

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Розробка гейміфіковано вебсимулятора
PokeGame»**

Виконав студент групи ІПЗк-22-1
ПЕТРУК Олександр Олександрович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПЕТРУК Олександр Олександрович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Розробка гейміфікованого вебсимулятора PokeGame»**,
затверджена Наказом закладу вищої освіти від **«23» лютого 2023 р., №74/с**
Вихідні дані для роботи: Об'єктом дослідження є гейміфікований вебсимулятор у всесвіті покемонів, де користувачі можуть досліджувати покемонів та проводити битви з іншими користувачами. Предметом дослідження є архітектура та компонентна структура гейміфіковано вебсимулятора. Дослідження спрямоване на вивчення технологій та методів, що дозволяють забезпечити зручне та ефективне проведення битв між гравцями та дослідження покемонів в рамках гейміфікованого вебсимулятора.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Роман САВІЦЬКИЙ	02.03.2024	02.03.2024
2	Роман САВІЦЬКИЙ	04.04.2024	04.04.2024
3	Роман САВІЦЬКИЙ	02.05.2024	02.05.2024

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу гейміфікованого вебсимулятора PokeGame та пояснювальної записки. Пояснювальна записка до випускної роботи містить 71 сторінок, 52 ілюстрацій та 5 таблиці.

Метою роботи є розробка гейміфікованого вебсимулятора. Вебсимулятор має включати модулі для проведення битв між гравцями або гравця з ботами, перегляд існуючих покемонів, придбання нових покемонів або покращення існуючих, систему досягнень і також соціальний модуль.

В роботі визначено основні завдання на розробку гейміфікованого вебсимулятора, проведений аналіз аналогічних систем. Було обрано архітектурний патерн Медіатор для написання для багатосторінкового вебсайту. Представлені сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис БД (MongoDB), алгоритми взаємодії модульної системи, діаграма базових варіантів використання, діаграму прецедентів та алгоритми роботи основних модулів. Реалізація серверної частини здійснена мовою TypeScript з використанням фреймворку Nest.js. Клієнтська частина реалізована на мові TypeScript з використанням фреймворку Next.js.

КЛЮЧОВІ СЛОВА: СИМУЛЯТОР, КОРИСТУВАЧ, АРЕНА, ГЕЙМИФІКОВАНИЙ ВЕБСИМУЛЯТОР, ВЕБСИМУЛЯТОР.

					ІПЗК.КР.Б-121-24-ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Петрук О.О.			Розробка гейміфіковано вебсимулятора PokeGame	Літ.	Арк.	Аркушів
Керівник		Савіцький Р.С.					3	
						Житомирська політехніка, група ІПЗк-22-1		
н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

Bachelor's thesis consists of a software suite for a gamified web simulator called PokeGame and an explanatory note. The explanatory note for the thesis contains 71 pages, 52 illustrations, and 5 tables.

The purpose of the work is to develop a gamified web simulator. The web simulator includes modules for conducting battles between players or between a player and bots, viewing existing Pokémon, acquiring new Pokémon or improving existing ones, an achievement system, and a social module.

The work defines the main tasks for developing the gamified web simulator and includes an analysis of similar systems. The Mediator architectural pattern was chosen for creating a multi-page website. The work presents the scenarios of the software suite, the overall structure of the software suite, a description of the database (MongoDB), interaction algorithms of the modular system, a basic use case diagram, a use case diagram, and the algorithms for the main modules. The server-side implementation is done in TypeScript using the Nest.js framework. The client-side is implemented in TypeScript using the Next.js framework.

KEYWORDS: SIMULATOR, USER, ARENA, GAMIFIED WEB SIMULATOR, WEB SIMULATOR.

					ИПЗК.КР.Б-121-24-ПЗ	
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ГЕЙМИФІКОВАНИХ ВЕБСИМУЛЯТОРІВ	10
1.1 Постановка задачі.....	10
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури геймифікованого вебсимулятора	19
1.4 Обґрунтування вибору інструментальних засобів.....	21
Висновки до розділу 1	24
РОЗДІЛ 2 ПРОЄКТУВАННЯ ГЕЙМИФІКОВАНОГО ВЕБСИМУЛЯТОРУ.....	26
2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи.....	26
2.2 Розробка бази даних системи.....	31
2.3 Проєктування та реалізація алгоритмів роботи системи	33
2.4 Реалізація геймифікованого вебсимулятора	37
Висновки до розділу 2	43
РОЗДІЛ №ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	45
3.1 Порядок встановлення та налаштування системи	45

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з гейміфікованим симулятором.	46
3.3 Тестування роботи програмного продукту.....	63
Висновки до розділу 3	67
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ	71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML – HyperText Markup Language

RAM – пам'ять із довільним доступом

REST API – архітектурний стиль взаємодії компонентів розподіленої програми у мережі.

SSD – комп'ютерний запам'ятовувальний пристрій на основі мікросхем пам'яті та контролера керування ними

БД – база даних

ПЗ – програмне забезпечення

					ІПЗК.КР.Б-121-24-ПЗ	
						7

ВСТУП

Актуальність теми. Актуальність теми гейміфікованого вебсимулятора у всесвіті покемонів можна розглянути з кількох аспектів: популярність франшизи, інтерактивність, технологічні тенденції.

Вебсимулятори можуть бути використані для навчання та розвитку навичок. У випадку покемонів, це може бути вивчення стратегій, логічного мислення та взаємодії з іншими гравцями. Гейміфіковані елементи можуть підвищити інтерес до навчання та зробити його більш ефективним.

Сучасні вебтехнології дозволяють створювати складні та багатофункціональні симулятори, які працюють у браузері. Це зручно для користувачів, оскільки не вимагає встановлення додаткового ПЗ та дозволяє доступ до симулятора з будь-якого пристрою, що має доступ до інтернету.

Розробка гейміфікованого вебсимулятора у всесвіті покемонів є перспективною та актуальною через популярність франшизи, зростаючий інтерес до гейміфікації.

Метою випускної роботи є розробка гейміфікованого вебсимулятора у всесвіті покемонів з метою забезпечення заохочення досліджувати всесвіт покемонів, знаходження індивідуальних тактик проведення бою, який буде відповідно відпрацьовувати у інших симуляторах або іграх по всесвіту покемонів. Основні завдання включають:

1. Аналіз потреб користувачів.
2. Проєктування архітектури та вибір технологій для вебсимулятора.
3. Розробка серверної частини додатку.
4. Розробка клієнтської частини додатку.

Об'єктом дослідження є гейміфікований вебсимулятор у всесвіті покемонів, де користувачі можуть досліджувати покемонів та проводити битви з іншими користувачами. Дослідження спрямоване на вивчення технологій та методів, що дозволяють зручне та зрозуміле проведення боїв з іншими гравцями.

Предметом дослідження є архітектура та компонентна структура гейміфікованого вебсимулятора. Дослідження спрямоване на вивчення технологій

та методів, що дозволяють забезпечити зручне та ефективне проведення битв між гравцями та дослідження покемонів в рамках гейміфікованого вебсимулятора.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано статтю: Савіцький Р.С., Петрук О.О. Розробка прототипу вебдоступного ігрового симулятора. Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки. Том 35 (74), №2, 2024. С. 143-149 [1].

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ГЕЙМИФІКОВАНИХ ВЕБСИМУЛЯТОРІВ

1.1 Постановка задачі

Основна мета цієї роботи полягає у розробці гейміфікованого вебсимулятора з метою забезпечення заохочення досліджувати всесвіт покемонів. Продукт використовується для залучення користувачів до бренду, а також може бути просто розважальним інструментом, який допомагає користувачам розважитися і відпочити.

1. Підготовка до створення проєкту.

1.1. Обрати концепцію, на якій буде будуватися симулятор.

1.2. Знайти API, яке буде надавати потрібні дані про персонажів. Для використання зображень та характеристик персонажів.

1.3. Застосовувати модульну систему для ефективного завантаження компонентів в браузер.

1.4. Створити дизайн зосереджений на зручності та простоті використання для користувачів.

2. Реалізувати серверну частину додатку.

2.1. Написати програмні модулі для: авторизації користувача можливість зміни даних користувачем, при умові що він авторизований.

2.2. Можливість зберігання в базі даних: рейтингу, монет, список персонажів, активного персонажа, кількість зілля, досягнення на акаунті, список боїв в яких користувач приймав участь.

2.3. Можливість переглянути весь список героїв, пошук героя по імені або по номеру, можливість обрати активного персонажа, можливість купляти персонажів у випадковому порядку, можливість покращити існуючого персонажа, можливість використовувати зілля для покращення героя на одну гру, можливість придбання зілля, можливість пошуку гравців які готові для битви. А також отримання монет та рейтингу при виграші або втрата рейтингу

при програшу, можливість битви в оффлайн арені де користувач отримує монети, реалізувати структуру збереження даних.

3.1. Дані користувача які зберігаються в БД: ім'я користувача, електрона адресу користувача, пароль у зашифрованому вигляді, кількість персонажів, кількість монет, кількість очків рейтингу, інформація про бої, активного персонажа у користувача, кількість зілля, досягнення, фото профілю користувача, рівень в оффлайн арені.

4. Створити тестовий контент для сайту.

5. Розробити план тестування системи управління контентом. Зокрема передбачити наступні види тестування: функціональне, кросбраузерне та адаптивне, тестування безпеки даних.

Результатом реалізації поставленого завдання є гейміфікований вебсимулятор, що містить в своєму складі серверну структуру збереження даних, однокористувацький клієнтський додаток для реалізації функціональних можливостей.

1.2 Аналіз аналогів програмного продукту

Основною метою для гейміфікованого симулятору є зацікавлення користувача до дослідження додатку та його концепції.

Ось кілька прикладів існуючих схожих рішень:

Foldit [2] – це онлайн-гра, створена для участі гравців у вирішенні складних проблем біологічного характеру. Вона була розроблена вченими з Університету Вашингтона в Сіетлі та Інституту фундаментальної біології. Геймплей онлайн-гри представлено на рисунку 1.1.

Гра використовує методи гейміфікації для залучення гравців до передбачення та моделювання тривимірних структур білків. Білки – це основні будівельні блоки живих організмів і відіграють ключову роль в біологічних процесах [3]. Правильне передбачення структури білка може допомогти у розробці нових лікарських препаратів та розумінні біологічних механізмів.

Учасники Foldit використовують інтуїтивний інтерфейс для складання та перестановки амінокислот – елементів, з яких складаються білки. Метою гри є досягнення найкращої структури білка з мінімальною енергією. Гравці можуть змагатися один з одним або працювати в команді, щоб вирішити наукові завдання.

Foldit вже досяг значних успіхів, зокрема допомігши в розв'язанні складних наукових проблем та оптимізації дизайну білків для різних цілей, таких як створення ефективних лікарських препаратів та біопалива. Гра дозволяє людям без спеціальної освіти в галузі біології та хімії внести внесок у науку та прийняти участь в розв'язанні глобальних проблем.

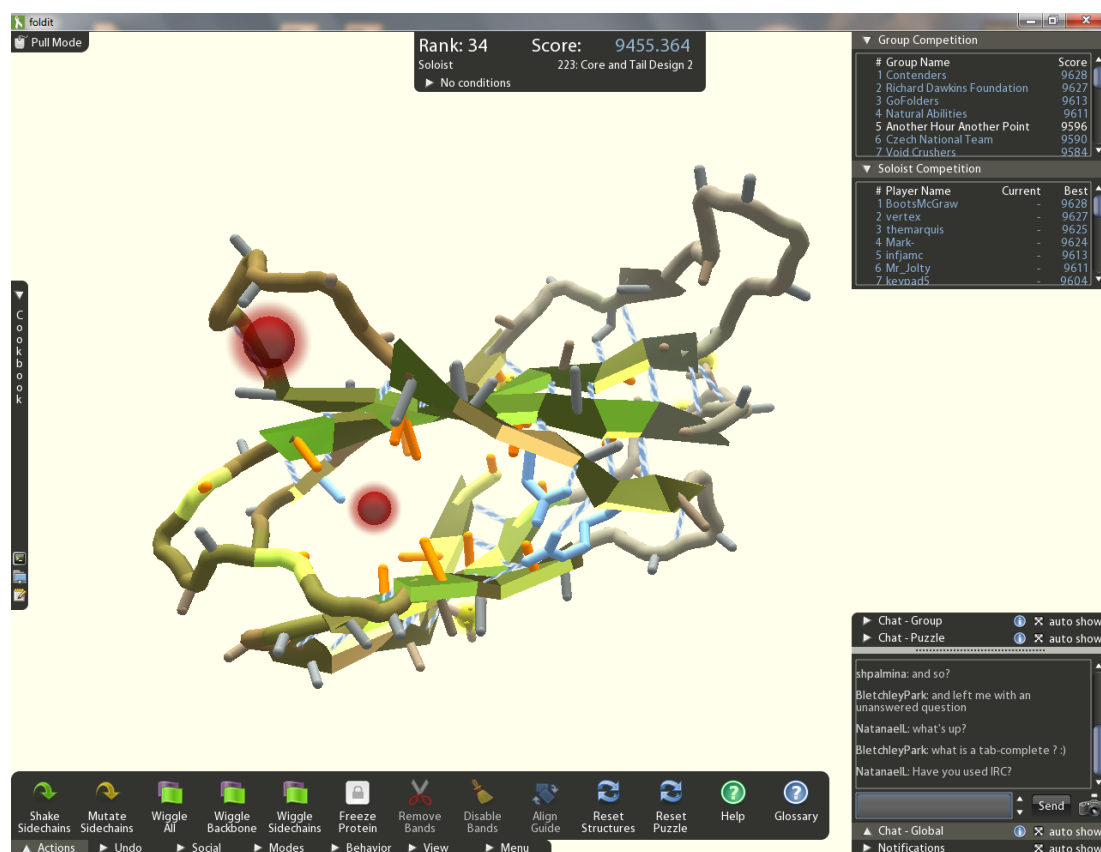


Рисунок 1.1. Геймлей українського аналогу Foldit

SimCity [4] – це гейміфікований вебсимулятор, який дозволяє гравцям будувати та керувати містами. Гра навчає гравців основам міського планування та економіки. SimCity є дуже популярною грою і використовується в освітніх цілях. Геймплей представлений на рисунку 1.2.

Основна мета гри – побудувати і розвинути успішне місто, забезпечивши його мешканців всім необхідним: від електроенергії та водопостачання до шкіл і розважальних закладів. Гравці можуть планувати розташування зон для житлової забудови, комерційних підприємств і промислових об'єктів, а також розвивати інфраструктуру, таку як дороги, міська залізниця та система громадського транспорту.

Одним із ключових елементів гри є збалансованість між потребами мешканців і економічними обмеженнями. Гравці повинні приймати стратегічні рішення щодо розподілу ресурсів і розвитку міста, враховуючи різні фактори, такі як екологія, безпека і задоволення потреб мешканців.

SimCity став класикою в жанрі міського будівництва і отримав широку популярність серед гравців у всьому світі. Ця серія вплинула на розвиток комп'ютерних ігор і залишається важливою частиною історії відеоігор до цього дня.



Рисунок 1.2. Геймплей SimCity

3. Freerice [5] – це гейміфікований вебсимулятор, який допомагає людям вивчити нову словниковий запас. Гравці відповідають на запитання з

множинним вибором, щоб вивчити нові слова. За кожну правильну відповідь Freerice жертвує 10 зерен рису Світовій продовольчій програмі, геймплей додатку представлений на рисунку 1.3.

У грі Freerice гравцям пропонується відповідати на питання з різних предметів, таких як мова, географія, наука, література та мистецтво. За кожну правильну відповідь гравець отримує зерно рису, а благодійна організація, яка співпрацює з Freerice, перераховує кошти, щоб допомогти людям, що страждають від голоду. Загальна сума зароблених зерен рису використовується для закупівлі продуктів харчування для тих, хто потребує допомоги.

Гра стала дуже популярною через свою простоту та благородну мету. Люди з усього світу можуть грати у Freerice безкоштовно і одночасно допомагати тим, хто опинився в скрутному становищі. Freerice надає можливість не лише вдосконалювати свої знання, а й робити світ кращим місцем через активну благодійну участь.

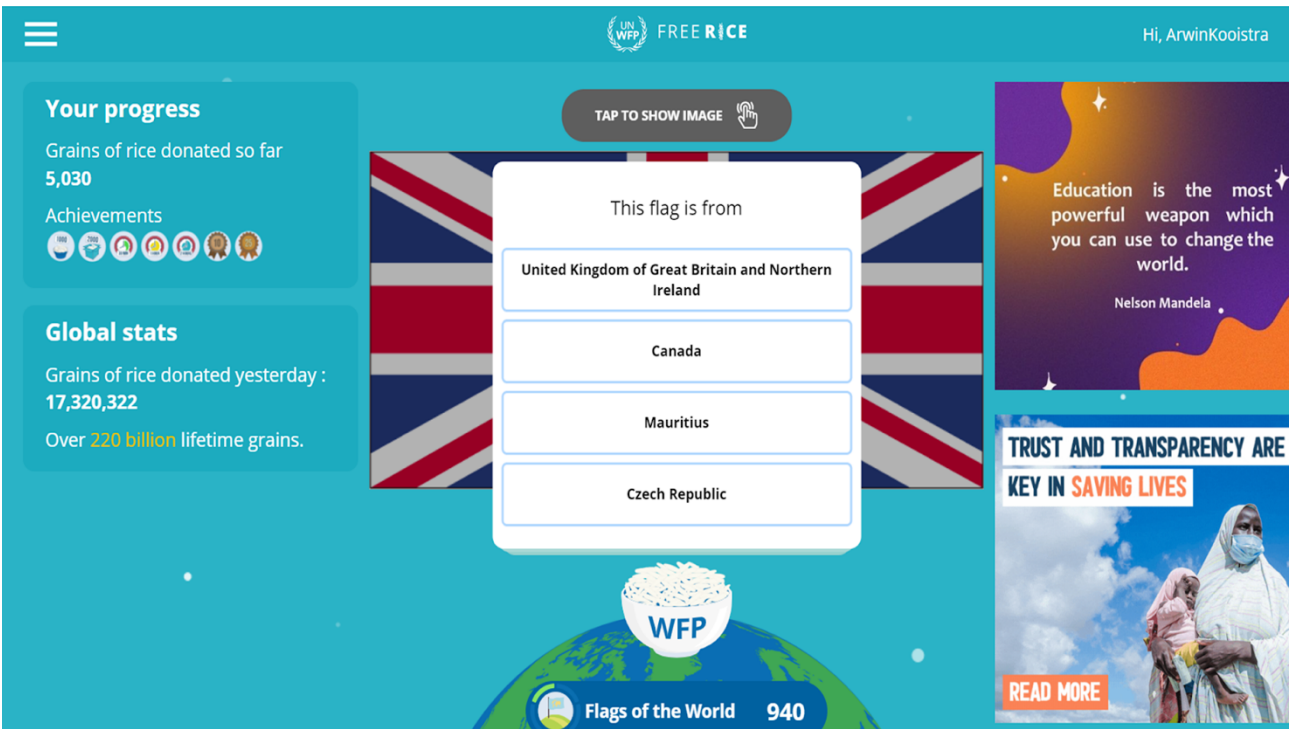


Рисунок 1.3. Геймплей Freerice

DragonBox Algebra 5+ [6] – це гейміфікований вебсимулятор, який допомагає дітям вивчати алгебру. Гра використовує ігри та головоломки, щоб навчити дітей основним алгебраїчним поняттям. DragonBox Algebra 5+ – це дуже популярна гра, яку використовують у школах по всьому світу. Геймплей гри представлений на рисунку 1.4.

У DragonBox Algebra 5+ гравцям пропонується розв'язувати математичні завдання, використовуючи різні ігрові елементи, такі як дракони та скрині. Головна ідея полягає в тому, щоб за допомогою різних дій, таких як додавання, віднімання, множення та ділення, «випроваджувати» драконів і отримувати правильні відповіді. Цей підхід допомагає дітям зрозуміти алгебраїчні концепції на практичному рівні, без необхідності вивчати складні формули.

DragonBox Algebra 5+ рекомендується для дітей від 5 років і старших, проте вона може бути корисною і для дорослих, які хочуть оновити свої математичні навички. Гра отримала велику популярність серед батьків та вчителів як ефективний інструмент для навчання алгебри в захопливій та доступній формі.

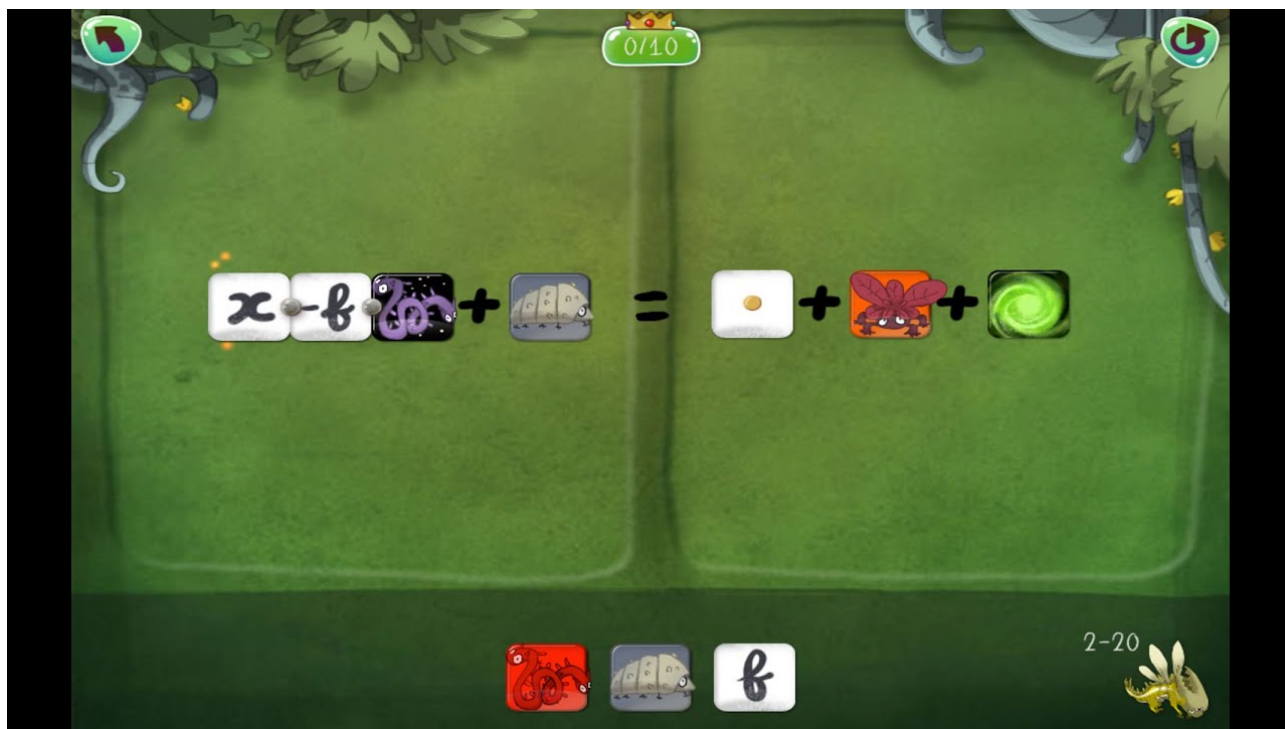


Рисунок 1.4. Геймплей DragonBox Algebra 5+

5. CodeCombat [7] – це гейміфікований вебсимулятор, який допомагає людям вивчати кодування. Гравці проходять рівні, пишучи код для керування персонажем. CodeCombat – це дуже популярна гра, яку використовують у школах та вдома. Геймплей представлений на рисунку 1.5.

На платформі CodeCombat гравці можуть виконувати завдання, розв'язувати головоломки та програмувати власний персонаж, щоб просуватися через ігровий світ. У процесі гри гравці навчаються основам програмування, таким як умовні оператори, цикли, функції та багато іншого, вирішуючи завдання, які вимагають програмувальних навичок для їх вирішення.

CodeCombat має широкий спектр курсів, які призначені для різних рівнів вивчення, від початкового до просунутого. Платформа підходить як для початківців, які тільки починають вивчати програмування, так і для більш досвідчених розробників, які хочуть поглибити свої знання або вивчити нові мови програмування.

Завдяки інноваційному підходу до навчання програмування через ігри, CodeCombat став популярним інструментом для навчання в школах, університетах і серед самостійних учнів.

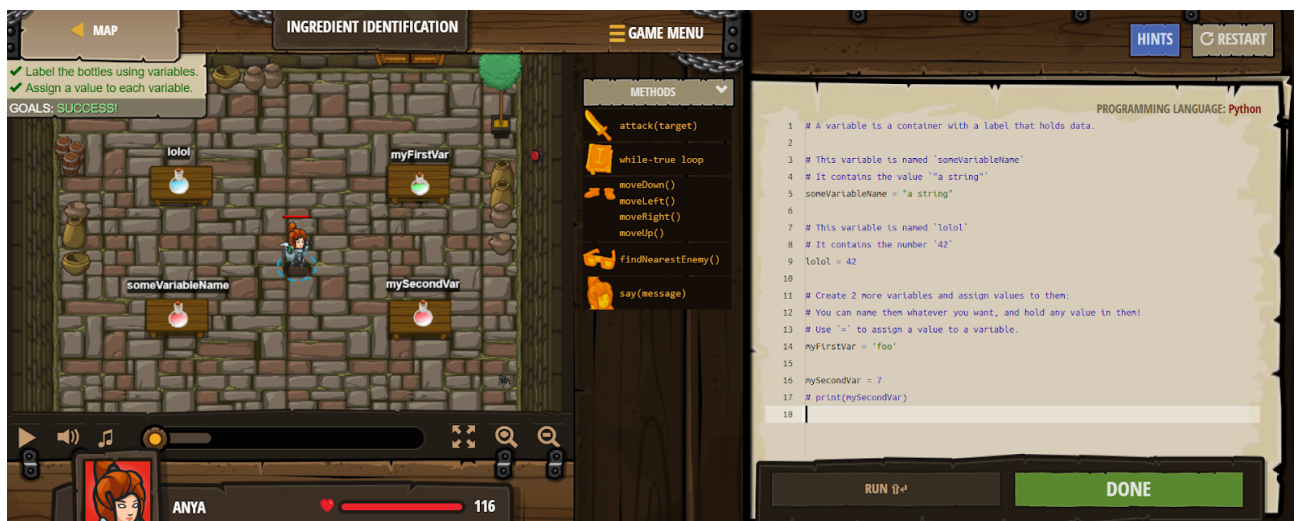


Рисунок 1.5. Геймплей одного рівня CodeCombat

Перелік аналогів системи

Назва	Розробник	Платформа	СУБД	Умови ліцензії
Foldit	Rosetta@home	Веббраузер	Python, MySQL	Ліцензія MIT
SimCity	Maxis	Мобільні пристрої	C++, PostgreSQL	Ліцензія Electronic Arts
Freerice	John Cook	Веббраузер	JavaScript, MongoDB	Creative Commons Attribution- ShareAlike 3.0 Unported
DragonBox Algebra 5+	DragonBox	Веббраузер	JavaScript, SQLite	Creative Commons Attribution- NonCommercial- ShareAlike 4.0 International
CodeCombat	CodeCombat	Веббраузер	Python, MySQL	Ліцензія CodeCombat
PokeGame	Олександр Петрук	Веббраузер	Typescript, Next.js, Nest.js, MongoDB	Ліцензія MIT

Функціональні можливості

	Foldit	SimCity	Freerice	DragonBox Algebra 5+	CodeCombat	PokeGane
Нагороди	Так	Так	Ні	Так	Так	Так
Соціальні функції	Так	Так	Ні	Так	Так	Так
Списки лідерів	Так	Так	Так	Так	Ні	Так
Підказки та інструкції	Так	Так	Ні	Ні	Ні	Так
Інтерактивні візуалізації	Так	Так	Так	Так	Так	Так

Всі представлені гейміфіковані вебсимулятори (див. табл. 1.1) мають свої особливості та переваги(див. табл. 1.2). Foldit та SimCity більше підходять для людей, які мають певні знання з біології або міського планування. Freerice та DragonBox Algebra 5+ є більш простими та доступними для людей з різним досвідом. CodeCombat може бути складним для людей без досвіду програмування, але він є чудовим способом навчитися програмувати в ігровій формі. Гейміфіковані вебсимулятори – це потужний інструмент, який може використовуватися для різних цілей. Вони можуть зробити навчання більш цікавим, захоплюючим та мотивуючим [8].

1.3 Вибір архітектури гейміфікованого вебсимулятора

Для реалізації системи управління гейміфікованим вебсимулятором було обрано клієнт-серверну архітектуру. У цій архітектурі клієнтські додатки взаємодіють з сервером, що забезпечує централізоване управління та обробку даних.

В клієнт-серверній архітектурі система розподіляється між клієнтською та серверною частинами. Клієнтська частина відповідає за інтерфейс користувача та збирає введені дані від користувачів. Серверна частина обробляє та зберігає дані, а також виконує бізнес-логіку системи. Приклад архітектури представлено на рисунку 1.6.

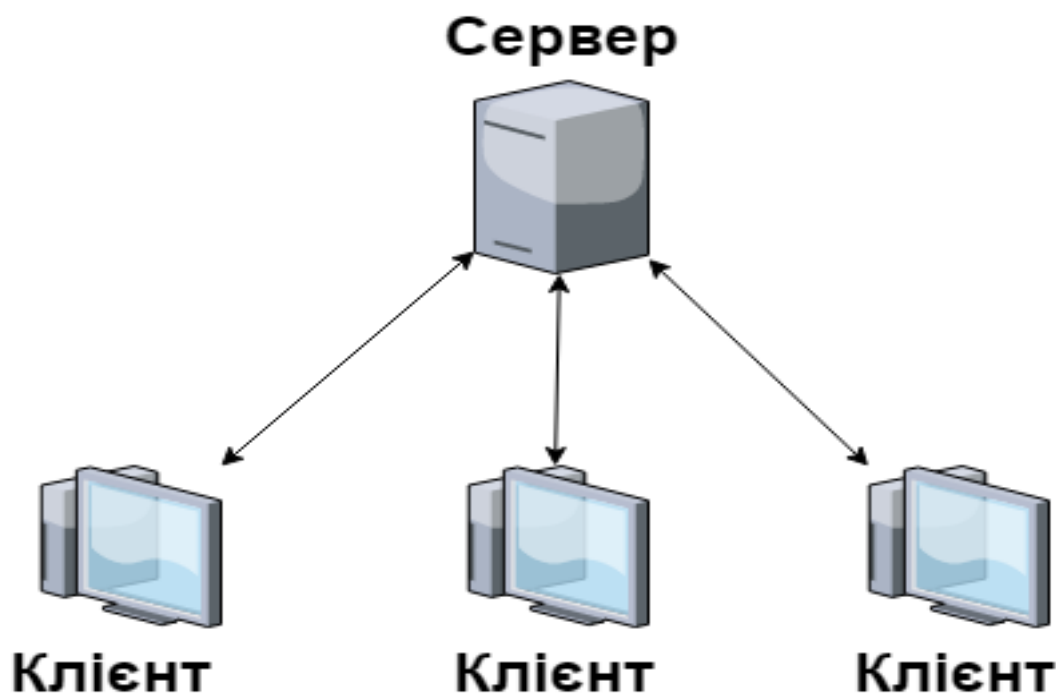


Рисунок 1.6. Клієнт-серверна архітектура

Гейміфікований вебсимулятор використовує клієнт-серверну архітектуру, де клієнтська частина взаємодіє з сервером для обміну даними та виконання бізнес-операцій. При розробці використовується патерн Медіатор.

Патерн Медіатор – це патерн проєктування ПЗ, який використовується для покращення взаємодії між об'єктами, що взаємодіють, шляхом введення

посередника, який координує їх спільну роботу. Основна ідея полягає в тому, щоб зменшити залежності між компонентами системи, перекладаючи комунікацію між ними через спеціальний об'єкт-медіатор, замість безпосередньої взаємодії.

Медіатор відокремлює компоненти системи один від одного, що робить код більш зрозумілим та підтримуваним. Він може обробляти всі повідомлення та події, що передаються між об'єктами, і вирішувати, як вони взаємодіють, тим самим зменшуючи залежності між ними. Цей патерн особливо корисний у великих системах, де збереження низького рівня залежностей допомагає забезпечити легку модифікацію та розширення програми [9].

Серверна модель відповідає за реалізацію бізнес-логіки симулятору на сервері. Вона обробляє запити від клієнтської частини і виконує необхідні операції, включаючи доступ до БД, обчислення та маніпуляції з даними. Серверна модель може використовувати підхід, який найкраще підходить для мого додатку, такий як ORM для роботи з базою даних.

На серверній стороні використовується патерн Медіатор для забезпечення взаємодії між різними компонентами системи. Медіатор виступає як центральний об'єкт, який отримує запити від клієнтського представлення та керує комунікацією між різними модулями на сервері, такими як модуль авторизації, модуль отримання даних користувачів та інші. Це дозволяє зменшити залежності між компонентами та полегшує розширення системи

Клієнтське представлення відповідає за відображення даних та інтерфейсу для користувача. Воно взаємодіє з сервером, надсилаючи запити та отримуючи відповіді. Клієнтське представлення може бути реалізоване у вигляді вебсторінок.

Використання клієнт-серверної архітектури та патерна Медіатор на серверній частині симулятору допомагає забезпечити гнучкість та легкість розширення сайту, а також дозволяє зручно враховувати зміни вимог користувачів та обробки інформації.

1.4 Обґрунтування вибору інструментальних засобів

Для створення гейміфікованого вебсимулятора будуть використовуватися наступні технології:

- мова розмітки гіпертекстових документів: HTML5;
- мова для стилів: CSS, Tailwind;
- TypeScript фреймворк: Next.js;
- серверна частина: мова програмування Node.js та фреймворк Nest.js;
- БД: MongoDB.

Ці технології дозволять розробляти вебсайт системи з використанням сучасних технологій та інструментів. HTML5 забезпечить семантичну розмітку вебсторінок, а Next.js – потужний фронтенд для інтерактивної взаємодії з користувачем та швидким роутингом.

Серверна частина буде побудована на базі мови програмування Node.js та фреймворка Nest.js, що дозволить створити масштабований серверний додаток. БД MongoDB забезпечить надійне управління даними.

HTML5 [10] – це мова розмітки гіпертексту, яка використовується для створення та відображення вебсторінок у браузерях. Вона є еволюційним кроком у розвитку мови HyperText Markup Language і включає в себе нові елементи, атрибути та можливості, які розширюють можливості веброзробки. HTML5 включає підтримку відео, аудіо, графіки, анімації, розробки ігор та багато іншого без необхідності використання додаткових плагінів або розширень браузера.

Tailwind CSS [11] – це потужна бібліотека Cascading Style Sheets, яка дозволяє швидко створювати користувацький інтерфейс, використовуючи набір готових класів. Замість написання власних CSS стилів, розробники можуть використовувати класи Tailwind для вирішення різних завдань стилізації. Наприклад, замість того, щоб написати CSS для задання розміру тексту або вирівнювання елементів, ви можете просто використати відповідні класи Tailwind. Tailwind також надає можливість розширити свій набір класів за допомогою конфігурації, щоб відповідати потребам конкретного проєкту. Це

робить процес розробки швидшим та ефективнішим, особливо для великих та складних проєктів.

TypeScript [12] – це мова програмування, розроблена компанією Microsoft, яка є розширенням JavaScript. Вона додає до JavaScript статичну типізацію, що дозволяє виявляти помилки в коді на етапі розробки та покращує зручність роботи з великими проєктами. Основні особливості TypeScript: статична типізація, об'єктно-орієнтоване програмування, інтеграція з JavaScript,

Загалом, TypeScript допомагає розробникам покращити якість свого коду, зробити його більш зрозумілим та ефективним, що робить його популярним в великих та серйозних проєктах.

React [13] – є JavaScript бібліотекою для створення інтерфейсів користувача. Вона дозволяє розробникам будувати вебзастосунки, які мають динамічний інтерфейс, шляхом розбиття його на компоненти. Кожен компонент в React може мати свою власну логіку та стан, і вони можуть бути вкладені один в одного для створення складних інтерфейсів.

Однією з ключових особливостей React є використання віртуального DOM (Document Object Model), що дозволяє ефективно маніпулювати станом інтерфейсу та автоматично оновлювати відображення, коли змінюється стан даних.

Next.js [14] – це фреймворк для реактивних вебзастосунків на базі React. Він надає набір інструментів і зручних функцій для швидкої та ефективної розробки вебзастосунків. Основні особливості Next.js включають: Роутинг, Клієнт-сайд рендерінг.

Node.js [15] – це вільне, відкрите середовище виконання JavaScript, яке дозволяє розробникам створювати серверні та мережеві застосунки з використанням JavaScript. Node.js використовує механізм подій для асинхронного вводу/виводу (I/O), що дозволяє створювати ефективні та масштабовані застосунки.

Основні особливості Node.js: асинхронність, платформонезалежність, потужний пакетний менеджер.

Node.js дуже популярний серед розробників і знаходить широке застосування у різних сферах, від веброзробки до розробки мобільних та настільних застосунків.

Nest.js [16] – це прогресивний фреймворк для побудови масштабованих та ефективних серверних застосунків на Node.js. Він базується на TypeScript і використовує концепції і шаблони проєктування, такі як Dependency Injection, Middleware та модульна структура, щоб допомогти розробникам створювати чистий, організований та легко розширюваний код.

Основні особливості Nest.js: модульність, middleware, підтримка TypeScript.

Загалом, Nest.js дозволяє розробникам швидко створювати масштабовані, ефективні та легко підтримувані серверні застосунки на Node.js.

MongoDB [17] – це документо-орієнтована БД, яка використовує NoSQL (неструктурований) підхід до зберігання та організації даних. У відмінність від традиційних реляційних баз даних, таких як MySQL або PostgreSQL, MongoDB зберігає дані у вигляді документів у форматі BSON (Binary JSON), що є розширенням формату JSON.

Основні особливості MongoDB: документо-орієнтована модель даних, гнучкість, висока швидкодія, масштабованість, доступність та надійність.

MongoDB широко використовується в сучасних веб та мобільних додатках, а також в різних сферах, від аналітики даних до систем керування контентом.

WebStorm [18] – це інтегроване середовище розробки (IDE) від компанії JetBrains, призначене для веброзробки на базі JavaScript, HTML та CSS, а також для роботи з іншими технологіями, такими як TypeScript, Angular, React і Node.js. WebStorm надає розширені можливості для написання коду, відлагодження, аналізу проєктів та управління версіями.

Основні особливості WebStorm: підтримка великої кількості мов програмування та технологій, автодоповнення та інспекція коду, інтеграція з системами керування версіями, розширені можливості рефакторингу.

Для розробки та тестування React додатків знадобиться комп'ютер, який може ефективно працювати з різними інструментами розробки ПЗ. Ось кілька ключових вимог:

Продуктивний процесор: Рекомендується мінімум чотирьохядерний процесор. Швидкий процесор допоможе покращити час компіляції та завантаження проєктів.

Пам'ять (RAM): Мінімум 8 гігабайт оперативної пам'яті. Більше пам'яті дозволить вам запускати більше програм одночасно та працювати з великими проєктами без затримок.

Накопичувач (SSD): Рекомендується використовувати твердотільний накопичувач (SSD), оскільки він пропонує швидкий доступ до даних та покращить час завантаження проєктів.

Графічна карта: Для розробки React додатків не потрібна велика потужність графічної карти. Вбудована графіка або базова відеокарта буде достатньою.

Операційна система: Більшість інструментів розробки, таких як Visual Studio Code, Node.js, і браузер, підтримуються на Windows, macOS та Linux. Виберіть операційну систему, з якою ви найбільш зручно працюєте.

Інтернет-з'єднання: Хоча необов'язкове, швидке Інтернет-з'єднання допоможе вам швидше завантажувати бібліотеки та пакети з Інтернету, що може бути важливим під час розробки.

Висновки до розділу 1

Проведений аналіз предметної області дозволив визначити основні напрямки реалізації вебсимулятора. Необхідно реалізувати збереження даних в БД про користувачів, користувацьких боїв, список найкращих гравців, а також зміна в реальному часі рейтингу, кількості монет.

Основними вимогами є зручне управління, зручний інтерфейс, масштабованість проєкту. Розробка гейміфікованого вебсимулятора має на меті забезпечити вимоги та покращити враження від симулятора.

					ІПЗК.КР.Б-121-24-ПЗ	
						24

В роботі обґрунтовано вибір архітектури для реалізації системи:

- БД: MongoDB;
- мова програмування на серверній частині: Node.js;
- фреймворк: Nest.js;
- мова програмування на клієнтській частині: TypeScript;
- фреймворк на клієнтській частині: Next.js;
- мова процесора для стилів: CSS.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ГЕЙМИФІКОВАНОГО ВЕБСИМУЛЯТОРУ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Гейміфікований вебсимулятор має наступні основні цілі: створення захопливого середовища, яке стимулює користувачів до взаємодії та повернення до симулятору, встановлення цілей, нагород та рівнів, що стимулюють користувачів до активності та досягнення результатів, забезпечення користувачам можливості розважатися та відпочивати від повсякденних справ через ігровий процес та дослідити всесвіт покемонів. Симулятор зберігає інформацію про користувачів та їх активність. Крім того, симулятор має бути зручним та доступним, з інтуїтивним інтерфейсом, який можна використовувати з різних пристроїв, та платформ. На рисунку 2.1. зображено побудовану діаграму базових варіантів використання.

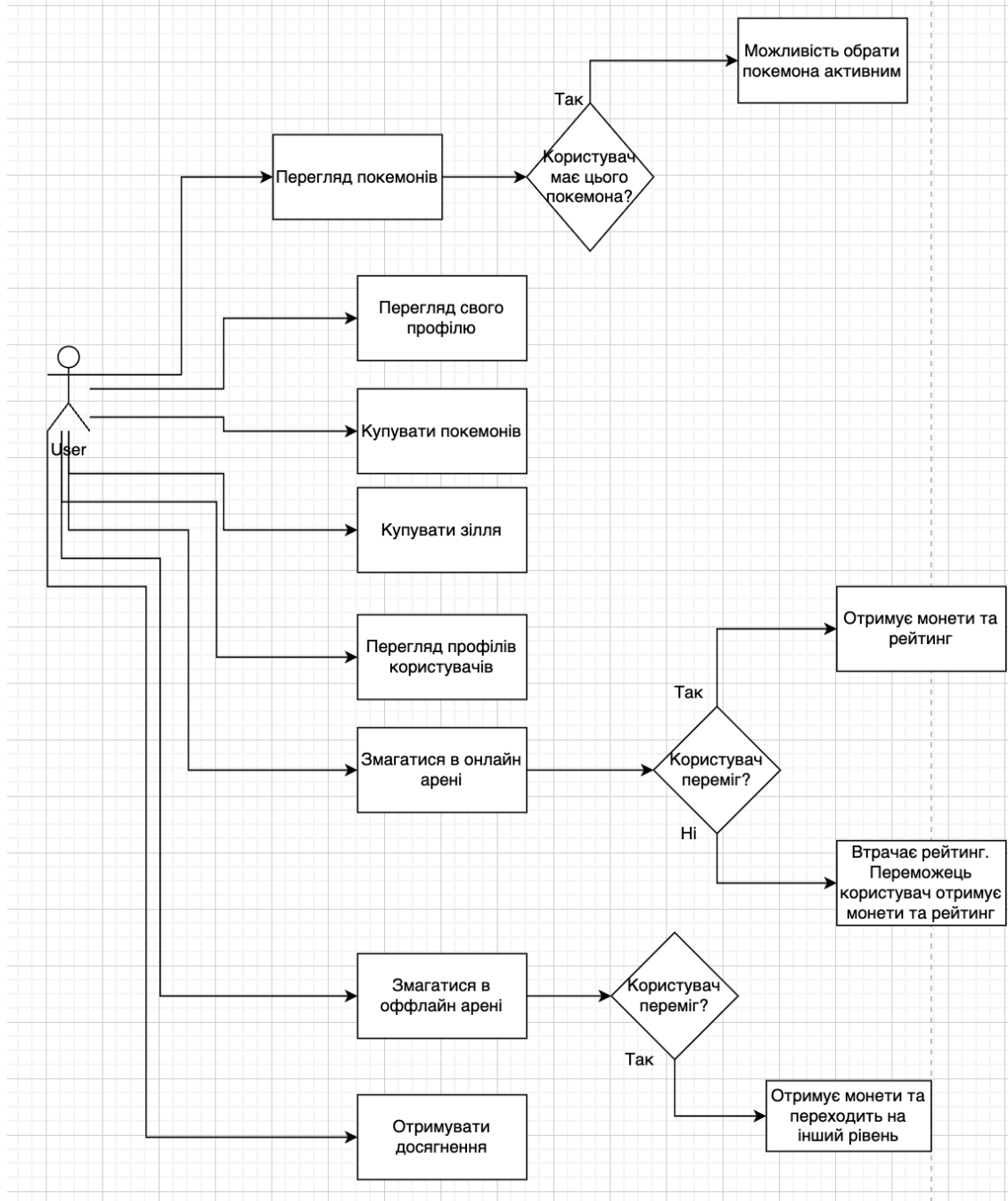


Рисунок 2.1. Діаграма базових варіантів використання

Користувачам надається ряд функцій що включає:

1. Можливість переглядати покемонів по списку, або по типам. Також кожного покемона можна обрати і переглянути його здібність, здоров'я та силу. Якщо покемон який був обраний є в бібліотеці в користувача, але він не є активний до битви, користувач може активувати його.

2. Можливість перегляду особистого профілю. В користувача є можливість подивитися загальну кількість досягнень користувачу яку залишилося виконати, активного покемону, загальну кількість покемонів в бібліотеці, очки рейтингу, історію боїв. Також є можливість зміна імені, паролю, пошти, та фото аккаунту. В користувача є можливість видалити незворотно аккаунт.

3. Можливість купувати покемонів. Коли користувач купує покемона йому дається можливість отримати будь-якого покемона. Якщо користувач має схожого покемона який є в колекції він отримує додаткові монети.

4. Можливість купування зілля. Користувач може покращити свого активного покемона на одну гру в арені придбавши зілля в магазині.

5. Можливість перегляду профілю користувача. Можливість користувачеві дивитися на особисті кабінети інших користувачів де знаходиться загальна кількість досягнень користувача, активного покемону, загальну кількість покемонів, кількість рейтингу, історію боїв.

6. Змагання в онлайн арені. Користувач може обрати гравця проти якого він хоче почати битву. Під час битви користувач може втекти і не втратити ніякі ресурси або програти битву і втратити рейтинг або перемогти і отримати рейтинг і монети.

7. Змагання в оффлайн арені дає можливість отримати додаткові монети якщо в онлайн арені користувач не хоче втратити рейтинг. В оффлайн арені представлені всі покемони зі всесвіту покемонів. При перемозі користувач отримує монети.

8. Користувач отримує досягнення за активність в симуляторі. Користувач може переглянути які в нього залишились невиконані досягнення та які він вже виконав, під час різних дій в симуляторі користувач може отримати досягнення.

Симулятор складається з трьох незалежних сервісів, кожен з яких налічує 3-4 модулі. Загалом імплементовано 10 модулів. Кожен з модулів відповідає

функціоналу на діаграмі прецедентів, а також реалізовує критерії доступності WCAG на рівні 2.1 (AA) детальна діаграма на рисунку 2.2.

Цікавими модулями є модуль авторизації та аутентифікації. Новий користувач може зареєструватися у вебдоступний ігровий симулятор, і отримати випадкового покемона і 20 монет для подальшого розвитку.

Зареєстрованому користувачеві дається змога детальніше дізнатися про всесвіт покемонів, вивчаючи і вдосконалюючи його. Зареєстрований користувач може відвідувати симуляції онлайн арени, де ознайомитись з стратегіями розвитку персонажу (покращення за рахунок бою та тактик), користувач може змагатися проти інших користувачів, і отримувати нагороди за перемоги у битвах у вигляді рейтингу та монет. Якщо користувач не хоче втрачати рейтинг в арені, є можливість тренування навичок у оффлайн арені, де представленні покемони різних типів та здібностей. Симуляція тренування у оффлайн арені спрямована на покращення та розвиток механік бою. Існує система винагород у вигляді монет. Монети дозволяють покращувати властивості покемонів, або ж придбати нового в залежності від розвитку стратегічних цілей. Одним з варіантів використання є зілля, яке покращує характеристики на один матч. Також існує система винагород та статистики, яка відображає різноманітну інформацію, зокрема, рейтинг покемонів, найкращих гравців, відсоток виграних битв тощо.

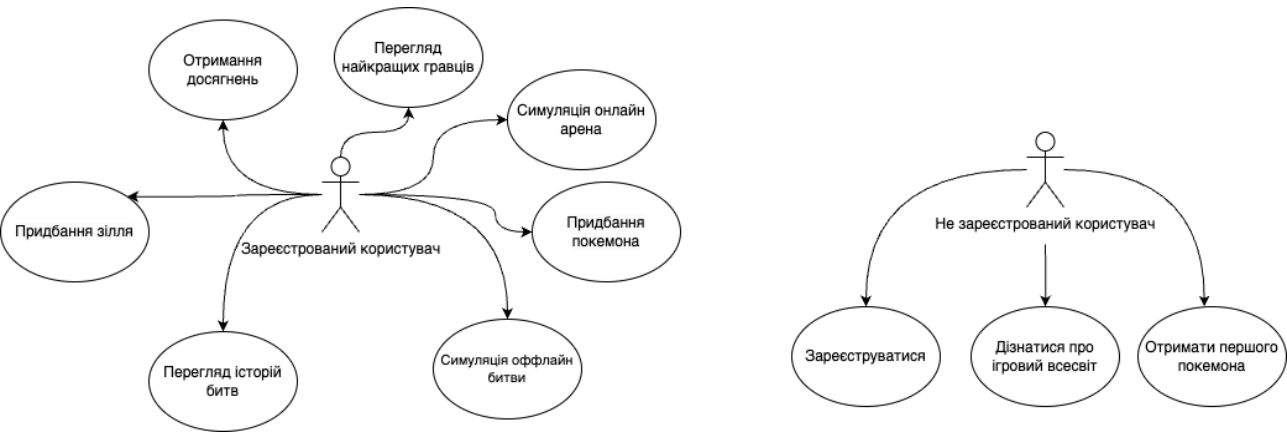


Рисунок 2.2. Діаграма прецедентів

Функціональні вимоги:

Гейміфікований вебсимулятор надає користувачам такі можливості:

1. Реєстрація аккаунту та логізація.
2. Отримання першого покемону.
3. Перегляд покемонів.
4. Пошук покемонів.
5. Відвідування особистого аккаунту.
6. Відвідування кабінетів інших користувачів.
7. Перегляд найкращих користувачів.
8. Перегляд історії битв.
9. Зміна паролю, логіну, пошти, фото.
10. Видалення профілю.
11. Купівлю покемонів та зілля.
12. Отримання досягнень.
13. Відвідувати оффлайн арену.
14. Заробляння монет.
15. Заробляння рейтингу.
16. Змагання в онлайн арені.
17. Використання зілля.
18. Отримання сповіщення за отримане досягнення.

Нефункціональні вимоги:

Навчання роботи з системою:

- Час навчання основних механік: для звичайних користувачів 1 година для досвідчених 0.3 години;
- Неперервна робота: безперебійна робота на протязі 30 робочих днів.
- Інтуїтивний інтерфейс: легкий у використанні, не потребує додаткової підготовки користувача;
- Норма помилок: Максимальна кількість 1 помилка на 10000 запитів користувачів;
- Надійність: стабільна та надійна робота системи без збоїв;

					ІПЗК.КР.Б-121-24-ПЗ	
						30

- **Продуктивність:** Продуктивність системи включає в себе ряд параметрів, таких як швидкість реакції, ефективна обробка даних, масштабованість, економне використання ресурсів, оптимізація запитів, паралельна обробка, надійність і ефективне управління ресурсами.

Очікується, що система буде забезпечувати відповідь протягом 1-3 секунд, ефективно використовувати ресурси та забезпечувати надійну роботу. Також важливою є масштабованість системи для оптимальної роботи з великими обсягами даних і багатьма користувачами.

Крім того, система має мати інтуїтивно зрозумілий інтерфейс, який не потребує додаткової підготовки користувача. Час, необхідний для освоєння інструментів роботи з системою, для звичайних користувачів складає 1-2 години, а для досвідчених – 1 година.

2.2 Розробка бази даних системи

Була реалізована БД MongoDB яка складається з 2 таблиць. В них зберігається інформація про користувачів та про битви.

Таблиця users включає в себе основну інформацію про користувачів (див. табл. 2.1).

Таблиця 2.1

Опис полів таблиці Users

Назва	Тип даних	Опис поля
_id	ObjectId	Унікальний номер для користувача яке видає БД.
name	string	Унікальне ім'я користувача яке користувач вводить при реєстрації. Влюбий момент може змінити, тільки на унікальне ім'я.

Продовження Таблиця 2.1. Опис полів таблиці Users

email	string	Унікальна пошта яку користувач вводить при реєстрації. Користувач має можливість змінити пошту в будь-який момент при умові того що пошта буде унікальна.
password	string	Пароль користувач вводить при реєстрації. Далі пароль шифрується щоб шахраї не мали змоги дізнатись пароль. Також можна змінити.
img	string	Фото профіля користувача. Користувач при реєстрації отримує фото але в будь-який момент користувач може змінити його.
selectedPokemon	string	Обраний покемон користувача. В будь-який момент користувач може його змінити.
coins	number	Число монет які користувач має.
rang	number	Кількість рангу якого має користувач.
stageOfOfflineArena	number	Рівень оффлайн арени користувача.
arrPokemons	string[]	Масив покемонів які має користувач.
arrAchives	number[]	Масив досягнень які має користувач.
arrPotions	object[]	Масив зілей які має користувач, туди входить, унікальний номер, назва, та кількість зілей.

Таблиця fights включає в себе основну інформацію про битви користувачів (див. табл. 2.2).

Таблиця 2.2

Опис полів таблиці Fights

Назва	Тип даних	Опис поля
_id	ObjectId	Унікальний номер для користувача яке видає БД.
whoStart	string	Ім'я гравця хто почав битву
whoDefence	string	Ім'я гравця хто захищався
whoWin	string	Ім'я гравця хто переміг
timeFight	Date	Час коли була битва

Отже реалізована БД у відповідності до вимог моделі надає зручний та ефективний спосіб збереження інформації про користувачів та боїв. З використанням таблиць, таких як Users, Fights можна організувати та управляти даними, забезпечуючи зручний інструмент для роботи з інформацією про користувачів та бої. БД складається з 2 таблиць.

2.3 Проєктування та реалізація алгоритмів роботи системи

Так як обраною архітектурою для вебсимулятора є клієнт-серверна, то вебсимулятор має логічне розподілення на клієнтську та серверну частини і передбачає використання RESTfull API.

RESTful API – це архітектура інтерфейсу прикладних програм (API), що використовує HTTP-запити для доступу до ресурсів та їхнього використання. Такими HTTP-запитами є GET, PUT, POST і DELETE. Вони дають змогу, відповідно, читати, змінювати, створювати й видаляти ресурси [19].

Основні принципи REST API включають: клієнт-серверна модель, безстандартність, кешування, інтерфейс за запитами, шари системи, код за запитом,

REST API використовується для побудови вебсервісів, що надають доступ до даних та функціональності програм через Інтернет. Він дозволяє різним програмним компонентам, розробленим на різних мовах програмування та працюючим на різних платформах, спілкуватися між собою, обмінюючись даними за стандартами вебпротоколів. Детальна схема роботи REST API представлено на рисунку 2.3.

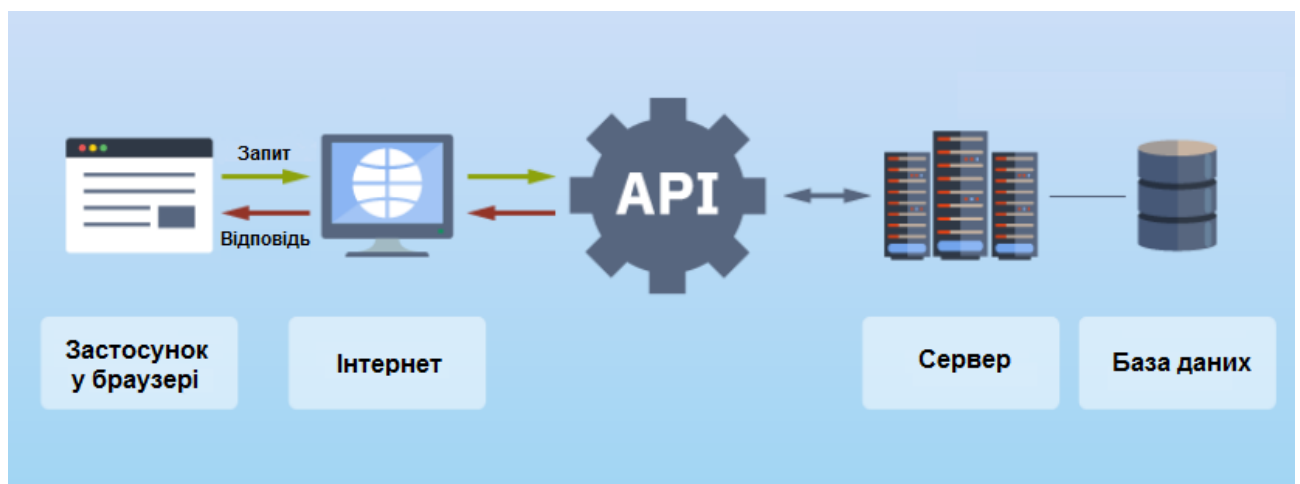


Рисунок 2.3. Відображення роботи REST API

Також для зображення алгоритму роботи вебсимулятора створимо структурну діаграму.

Структурна діаграма – це графічне зображення структури системи, що відображає складові її частини та зв'язки між ними. Цей тип діаграми використовується для моделювання взаємозв'язків між об'єктами або елементами системи, їх ієрархії та організації. Структурні діаграми можуть бути використані в різних галузях, включаючи ПЗ, інформаційні системи, бізнес-аналіз, технічне проектування та інші області. Основними елементами структурної діаграми можуть бути об'єкти, класи, модулі, компоненти або будь-які інші складові частини системи, а також зв'язки між ними. За допомогою

структурних діаграм можна відобразити, наприклад, ієрархію класів в програмному забезпеченні, структуру БД, організаційну структуру компанії, зв'язки між компонентами апаратного забезпечення тощо. Структурні діаграми допомагають зрозуміти складові частини системи та їх взаємозв'язки, що полегшує процес розробки, аналізу та управління системою. Структурна діаграма проєкту зображена на рисунку 2.4.

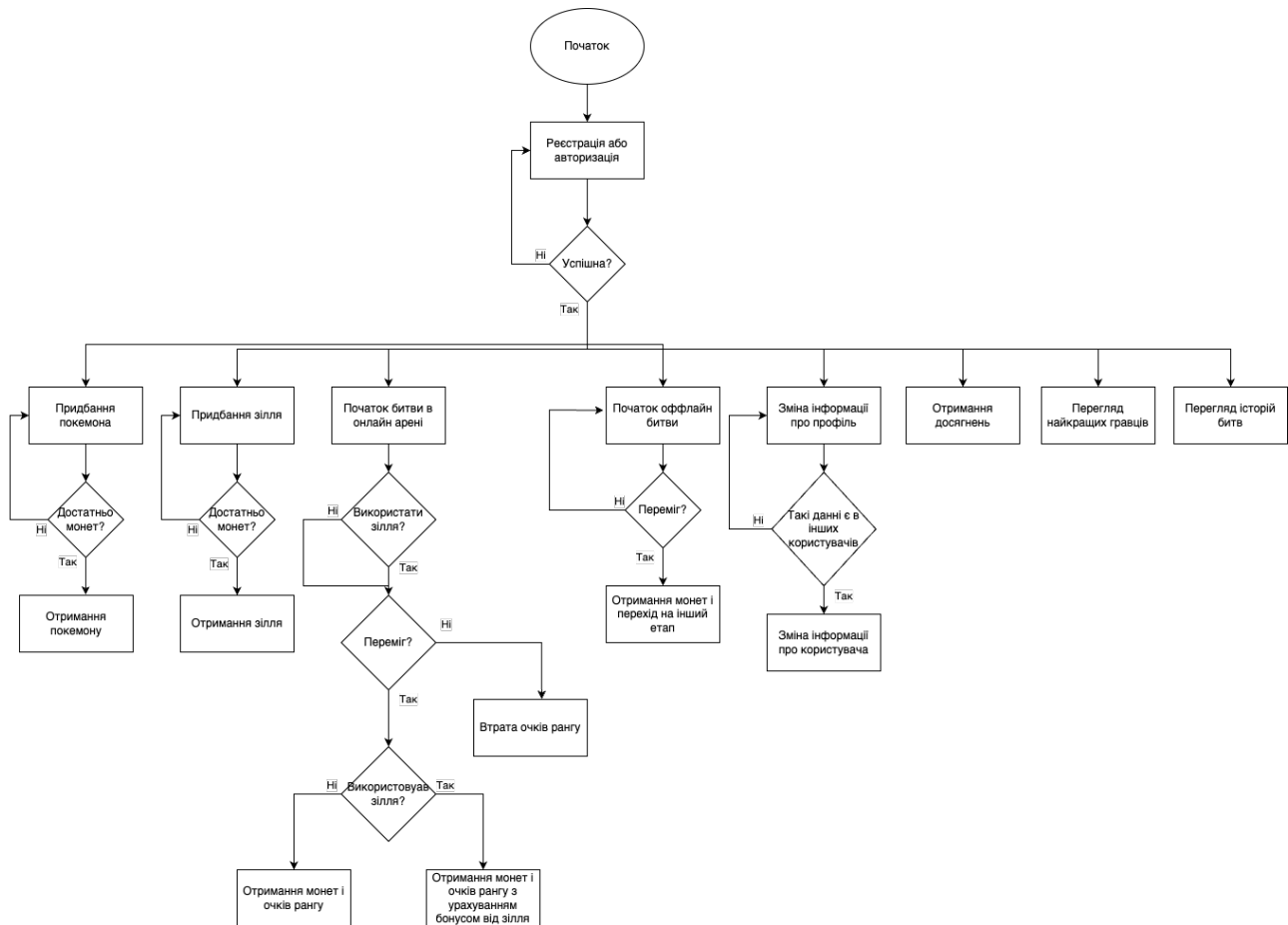


Рисунок 2.4. Структурна діаграма проєкту

Опис діаграми:

1. Користувач з’являється на сторінці логіну, якщо користувач не має аккаунту може перейти на сторінку реєстрації.
2. Вхід користувача(так – данні валідні користувач перехід даліше, ні – з’являється повідомлення користувачеві чому виникла помилка).
3. Потрапляння до головного меню.
4. Придбання покемона – користувач може придбати покемона в магазині якщо достатньо монет.

5. Придбання зілля – користувач може придбати зілля в магазині якщо достатньо монет.

6. Початок битви в онлайн арені – користувач обирає гравця проти якого хоче почати битву, перед початком битви переглядає покемона противника.

7. Втрата очків рангу – якщо користувач програв битву, він втрачає очки рангу, при цьому противник отримує ці очки рейтингу та монети.

8. Отримання монет і очків рангу – при перемозі користувач отримує очки рейтингу та монети, противник втрачає очки рейтингу.

9. Отримання монет і очків рангу з урахуванням бонусу від зілля – якщо під час битви користувач може прийняти зілля яке збільшує кількість рейтингу або монет за перемогу. При виграші користувач отримує збільшену нагороду.

10. Початок оффлайн битв – користувач бачить противника бота проти якого буде битва. Якщо користувач програв битву, дається змога почати знову битву проти противника.

11. Отримання монет та перехід на інший етап – при перемозі в оффлайн арені користувач отримує наступного противника, та монети за перемогу.

12. Зміна інформації про профіль – користувач може влюбий момент змінити інформацію в профілі, мову вебсимулятора, ім'я користувача, фото профілю, пароль, електрону адресу. При зміні імені, пошти, відбувається перевірка на існуючого користувача, якщо користувач з такими даними існує, видається помилка.

13. Зміна інформації про користувача – якщо при змінні даних в користувача було введено все коректно данні користувача зміняться в БД і автоматично на вебсайті.

14. Отримання досягнень – під час проходження вебсимулятора користувач може отримувати досягнення, наприклад за переможені бої, покемонів в колекції і тд.

15. Перегляд найкращих гравців – при проходження геймифікованого вебсимулятора користувач отримує рейтинг, та нових покемонів. Список

найкращих користувачів які мають найбільшу кількість покемонів або рейтингу можна переглянути на відповідній вебсторінці.

16. Перегляд історії боїв – в профілі користувача можна переглянути повну історію боїв в онлайн арені.

2.4 Реалізація гейміфікованого вебсимулятора

На самому початку стало створення реєстрації користувача, код представлено на рисунку 2.5. При реєстрації перевіряється унікальність імені та електронної пошти користувача. Якщо з введеними даними є проблема, сервер відправляє помилку до вебсимулятору яка буде відображатися користувачеві. Якщо з унікальністю даних все коректно, пароль користувача шифрується та записується в БД. Після успішного записування даних в БД користувач отримує на виході профіль разом з токеном, для подальших запитів.

```
15  async create(dto: CreateUserDto): Promise<{user: User, access_token: string }>{
16      const isUnicalName = await this.userModel.findOne({name: dto.name})
17      const isUnicalEmail = await this.userModel.findOne({email: dto.email})
18      if(isUnicalName){
19          throw new HttpException({
20              status: HttpStatus.NOT_ACCEPTABLE,
21              error: 'This name is used',
22          }, HttpStatus.NOT_ACCEPTABLE);
23      }
24      if(isUnicalEmail){
25          throw new HttpException({
26              status: HttpStatus.NOT_ACCEPTABLE,
27              error: 'This email is used',
28          }, HttpStatus.NOT_ACCEPTABLE);
29      }
30      if(isUnicalName === null && isUnicalEmail === null){
31          const hashPassword = await bcrypt.hash(dto.password, 3)
32          const {password, ...result} = dto;
33          const user = await this.userModel.create({...result, password: hashPassword, ...INIT_USER})
34          const payload = {sub: user._id, username: user.name}
35          const token = await this.jwtService.signAsync(payload)
36          return {
37              user,
38              access_token: token
39          }
40      }
41  }
```

Рисунок 2.5. Код створення користувача.

Якщо користувач вже має аккаунт він може авторизуватися, код представлено на рисунку 2.6. Якщо пароль користувача не буде відповідати тому

що користувач надіслав, він отримає помилку. Також як і з реєстрацією користувач отримує токен та інформацію про профіль.

```
42 | async login(dto: CreateUserDto): Promise<{
43 |   user: User,
44 |   access_token: string }> {
45 |   try{
46 |     const user = await this.userModel.findOne({name: dto.name});
47 |     const isMatch = await bcrypt.compare(dto.password, user.password);
48 |     const isTrueEmail = user.email === dto.email;
49 |     if(!isTrueEmail){
50 |       throw new HttpException({
51 |         status: HttpStatus.NOT_ACCEPTABLE,
52 |         error: 'Wrong email',
53 |       }, HttpStatus.NOT_ACCEPTABLE);
54 |     }
55 |     if(!isMatch){
56 |       throw new HttpException({
57 |         status: HttpStatus.NOT_ACCEPTABLE,
58 |         error: 'Wrong password',
59 |       }, HttpStatus.NOT_ACCEPTABLE);
60 |     }
61 |     const payload = {sub: user._id, username: user.name}
62 |     const token = await this.jwtService.signAsync(payload)
63 |     return {
64 |       user,
65 |       access_token: token
66 |     }
67 |   }
68 |   catch (e){
69 |     throw new HttpException({
70 |       status: HttpStatus.NOT_ACCEPTABLE,
71 |       error: 'Wrong credentials',
72 |     }, HttpStatus.NOT_ACCEPTABLE);
73 |   }
74 | }
```

Рисунок 2.6. Код авторизації користувача.

Якщо користувач хоче почати профіль з початку або просто не хоче щоб його аккаунт існував в вебсимуляторі. Користувач може в будь-який момент видалити свій профіль. Код представлений на рисунку 2.7.

```
75 | async delete(id: ObjectId): Promise<ObjectId>{
76 |   const user = await this.userModel.findByIdAndDelete(id);
77 |   return user.id
78 | }
```

Рисунок 2.7. Код видалення аккаунту користувача

Якщо користувач не впевнений у безпеці свого паролю в будь-який момент користувач може змінити свій пароль. Код представлений на рисунку 2.8.

```

79   async changePassword(id: ObjectId, password: string): Promise<User> {
80       const updatedPassword: string = await bcrypt.hash(password, 3);
81       const user = await this.userModel.findById(id);
82       user.password = updatedPassword;
83       user.save()
84       return user
85   }

```

Рисунок 2.8. Код зміни паролю користувача

Якщо користувач хоче змінити електронну адресу свого профілю. Він зможе змінити її з умовою того що це буде унікальна електронна адреса. Код представлений на рисунку 2.9.

```

101  async changeEmail(id: ObjectId, email: string): Promise<User> {
102      const isUnicalUser = await this.userModel.findOne({email: email});
103      if(isUnicalUser){
104          throw new HttpException({
105              status: HttpStatus.NOT_ACCEPTABLE,
106              error: 'This email is used',
107          }, HttpStatus.NOT_ACCEPTABLE);
108      }
109      if(isUnicalUser === null){
110          const user = await this.userModel.findById(id);
111          user.email = email;
112          user.save()
113          return user
114      }
115  }

```

Рисунок 2.9. Код зміни електронної адреси користувача

Якщо користувач хоче змінити фото свого профілю. Він зможе зміни її вказавши посилання на фото. Код представлений на рисунку 2.10.

```

116  async changeImg(id: ObjectId, img: string): Promise<User> {
117      const user = await this.userModel.findById(id);
118      user.img = img;
119      user.save();
120      return user
121  }

```

Рисунок 2.10. Код зміни фото профілю користувача

В гейміфікованому вебсимуляторі є система рейтингу по якому вибираються найкращі користувачі з найбільшим рейтингом. Рейтинг може змінюватися якщо користувач програв або виграв битву. Битва може бути почата

користувачем, так і на користувача може напасти інший користувач. По результатам битви переможений втрачає рейтинг а той хто переміг отримує рейтинг. Код представлений на рисунку 2.11.

```

122     async changeRangById(id: ObjectId, rang: number): Promise<User>{
123         const user = await this.userModel.findById(id);
124         if(user.rang === 0 && rang < 0){
125             return user
126         }
127         user.rang = rang;
128         user.save()
129         return user
130     }

```

Рисунок 2.11. Код зміни рейтингу користувача

Однією з основних механік є додавання покемона користувачеві в бібліотеку, код представлено на рисунку 2.12. При доданні сервер перевіряє чи користувач не має такого самого покемона в бібліотеці.

```

131     async addPokemon(id: ObjectId, idPokemon: string): Promise<User>{
132         const user = await this.userModel.findById(id);
133         if(user.arrPokemons.findIndex(pokemon => pokemon === idPokemon) === -1){
134             user.arrPokemons.push(idPokemon);
135             user.save();
136             return user;
137         }
138     }

```

Рисунок 2.12. Код додавання користувачеві нового покемона

Під час битви, проти реальних користувачів та ботів, користувач може застосувати зілля, . Код представлений на рисунку 2.13.

```

148     async setPotions(id: ObjectId, potion: Potions[]): Promise<User>{
149         const user = await this.userModel.findById(id);
150         user.arrPotions = potion;
151         user.markModified('arrPotions');
152         await user.save();
153         return user
154     }

```

Рисунок 2.13. Код додавання користувачеві зілля

Однією з основних механік в оффлайн арені є зміна етапу, при перемозі бота в арені користувач переходить на наступний етап, ці дані записуються в БД. Код представлений на рисунку 2.14.

```

155     async changeStage(id: ObjectId, stage: number): Promise<User>{
156         const user = await this.userModel.findById(id);
157         user.stageInOfflineArena = stage;
158         user.save();
159         return user
160     }

```

Рисунок 2.14. Код переходу користувача на наступний етап арені

Також основою вебсимулятору є застосування активного покемону, активний покемон буде використовуватися в битвах та показуватися в профілі користувача. Код представлений на рисунку 2.15.

```

161     async changeCurrentPokemonById(id: ObjectId, pokemonId: string): Promise<User>{
162         const user = await this.userModel.findById(id);
163         user.selectedPokemon = pokemonId;
164         user.save();
165         return user
166     }

```

Рисунок 2.15. Код встановлення активного покемона

Головною валютою в вебсимуляторі є монети, за які користувач придбає зілля, нових покемонів або покращення існуючого покемона. Ця функція буде зручною як для додавання так і відняття монет. Код представлений на рисунку 2.16.

```

167     async changeCountOfMoney(id: ObjectId, money: number): Promise<User>{
168         const user = await this.userModel.findById(id);
169         user.coins = money;
170         user.save();
171         return user
172     }

```

Рисунок 2.16. Код зміни числа монет в користувача

Для заохочення користувачів більше проводити час в симуляторі, була додана механіка досягнень. Користувач може отримати досягнення за будь-яку активність. Також інші користувачі можуть побачити в кабінеті відсоток прогресії досягнень. Код представлений на рисунку 2.17.

```

173     async addAchivesById(id: ObjectId, index: number): Promise<User> {
174         const user = await this userModel.findById(id);
175         if(user.arrAchives.findIndex(achive => achive === index) === -1){
176             user.arrAchives.push(index);
177             user.save();
178             return user
179         }
180     }

```

Рисунок 2.17. Код додавання досягнень користувачеві

Для побудови найкращих користувачів або списку користувачів для арени була створена функція отримання всіх активних користувачів, які мають активного покемона. Код представлений на рисунку 2.18.

```

181     async getAll(): Promise<User[]> {
182         const users = await this userModel.find()
183         const usersWithPokemons: User[] = []
184         users.map((user) => {
185             if(user.selectedPokemon !== '' && user.rang >= 0){
186                 usersWithPokemons.push(user);
187             }
188         })
189         return usersWithPokemons
190     }

```

Рисунок 2.18. Код отримання всіх користувачів.

Для отримання загальної статистики битв, була створена функція яка сортує битви по часові проведення і видає результат. Код представлений на рисунку 2.19.

```

9         async getAllFights(): Promise<Rang[]>{
10             const fights = await this.rangModel.find();
11             fights.sort((a, b) =>{
12                 return a.timeFight.getTime() - b.timeFight.getTime()
13             })
14             return fights
15         }

```

Рисунок 2.19. Код отримання всіх битв.

Якщо користувач увійде в кабінет він може побачити список битв певного користувача. Для оптимізації роботи на серверній частині було вирішено

написати окрему функцію яка шукає в яких битвах приймав участь потрібний користувач, і видає відфільтрований результат. Код представлений на рисунку 2.20.

```
16   async getFightForUserByName(name: string): Promise<Rang[]> {
17       const userFightStart = await this.rangModel.find({whoStart: name});
18       const userFightDefense = await this.rangModel.find({whoDefence: name});
19       const allInfo = userFightStart.concat(userFightDefense);
20       allInfo.sort((a, b) =>{
21           return b.timeFight.getTime() - a.timeFight.getTime()
22       })
23       return allInfo
24   }
```

Рисунок 2.20. Код отримання битв певного користувача.

Після закінчення битви результат записується в БД. Вноситься ім'я того хто почав битву і той хто програв. Відповідно результат битви, хто виграв і о котрій годині це було. Код представлений на рисунку 2.21.

```
25   async setFight(dto: any): Promise<Rang>{
26       const fight = await this.rangModel.create({...dto.user,
27           timeFight: new Date()
28       });
29       return fight
30   }
31 }
32 |
```

Рисунок 2.21. Код додавання в базу даних результатів битви.

Висновки до розділу 2

Гейміфікований вебсимулятор надає широкий функціонал пізнання та занурення у всесвіт покемонів для користувача. Симулятор дозволяє переглядати та досліджувати покемонів. Отримувати або покращувати покемонів за монети. Проводити битви в оффлайн арені проти великого списку покемонів які є у всесвіті покемонів. Або проводити битви проти справжніх гравців отримуючи в нагороду монети та рейтинг для подальшого просування по списку лідерів. Для більшого зацікавлення користувач може проходити нелегкі досягнення які будуть відображатися в профілі користувача.

У цьому розділі було реалізована структура для БД з відповідними вимогами до нереляційної моделі БД, що забезпечує швидку відповідь та доступність для користувачів симулятору. БД складається з 2 таблиць. Ця БД забезпечує збереження даних про користувачів, історію їхніх битв також БД забезпечує зручний та швидкий доступ до неї для використання у функціях гейміфікованого вебсимулятора.

В останньому пункті детально розглядається реалізація серверного коду додатку. Основна увага зосереджена на функціях збереження, зміни та видалення даних з БД та їх відправлення на клієнтську частину. Функціонал забезпечує ефективну та зручну роботу зі збереження даних у гейміфікованому вебсимуляторі.

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання – це візуальне уявлення конфігурації вузлів обробки та компонентів системи, коли вони розміщені. Цей вид структурних діаграм використовується для моделювання фізичних аспектів об'єктно-орієнтованої системи, зокрема, її апаратної топології. Діаграма розгортання дозволяє візуально представити розташування компонентів системи на обчислювальних вузлах та їх взаємодію.

На рисунку 3.1 представлена діаграма розгортання, яка ілюструє конфігурацію вузлів та компонентів системи під час розгортання.

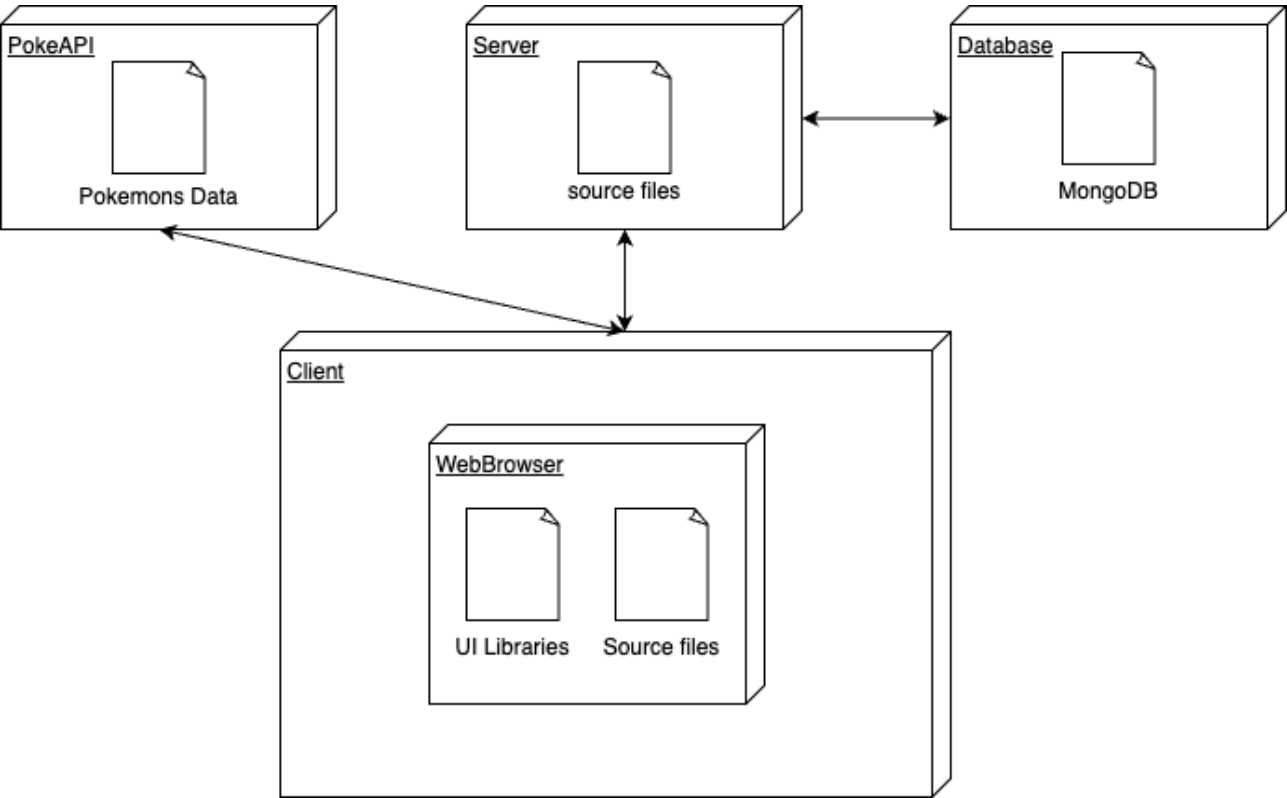


Рисунок 3.1. Діаграма розгортання проекту

Після побудови даної діаграми розгортання можна зробити наступні висновки щодо порядку встановлення та налаштування параметрів системи:

1. Серверна частина розгортається на порті 3001, в той час клієнтська частина розгортається на 3000 портові.
2. Сервер взаємодіє з базою даних для отримання даних та передає їх в клієнтську частину додатку.
3. Коли зі серверної частини потрапляють дані які пов’язані з API покемонів, клієнтська частина звертається до серверів покемон API та отримує потрібну інформацію.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з гейміфікованим симулятором.

Коли користувач відкриває вебсторінку його зустрічає сторінка авторизації приклад представлено на рисунку 3.2. На цій сторінці користувач може обрати мову на якій буде додаток на рисунку 3.3 представлено сторінку зі зміненою мовою вебсторінки. При бажанні користувача, він може перейти до сторінки реєстрації та зареєструватися, приклад сторінки реєстрації представлено на рисунку 3.4.

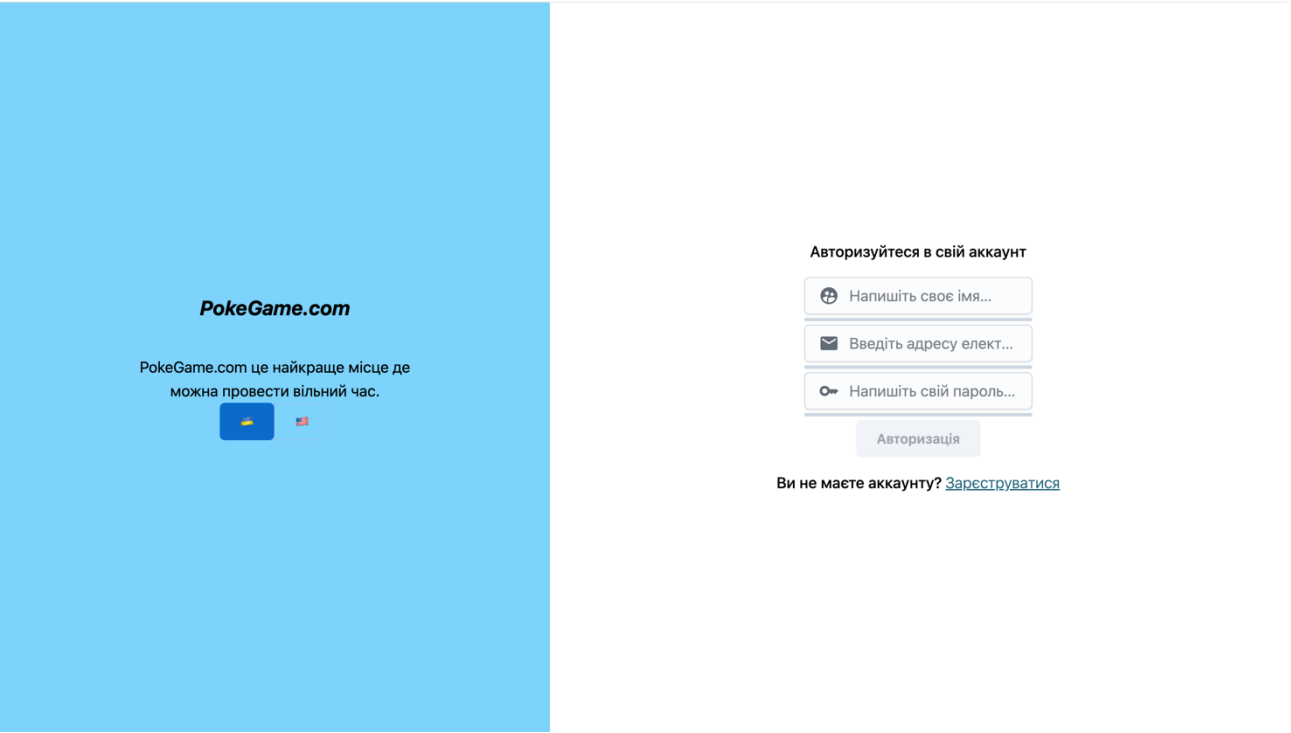


Рисунок 3.2. Сторінка авторизації

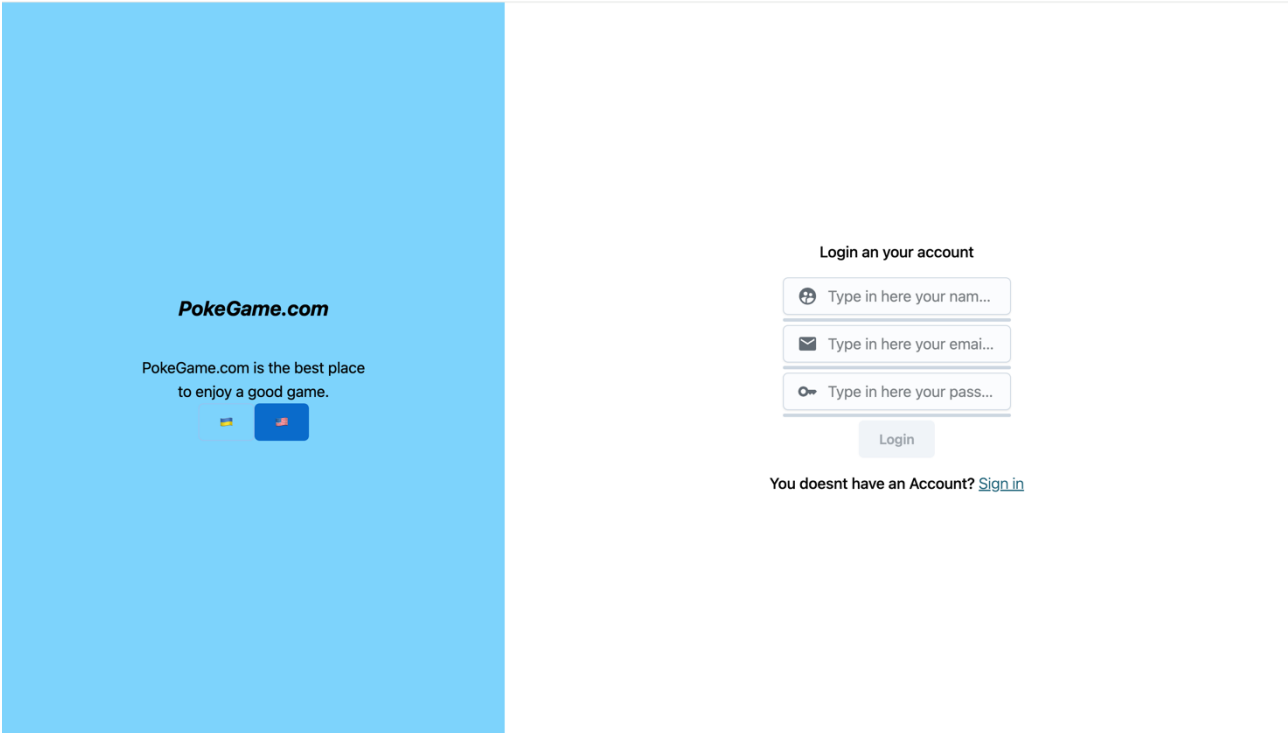


Рисунок 3.3. Сторінка авторизації зі зміненою мовою

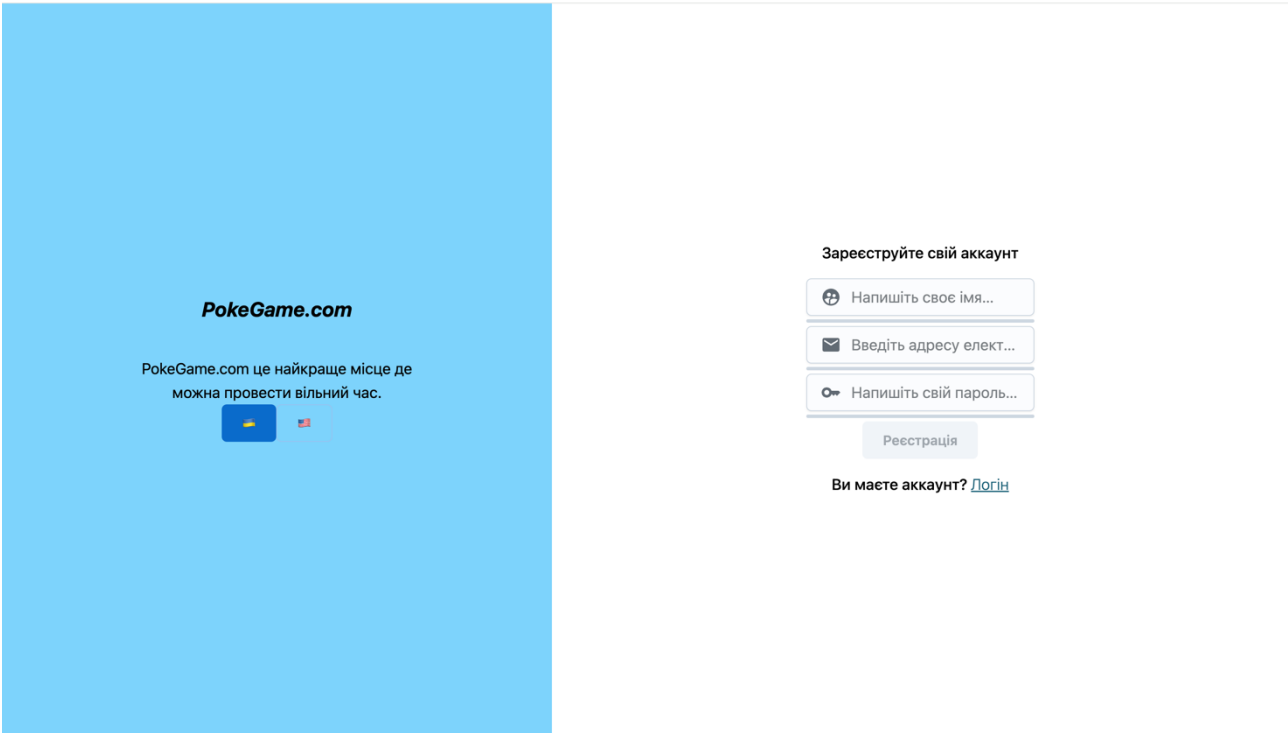


Рисунок 3.4. Сторінка реєстрації

Коли користувач реєструється в вебсимуляторі, йому розповідається про представлення всесвіту покемонів, правила та взагалі головну ціль користувача. Приклад одного з моменту тексту представлено на рисунку 3.5.. Після ознайомлення зі всесвітом користувачу надається можливість отримати свого

першого покемона, момент отримання першого покемона представлено на рисунку 3.6.

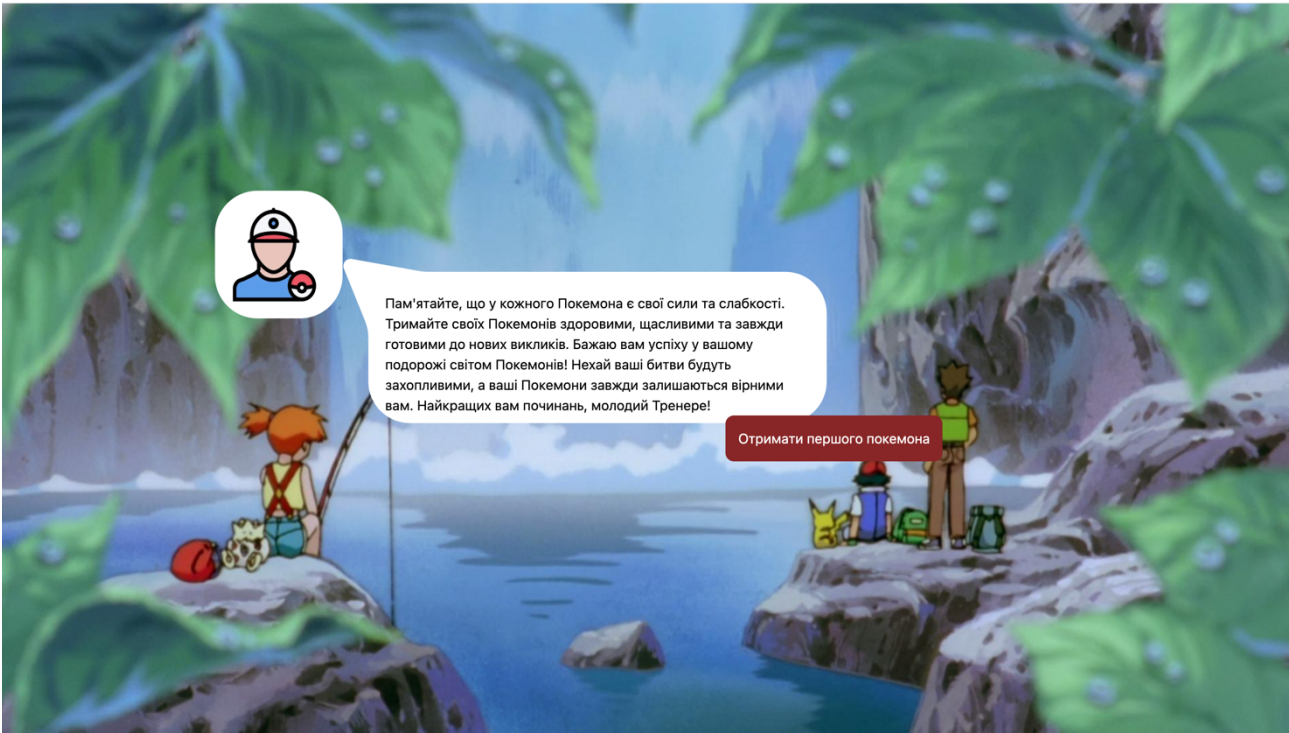


Рисунок 3.5. Представлення всесвіту покемонів

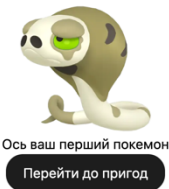


Рисунок 3.6. Сторінка отримання першого покемону

Авторизувавшись чи зареєструвавшись користувач переміщається на головне меню, де є основна навігація вебсимулятора, меню представлено на рисунку 3.7.

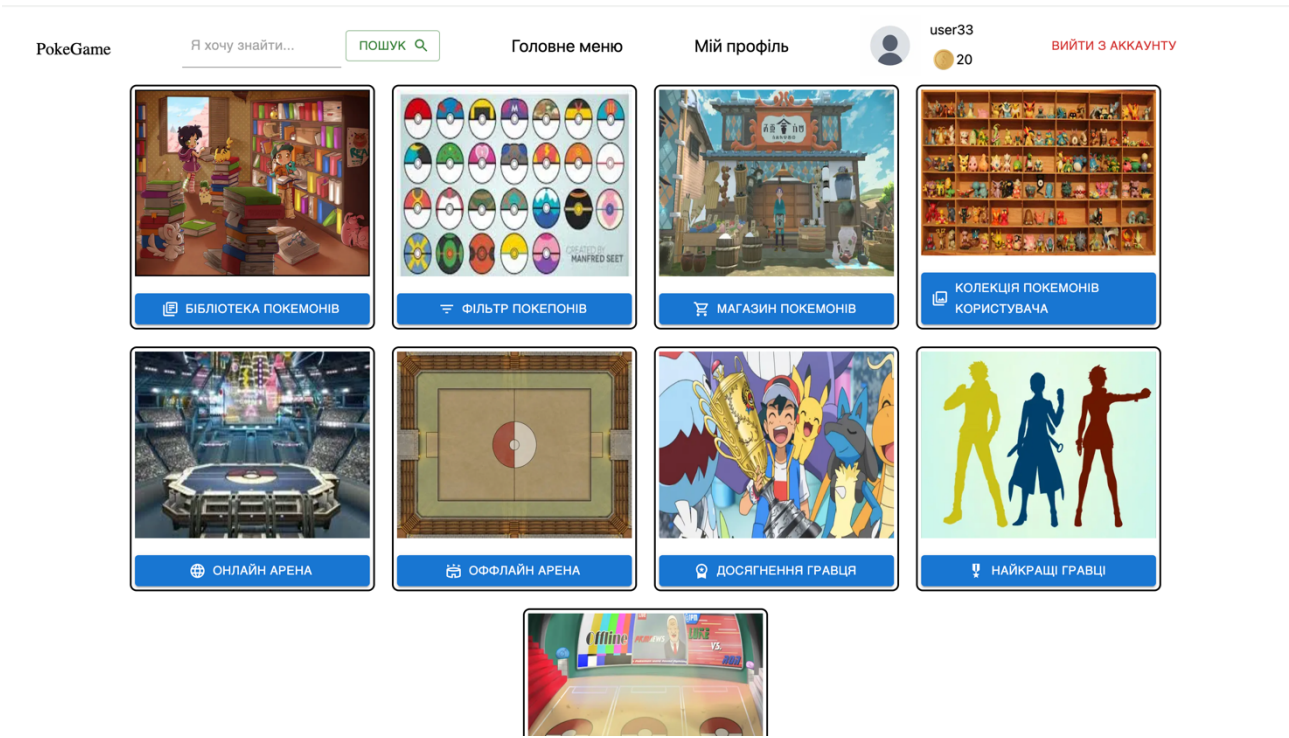


Рисунок 3.7. Головне меню

Першим елементом в меню є бібліотека покемонів де представлений список всіх існуючих покемонів. На одній сторінці представлено 20 покемонів. Пагінація відбувається за допомогою кнопки «наступна сторінка», «минула сторінка». Приклад сторінки представлений на рисунку 3.8.

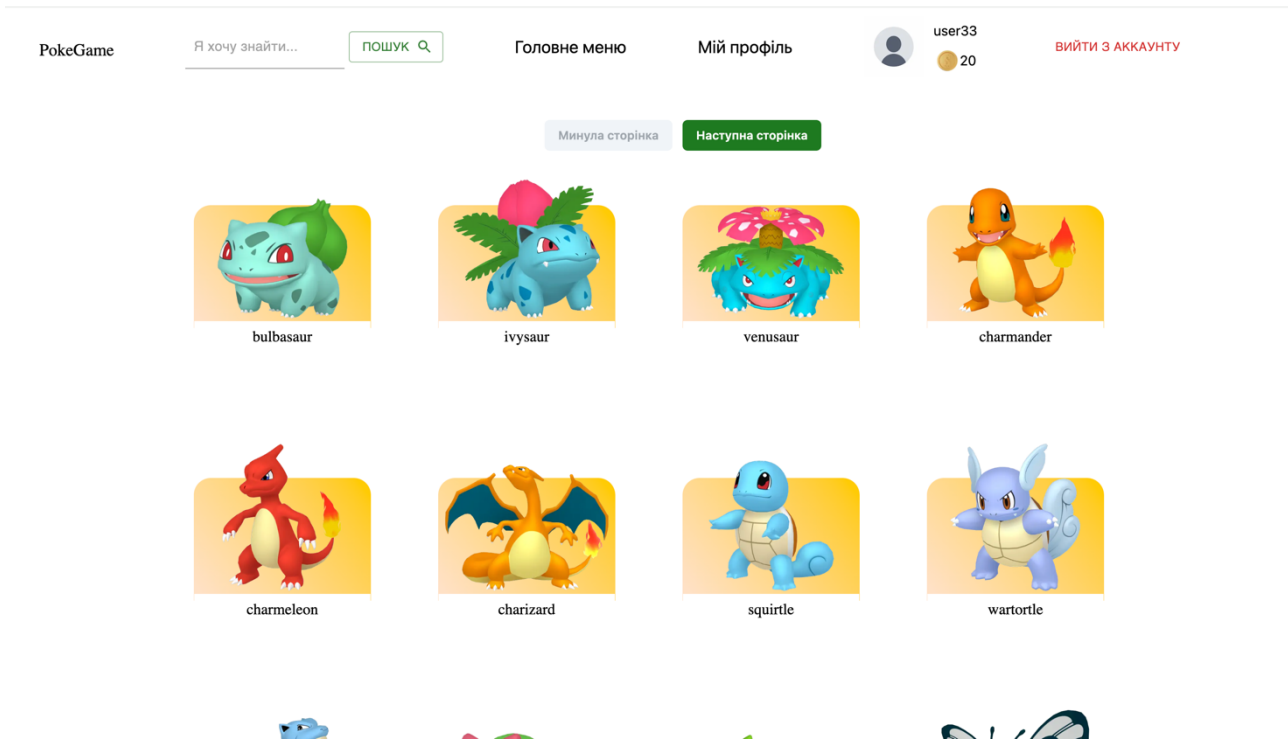


Рисунок 3.8. Бібліотека покемонів

В бібліотеці при обранні покемона, користувач перенаправляється на сторінку покемона. На даній сторінці представлено фото покемона, пагінація до наступного покемона по списку або минулого. Також фон сторінки залежить від стихії покемона, якщо стихій декілька, фон можна змінити. На сторінці є загальна інформація про покемона: його типи, ім'я, вага, висота, та базові статистики(здоров'я, сила атаки, кількість броні, сила спеціальної атаки, сила спеціальної броні, швидкість руху), еволюційні етапи покемона. Якщо в користувача є даний покемон в нього є можливість зробити покемона активним для битви, або еволюціонувати покемона до наступної стадії за допомогою монет. Приклад сторінки покемона представлено на рисунку 3.9.

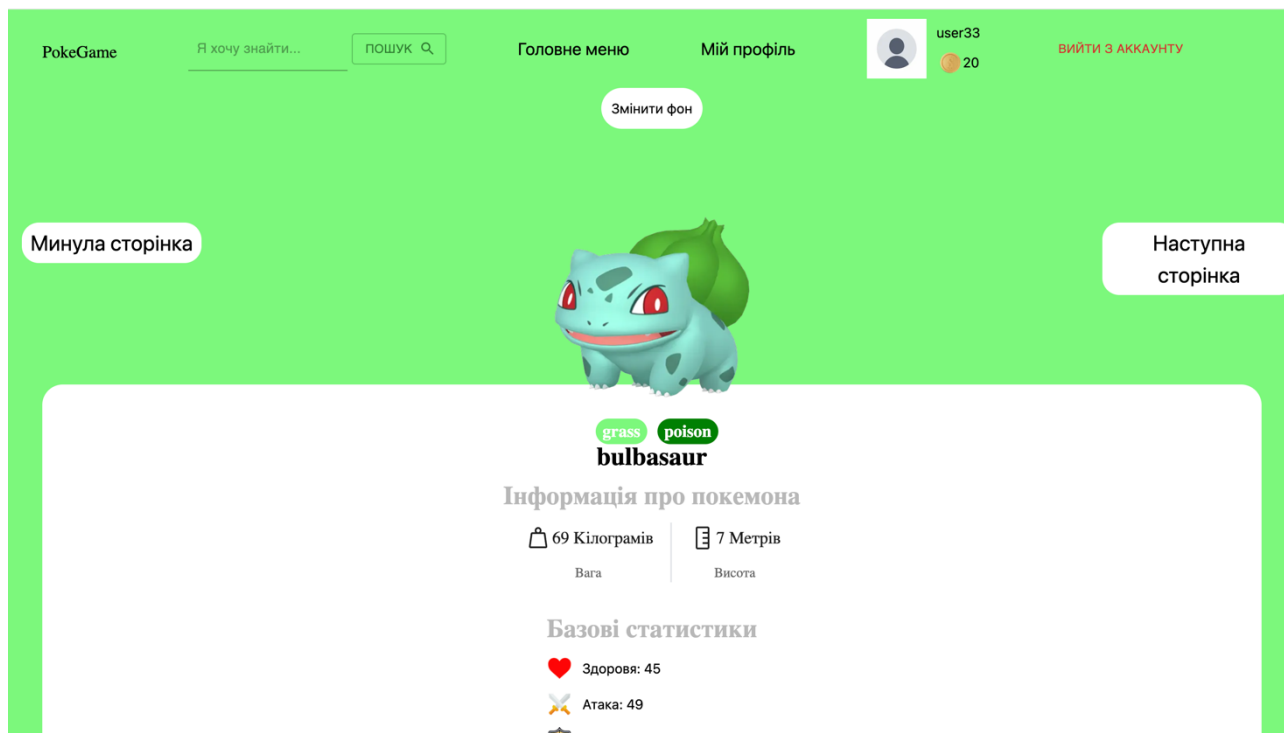


Рисунок 3.9. Приклад сторінки покемона

В пункті фільтрів покемонів в користувача є можливість обрати тип покемона який його цікавить, сторінка типів представлено на рисунку 3.10. Обравши тип зі списку користувача перенаправляє на сторінку де представленні покемони того типу. Приклад списку покемонів представлений на рисунку 3.11.



Рисунок 3.10. Сторінка типів покемонів

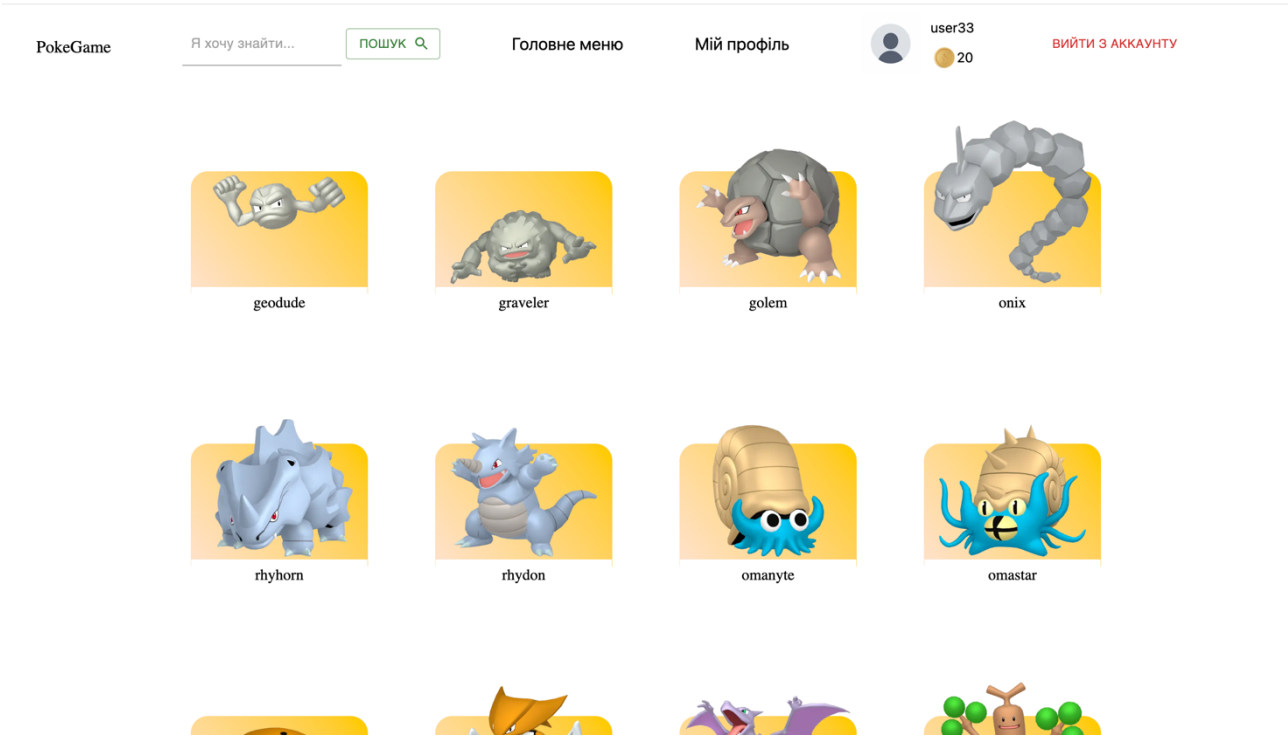


Рисунок 3.11. Приклад сторінки певного прикладу покемонів

В магазині користувач може придбати собі предмети за внутрішню валюту гейміфікованого симулятору. Користувач може придбати: будь-якого покемона, зілля для втечі з битви, зілля для подвоєння здоров'я покемона в битві, зілля для подвоєння сили удару покемона, зілля для подвоєння рангу при перемозі в онлайн арені, зілля для автоматичної перемоги в битві, зілля для подвоєння монет при перемозі в онлайн битві, зілля для додання шипів в битві яке працює по принципу повернення противнику шкоди яку він надав. Сторінка магазину представлено на рисунку 3.12.



Рисунок 3.12. Сторінка магазину

На сторінці колекції покемонів, в користувача є можливість переглянути активного покемона користувача, та покемонів які є в користувача в наявності, з можливістю перегляду покемона приклад сторінки представлений на рисунку 3.13.

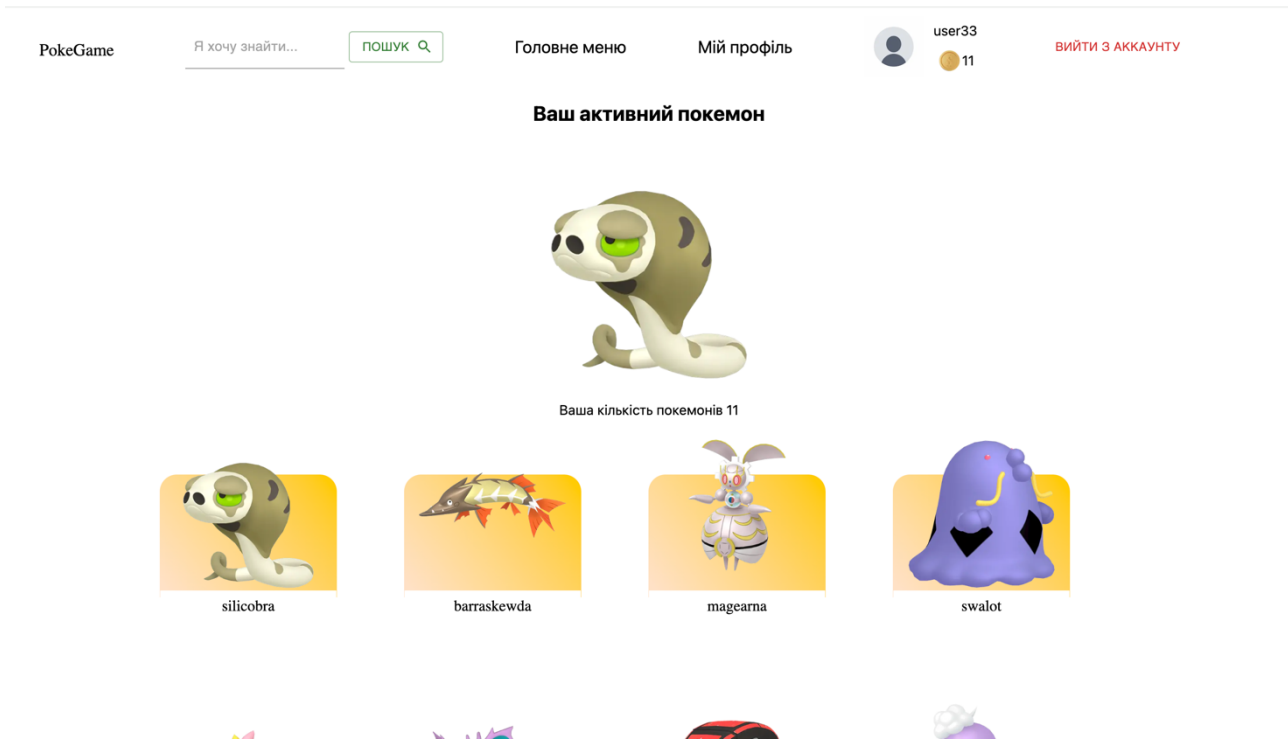


Рисунок 3.13. Сторінка колекції покемонів

На сторінці онлайн арени користувач може починати битви з іншими гравцями. Список користувачів з якими є можливість почати битву фільтрується по кількості рейтингу користувача, перед початком битви користувач може побачити фото користувача, його ім'я, та кількість рейтингу, також покемон який в користувача активний з описом характеристиками приклад сторінки представлено на рисунку 3.14.

При нажаті на початок битви, з'являється модальне вікно де представлена інформація про покемона, його типи і ім'я, приклад модального вікна представлено на рисунку 3.15.

При початку битви користувач переходить до вікна битви приклад модального вікна з битвою представлено на рисунку 3.16.

В битві користувач може здійснити удар при якому покемони отримують шкоду. При ударові відбувається анімація. При нажаті на кнопку втечі йде перевірка чи в користувача достатньо швидкості щоб втекти з битви, при втечі користувач нічого не втрачає. Користувач може виконати спеціальні дії яка прив'язана до типу покемона. Наприклад сильна атака або підняття здоров'я в

покемона. Також користувач може прийняти зілля для покращення характеристик покемона, лише на один бій. При перемозі користувач отримує очки рангу та монети, приклад перемоги представлено на рисунку 3.17. При поразці користувач втрачає очки рангу, приклад поразки представлено на рисунку 3.18.

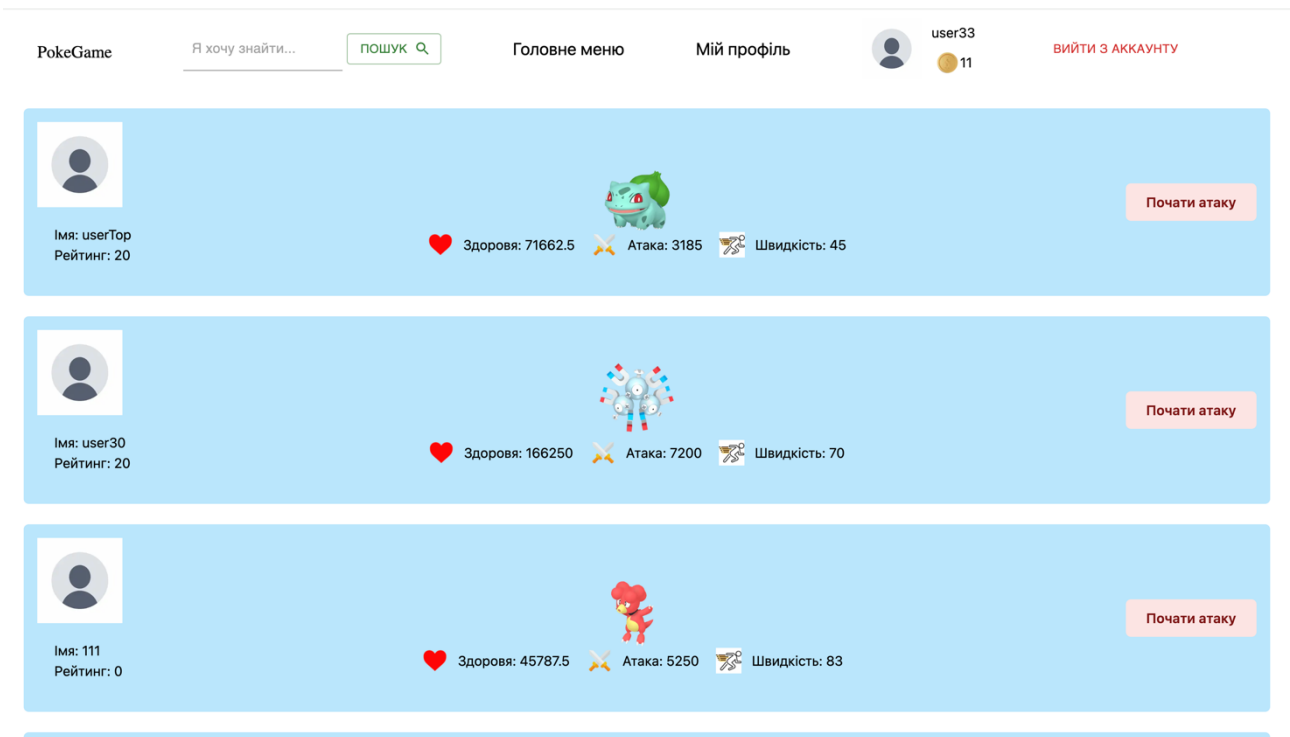


Рисунок 3.14. Сторінка арени

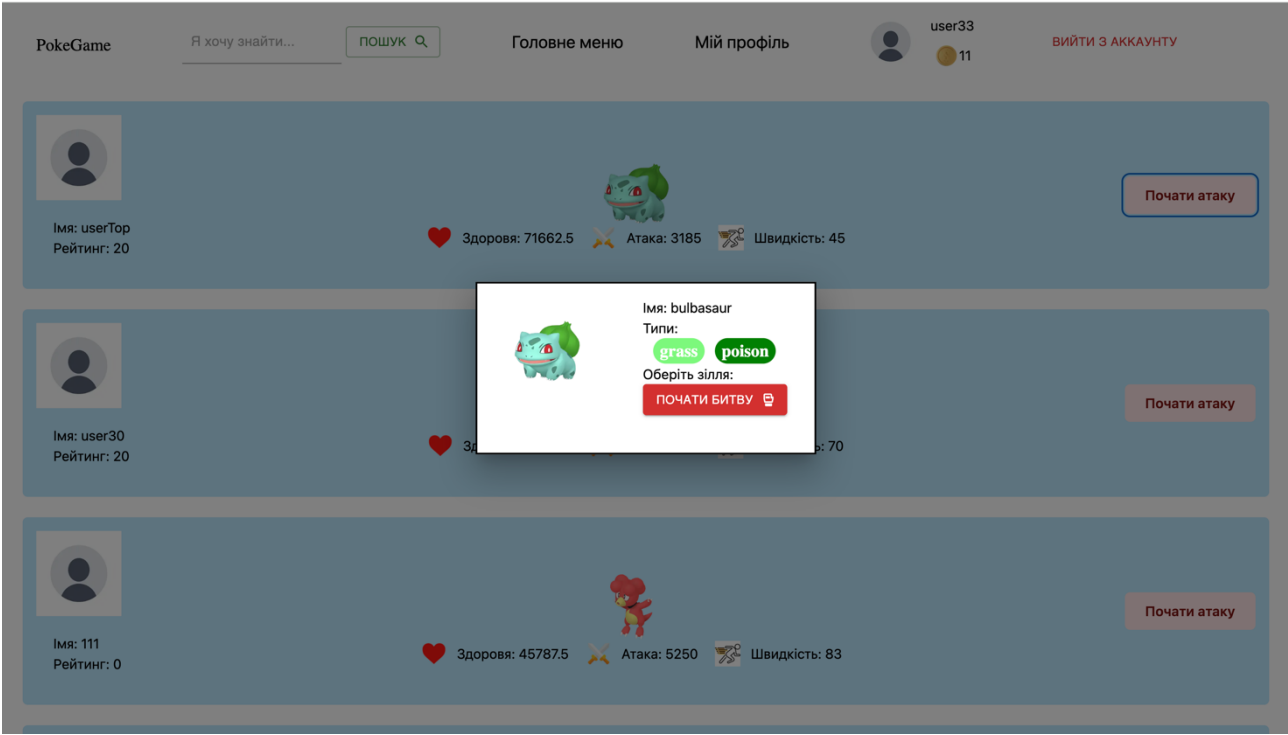


Рисунок 3.15. Обраний користувач перед битвою

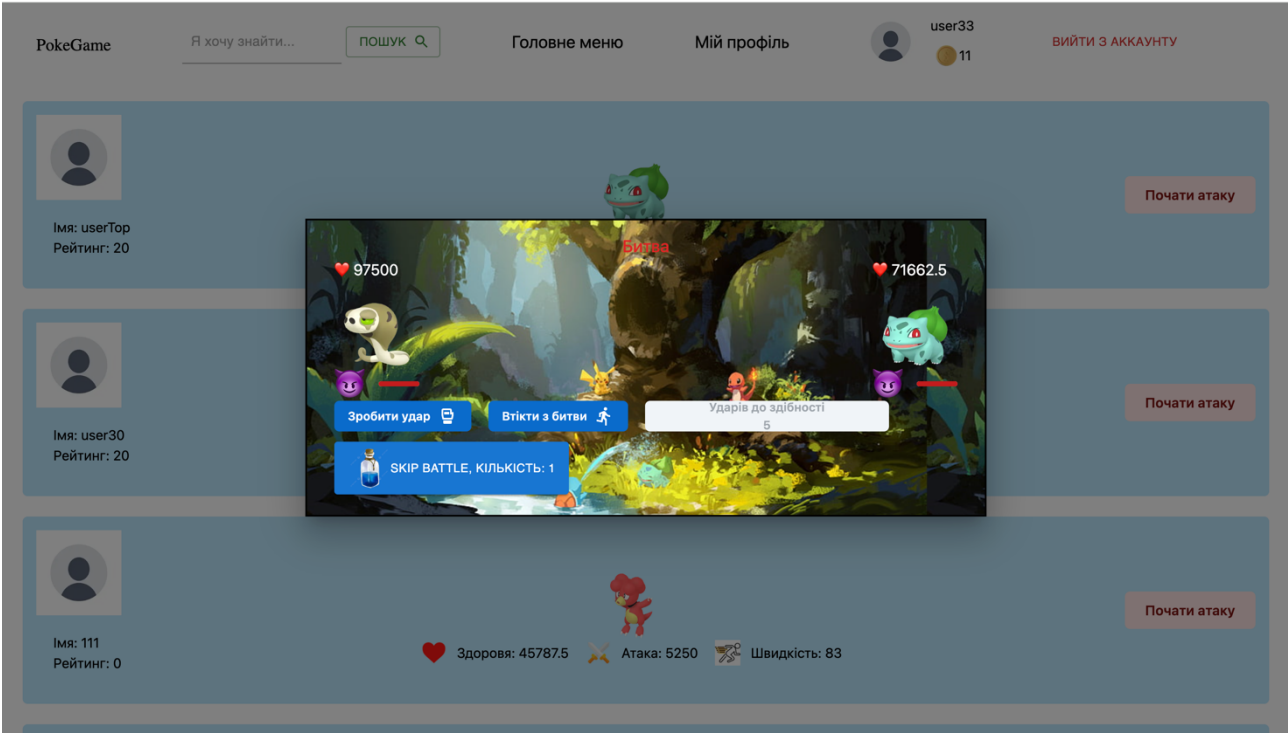


Рисунок 3.16. Битва з користувачем

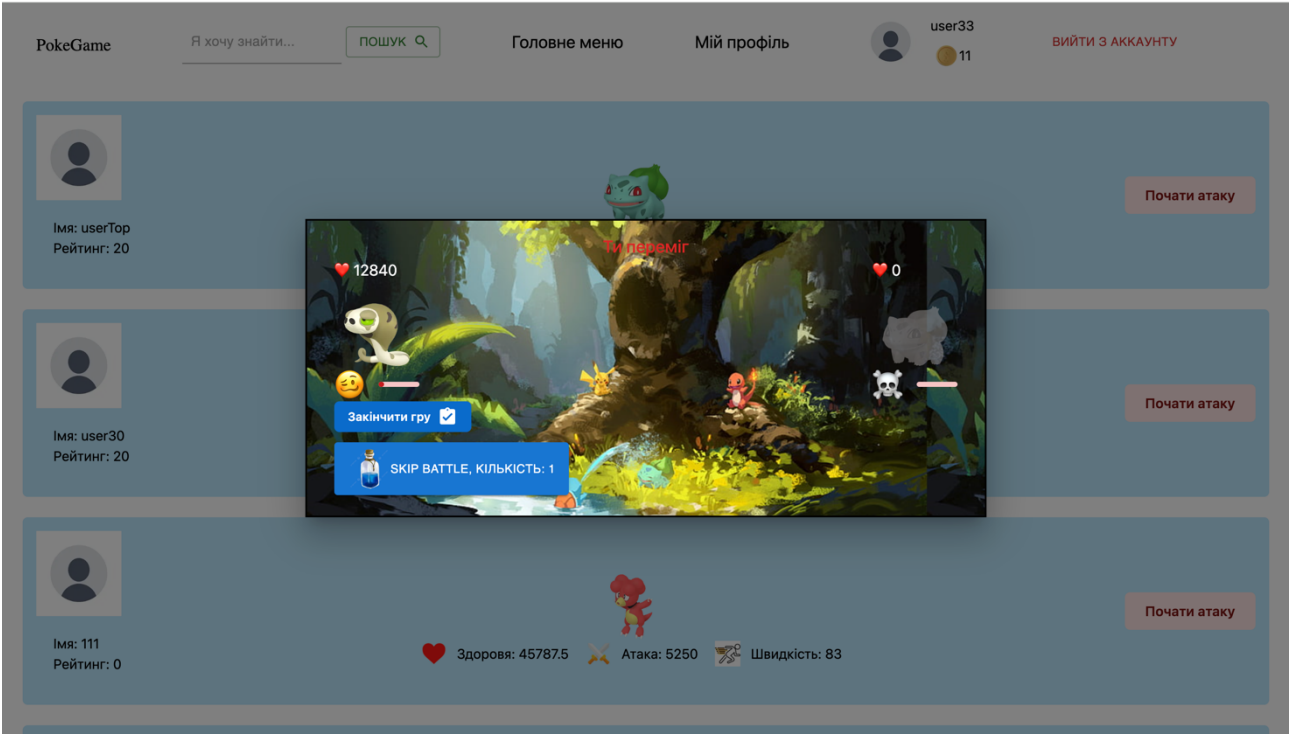


Рисунок 3.17. Перемога користувача

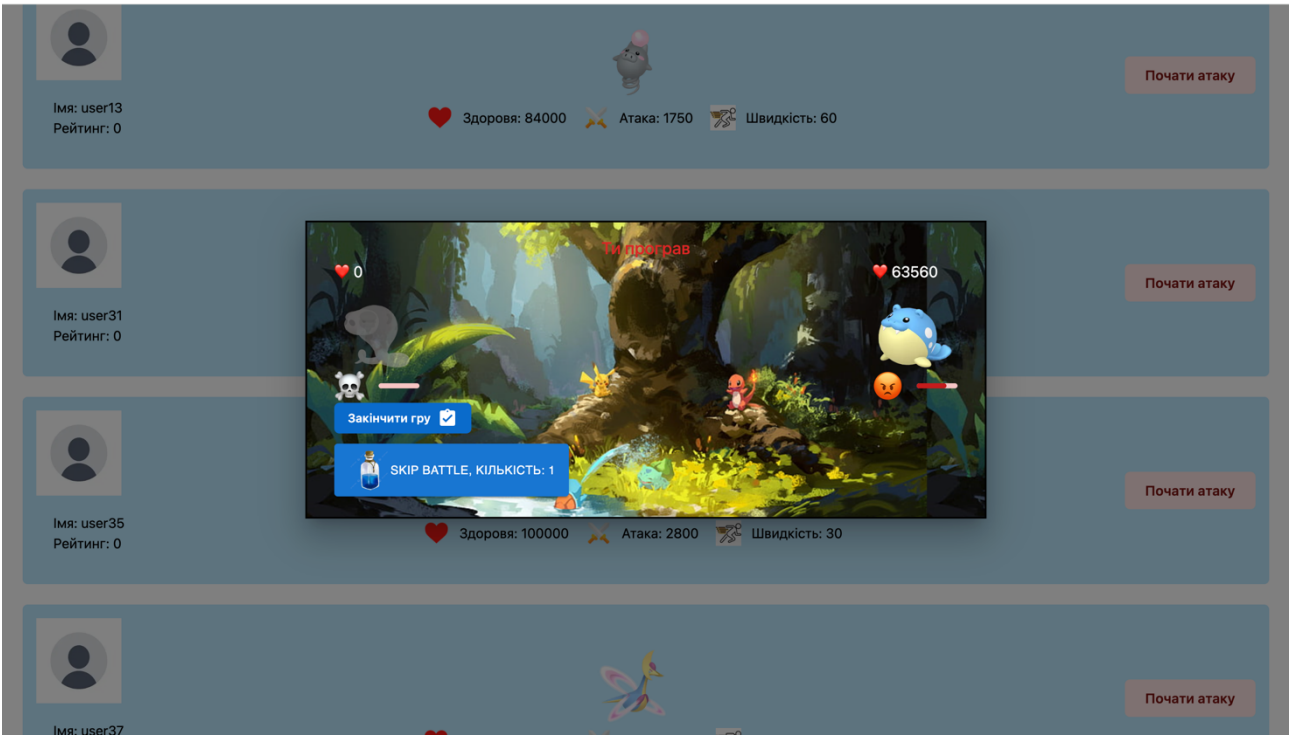


Рисунок 3.18. Програш користувача

В оффлайн арені користувач може проводити битви з покемонами з самого першого до самого останнього покемону по списку. На сторінці представлено активного покемона користувача, його характеристики, етап арени, інформація про противного покемона, приклад сторінки на рисунку 3.19.

В битві користувач може здійснити удар при якому покемони отримують шкоду, приклад модального вікна де відбувається битва на рисунку 3.20.

При перемозі користувач отримує монети та перехід на наступний етап приклад перемоги користувача на рисунку 3.21. Перехід до наступного етапу арени представлено на рисунку 3.22.

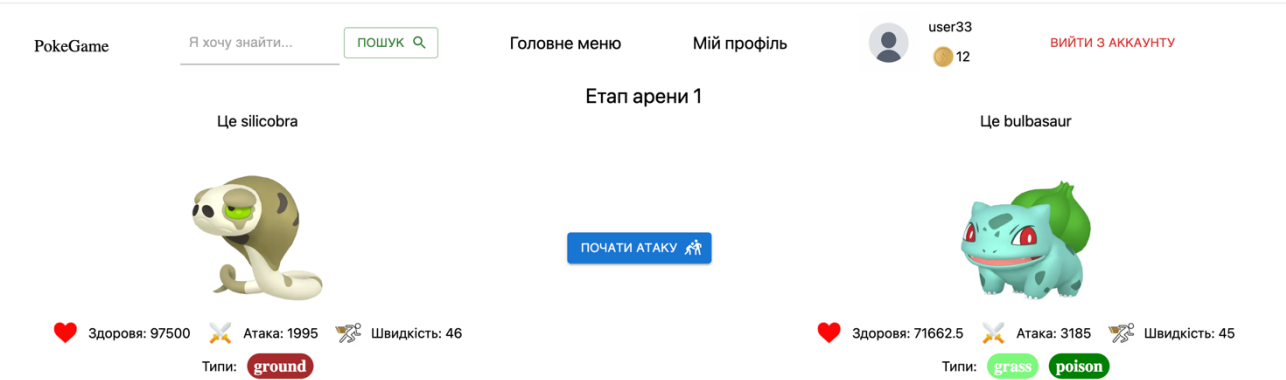


Рисунок 3.19. Сторінка оффлайн арени

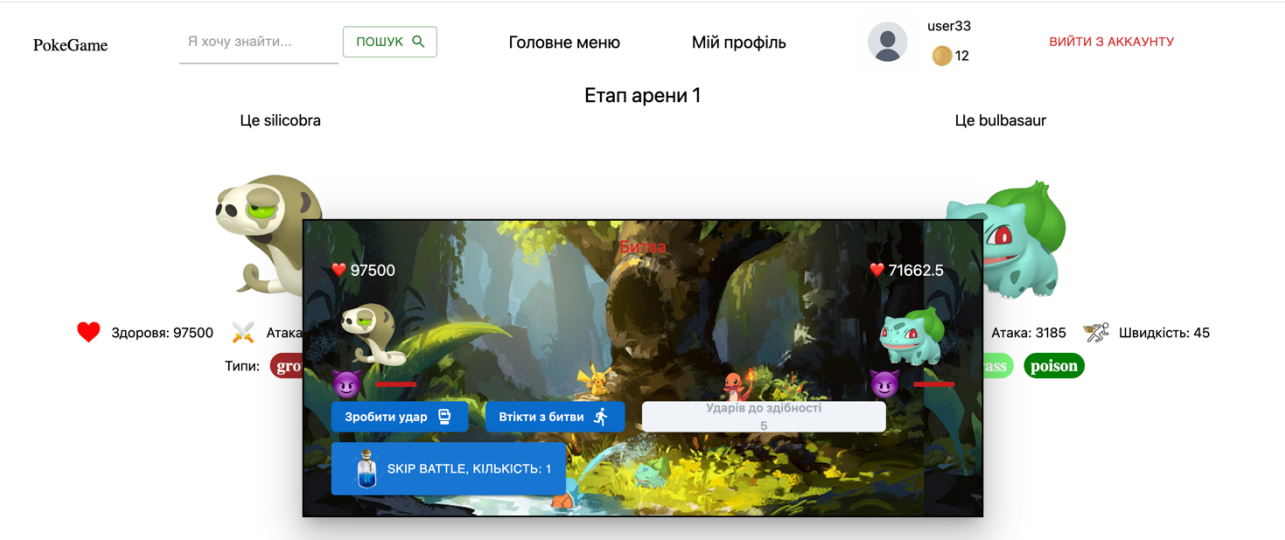


Рисунок 3.20. Битва оффлайн арени

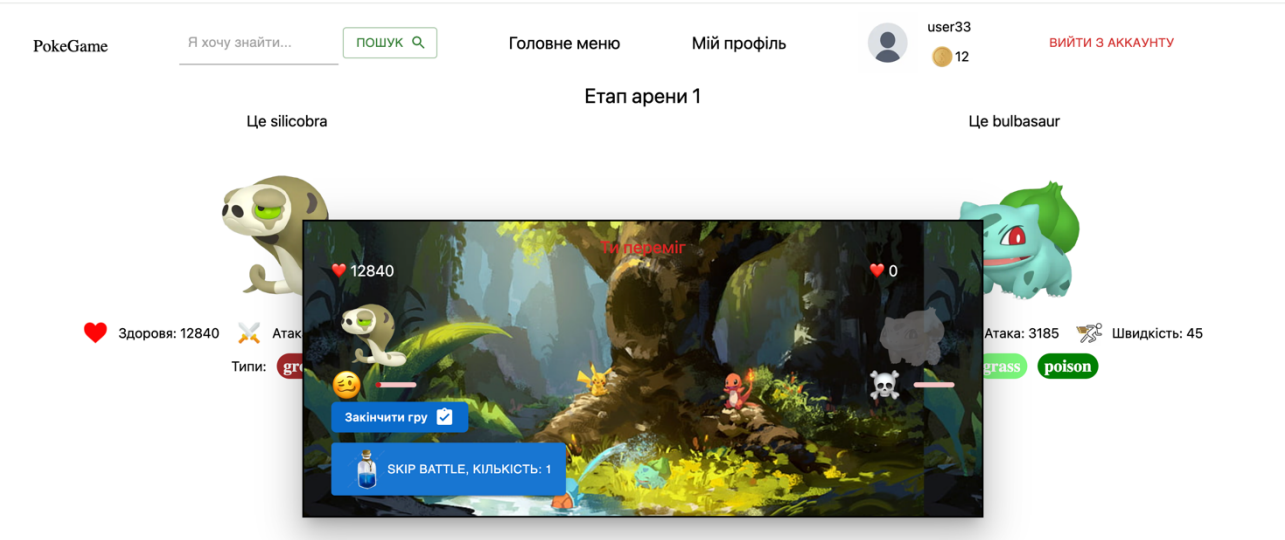


Рисунок 3.21. Перемога в оффлайн арени

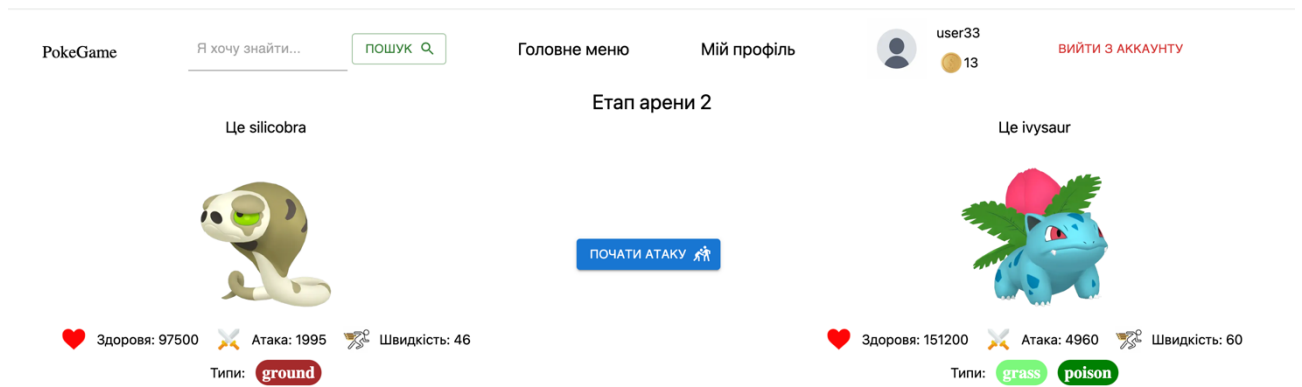


Рисунок 3.22. Наступний етап в оффлайн арени

При проходженні гейміфікованого вебсимулятора користувач отримує досягнення, наприклад за удари гравців, або кількість покемонів. Коли користувач отримує досягнення з'являється модальне вікно яке повідомляє про отримання досягнення. На сторінці досягнень користувач може переглянути свої отримані досягнення та ті які користувач може отримати. Сторінка досягнень представлена на рисунку 3.23.

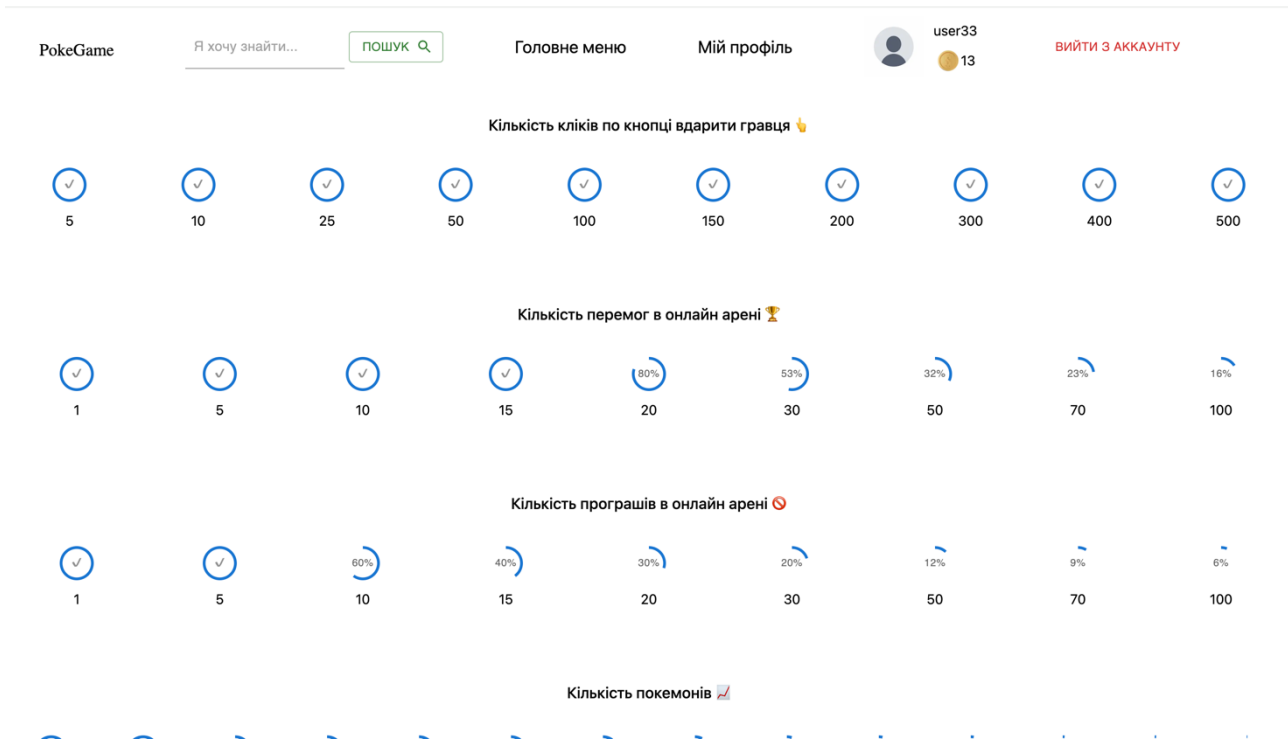


Рисунок 3.23. Сторінка досягнень гравця

В гейміфікованому вебсимуляторі користувач має рейтинг та кількість покемонів. Щоб переглянути список найкращих користувачів які мають найбільшу кількість покемонів та найбільшу кількість рейтингу, користувач може перейти на сторінку список найкращих гравців. Дана сторінка представлена на рисунку 3.24.

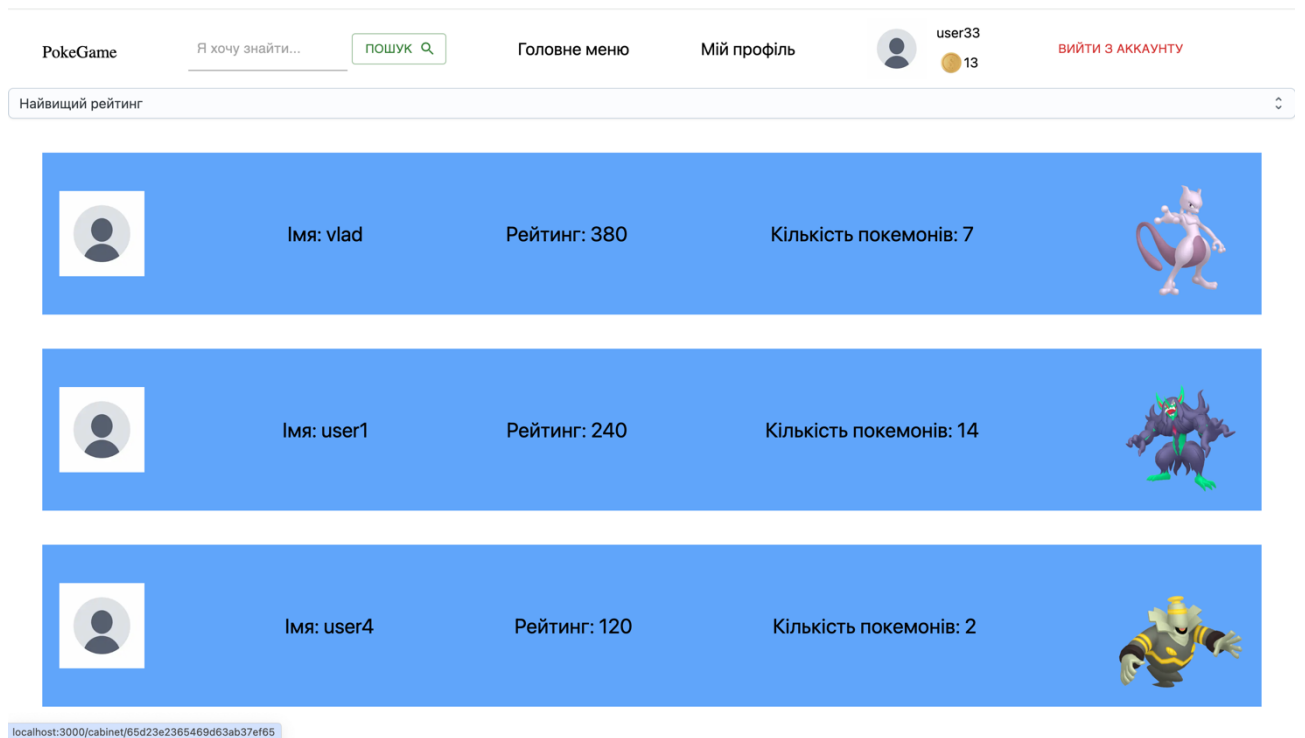


Рисунок 3.24. Сторінка найкращих гравців

На сторінці особистого профілю користувач бачить: історію своїх битв, список покемонів, відсоток виконаних досягнень, електронну пошту, кількість рейтингу, ім'я користувача. Користувач може змінити деяку інформацію: мову гейміфікованого вебсимулятора, ім'я, адресу, пароль, зображення профілю, та можливість видалення аккаунту користувача. Приклад сторінки користувача представлені на рисунку 3.25.

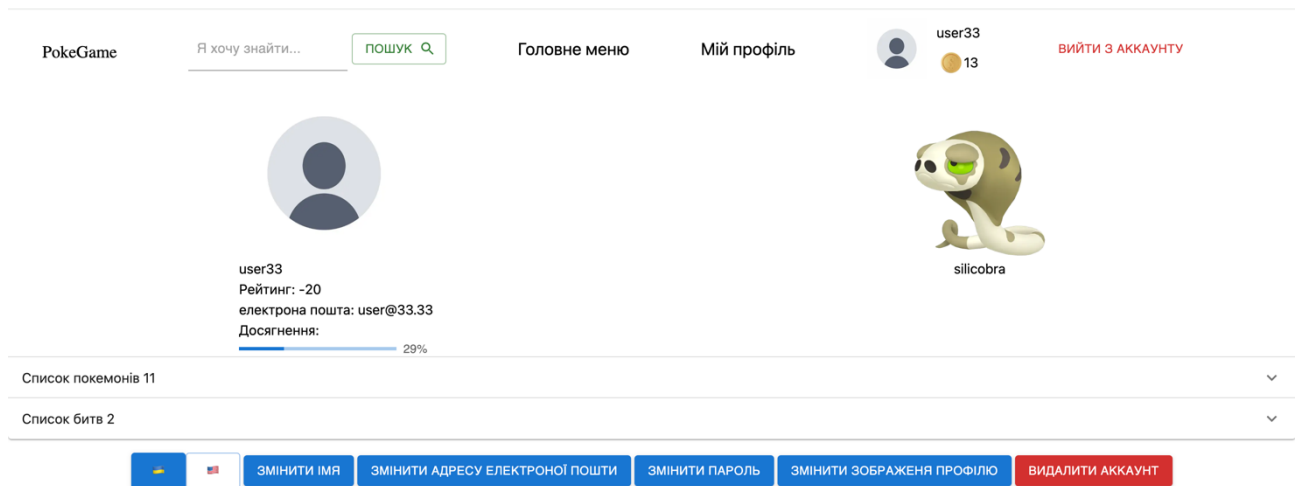


Рисунок 3.25. Профіль користувача

3.3 Тестування роботи програмного продукту

Розробка будь-якого програмного продукту припускає можливість виникнення помилок. Тому без проведення тестування неможливо забезпечити високу якість додатка. Під час тестування аналізуються різні аспекти програми, такі як правильність введених та виведених даних, коректність роботи функцій та алгоритмів, обробка помилок і відновлення роботи після збоїв. Головною метою тестування є виявлення помилок і дефектів програмного продукту перед його випуском на ринок або в експлуатацію. Для цього спочатку створюються тестові сценарії, які представляють собою послідовність дій та/або умов, необхідних для перевірки роботи певної функціональності програмної системи. Ці сценарії можна подати у вигляді таблиці (див. табл. 3.1).

Таблиця Test Cases

Test Case Description	Test Steps	Test Data	Expected Results
Перевірка авторизації з дійсними даними	1. Перейти на сторінку localhost/login 2. Ввести ім'я користувача 3. Ввести пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти	Пошта user@1.1 ім'я користувача = user1 Пароль користувача = user1	Користувач увійде у головне меню
Перевірка авторизації з не дійсним паролем	1. Перейти на сторінку localhost/login 2. Ввести ім'я користувача 3. Ввести пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти	Пошта user@1.1 ім'я користувача = user1 Пароль користувача = user123	Користувач отримус повідомлення про неправильні введені дані
Реєстрація користувача з унікальним іменем і поштою	1. Перейти на сторінку localhost/registration 2. Ввести унікальне ім'я користувача 3. Ввести унікальну пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти	Пошта user@100.100 ім'я користувача = user100 Пароль користувача = user100	Користувач перейде на сторінку представлення всесвіту покемонів

Продовження таблиці 3.1. Таблиця Test Cases

Реєстрація користувача з не унікальним іменем і поштою	1. Перейти на сторінку localhost/registration 2. Ввести унікальне ім'я користувача 3. Ввести унікальну пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти	Пошта = user@1.1 ім'я користувача = user102 Пароль = user102	Користувач отримає повідомлення що такі дані використовує інший користувач.
Отримання першого покемона	1. Перейти на сторінку localhost/registration 2. Ввести унікальне ім'я користувача 3. Ввести унікальну пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти 6. Пройти ознайомлення зі всесвітом 7. На сторінці отримання покемону натиснути «Отримати покемона» 8. Попасти в меню. 9. Перейти на сторінку колекції покемонів. 10. Впевнитись що саме цей покемон був отриманий	Пошта = user@103.103 ім'я користувача = user103 Пароль = user103	Користувач отримав покемона та він автоматично став активним

Продовження таблиці 3.1. Таблиця Test Cases

Купівля покемона в магазині	1. Перейти на сторінку localhost/login 2. Ввести ім'я користувача 3. Ввести пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти 6. Перейти на сторінку магазину. 7. Натиснути на придбати покемона. 8. Перейти на колекцію покемонів. 9. Впевнитись що покемон додався	Пошта = user@1.1 ім'я користувача = user1 Пароль = user1	Користувач має придбати покемона та отримати його собі.
-----------------------------	---	--	---

Продовження таблиці 3.1. Таблиця Test Cases

Перемога в арені	1. Перейти на сторінку localhost/login 2. Ввести ім'я користувача 3. Ввести пошту користувача 4. Ввести пароль користувача 5. Натиснути увійти. 6. Перейти на сторінку онлайн арени. 7. Обрати користувача з слабшим покемоном ніж у перевіряючого. 8. Натиснути почати битву. 9. Перемогти. 10. Перевірити чи додалися монети. 11. Перевірити що у програшного користувача стало менше рейтингу.	Пошта = user@1.1 ім'я користувача = user1 Пароль = user1	Перевірка працездатності зарахування нагороди за перемогу в онлайн арені, та відняття рейтингу у програшного.
------------------	---	--	---

Висновки до розділу 3

У цьому розділі було побудовано діаграму розгортання та зібрано дані щодо процесу встановлення та налаштування параметрів системи.

Також, було проілюстровано структуру інтерфейсу та описано порядок взаємодії з гейміфікованим вебсимулятором. Детально розглянуто функціональні можливості додатку для користувачів.

ВИСНОВКИ

Під час виконання бакалаврської роботи було розроблено гейміфікований веб симулятор з наступними завданнями:

Було проведено аналіз схожих симуляторів з метою порівняння. Кожен з цих симуляторів мають свої унікальні особливості та механіки, але для нового симулятору ключовими аспектами мають бути наявність інтерфейсу для зручного доступу, та цікаві механіки для заохочення користувача більше часу проводити в симуляторі, цікавий і впізнаваний всесвіт щоб легше користувачі адаптувалися до симулятору.

Було обрано компоненти для реалізації симулятору: БД MongoDB, основна мова програмування TypeScript, для серверної частини фреймворк Nest.JS, для клієнтської частини фреймворк Next.js. Використано було середовище розробки WebStorm.

Було проведено аналіз вимог до системи як функціональних, так і нефункціональних. Також, було створено діаграму варіантів використання та діаграму класів для докладного вивчення структури системи.

Реалізовано базу даних з 2 таблицями, що забезпечує зручне збереження та розділення даних на користувачів, та історію битв.

Розроблено алгоритми функціонування симулятору, реалізовано гейміфікований вебсимулятор.

Реалізовано інтерфейс симулятору для користувача. Користувач може створювати профіль, переглядати покемонів, проводити онлайн та оффлайн битви де може отримати нагороди та досягнення. За отриману нагороду може придбати зілля для битв або нових покемонів або покращити вже існуючих.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савіцький Р.С., Петрук О.О. Розробка прототипу вебдоступного ігрового симулятора. Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки. Том 35 (74), №2, 2024. С. 143-149. Режим доступу до ресурсу: https://www.tech.vernadskyjournals.in.ua/journals/2024/2_2024/22.pdf
2. Foldit [Електронний ресурс] – Режим доступу до ресурсу: <https://fold.it/>
3. М. М. Макрушин, Є. М. Макрушина, Н. В. Петерсон, М. М. Мельников Фізіологія рослин Вінниця 2006 Режим доступу до ресурсу: https://snvlnk.at.ua/_ld/0/2_Fisiologi_m.pdf
4. SimCity [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ea.com/games/simcity>
5. Freerice [Електронний ресурс] – Режим доступу до ресурсу: <https://freerice.com/>
6. Макаревич О. О. Гейміфікація як невід’ємний чинник підвищення ефективності елементів дистан-ційного навчання. Молодий вчений. № 2(6). 2015. С. 279-282. Режим доступу до ресурсу: http://nbuv.gov.ua/UJRN/molv_2015_2%286%29__73
7. DragonBox Algebra 5+ [Електронний ресурс] – Режим доступу до ресурсу: <https://dragonbox.com/products/algebra-5>
8. CodeCombat [Електронний ресурс] – Режим доступу до ресурсу: <https://codecombat.com/>
9. Патерн Медіатор [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/mediator>
10. HTML5 [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/html/html5_intro.html#gsc.tab=0
11. Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/>
12. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/>

					ІПЗК.КР.Б-121-24-ПЗ	
						69

13. React [Електронний ресурс] – Режим доступу до ресурсу:
<https://uk.legacy.reactjs.org/>

14. Next.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://nextjs.org/>

15. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/en>

16. Nest.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nestjs.com/>

17. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com>

18. WebStorm [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/webstorm>

19. Що таке REST API [Електронний ресурс] – Режим доступу до ресурсу:
<https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi>

ДОДАТКИ

					ІІЗК.КР.Б-121-24-ІЗ	
						71