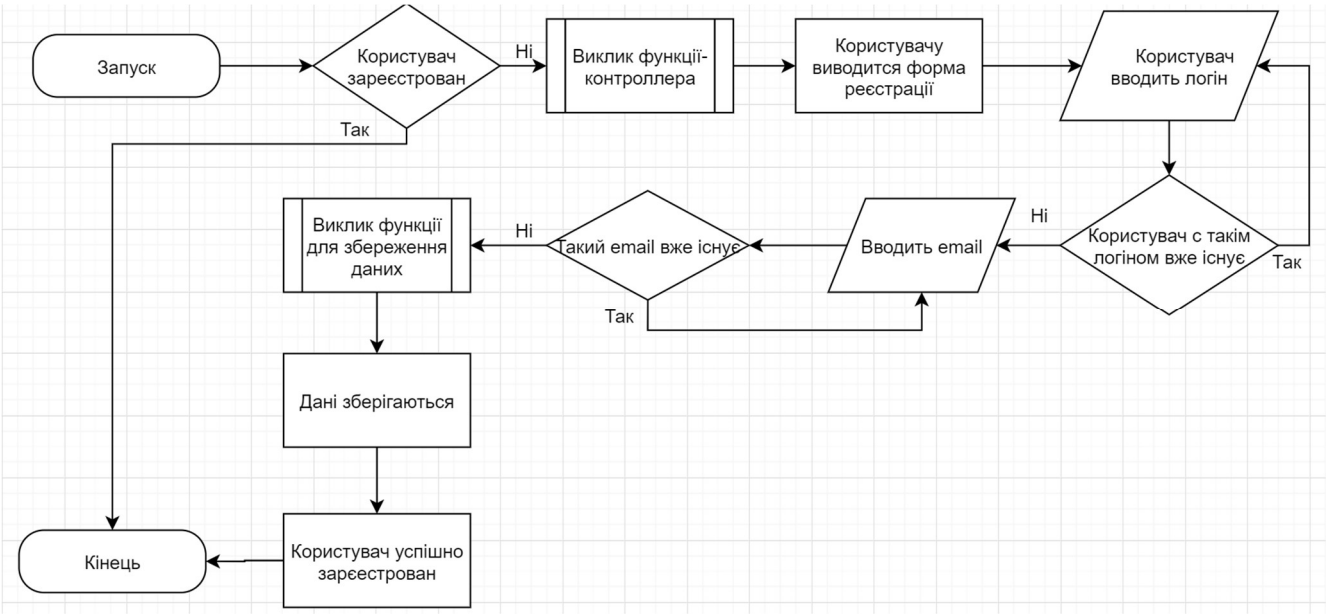


Рисунок 4.6 – Блок-схема варіанту використання «Реєстрація»

Під час виконання процесу реєстрації програма перевіряє чи зареєстрований користувач с таким ідентифікатором. Якщо користувач зареєстрований, він буде переправлений на сторінку авторизації. Далі йде перевірка по електронній пошті або логіну. Якщо користувач з такою поштою вже існує у системі, то буде виведено повідомлення про це.

4.7 Діаграми послідовності

Діаграма послідовності – це різновид діаграми в UML, яка відображає взаємодії об’єктів, впорядкованих за часом. Діаграма допомагає визначити об’єкти та їх взаємодію у графічному представлені [8].

На рисунках 4.7 – 4.10 наведені приклади діаграм послідовностей для варіантів використання веб-сервісу.

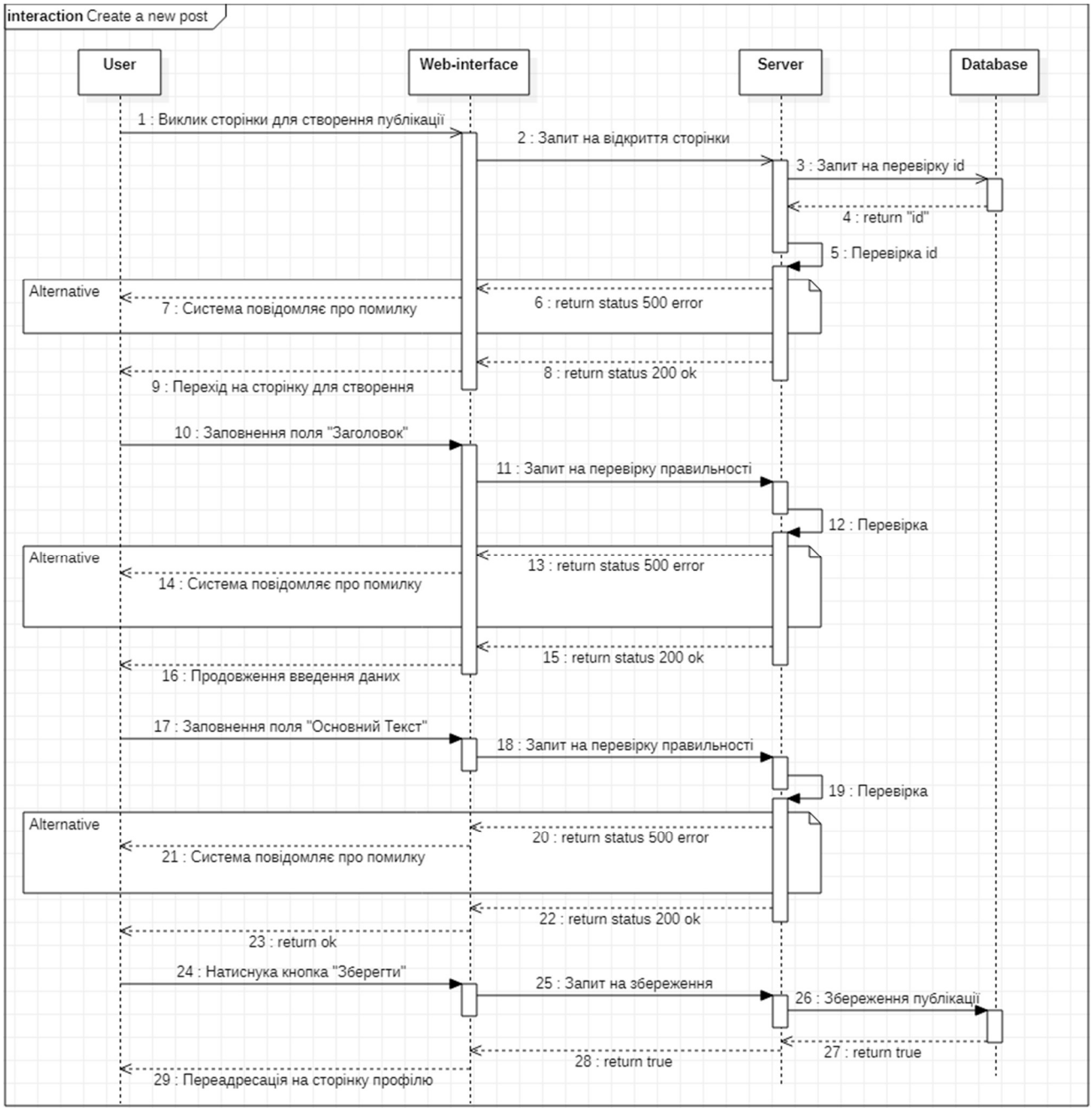


Рисунок 4.7 – Діаграма послідовності для варіанту використання «Створити новий пост»

Під час виконання цього варіанту використання система звертається до БД два рази, коли потрібно перевірити стан користувача по його ідентифікаційному номеру, а також при збережені даних. Дані, які вводить користувач перевіряються на сервері через запити. При помилковому вводі, система повідомляє користувача про це, і користувач повинен ввести дані правильно для успішного збереження.

Після успішного збереження даних, система переадресує користувача на сторінку його профілю, де вже присутня нова публікація.

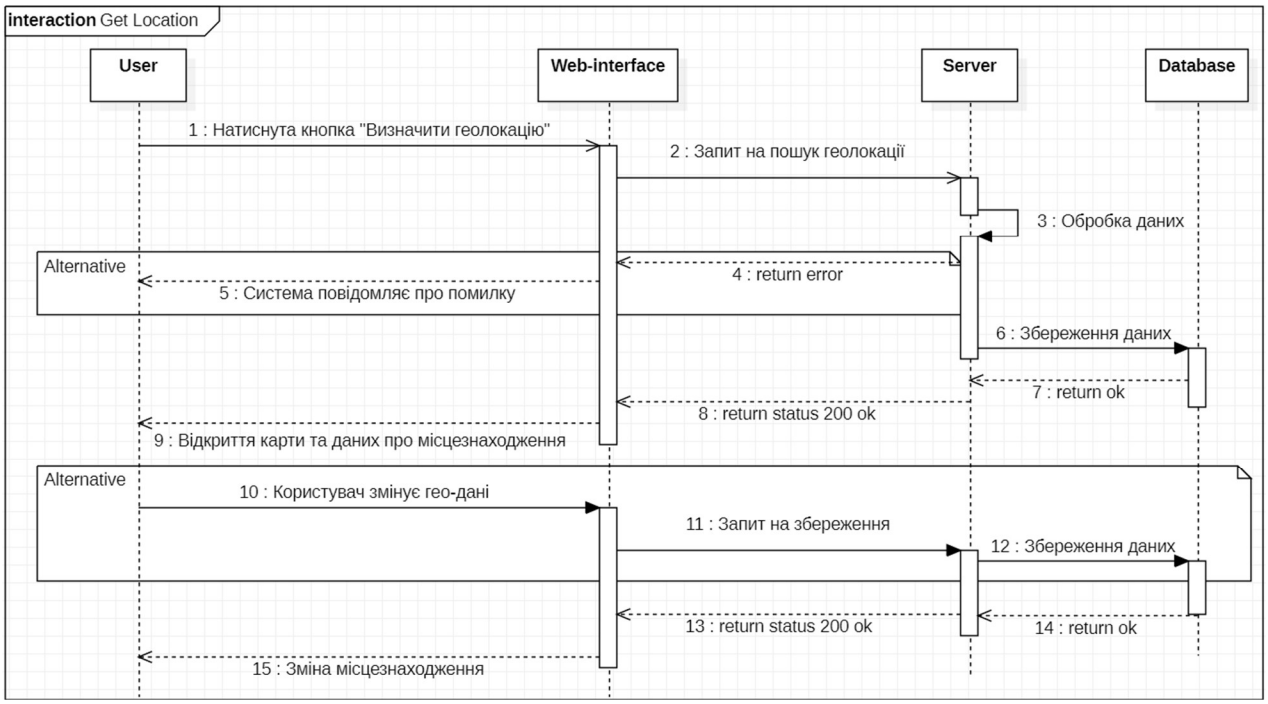


Рисунок 4.8 – Діаграма послідовності для варіанту використання «Визначити геолокацію»

Під час геолокації виконується запит на сервер для обробки помилок, які можуть вплинути на пошук місцезнаходження – це перший та обов’язковий етап. При успішному визначенню місця, система відображає користувачу карту та його дані. Якщо користувач бажає змінити локацію, він має змогу обрати місце на карті або використати пошуковий рядок. Надалі він повинен перейти на сторінку пошуку діяльностей та користувачів, у радіусі, який він визначив.

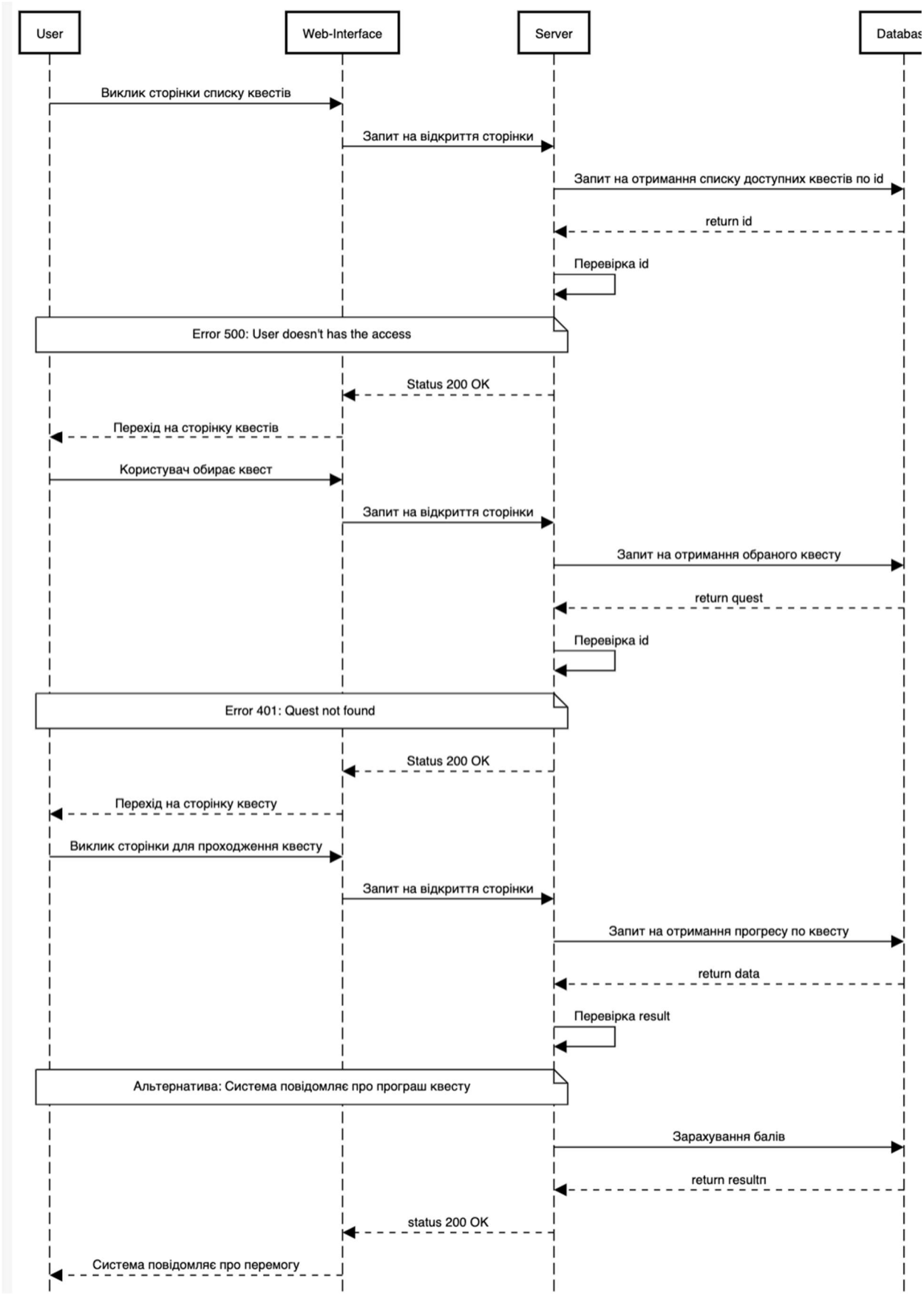


Рисунок 4.9 – Діаграма послідовності для варіанту використання «Пройти квест»

Для проходження квесту користувач обов’язково має бути авторизованим. Після переходу на сторінку профілю, користувач отримує список щоденних квестів, які доступні тільки для нього, тобто, для кожного користувача, список квестів унікальний. Далі, користувач обирає квест, який йому сподобався та натискає кнопку «Розпочати». Після чого відкривається вікно проходження квесту, де користучав отримує завдання та виконує його. Користувач має змогу бачити свій прогрес. Після того, як час проходження спливає система зчитує прогрес користувача та повідомля про перемогу або поразку, та відповідно зараховує бали або ні.

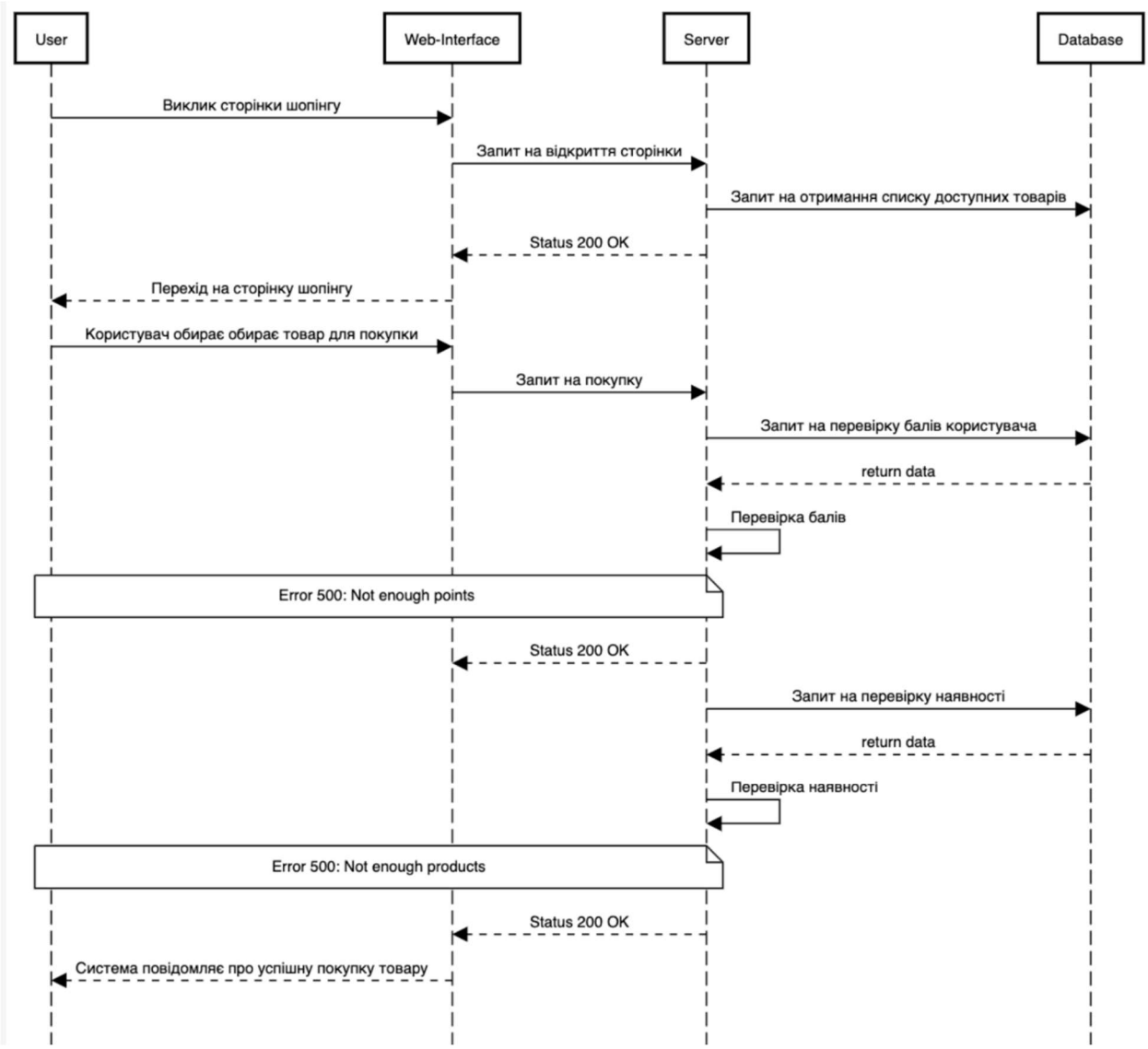


Рисунок 4.10 – Діаграма послідовності для варіанту використання «Придбання товарів»

Для того, щоб витратити бали, які користувач заробив під час проходження квестів, йому потрібно перейти на сторінку «Шопінг», де виводиться список доступних товарів для покупки. Користувач має змогу обрати будь-який продукт, подивитися його деталі та купити. Система перевірятиме кількість балів користувача та ціну продукту, а також наявну кількість, для того щоб користувач не потрапив у неприємну ситуацію із довгим очікуванням продуктів, якщо така є.

Продукти можуть бути різного типу та характеру, від книжок до спортивної атрибутики для активної діяльності. Продукти будуть надаватися партнерами веб-застосування та підпримствами, які активно рекламують себе у веб-застосуванні.

Ці діаграми використовуються розробниками програмного забезпечення та бізнес-фахівцями, щоб зрозуміти вимоги до нової системи або документувати існуючий процес. Діаграми послідовності іноді називають діаграмами подій або сценаріями подій.

Діаграми послідовності можуть бути корисними довідками для великих та маленьких проектів. Переваги діаграм послідовності:

- Представляють деталі варіанту використання UML.
- Моделюють логіку складної процедури, функції або операції.
- Розглядають, як об’єкти та компоненти взаємодіють один з одним, щоб завершити процес.
- Планують детальна функціональність існуючого або майбутнього сценарію.

Таким чином, за допомогою діаграм послідовносте було детально розглянуто основні варіанти використання веб-застосування для контролю ігрового часу користувачами таких, як: Створити пост, визначення місцезнаходження, проходження квестів та придбання товарів або продуктів. Було виявлено ряд альтернативних рішень для всіх варіантів, якщо система отримує помилку при обробці даних, що надає можливість повністю тримати користувача у курсі подій під час роботи у веюб-застосуванні.

4.8 Проектування інтерфейсу користувача

Інтерфейс користувача – це сукупність засобів, за допомогою яких користувач взаємодіє із системою. Стиль інтерфейсу користувача – це набір ознак, методів, прийомів діяльності, які характеризують індивідуальність [9].

Зручний та привабливий інтерфейс – це показник якості веб-сервісу. Інтерфейс повинен розроблюватися згідно із міжнародним стандартом ISO/IEC JTC 1/SC 35 [10].

Стандартизація в області інтерфейсів системи користувача в середовищі інформаційно-комунікаційних технологій (ІКТ) і підтримка цих інтерфейсів для обслуговування всіх користувачів, включаючи людей, які мають доступність або інші специфічні потреби, з пріоритетом відповідності вимогам JTC 1 для культурних та мовних адаптивність.

Це включає:

- доступність інтерфейсу користувача (вимоги, потреби, методи, прийоми та засоби);
- культурна та мовна адаптивність та доступність (наприклад, оцінка культурної та лінгвістичної адаптивності продуктів ІКТ, гармонізовані людські мовні еквіваленти, параметри локалізації, меню голосових повідомлень);
- об'єкти, дії та атрибути інтерфейсу користувача;
- методи та технології для управління та навігації в системах, пристроях і програмах у візуальних, слухових, тактильних та інших сенсорних модальностях (наприклад, голосом, зором, рухом, жестами);
- символи, функціональні можливості та взаємодії інтерфейсів користувача (такі як графічні, тактильні та слухові значки, графічні символи та інші елементи інтерфейсу користувача);
- візуальні, слухові, тактильні та інші сенсорні пристрої та методи введення та виведення в ІКТ середовищах (для таких пристроїв, як клавіатури, дисплеї, миші);

• користувацькі інтерфейси для мобільних пристроїв, портативних пристроїв і віддалених взаємодій.

Веб-інтерфейс надає можливість універсального віддаленого доступу до служб та пристроїв, у цьому технології практично нема альтернатив. Але водночас, оскільки такий інтерфейс доступний усім, постають серйозні питання забезпечення безпеки, зокрема авторизація користувачів, шифрування переданих даних від сторонніх очей, модерація вмісту тощо.

Прототип сторінки користувача веб-сервісу представлений на рис. 4.11. На прототипі можна побачити весь можливий функціонал сторінки профілю користувача, а саме створення та перегляд постів, створення та перегляд коментарів, редагування та налаштування профілю та персональної інформації користувача, список квестів для проходженя і отримання балів, бокове меню для пошуку потрібного функціоналу веб-застосування.

Доступний інтерфейс — це інтерфейс, який максимально зручний для широкого кола користувачів. Перша і, на мене, найважливіша мотивація — моральна. Люди з обмеженими можливостями здоров'я нічим не гірші за інших, вони теж хочуть і мають можливість користуватися повним функціоналом ресурсів, які ви розробляєте [11].

Люди з обмеженими можливостями - це клієнти, які купують ваші товари, користуються вашими послугами, тому забезпечення доступності також сприяє розширенню поточного ринку і виходу на нові ринки.

Технології, які допомагають людям користуватися інтерфейсами, можна розділити на два типи: апаратні та програмні. Людям з обмеженими можливостями допомагають використовувати інтерфейси спеціальні клавіатури, мишки, джойстики та інші пристрої введення [12].

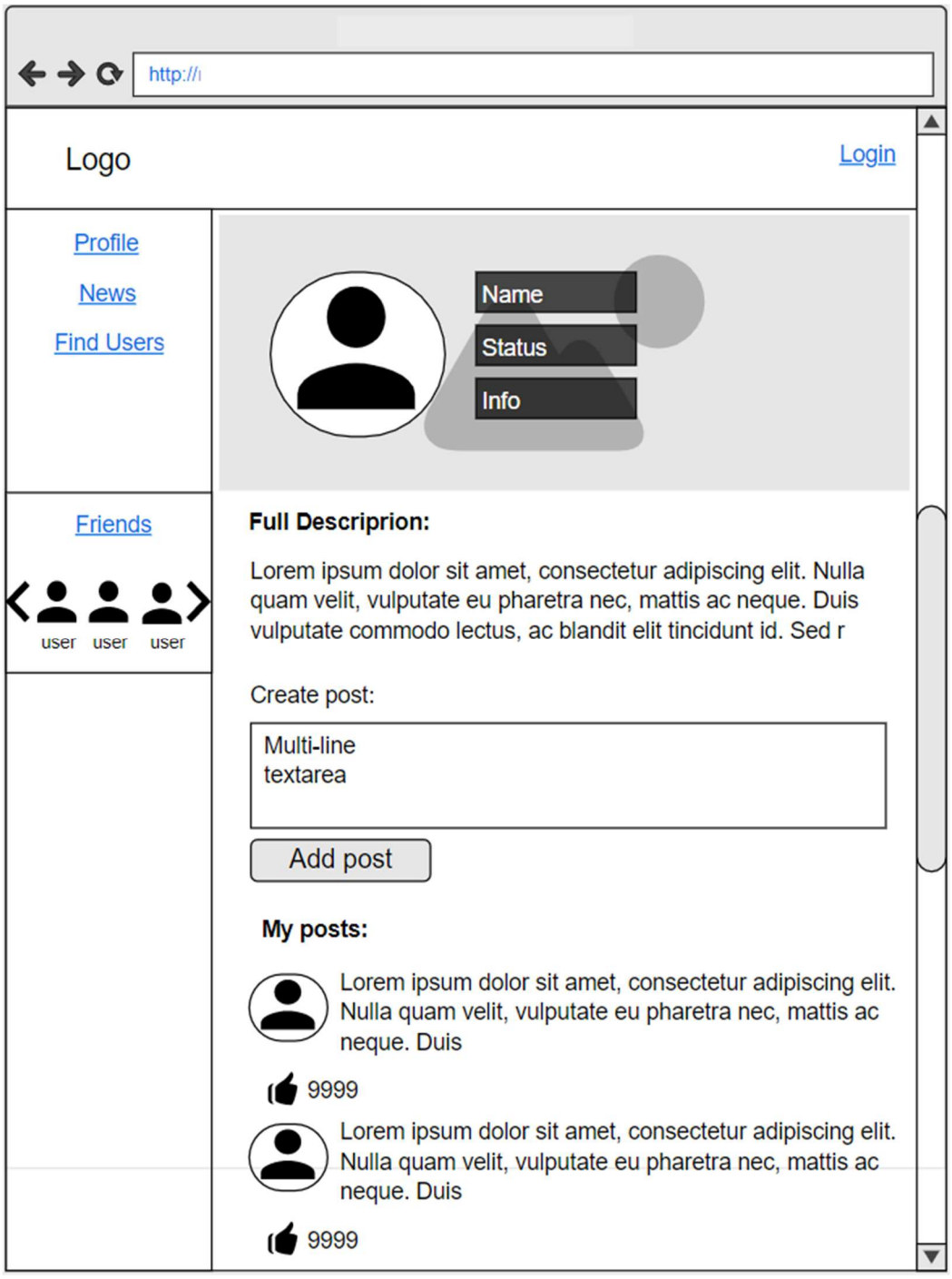


Рисунок 4.11 – Прототип сторінки «Профіль»

Сторінка проектувалась таким чином, щоб користувач мав змогу швидко орієнтуватися серед можливого функціоналу. Вся інформація подана у зручному форматі. Розмір шрифтів та рисунків дозволяє користувачу із задоволенням проводити час.

Моделі інтерфейсів користувача для взаємодії із функціоналом представлені на рис. 4.12 – 4.13.

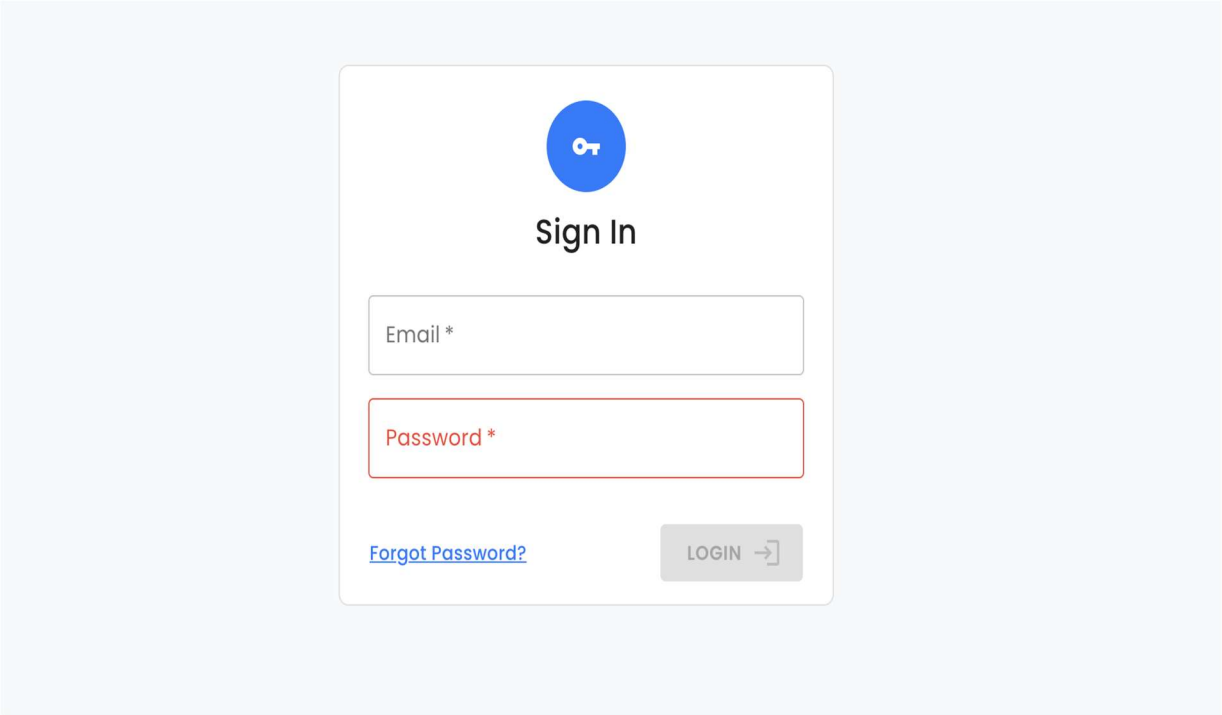


Рисунок 4.12 – Модель форми авторизації веб-сервісу

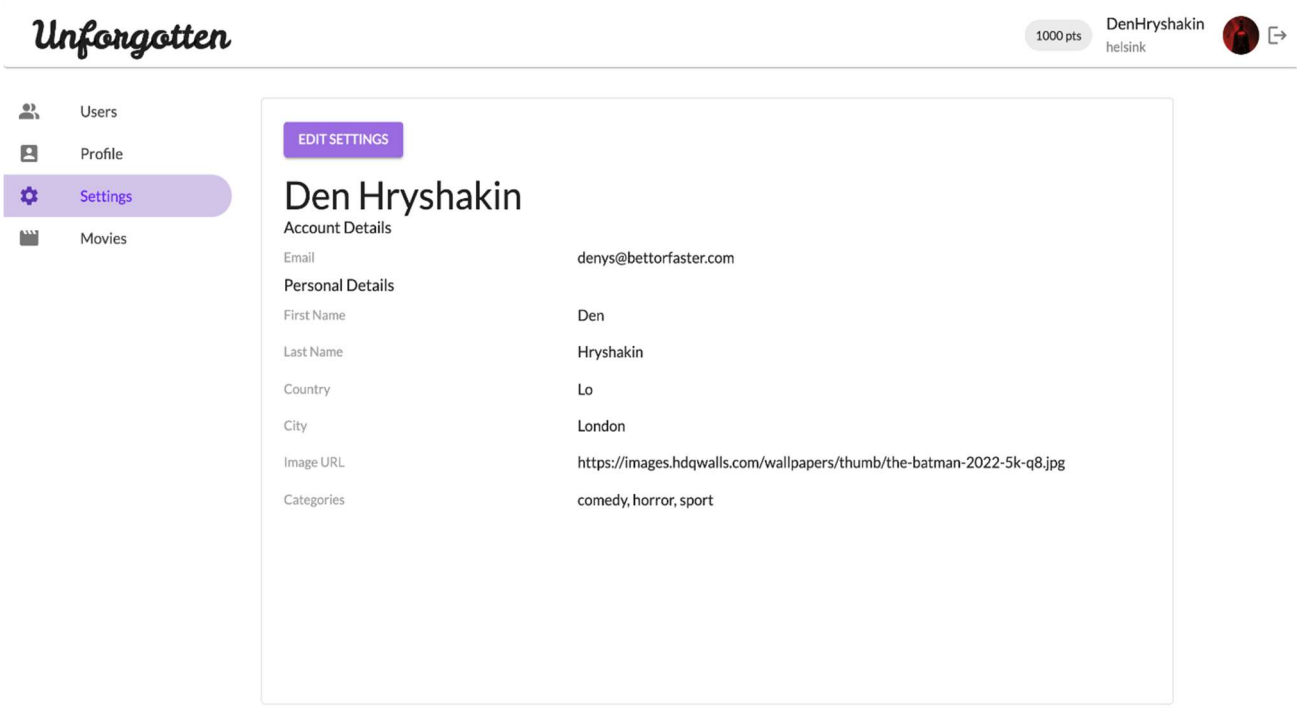


Рисунок 4.13 – Інтерфейс сторінки налаштувань користувача

5 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУВАННЯ

5.1 Опис програмних бібліотек та технологій

JavaScript —універсальна об'єктно-орієнтована мова сценаріїв (скриптів). JavaScript підключається до об'єктів свого середовища виконання (зазвичай, веб-переглядача) та надає можливість керування ними. Має динамічну типізацію та відмінно співпрацює із Node середовищем.

Мова JavaScript має стандартну бібліотеку об'єктів (таких як Array, Date та Math) і основний набір елементів мови програмування, таких як оператори, керівні структури та вирази. Мова підтримує об’єктно-орієнтовану, а також декларативну та імперичну стилі програмування.

Ядро JavaScript може бути розширене для різних потреб шляхом доповнення його додатковими об'єктами.

React – це бібліотека JavaScript, для створення інтерфейсів користувача, яка вирішує проблеми часткового оновлення сторінки. Розробляється Facebook та Instagram. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим [13].

Компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити [14].

CSS – спеціальна мова, яка використовується, щоб описати зовнішній вигляд сторінок, які написані за допомогою мов розмітки даних. CSS використовують для візуальної презентації сторінок, які написані за допомогою HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів [18].

Об'єктний запис JavaScript (JSON, JavaScript Object Notation) — це формат для подання даних. Синтаксис дуже схожий на JavaScript. Підтримка JSON є у багатьох мовах програмування, проте найкраще передається під час використання із файлами, які написані мовою JavaScript.

JSON може подавати числа, тип boolean, рядки, null, масиви (впорядковані послідовності значень), а також об'єкти (містять пари ключ-значення), які й собі можуть містити значення всіх наведених типів (зокрема, інші об'єкти чи масиви). Складніші дані, як-от функції, регулярні вирази чи дати, в JSON неможливо подати як є — їх доведеться попередньо перетворювати на дані підтримуваних типів. Початково об'єкти Date перетворювано на рядок, що містить запис дати у форматі ISO, тож не всі відомості буде загублено. Втім, якщо є потреба зберігати у JSON деякі додаткові типи даних, доведеться виконати належні перетворення до серіалізування та після десеріалізування.

React – це інструмент для створення користувацьких інтерфейсів. Його головне завдання - забезпечення виведення на екран того, що можна бачити на веб-сторінках. React значно полегшує створення інтерфейсів завдяки розподіленню кожної сторінки на невеликі фрагменти. Ми називаємо ці фрагменти компонентами [15].

Node.js – програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду [16].

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць [17]. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів. MongoDB підтримує зберігання документів в JSON-подібному

форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів.

npm (Node Package Manager) - це менеджер пакунків для мови програмування JavaScript. Для середовища виконання Node.js це менеджер пакунків за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через веб-сайт npm [18].

5.2 Програмна реалізація веб-інтерфейсу користувача

Як вже було зазначено у третьому розділі, інтерфейс користувача під час розробки веб-сервісу є одним із важливіших частин тому, що користувач взаємодіє із системою саме через інтерфейс. Тому важливо, щоб він був максимально зручним та привабливим.

Інтерфейс користувача веб-сервісу розроблюється за допомогою бібліотеки React, яка складається з розширень JSX, яка потрібна для виклику та опису функцій, безпосередньо у HTML.

Інтерфейс профілю користувача веб-сервісу представлена на рис. 5.1.

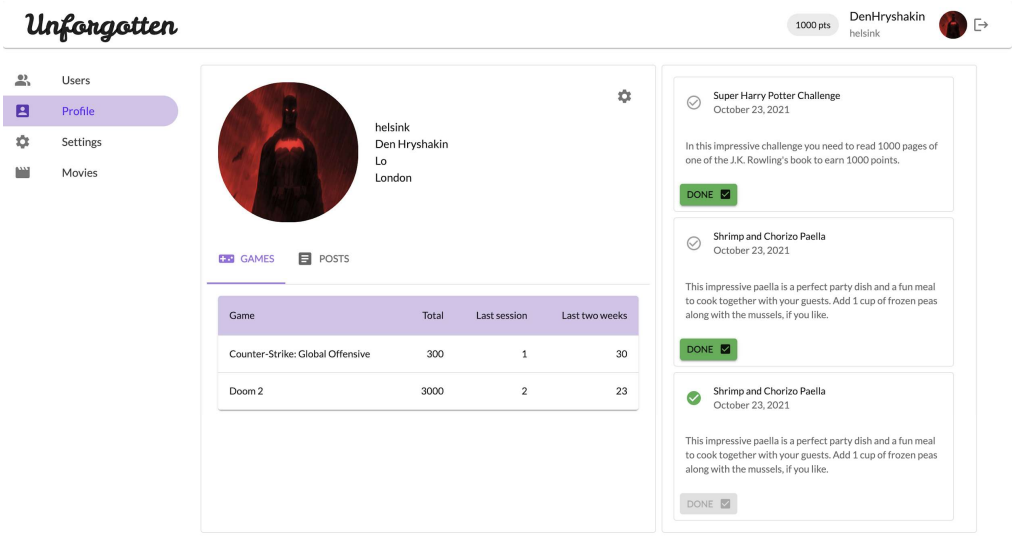


Рисунок 5.1 – Інтерфейс ігрового профілю користувача

Безпосередньо профіль представляє повний опис ігрової статистики користувача. Власник сторінки має повний доступ для створення, оновлення та видалення особистих коментарів. Сторінка списку всіх зареєстрованих користувачів представлена на рис. 5.2

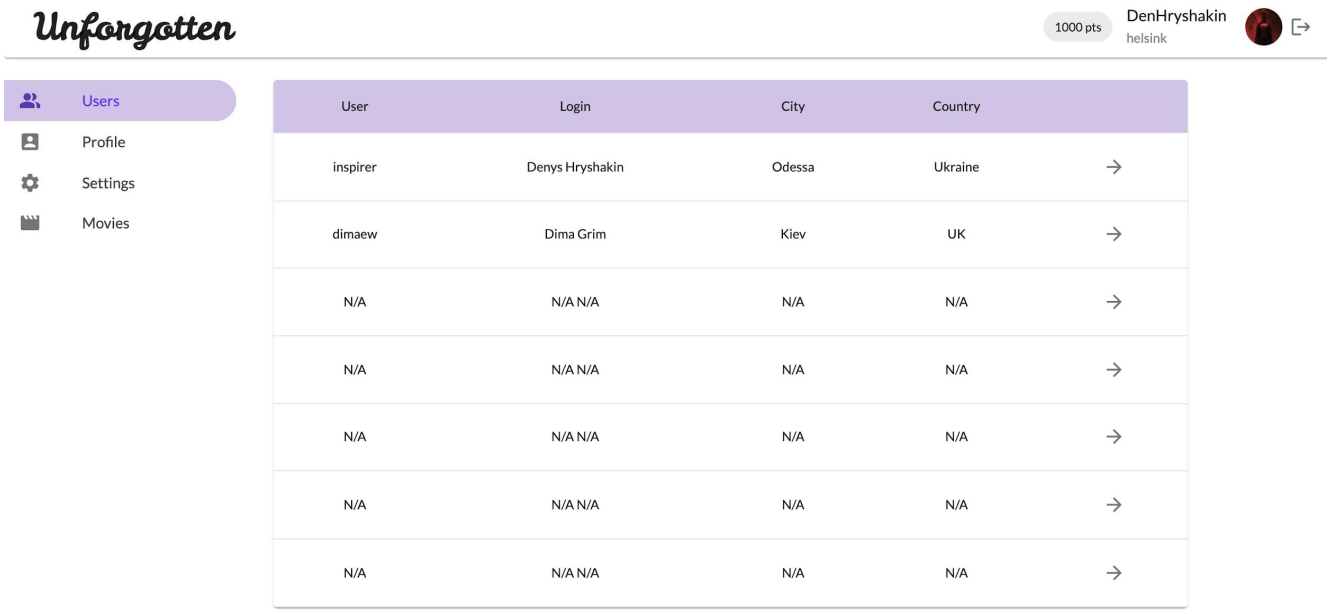


Рисунок 5.2 – Інтерфейс списку користувачів

Процес завантаження списку користувачів описаний у діаграмі активності (рис. 4.5). Сторінка представляє із себе звичайний список зареєстрованих користувачів системи і містить: персональну фотографію, ім’я та прізвище, статус та місцезнаходження. Тут реалізований «pagination», який дозволяє контролювати кількість користувачів на сторінці, щоб не перевантажувати сторінки. Користувач має можливість натиснути на будь-якого іншого та перейти у його профіль.

Спочатку викликається анонімна функція для отримання інформації із БД, а далі виконується по черзі дія по запису даних у локальне сховище. Змінна pagesCount дорівнює частці від ділення загальній кількості користувачів на розмір сторінки, який задається розробником. Метод Math.ceil округляє отримане число до найбільшого, так як сторінки можуть бути лише цілими числами. Далі ми

створюємо масив сторінок та за допомогою циклу for записуємо туди сторінки по черзі починаючи з одиниці. Запит до серверу представлено у додатку А.

Запит виконується за допомогою бібліотеки Axios, яка звертається до серверу із параметрами поточної сторінки та розміру сторінки. Параметри вказуються після знаку питання.

Запит має виконуватися у разі відкриття сторінки, коли компонент монтується вперше або, якщо дані оновилися і компонент монтується знову.

Перевага розробки із використанням бібліотеки React полягає в продуктивності. Усі компоненті монтуються у перше відвідування веб-сервісу та не потребують перезапуску при переході на інші сторінки. Якщо потрібно оновити дані, або користувач змінив дані, компонента монтується знову та оновлює дані, при цьому не перезапускаючи сторінку, що значно пришвидшує роботу системи та знижує витрати інтернет-трафіку.

Після перегляду списку користувачів системи, можна перейти у профіль будь-якого гравця (рис. 5.3.), та побачити його ігрову статистику і переглянути його пости та створити коментар. При цьому інтерфейс динамічно змінюються с анімаційний ефектом, що надає користувачу можливість без різких переходів та приємно для очей використовувати веб-застосування [19].

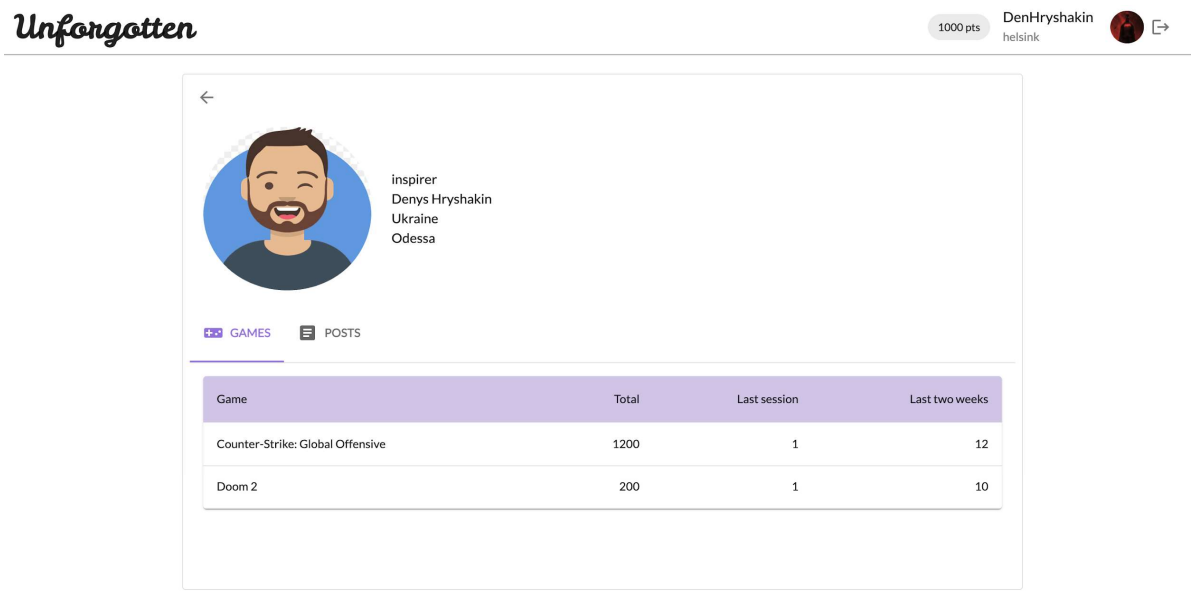


Рисунок 5.3 - Профіль іншого користувача системи

На цій сторінці профілю поточний користувач має змогу переглянути список ігор та статистику у них, а також переходити між «табами» та переглядати пости, а також має кнопку для повернення назад.

Інтерфейс авторизації представлений на рис. 5.4.

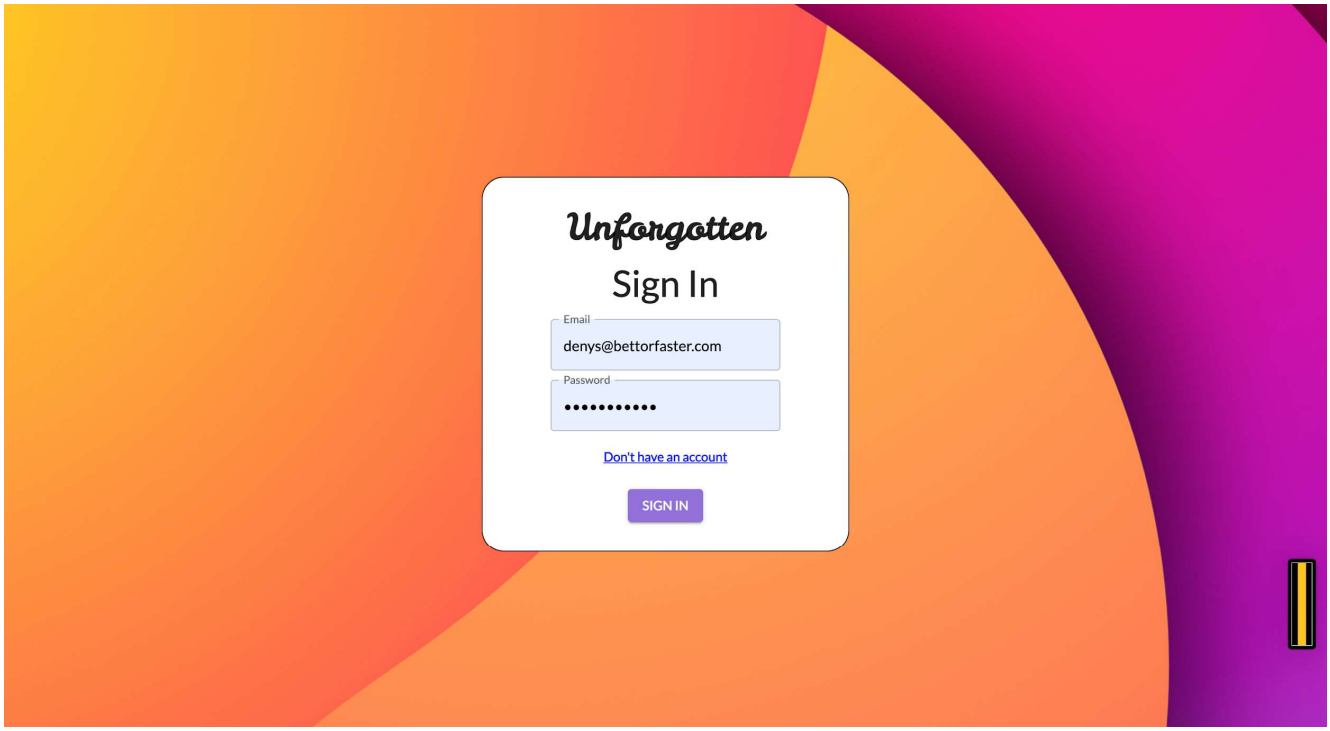


Рисунок 5.4 – Сторінка авторизації користувачів

Користувач повинен заповнити форму згідно без помилок, враховуючи тип кожного рядку. За перевірку правильності введених даних відповідає бібліотека validator. Приклад на перевірку представлено у додатку А.

Бібліотека дозволяє обмежувати кількість символів, правильність електронної пошти, збіг паролів та ін.

Інтерфейс визначення геолокації та пошуку місць на мапі наведено на рис. 5.5.

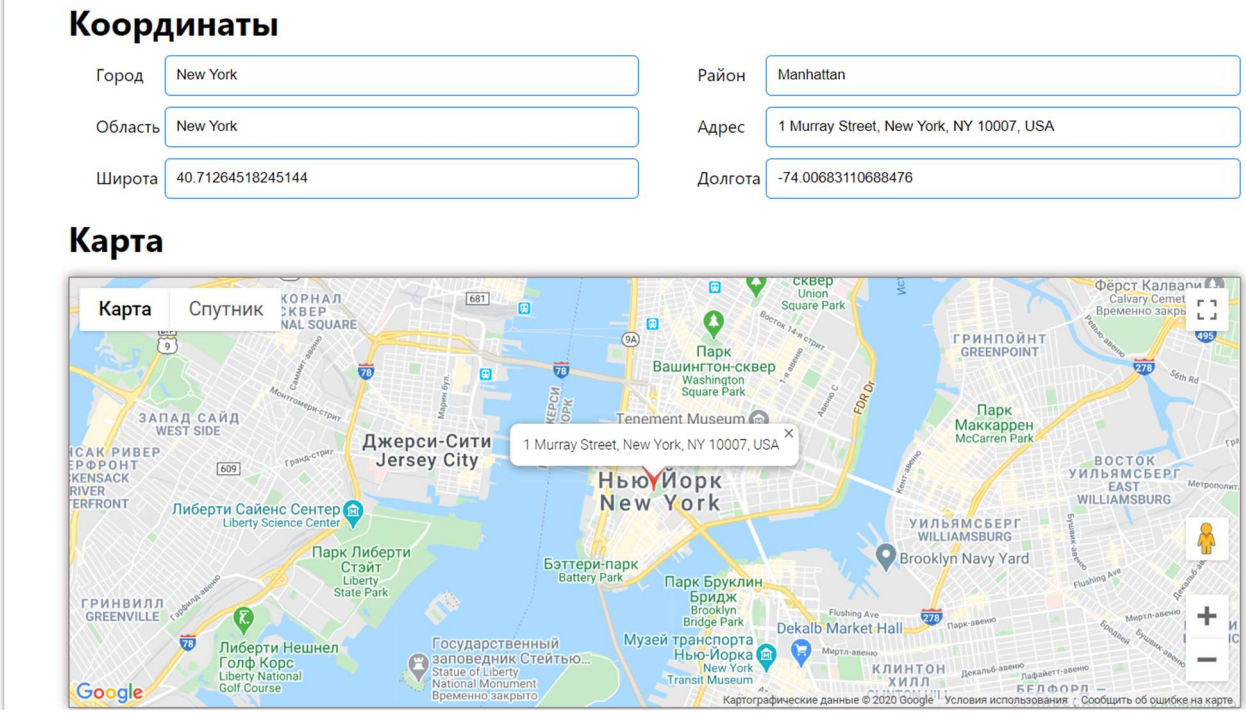
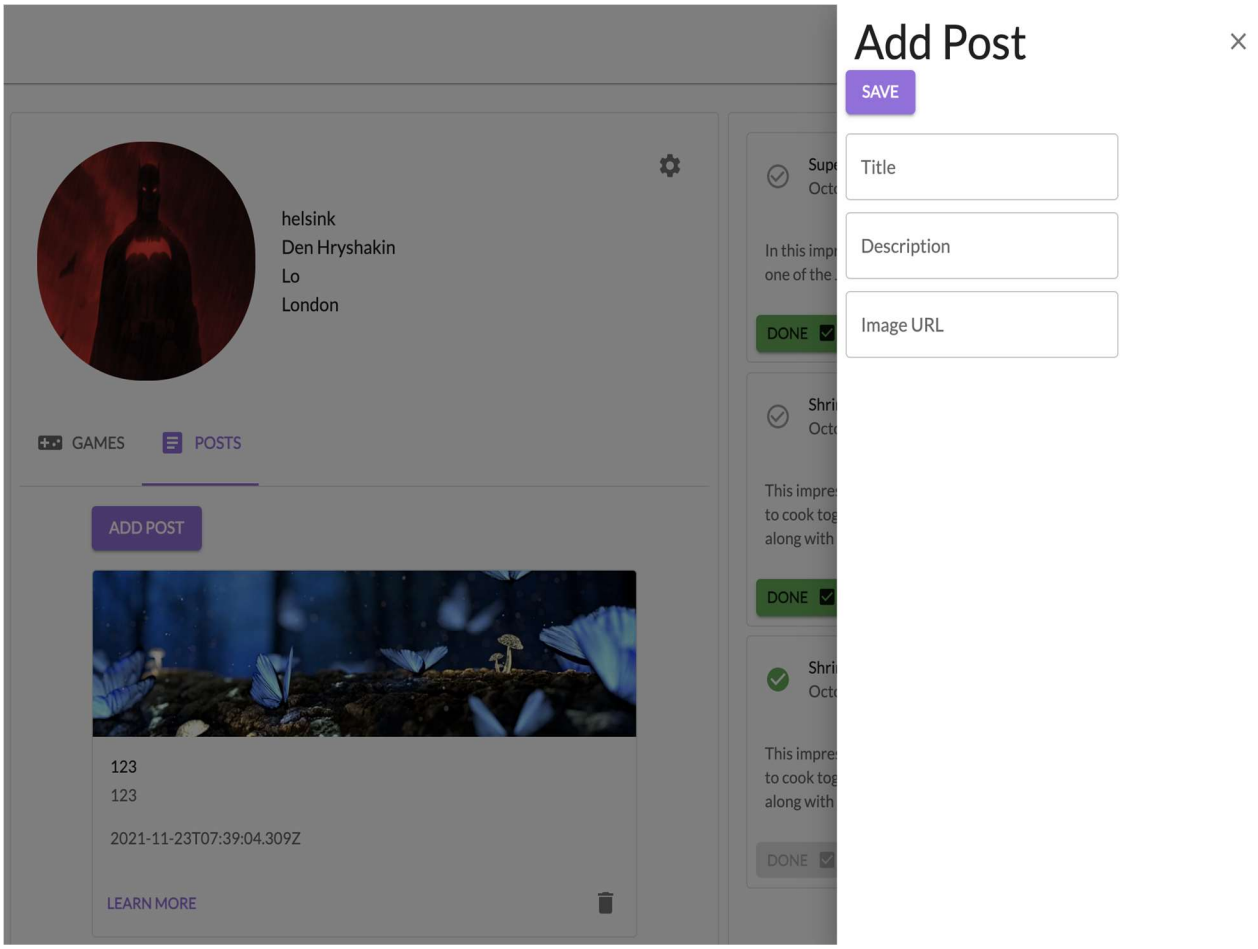


Рисунок 5.5 – Інтерфейс сторінки «Геолокація»

Користувач переходить на сторінку та натискає кнопку «Визначити геолокацію». За декілька секунд, використовуючи Google API, місцезнаходження користувача визначено, та показано на карті. Якщо місце не вірне, або користувач бажає самостійно обрати місце для пошуку діяльностей або інших користувачів, він використовує пошуковий рядок, або маркер на карті. Обрані дані автоматично записуються у базу даних та компонент монтується знову [20].

Інтерфейс сторінки створення користувальницької публікації представлений на рис. 5.6.



Рисунокс 5.6 – Інтерфейс сторінки «Створити новий коментар»

Якщо користувач бажає створити коментар, йому потрібно перейти на сторінку створення, де відкривається форма для заповнення даних. Користувач повинен обов’язково заповнити поля, інакше йому не вдасться створити публікацію. За перевірку відповідає бібліотека validator.

Запит до сервера від клієнта на створення нової публікації наведено у додатку А.

Запит виконується із використанням бібліотеки Axios, у параметрах анонімної функції виступають дані, які користувач ввів. У параметрах POST запиту використовується унікальний ідентифікатор користувача, який створює публікацію. За контроль помилок виступає редуктор GET_ERRORS [21].

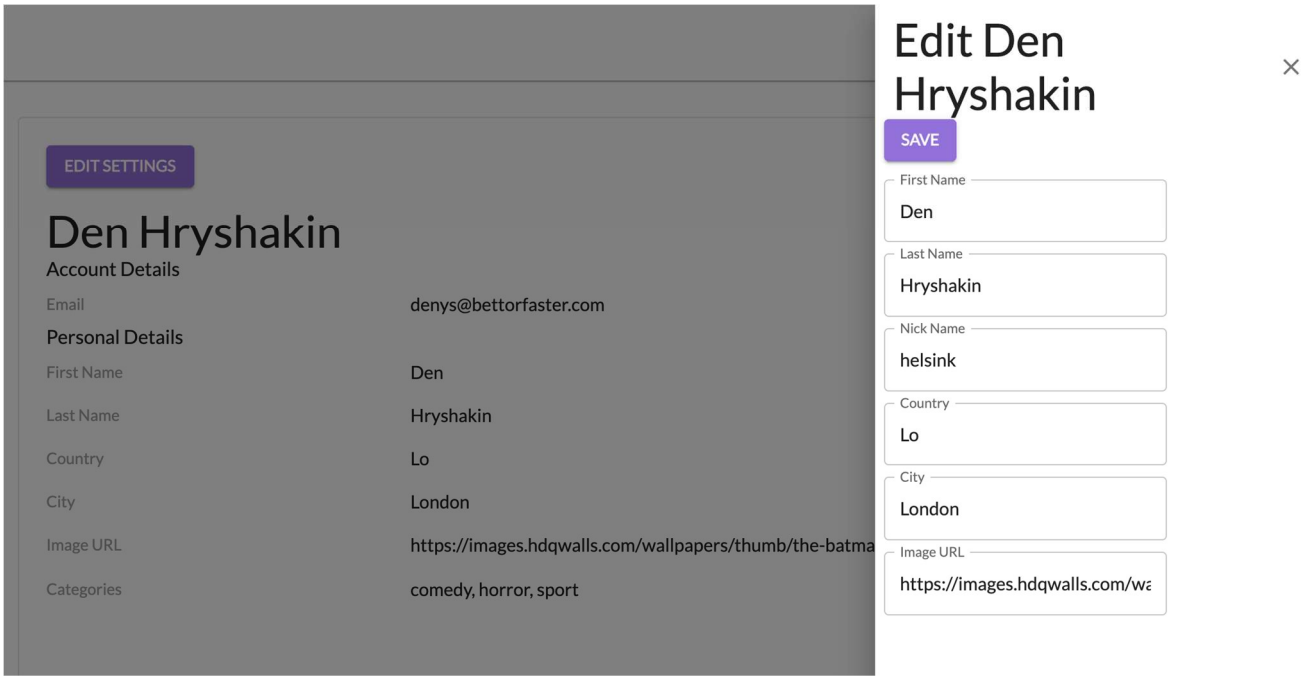


Рисунок 5.7 - Інтерфейс сторінки редагування користувача

Користувачу надається змога редагувати всю детальну інформацію про себе. Для редагування потрібно натиснути кнопку «Редагувати налаштування», після чого відкривається бокова форма для редагування, де користувач повинен внести зміни та зберегти. На даній сторінці використовуються персональні React-hooks для завантаження даних, такі як useCurrentUser() та updateUser(). Кожне поле перевіряється за допомогою валідації, щоб користувач не міг зберегти недостовірну інформацію чи інформацію, яка не відповідає правилам користування веб-застосуванням. Для цього використовуються методи getValidValue(), isValid(), isSubmitting() та isLoading(). Дані методи та хуки дозволяють швидко реагувати на зміни, які робить користувач, що підвищує зручність використання для поточного користувача. Такий ефект досягається за допомогою надшвидкого з’єднання із Node js сервером, проте система повинна відправляти велику кількість запитів на хвилину, що підвищує ризик перенавантаження серверу, саме тому створюється розподіл запитів на back-end.

6 ТЕСТУВАННЯ ВЕБ-СЕРВІСУ ДЛЯ ПОШУКУ НОВИН

Тестування програмного забезпечення – це процес пошуку помилок, недоробок, а також перевірка відповідності вимог та функцій. Процес тестування проходить у декілька етапів.

Цілями тестування є визначення:

1) часу, за який система опрацьовує дані
2) технічних помилок системи
3) недоробок та слабких сторін
4) правильності даних при роботі клієнт-сервера

6.1 Діаграма причинно-наслідкових зв’язків

Для прецеденту «Створення нової публікації» створимо діаграму причинно-наслідкових зв’язків. Для цього потрібно визначити наслідки та причини.

1) Причини:
    a. Присутнє з’єднання із сервером
    b. Запит до сервера відповідає вимогам
    c. Дані оброблені
    d. Дані збережені
2) Наслідки:
    a) Запит до серверу
    b) Передача даних до серверу
    c) Збереження даних на сервері
    d) Створення нової публікації

Таблиця 6.1 – Таблиця рішень

Причина	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Присутнє з’єднання із сервером	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Запит до серверу правильний	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Дані оброблені	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Дані збережені	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Наслідок																
Запит до серверу									1	1	1	1				
Передача даних до серверу													1	1		
Збереження даних на сервері															1	
Створення нової публікації																1

Для успішного створення публікації, потрібно виконання усіх причин. Запит до серверу виконується лише при умові присутнього з’єднання із сервером та БД. Передача даних може виконуватися при умові з’єднання із сервером та правильного запиту до серверу. Збереження даних можливе при виконані перших двох умов та при успішній обробці даних.

6.2 Тестування методів та функцій

Для проходження тестування методом симуляцій, потрібно створити тестові кейси із можливими діями користувача системі (табл. 6.2).

Таблиця 6.2 – Тестові кейси

№ тесту	Опис тесту
1	Користувач бажає видалити публікацію
2	Користувач бажає сформувати стрічку публікацій
3	Користувач заходить у систему і автоматично формується дані про його профіль.
4	Користувач намагається зайти у систему, проте вона повідомляє, що такий користувач не зареєстрований
5	Формування списку користувачів системи, даних для пагінації (номер сторінки, кількість користувачів та обмеження сторінки)

Реалізація тестування React функцій та компонент проводиться у рамках бібліотек Jest. Вони надають повний функціонал для тестування [22].

Jest — це чудова платформа для тестування JavaScript з акцентом на простоту.

Він працює з проектами, які використовують: Babel, TypeScript, Node, React, Angular, Vue. Jest прагне працювати з коробки, без налаштувань, у більшості проектів JavaScript.

При цьому типі тестування розглядаються окремі модулі або компоненти програмного забезпечення. Одиницею тестування може бути окрема функція, метод, процедура, модуль чи об'єкт. Модульний тест ізолює частину коду та перевіряє його правильність, щоб перевірити, що кожна одиниця коду програмного забезпечення працює так, як очікувалося.

При модульному тестуванні перевіряються окремі процедури або функції, щоб гарантувати їхню правильну роботу, а всі компоненти тестуються індивідуально. Наприклад, тестування функції чи визначення правильності роботи оператора чи циклу у програмі підпадають під визначення модульного тестування.

Тестування компонентів перевіряє функціонал окремої частини програми. Тестування проводиться для кожного компонента окремо від інших. Як правило, React-програми складаються з декількох компонентів, тому тестування компонентів пов'язане з індивідуальним тестуванням цих компонентів.

Переваги модульного тестування Jest:

- Запобігає несподіваній регресії.
- Дозволяє розробнику зосередитися на поточному завданні і не турбуватися про минулі.
- Дозволяє створювати модульну програму, яка інакше була б занадто складною для побудови.
- Зменшує потребу ручної перевірки.

Імітація функції – це достовірний дублікат об'єкта або модуля без будь-яких реальних внутрішніх дій. У нього може бути незначний функціонал, але, порівняно з реальним об'єктом, це макет. Він може бути створений Jest автоматично або вручну.

Імітація зменшує кількість залежностей, тобто кількість пов'язаних файлів, які мають бути завантажені та проаналізовані під час запуску тесту. Таким чином, використання великої кількості макетів змушує тести виконуватись швидше.

Імітації функції також відомі як «шпигуни», тому що вони дозволяють стежити за поведінкою функції, яка викликається іншим кодом, а не тільки перевіряти висновок.

Існує два способи імітації функції: або шляхом створення фіктивної функції, щоб її можна було використовувати в тестовому коді або шляхом ручного написання макета для перевизначення залежності модуля.

Для того, щоб протестувати функціонал сторінки профілю, бібліотека Jest виконує методи getUserProfile() та getPosts(), для отримання даних про профіль та публікації.

Надалі проводиться перевірка методів видалення публікацій. Викликається метод deletePost(1) із параметром ідентифікатора публікації, яку потрібно видалити.

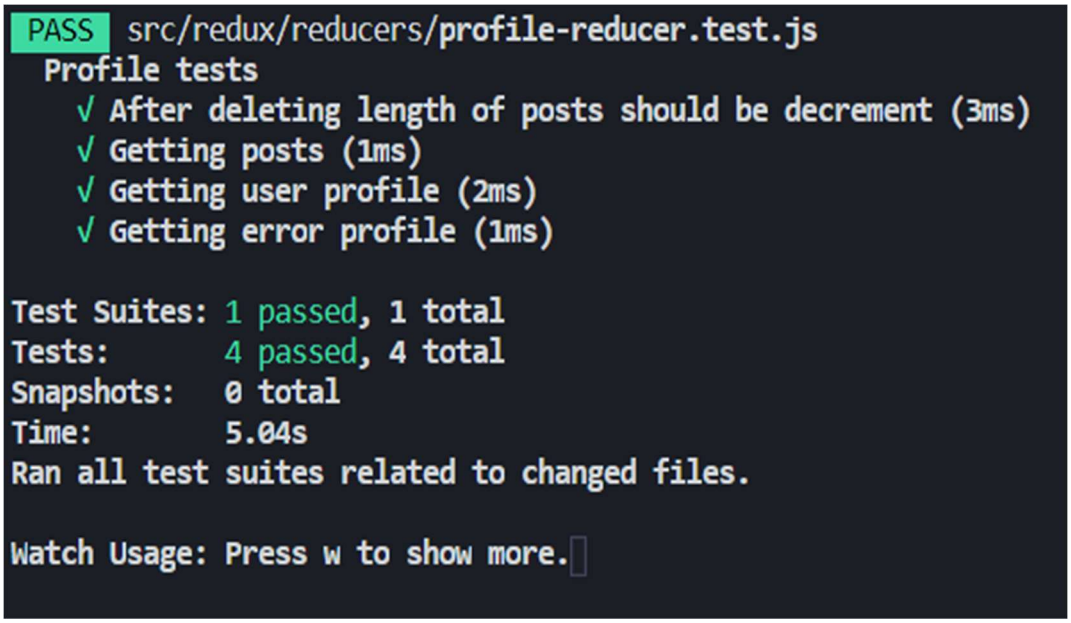


Рисунок 6.1 – Результат тестування

Для перевірки профілю користувача, виконується метод setUserProfile(), який перевіряє на збіги дані. Швидкість тестування 10ms, що є звичайною для таких модулів.

Для тестування класу користувачів, було обрано основні функції для отримання даних із серверу та запису їх у локальне сховище, для подальшої роботи веб-інтерфейсу користувача. Результати тестів наведено на Рисунку 6.2.

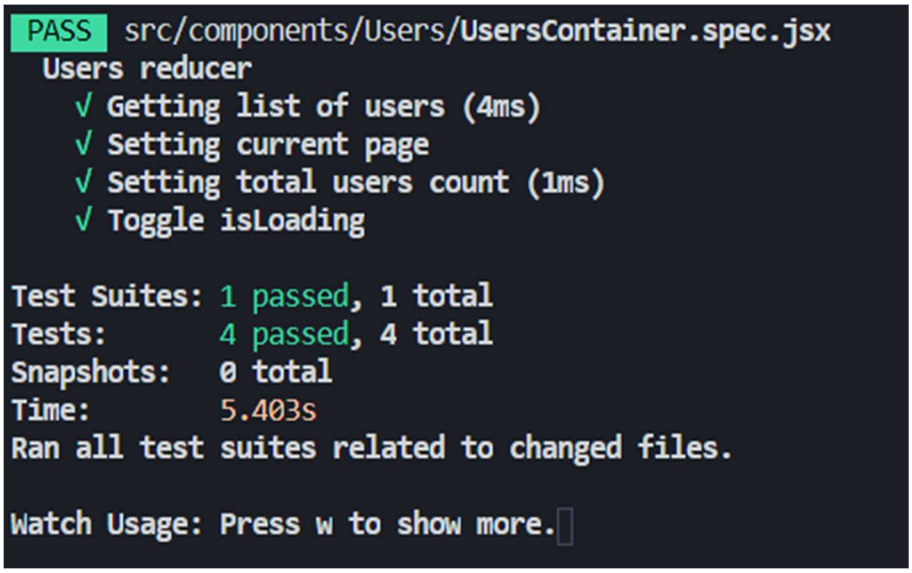


Рисунок 6.2 – Результати тестів класу Users

Під час перевірку було отримано правильні дані від сервера та успішно записано у локальне сховище.

6.3 Експлуатаційні випробування

Для проведення експлуатаційного тестування веб-застосування була зібрана статистика використання веб-сервісів та було обрано метод соціального опитування серед користувачів. Оцінка проводилась за 10-ти бальною шкалою, за допомогою чотирьох категорій:

- Середня кількість годин проведених за іграми – категорія, яка описує середню кількість годин у всіх іграх, проведену користувача під час користування веб-застосувань.
- Конверсія кількості покупок до кількості користувачів – категорія, яка описує статистику відношення кількості покупок до кількості користувачів.
- Простота та зручність інтерфейсу – категорія, яка описує якість методів, які забезпечують веб-інтерфейс.

- Функціональність системи – категорія, яка описує функціональні можливості системи, тобто кількість і якість функцій.

У цьому методі тестування брали участь 150 чоловік, які мали змогу віддавати голос один раз тільки за одну систему у кожній категорії. Результати тестування приведені у табл. 6.3.

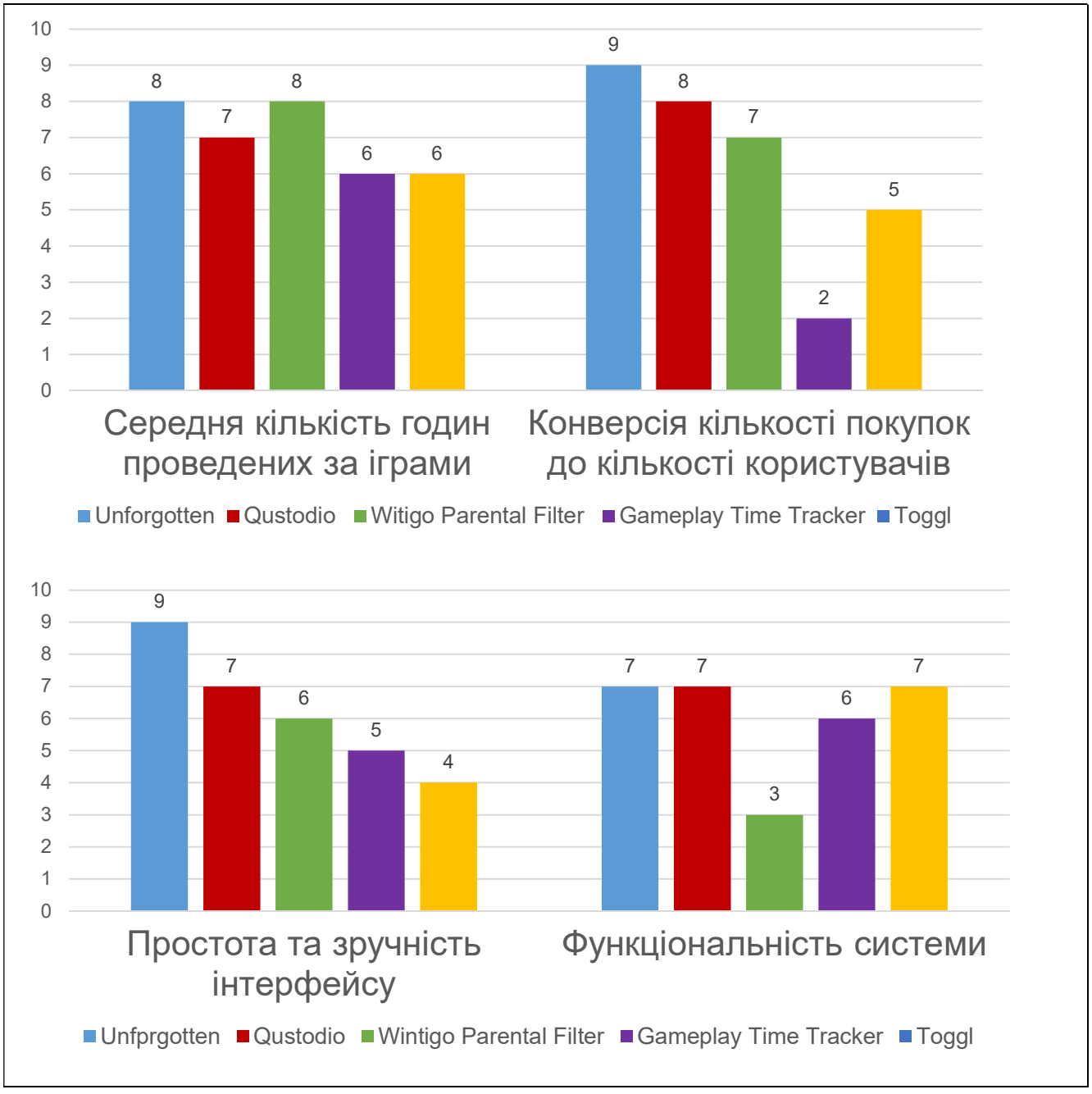


Рисунок 6.3 – Результати тестувння

Для тестування основної мети веб-застосування, а саме зниження часу проведеного за іграми, було запрошено 50 чоловік, які за останній час проводили за іграми більшу кількість часу на день у постійному режимі. Напопулярнішою грою була Counte-Strike: Global Offensive. Серед тестувальників середній час гри за два тижні складав 97 годин, що складало приблизно 7 годин на день. Це майже нормовий робочий день для більгшості людей. При цьому були серйозні проблеми зі здоров'ям через постійну активність зору та малорухливість.

Для перевірки тестувальники протягом трьох тижнів використовували розроблюваємий веб-сервіс, для контролю та зниженню часу, що проведено у іграх.

За цей період було пройдено майже 80 квестів різних рівнів та зароблено 25870 балів. Тестувальники проводили приблизно годину на день, розподілену по невеличких відрізках для більшої ефективності. На рисунку 6.4 представлений графік середнього часу проведеного за іграми у декілька періодів: до використання та після використання застосування.

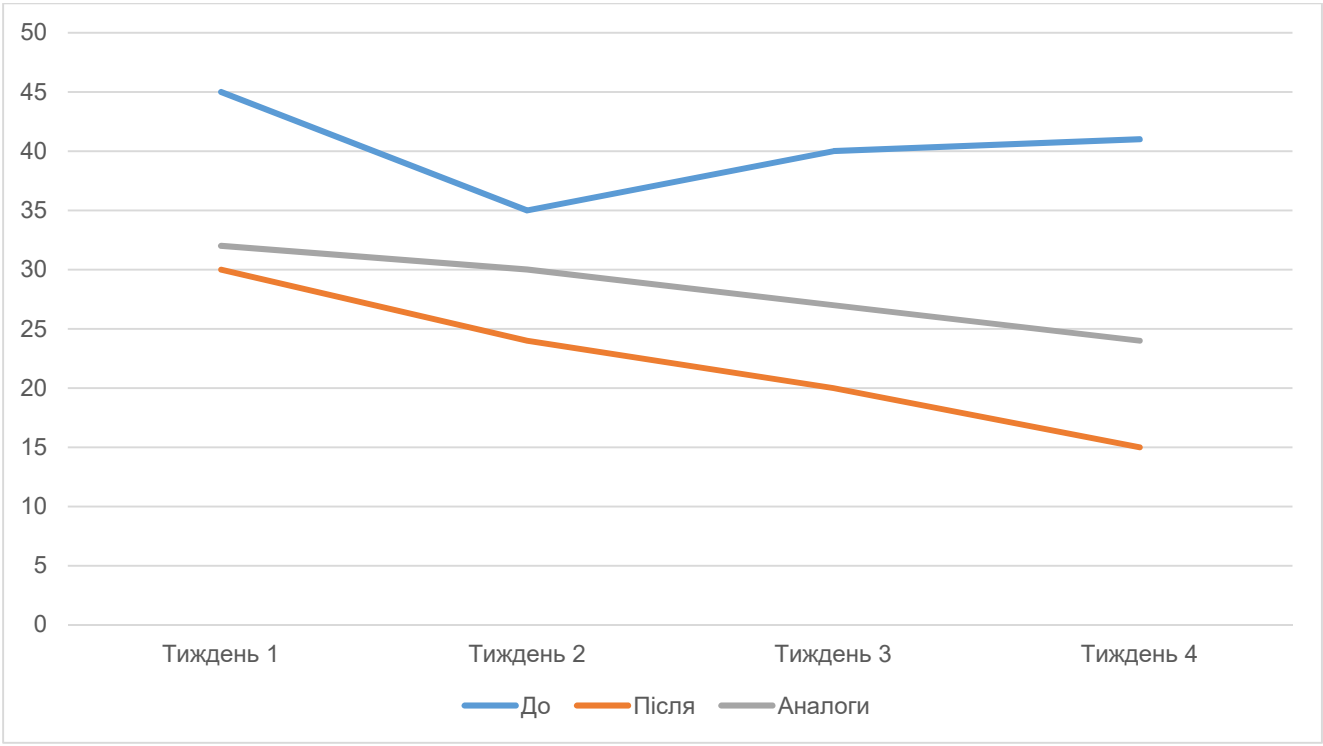


Рисунок 6.4 – Середня кількість годин, що проведено за ігрми в різні періоди

ВИСНОВКИ

В рамках кваліфікаційної роботи виконана робота по опису предметної області, формуванню основних вимог до системи, проектуванню, програмуванню та тестуванню основних програмних модулів застосування для контролю кількості часу, що проведено за іграми та його зниження шляхом мотиваційних квестів та завдань, які підвищать комунікаційні навички та допоможуть знайти нові знайомства та хобі.

Для детального опису системи під час проектування були використанні діаграми стандарту UML.

Стабільну та швидку роботу веб-сервісу забезпечує бібліотека для розробки веб-інтерфейсів React на базі мови програмування JavaScript, а також найновітніші розробки такі, як «React-Hook».

Надійність надає веб-сервер на базі Node js, який забезпечує безпечну та швидку передачу та обробку даних під час роботи системи, а також регулює помилки та підвищує надійність.

Проведено детальний аналіз аналогів системи та повний опис функціоналу веб-сервісу, у порівняні із аналогами, що допоможе створити систему, яка матиме суттєві переваги. Визначені обмеження, функціональні та нефункціональні вимоги. Проведено повний опис варіантів використання та побудовані сценарії варіантів використання.

Визначені основні етапи розробки під час планування системи, побудована таблиця із урахуванням основних етапів розробки веб-сервісу, а також представлена діаграма Ґанта, визначені основні ризики при розробці та проведена оцінка тривалості розробки веб-застосування.

Під час проектування системи були визначені і описані основні модулі системи, представлена діаграма програмних класів із повним описом самих класів, а також їх функцій та методів. Для представлення сутностей бази даних була побудована діаграма моделей а також структура даних, а також опис всіх моделей

та сутностей бази даних. Також були представлені прототипи веб-інтерфейсу користувача системи.

В якості програмних модулів була представлена реалізація веб-інтерфейсу користувача, за допомогою можливостей бібліотеки React. Також були розглянуті можливості роботи клієнт-серверу, в якості http запитів, були представлені рисунки, які описують програмний код та результати його виконання. Програмна структура написана на мові програмування JavaScript із підтримкою бібліотеки React, що дозволило зробити максимально продуктивну систему, за рахунок побудови веб-сервісу як SPA (Single Page Application). Це надало можливість знизити витрати трафіку та підвищити швидкість обробки даних та легку роботу.

Після опису програмних модулів та особливостей системи, було проведено тестування веб-сервісу у два етапи, за допомогою методу причинно-наслідкових зв’язків, симуляції дій користувача, а також проведено аналіз статистики використання веб-застосувань та проведено соціальне опитування. Ці методи тестування відмінно доповнюють оцінку про роботу веб-застосування, що надає змогу в повній мірі оцінити результат виконаної роботи.

Отже, можна зробити висновок, що веб-застосування для контролю кількості часу, що проведено за іграми наблизилос до мети зниження середньої кількості часу проведеного за іграми та підвищення рівня зацікавленості у інших сферах, а також підвищення відношення зацікавленості користувачів у покупках книжок, фільмів та спортивної атрибутики, до загальної кількості користувачів системи. Це означає, що під час використання даного веб-застосування більше мотивуються та зацікавлюються іншими діяльностями.

Методами тестування було виявлено значне зниження середньої кількості годин у іграх, шляхом збору ігрової статистики користувачів у періоди до та після використання веб-застосування. В середньому ігровий час знизився на 30%, порівняно із аналогами, з 40 до 27 годин на тиждень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kungurtsev, O. B., Zinovatnaya S. L & Potochniak, I. V. “Method of Searching Term Interpretations for Domain Dictionaries, Used for Developing Software”. *Applied Aspects of Information Technology. Publ. Science і Technical*. Odessa: Ukraine. 2019; Vol. 2 No.1: 11–19. DOI: https://doi.org/10.15276/aait.02.2019.1.

2. Функціональні вимоги. Інформаційний веб-сайт Quality assurance info: стаття. URL: https://www.quality-assurance-group.com/requirement-types/.

3. Rookee – Что такое Google Maps [Electronic resource] – Режим доступу: https://wiki.rookee.ru/google-maps/.

4. Model Structure. *Інформаційний портал*. URL: https://www.researchgate.net/figure/The-general-structure-of-a-model-development-sequence_fig1_316911393.

5. AJAX запити. *Інформаційний веб-портал MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started.

6. Структура БД. *Інформаційна документація*. URL: https://www.jetbrains.com/help/phpstorm/creating-diagrams.html.

7. Діаграма класів. *Інформацій портал Visual Paradigm*: стаття. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/.

8. Sequence diagrams. *Портал комп’ютерних наук*: стаття. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/.

9. Інтерфейс користувача. *Інформаційний портал Semantica*: стаття. URL: https://semantica.in/blog/veb-interfejs.html.

10. Стандарти інтерфейсів користувача. *Офіційний сайт міжнародної організації зі стандартизації.* URL: https://www.iso.org/committee/45382/x/catalogue/.

11. Принципи веб-доступності. URL: https://www.w3.org/WAI/fundamentals/accessibility-principles/ru#alternatives.

12. Оцінка веб-доступності. URL: https://www.w3.org/WAI/test-evaluate/.

13. Accomazzo, A., Murray N., Lerner A. Fullstack React: The Complete Guide to ReactJS. New York, NY: Fullstack.io, 2017. 836 p.

14. Налаштування React. *Офіційний сайт React*. URL: https://reactjs.org/.

15. Tielens M.T. React in action. New York, NY: Manning Publications, 2018. 332 p.

16. Node js. *Інформаційний портал Metanit*: стаття. URL: https://metanit.com/web/nodejs/3.1.php.

17. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. 3rd ed. Boston, MA: O`Reilly Media, 2019. 953 p.

18. NPM. *Офіційний веб-сайт менеджеру пакетів NPM*. URL: https://www.npmjs.com/.

19. Krisilov, V. A., Pysarenko, K. A. & Huy, Vu Ngoc. "Method of Dynamic Formation of Content in Conditions of Limited Resources". *Applied Aspects of Information Technology. Publ. Science і Technical*. Odessa: Ukraine. 2019; Vol. 2 No.2: 89–104. DOI: https://doi.org/10.15276/aait.02.2019.1.

20. Salcescu, C. Functional React: Quick start with React Hooks, Redux and MobX. 2nd ed. Seattle, WA: Amazon Services LLC, 2020. 148 p.

21. JSX. *Офіційний веб-сайт React.* URL: https://ru.reactjs.org/docs/introducing-jsx.html.

22. Jest. *Технічна документація бібліотеки Jest*. URL: https://jestjs.io/.