НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система аналізу ігрових даних та матчевої статистики Dota 2»

Виконав:
студент IV курсу, групи ІС-13
Білоконь Владислав Віталійович _____

Керівник:
Старший викладач
Проскура Світлана Леонідівна _____

Рецензент:
Доцент кафедри ІПІ, к.т.н, доцент
Лісовиченко Олег Іванович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Білоконю Владиславу Віталійовичу

1. Тема проєкту «Інформаційна система аналізу ігрових даних та матчевої статистики Dota 2», керівник проєкту Проскура Світлана Леонідівна, старший викладач кафедри ІСТ, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: «9» червня 2025 року

3. Вихідні дані до проєкту: мова програмування JavaScript, бібліотека React.js, база даних PostgreSQL, платформа Node.js мова написання запитів GraphQL.

4. Зміст пояснювальної записки: опис предметної області, аналіз існуючих рішень, формування вимог до системи, математичне забезпечення, вибір технологій розробки, розробка інформаційної системи, тестування системи.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): діаграма варіантів використання, діаграма взаємодії компонентів, діаграма розгортання, діаграма структури бази даних.

6. Дата видачі завдання «17» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Ознайомлення з ТЗ	16.04.2025	Виконано
2	Аналіз предметної області	18.04.2025	Виконано
3	Огляд існуючих аналогів	23.04.2025	Виконано
4	Розробка функціональної моделі	25.04.2025	Виконано
5	Формування цілей і задач розробки	29.04.2025	Виконано
6	Розробка дизайну ПЗ	01.05.2025	Виконано
7	Проєктування структури бази даних	02.05.2025	Виконано
8	Дослідження і розробка математичного забезпечення	07.05.2025	Виконано
9	Розробка алгоритму аналізу даних	10.05.2025	Виконано
10	Розробка клієнтської частини ПЗ	15.05.2025	Виконано
11	Тестування ПЗ	17.05.2025	Виконано

Студент Владислав БІЛОКОНЬ

Керівник Світлана ПРОСКУРА

АНОТАЦІЯ

Інформаційна система аналізу ігрових даних та матчевої статистики Dota 2.

Проєкт містить 60 с. тексту, 19 рисунків, 17 таблиці, посилання на 15 літературних джерел, додатки та 4 конструкторських документа.

АНАЛІЗ МАТЧІВ, ОПТИМІЗАЦІЯ ВИБІР ГЕРОЇВ, PDF-ПРОТОКОЛИ, КІБЕРСПОРТ, DOTA 2, ГЕНЕТИЧНИЙ АЛГОРИТМ, ВЕБЗАСТОСУНОК.

Об’єктом розробки є інформаційна система для аналізу та оптимізації ігрових даних матчів кіберспортивної дисципліни Dota 2.

Мета розробки – розробка додатку, що дозволяє користувачу зручно переглядати статистику і аналітику обраного зіграного матчу, за допомогою ідентифікатора матчу або перегляду профілю. Після перегляду цієї інформації при необхідності можна скористатися функцією оптимізації вибору героїв або згенерувати PDF-протокол. Отримана система може бути використана як навчальний ресурс для новачків так вже і розвинутими гравцями для різних цілей

У дипломному проєкті розроблено вебзастосунок, що складається з клієнтської частини, серверної частини та бази даних. Дані отримуються за допомогою відкритого API. Система дає змогу користувачу робити усе, що йому може потрібно для комфортного користування, а саме: реєстрація через додаток Steam, перегляд інформації користувачів та їх останні матчі, перегляд та аналіз зіграних матчів, перегляд найсильніших героїв за останній час, формування PDF-протоколу, генерація рекомендацій щодо вибору героїв за допомогою генетичного алгоритму.

SUMMARY

Information system for analyzing game data and match statistics in Dota 2.

The project includes 60 pages of text, 19 figures, 17 tables, links to 15 literary sources, annexes and 4 design documents.

Keywords: match analysis, hero pick optimization, pdf reports, esports, dota 2, genetic algorithm, web system.

The object of the development is an information system for analyzing and optimizing game data from Dota 2 matches.

The goal of the development is to create an application that enables users to conveniently view detailed analytics of a selected match using either a match ID or profile search. Based on this information, users can utilize hero pick optimization or generate a structured PDF protocol. The resulting system can serve both as an educational tool for beginners and as a practical resource for experienced players.

The diploma project involves the development of a web application consisting of a frontend, backend, and a database. Data is retrieved via a public API. The system allows users to: log in via Steam, view user profiles and match history, analyze match metrics, view current meta heroes, generate PDF protocols, and receive hero pick recommendations powered by a genetic algorithm.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС13.020БАК.005 ПЗ	Пояснювальна записка	61		
6	А3	ІС13.020БАК.005 Д1	Інформаційна система аналізу	1		
7			ігрових даних та матчевої			
8			статистики Dota 2. Діаграма			
9			варіантів використання			
10	А3	ІС13.020БАК.005 Д2	Інформаційна система аналізу	1		
11			ігрових даних та матчевої			
12			статистики Dota 2. Діаграма			
13			взаємодії компонентів			
14	А3	ІС13.020БАК.005 Д3	Інформаційна система аналізу	1		
15			ігрових даних та матчевої			
16			статистики Dota 2. Діаграма			
17			розгортання			
18	А3	ІС13.020БАК.005 Д4	Інформаційна система аналізу	1		
19			ігрових даних та матчевої			
20			статистики Dota 2. Діаграма			
21			структури бази даних			
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІС13.020БАК.005 ТП		
Розроб.		Білоконь В.В.			Інформаційна система аналізу ігрових даних та матчевої статистики Dota 2. Відомість дипломного проєкту	Літ.	Аркуш	Аркушів
Керівн.		Проскура С.Л				Т	1	1
						КПІ ім. Ігоря Сікорського Група ІС-13		
Затв.								

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система аналізу ігрових даних та матчевої статистики Dota 2»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Варди – віртуальні об'єкти в грі, що розміщуються гравцями на мапі для забезпечення огляду певних ділянок території, надаючи тактичну перевагу завдяки розширенню видимого простору.

Мета – це поняття, яке описує найбільш ефективні стратегії, тактики, вибір персонажів та стилі гри, що на даний момент дають найкращі результати у змаганнях.

Руни – спеціальні елементи, що періодично з'являються на ігровій мапі та надають персонажам тимчасові посилення або переваги при їх підборі.

Стаки – процес групування нейтральних ворожих істот на мапі шляхом виведення їх з визначеної території у певний момент часу, що дозволяє збільшити кількість ресурсів, отриманих при їх подальшому знищенні.

API (Application Programming Interface) – набір визначених методів та протоколів, що дозволяє різним програмним компонентам взаємодіяти між собою, забезпечуючи обмін даними та функціональністю.

Dire і Radiant – дві протилежні сторони віртуального ігрового поля, за одну з яких виступає команда гравців; вони відрізняються візуальним оформленням та розташуванням на мапі.

GraphQL – мова запитів до API та середовище виконання цих запитів, що дозволяє клієнтам точно вказати, які саме дані їм потрібні, оптимізуючи кількість інформації, що передається через мережу.

Steam – цифрова платформа для розповсюдження комп'ютерних програм та ігор через Інтернет, що дозволяє користувачам завантажувати ігровий контент та спілкуватися з іншими учасниками спільноти.

ВСТУП

У сучасну епоху цифрових технологій відеоігри вже давно перестали бути виключно розвагою. Кіберспорт перетворився на багатомільйонну індустрію, яка охоплює не лише змагання, а й глибоку аналітику, прогнозування та оптимізацію ігрових стратегій. Однією із найпопулярніших дисциплін кіберспорту є Dota 2 – командна стратегія в реальному часі, де аналітика відіграє вирішальну роль у підготовці та проведенні матчів.

Стрімкий розвиток кіберспортивної індустрії та постійне зростання обсягів ігрових даних, створюють запит на інструменти, які здатні наочно представляти, аналізувати та інтерпретувати матчеву статистику. Особливо актуальною є проблема аналізу зіграних матчів та інтерпретування матчевої статистики. Метою може бути як самовдосконалення гравців, так і підвищення командної взаємодії.

Умови війни також сприяють зростанню молоді до віддалених сфер, таких як кіберспорт, адже ці напрями дозволяють реалізувати себе навіть в умовах обмеженої мобільності. Крім того, українська геймерська та кіберспортивна спільнота неодноразово долучалась до зборів на потреби збройних сил, спрямовуючи частину призових фондів, донатів чи організовуючи благодійні турніри. Це робить тематику аналізу ігрових даних не лише актуальної в технічному сенсі, а й у соціально значущою в умовах воєнного стану. Такі проєкти сприяють як розвитку ІТ-індустрії, так і формуванню свідомої, відповідальної молоді, яка поєднує хобі з громадянською позицією.

Хоча існує ряд платформ, які надають доступ до ігрової статистки, більшість із них або занадто узагальнені, або не надають структурованого аналізу. До того ж, не всі сервіси надають можливості зберігати певні матчі користувачам, чи застосування евристичкних методик для аналізу певних аспектів гри.

В рамках цієї дипломного проєкту було створено інформаційну систему, що вирішує вищеперераховані проблеми. Розроблена система, використовуючи API для отримання статистики, не лише дозволяє користувачам аналізувати зіграні матчі, а й надає рекомендації для вдосконалення вибору героїв. Використання

евристичних алгоритмів на практиці демонструє перспективність об'єднання геймдизайну та аналітики. Підсумком реалізації проєкту стане інструментарій, який дозволить користувачам від аматорів до професійних команд, ефективно аналізувати ігровий процес, розвиватись та досягати кращих показників. Така система стена корисною як з навчальною метою так і при командній підготовці до турнірів.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис процесу діяльності

Кіберспорт перетворився на одну з найдинамічніше зростаючих галузей цифрового простору, об’єднуючи мільйони ентузіастів, аналітиків і гравців по всій планеті. Особливо за останні п’ять років індустрія зазнала безпрецедентного зростання – загальний призовий фонд турнірів з кіберспорту досяг кількох десяток мільйонів доларів, а аудиторія основних змагань перевищила навіть показники деяких традиційних видів спорту.

Однією з найпопулярніших дисциплін є Dota 2 – це багатокористувацька стратегічна гра в режимі реального часу. Гра відзначається надзвичайною складністю, глибоким стратегічним і тактичним потенціалом та унікальним балансом. Основи ігрового процесу полягають у протистоянні двох команд по п'ять гравців, кожен з яких керує унікальним героєм із власними унікальними характеристиками та здібностями. Мета ж звучить набагато простіше, просто зруйнувати головну споруду протилежної команди, захищаючи при цьому власну. Кожен матч є джерелом великого обсягу статистичних даних, обрані герої, їхні ролі, результативність гравців тощо [1].

Серед професіоналів та початківців дедалі більшу актуальність набуває глибокий аналітичний розбір ігрових зустрічей. Такий аналіз дає змогу:

– визначити слабкі сторони команди;

– оцінити ефективність вибору героїв;

– визначити фактори, що вплинули на результат гри;

– відстежувати тренди поточної мети та адаптуватися до змін.

На сьогодні значна частина подібної роботи здійснюється вручну або за допомогою сторонніх інструментів із обмеженим функціоналом, які не забезпечують достатньої гнучкості та не пропонують аналізу. Інформаційна система, що представлена у цій роботі, дозволяє вирішити вище згадані проблеми. Можна виділити основні бізнес-процеси, які підлягають автоматизації:

– зчитування та структурування ігрових даних за ID матчу;

– побудова блочного післяматчевого аналізу;

– генерація PDF-протоколу з повним звітом для друку або поширення;

– рекомендація кращого вибору героїв з використанням евристичного алгоритму;

– перегляд актуальної мети героїв;

– авторизація користувача через власний Steam аккаунт;

– перегляд профілю гравця та його останніх матчів.

1.2 Постановка задачі

1.2.1 Призначення системи

Система була розроблена для аналітичної підтримки гравців шляхом автоматизації розбору матчів, відображення статистичних даних та оптимізації вибору героїв за допомогою алгоритмів. Основним напрямом діяльності є управління та аналіз ігрового процесу з метою навчання та розвитку гравців на їх помилках. Ця система може з легкістю застосовуватись у різних контекстах:

– індивідуальний аналіз власних матчів для вдосконалення навичок;

– командний аналіз для розробки стратегій та тактик;

– тренерська діяльність для виявлення помилок;

– аналітика для коментаторів та контент-мейкерів;

– навчальний інструмент для новачків.

1.2.2 Цілі та задачі розробки

Мета проєкту – спрощення та вдосконалення процесу аналізу матчів по кіберспортивній дисципліні Dota 2 шляхом створення системи, що автоматично виконує збір, обробку та візуалізацію ігрових даних, а також дозволяє генерувати звіти у форматі PDF для подальшого використання. Для реалізації цієї мети було поставлено такі завдання:

– провести детальний аналіз предметної області, включаючи механіки, стратегії, взаємодії героїв та знайти доступні API;

– розробити надійну архітектуру клієнт серверного застосунку з можливим майбутнім масштабуванням;

– реалізувати модулі для роботи з API та збереженням інформації;

– інтегрувати алгоритм для оптимального підбору героїв, враховуючи синергії героїв, для підвищення ймовірності перемоги;

– реалізувати систему авторизації користувача через додаток Steam;

– створити модуль для PDF генерації звітів післяматчевої статистики;

– провести тестування системи, використовуючи реальні дані, та оцінити ефективність аналізу у різних сценаріях матчів.

Висновок до розділу 1

У цьому розділі розглянуто предметну область, що стосується аналізу післяматчевої статистики матчів з кіберспортивної дисципліни Dota 2, а також сформульовано призначення, мету, цілі та задачі розробки інформаційної системи, що спрямовані на усунення виявлених недоліків. Запропонована система не лише автоматизує обробку матчевої статистики, а й сприяє розвитку гравців завдяки аналітиці та має стати у нагоді як початківцям, так і професійним або напівпрофесійним командам. Для зручності присутній приємний інтерфейс та здатність генерувати PDF протоколи, що робить цю систему потужним інструментом.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

На даний момент у сучасному кіберспортивному середовищі існує значна кількість програмних рішень, які особливо орієнтовані конкретно на дисципліну Dota 2, що надають користувачам різноманітну статистику та інструменти для удосконалення ігрового процесу. Цей розділ містить перелік та аналіз найпопулярніших таких платформ та також виявлення їх сильних і слабких сторін, що буде слугувати фундаментом для обґрунтування унікальності, а також необхідності розробки власної інформаційної системи.

Із усіх сервісів можна виділити зараз такі найпопулярніші ресурси, як:

– OpenDota;

– Dotabuff;

– Stratz;

– Overwolf.

2.1 Вебсторінка «OpenDota»[2]

OpenDota – це один із найвідоміших ресурсів у спільноті Dota 2, що в свою чергу забезпечує доступ до широкого спектру ігрових даних, включаючи детальну статистику матчів, рейтинги гравців та інші метрики. Платформа працює на основі відкритого API, наданого компанією розробників. В свою чергу ця платформа надає свій розширений API, який дозволяє отримувати доступ до великого обсягу даних для створення власних інструментів або інтеграції з іншими системами. Також для зручності користувачів доступна мобільна вебсторінка. На рисунку 2.1 представлено, як виглядає інтерфейс сторінки гравця. Якщо підсумувати то можна виділити такі переваги:

– обширний масив доступних даних;

– інтуїтивно зрозумілий інтерфейс користувача;

– відкритий програмний інтерфейс API для інтеграції;

– функція пошуку публічного матчу, використовуючи комбінації героїв.

Але попри переваги присутні також і кілька недоліків, а саме:

– відсутність аналізу;

– відсутність функціоналу для формування та експорту звітів.

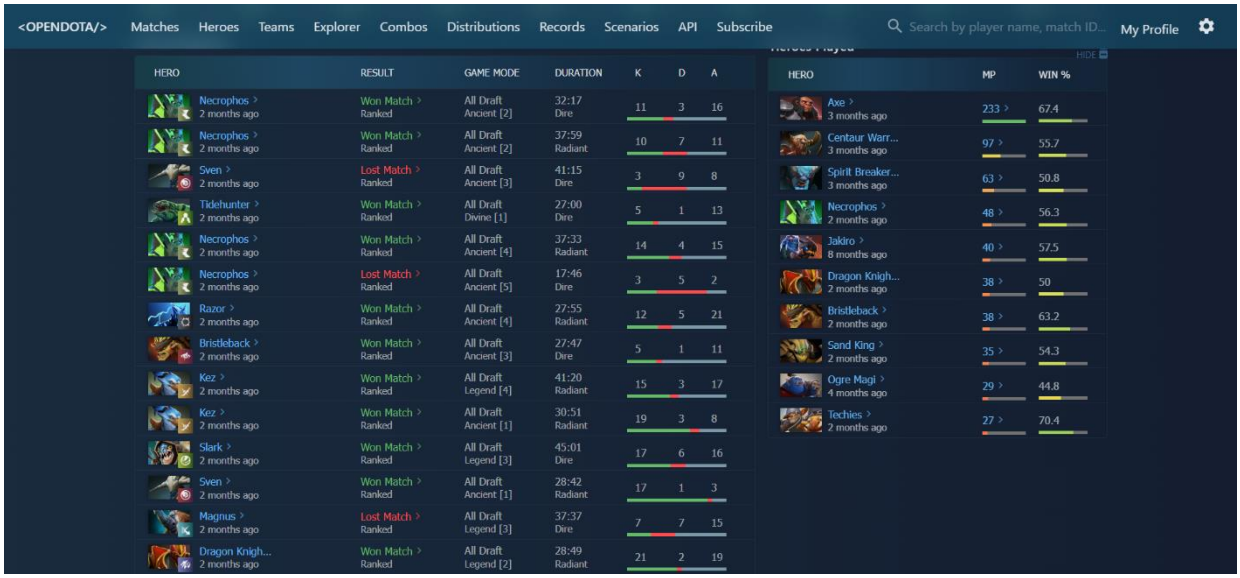

Рисунок 2.1 – Інтерфейс профілю користувача на сторінці OpenDota

2.2 Вебсторінка «Dotabuff»[3]

Dotabuff – це напевно найвідоміший і найстаріший ресурс, який пропонує огляд статистики гравців та історії матчів. Ця платформа широко відома завдяки її простоти. Але дизайн не оновлювався вже дуже давно та варто зазначити, що значна частина розширених функцій доступна виключно для користувачів із преміум-підпискою. На рисунку 2.2 представлено, як виглядає інтерфейс сторінки гравця. Тому як переваги цього ресурсу потрібно відмітити:

– інтерфейс з логічною та зрозумілою структурою;

– велика база історичних даних;

– система відстеження трендів героїв.

В свою чергу також потрібно зазначити і недоліки цієї платформи:

– суттєві обмеження фукціоналу для користувачів без підписки;

– відсутність публічної API для інтеграції з іншими системами;

– неможливість експорту аналітичних даних у форматі звіту.

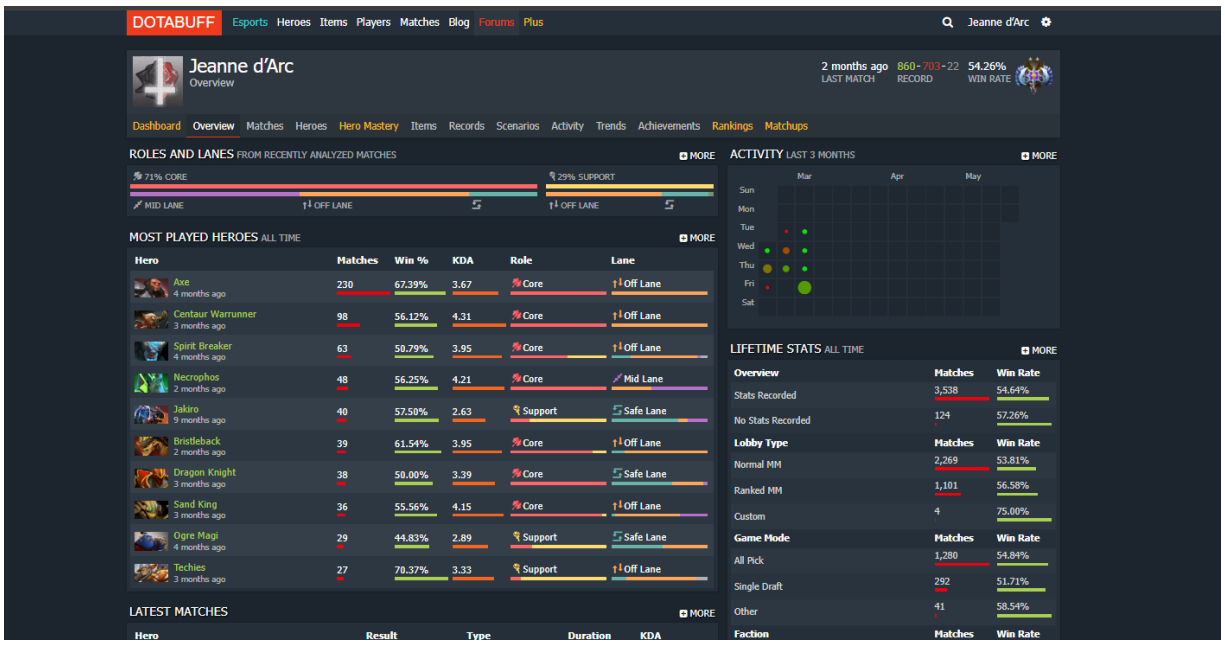

Рисунок 2.2 – Інтерфейс профілю користувача на сторінці Dotabuff

2.3 Вебсторінка «Stratz»[4]

Stratz – це вже одна із найсучасніших аналітичних платформ збору статистичних даних матчів Dota 2. Має сучасний дизайн та зручний користувацький інтерфейс. Цей сервіс надає свій власний гнучкий API, що дозволяє усім користувачам отримувати певні дані для аналізу. Також ця платформа надає можливість перегляду певної інформації з різних турнірів цієї кіберспортивної дисципліни. На рисунку 2.3 представлено, як виглядає інтерфейс сторінки одного із такого турніру. Підсумовуючи можна виділити такі переваги:

– потужний та гнучкий API для розробників;

– комплексна статистика гравців;

– сучасний та зручний дизайн;

– перегляд інформації про наявні турніри.

Із недоліків можна безперечно виділити:

– відсутність функціоналу створення звітів або протоколів для зручного поширення;

– відсутність детального аналізу певних ігрових дій;

– висока навантаженість під час обробки великих масивів даних.

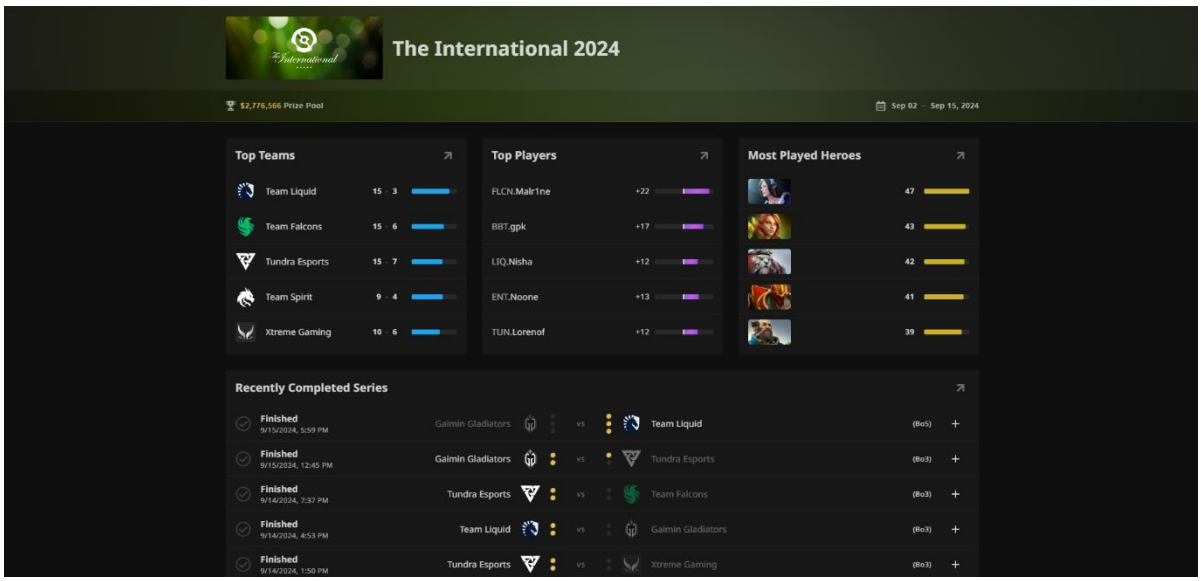

Рисунок 2.3 – Інтерфейс сторінки одного із турнірів на платформі Stratz

2.4 Програмне забезпечення «Overwolf»[5]

Overwolf – це програмне забезпечення, що слугує як допоміжний інструмент в режимі реального часу, надаючи гравцям певні рекомендації. Принцип роботи пов'язаний з ігровими підказками, що пов'язані з певними діями на мапі. Такий інструментарій є корисним для будь-яких гравців, включаючи новачків та навіть досвідчених користувачів. Ця система також дає можливість переглянути невелику статистику перед матчем. На рисунку 2.4 представлено, як виглядає інтерфейс цього додатку. Тому можна виділити такі основні переваги:

– оперативні поради та рекомендації під час гри;

– перегляд невеликої статистики;

– доступність та корисність для гравців різного рівня майстерності;

– безшовна інтеграція з ігровим клієнтом.

Але ця платформа має також значні недоліки, такі як:

– обмежений функціонал для детального аналізу;

– брак можливостей для формування звітів;

– Обмежений функціонал для користувачів без підписки.

Рисунок 2.4 – Інтерфейс профілю користувача додатка Overwolf

2.5 Порівняння існуючих рішень

Після детального розгляду популярних сервісів, наведено таблицю порівнянь вже існуючих рішень (таблиця 2.1).

Таблиця 2.1 – Порівняння існуючих рішень

Функціональні можливості	OpenDota	Stratz	Dotabuff	Overwolf
Перегляд повної статистики	+	+	+	-
Аналіз матчів	-	-	-	-
Рекомендація актуальних героїв	+	+	-	-
Гнучкий API	+	+	-	-
Генерація PDF-звітів	-	-	-	-

Продовження таблиці 2.1

Авторизація через Steam	+	+	+	+
Пропозиція для підбору героїв	-	-	-	+

Висновок до розділу 2

У цьому розділі проведено аналіз найпопулярніших та найактуальніших рішень, для перегляду статистики матчів із кіберспортивної дисципліни Dota 2, таких як: OpenDota, Statz, Dotabuff, Overwolf. Під час аналізу сучасних рішень, було з’ясовано, що кожна платформа має свої переваги та недоліки. Певні системи мають зручний інтерфейс, доступ до відкритого API, пошук матчів по комбінаціям героїв та демонстрація певних порад під час гри. Водночас жоден із цих сервісів не пропонує повного спектра усіх необхідних функцій для користувача у рамках однієї системи, таких як: детальний багаторівневий аналіз, генерація звітів, використання алгоритмів, пропозиція підбору героїв. Також слід зазначити, що деякі рішення, які були розглянуті також обмежуються. Корисні функції можуть бути платними, а використання сторонніх додатків загрожують назавжди блокуванням облікового запису гравця. Це ще раз доводить актуальність розробки власної інформаційної системи, яка буде відповідати усім запитам сучасного користувача. Розроблена система пропонує саме такий підхід, поєднання детального аналізу, перегляду статистики, генерації звітів та зручного інтерфейсу.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Розроблювана інформаційна система призначена для комплексного збору, обробки, аналізу та оптимізації ігрової статистики матчів кіберспортивної дисципліни Dota 2. Можна виділити такі функціональні підсистеми в цій системі:

– підсистема аналітики матчів, що забезпечує обробку зібраних даних з матчу, формування аналізу після матчу, такі як: оцінка вардів, рун, стаків, ліній та командної переваги; мета цієї підсистеми – це дати зрозуміти користувачу проблемні аспекти матчу та можливої причини певного результату матчу;

– підсистема оптимізації вибору героїв, що реалізовує механізм підбору оптимальної комбінації героїв на основі генетичного алгоритму; цей алгоритм враховує різні показники, такі як: ймовірність перемоги проти іншого героя та синергію з іншими героями в одній команді – для формування команди з максимальною ймовірністю перемоги;

– підсистема авторизації та персонального профілю, що надає можливість для авторизації до системи за допомогою вже існуючого акаунту платформи цієї кіберспортивної дисципліни; користувач буде мати можливість бачити свої останні матчі або зберігати певні матчі, щоб пізніше їх знову передивитися;

– підсистема формування PDF-протоколів, що повністю відповідає за створення структурованого протоколу у PDF форматі; це робиться для того щоб користувачі могли з легкістю надіслати їх для розбору гри або роздрукувати та взяти з собою і використовувати під час професійних змагань;

Архітектура передбачає масштабованість для розширення функціоналу в майбутньому. Це може бути, як розширення аналітичних можливостей користувацького профілю з візуалізацією індивідуального прогресу користувача, використання механізмів машинного навчання для поглибленого аналізу на основі мільйони минулих матчів, відстежування поточних турнірів та адаптування архітектури системи для підтримки інших кіберспортивних дисциплін.

Сама інформаційна система реалізується, як односторінковий вебзастосунок (SPA) за допомогою використання JavaScript бібліотеки React.js для клієнтської частини та Node.js для серверної частини. Зберігання даних забезпечується реляційною базою даних PosgresSQL, а обмін між компонентами системи відбувається за допомогою GraphQL API.

3.1.2 Вимоги до надійності

Інформаційна система повинна відповідати певним критеріям надійності та гарантувати безпечне зберігання даних. Це реалізовується завдяки багатьох факторів. Наприклад, автентифікація користувача через захищений протокол Steam OpenID, шифрування id облікового запису користувача, валідація вхідних даних та перевірка бази даних. Система має мати також зручний та зрозумілий інтерфейс для нових користувачів. Прогнозованість роботи досягається за допомогою стабільного часу відгуку системи, забезпечення детермінованості алгоритмів та прозорого масштабування системи без впливу на користувацький досвід.

3.1.3 Вимоги до збереження інформації

За допомогою використання реляційної моделі, дані зберігаються у структурованому вигляді. Також потрібно забезпечити резервні копіювання даних, у випадках відмов або певних помилок, для швидкого відновлення та захист від атак та витоку даних. Система повинна забезпечувати автоматичне відновлення інформації після аварій або помилок. Для гнучкого зберігання даних та обміну через API застосовується JSON.

3.2 Вимоги до функціональних характеристик

Для виконання завдань і досягнення цілей, що були визначені в минулих розділах, система повинна реалізовувати функціонал:

– збір та обробка ігрових даних за допомогою відкритого API; це має включати в себе, як отримання детальної інформації матчів так і синхронізація актуальних даних про героїв;

– проведення післяматчевого аналізу, що має охоплювати різні частини ігрового процесу, а саме: аналіз контролю рун, аналіз розміщення вардів, аналіз стадії ліній, аналіз стаків.

– оптимізація вибору героїв з використанням генетичного алгоритму для формування оптимальних комбінацій, який враховує відсоток перемог з усіма героєм та проти усіх героїв;

– впровадження механізму автентифікації та авторизації користувача, що має бути реалізована з системою Steam OpenID для безпеки та збереження профілю;

– генерація та експорт PDF-звітів з структурованою інформацією на основі проведеного аналізу матча та зручним шаблоном;

– виведення актуальних героїв за останні два тижня з розрахунком його відсотка перемог та кількості матчів; також цей модуль повинен відновлюватись кожен день;

– виведення інформації профіля гравця разом з його останніми зіграними матчами та індивідуальною статистикою.

Функціональні можливості мають виконуватись і працювати коректно, надійно, без збоїв або помилок, згідно визначеним вимогам. Користувацький інтерфейс має бути інтуїтивно зрозумілим та відповідати усім сучасним нормам. Час відгуку основних запитів системи не повинен перевищувати двох секунд. Усі розрахунки, а саме: загальна вартість, золото на хвилину, досвід на хвилину та інші статистичні показники – мають бути точними та відповідати дійсності. Звіти, що будуть формуватися, повинні мати чітку структуру.

Тому можна виділити такі основні вимоги до реалізації функцій:

– коректність усіх функцій, щоб вони працювали, як попередньо планувалось та без помилок;

– зручний інтерфейс, для того щоб користувач інтуїтивно та швидко орієнтувався та був адаптований під інші пристрої;

– точність розрахунків, що має відбуватись із похибкою не більше одного або двох знаків після коми;

– продуктивність, час відповіді на запити не повинен дратувати користувача та відволікати його від користуванням системи.

3.3 Вимоги до видів забезпечення

Інформаційна система аналізу ігрових даних, що охоплює матчеву статистику, базується на ряді ключових складових, кожна з яких є вирішальною для забезпечення працездатності, надійності та коректності роботи програми. Для цього потрібно детально описати вимоги до математичного інформаційного, програмного та технічного забезпечення.

3.3.1 Математичне забезпечення

Математична складова передбачає реалізацію генетичного алгоритму, що використовується для визначення найоптимальнішого вибору героїв для команди. При цьому враховується певна кількість параметрів, наприклад, індивідуальний відсоток перемог для кожного героя та показники ефективності проти героїв. Для покращення результатів застосовується механізми виборів найкращих рішень, кросовер та мутація. Додатково застосовується статистичні методи, такі як обчислення середніх значень та стандартного відхилення. Ці розрахунки є важливими для створення оптимального вибору героїв для кожного матчу.

3.3.2 Інформаційне забезпечення

В основі інформаційного забезпечення лежить реляційна база даних PostgresSQL, що призначена для зберігання структурованої інформації, таких як інформація про героїв, профілі гравців, історія ігор. Отримання даних відбувається за допомогою відкритого API сервісу Stratz, що пропонує великі обсяги даних та

дуже гнучкі запити за допомогою GraphQL, після чого вони піддаються обробці та збереженню у відповідних таблицях, для того щоб не робити постійно запити. Структура таблиць бази даних передбачає логічну організацію інформації за типами. Також реалізовано базові механізми кешування та оновлення даних, що знову таки сприяє зменшенню навантаження на API та підвищує швидкість системи.

3.3.3 Програмне забезпечення

Програмне забезпечення складається з декількох частин. Система використовує для клієнтської частини бібліотеку React.js і для серверної частини Node.js та GraphQL. Для взаємодії з API застосовуються різні бібліотеки такі як pg. Також для генерації PDF-протоколу використовується такий інструмент як pdfmake, що дає змогу легко додавати потрібну інформації до звіту. Усі програмні засоби, що були використані під час роботи над системою є безкоштовними та мають відкриті ліцензії, це дає розуміння, що вони придатні для використання в освітніх проєктах.

3.3.4 Технічне забезпечення

Для роботи клієнтської частини передбачається використання браузера з підтримкою мови програмування JavaScript, наприклад Google Chrome або Mozila Firefox. Користувачі також отримують функціонал з комп’ютера, планшета або смартфона, тому що система є адаптивною, але вона повинна мати доступ до стабільного інтернет з’єднання для найкращої роботи системи. Ну і також пристрій має мати достатню обчислювальну потужність. Для роботи серверної інфраструктури потрібно забезпечити стабільний зв’язок із API та проходження захисту CloudFlare. Сервер може бути розміщений як на локальному сервері або бути задеплоєним з підтримкою Node.js. Також важливо забезпечити створювання резервних копій бази даних для запобігання втрати інформації.

Висновок до розділу 3

У цьому розділі сформульовано вимоги до інформаційної системи аналізу ігрових даних та матчевої статистики кіберспортивної дисципліни Dota 2. Визначено архітектурні принципи побудови системи та описано функціональні підсистеми: аналітики матчів, оптимізації вибору героїв, авторизації та персонального профілю, формування PDF-протоколів.

Встановлено критерії надійності системи, що включають захист інформації через автентифікацію Steam OpenID, валідацію вхідних даних та забезпечення стабільного часу відгуку. Визначено вимоги до збереження інформації з використанням реляційної моделі PostgreSQL та JSON-формату для API.

Сформульовано функціональні характеристики системи: збір та обробка даних через API, післяматчевий аналіз, оптимізація вибору героїв генетичним алгоритмом, автентифікація користувачів, генерація PDF-звітів, відображення актуальної статистики героїв.

Визначено вимоги до математичного забезпечення (генетичний алгоритм), інформаційного забезпечення (PostgreSQL, API Stratz) та програмного забезпечення (React.js, Node.js, GraphQL).

Сформовані вимоги становлять технічну основу для проектування та реалізації системи, що відповідає сучасним стандартам розробки вебзастосунків.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

4.1 Вибір архітектури та загальний підхід

Сучасна розробка вебзастосунків орієнтується на рішеннях, що забезпечують швидку взаємодію з користувачем без потреби повного оновлення сторінки. У зв’язку з цим дедалі частіше створюються односторінкові додатки (SPA, Single Page Application). Саме така архітектура дозволяє реалізувати взаємодію, подібну до класичного програмного забезпечення. Односторінковий підхід, дозволяє користувачам безперешкодно взаємодіяти з застосунком, подібно до використання програмного забезпечення. Односторінковий підхід забезпечує надсилання браузеру лише необхідного коду або контенту у відповідь на дії користувача, без повного перезавантаження сторінки. Завдяки цьому досягається більш плавна й швидка робота інтерфейсу. Такий підхід істотно відрізняється від традиційної моделі, де кожен запит до сервера супроводжується завантаженням нової сторінки.

З урахуванням плюсів односторінкової архітектури, саме її було обрано для розробки інформаційної системи аналізу та перегляду матчевої статистики для кіберспортивної дисципліни Dota 2, оскільки вона буде гарантувати високу продуктивність, зручність, а також гарантує легке розширення функціоналу. В основі архітектури лежить класична модель клієнт-сервер, де фронтенд та бекенд реалізовані окремо, як окремі розподілені частини, що мають свої задачі.

4.2 JavaScript

JavaScript – це одна з найпопулярніших мов програмування у сфері розробки вебзастосунків, яка підтримується всіма сучасними браузерами та має повністю інтегрування з HTML/CSS. Вона було створена для того щоб “оживити вебсторінки”. Ця мова також підтримує об’єктно орієнтоване та функціональне програмування. Сучасний JavaScript не надає низькорівневого доступу до пам’яті чи процесора, оскільки була створена для браузерів, які цього не потребують. Вбудований в браузер JavaScript може робити все, що пов’язано з управлінням

вебсторінками, взаємодією з користувачем та сервером. Спочатку вона була створена тільки лише як мова для браузера, але сьогодні її використовують і в інших задачах. JavaScript може виконуватися на сервері, що має спеціальну програму, що буде читати скрипт, компілювати його у машинний код, а потім виконувати цей код. Ця програма називається рушій і вона оптимізує процес на кожному із раніше згаданих етапів та навіть слідкує за скомпільованим скриптом під час його виконання [6].

Тому нашій системі буде використовуватися саме ця мова. Вона застосовується як для клієнтської частини так і для серверної логіки, про це детальніше буде розповідатися пізніше. Також велика перевага в тому, що ця мова має розвинену екосистему бібліотек та здатність працювати на різних рівнях програмного стеку, що робить її ідеальним вибором для застосунків, що схожі на нашу систему.

4.3 Node.js

Node.js – це однопоточне середовище виконання JavaScript на сервері, з відкритим вихідним кодом для створення швидких і масштабованих серверних і мережевих програм. Це середовище дає можливість створювати високопродуктивні асинхронні рішення. Node.js використовує архітектуру "Single Threaded Event Loop" для одночасної обробки кількох клієнтів [7]. Оскільки він використовує менше потоків, він використовує менше пам’яті, що підвищує продуктивність та пришвидшує виконання поставлених задач.

Тому для наших цілей ця однопотокова архітектура еквівалентна багатопотоковій архітектурі. У рамках нашого додатку Node.js, використовується для реалізації серверної логіки, взаємодії з базою даних і роботи з GraphQL. Завдяки Node.js також вдалось спростити розробку та підтримку системи, тому що на фронтенді та бекенді використовується одна мова програмування.

4.4 React.js

React.js – це популярна JavaScript бібліотека, що була розроблена компанією Facebook. Вона була створена для розробки динамічних користувацьких інтерфейсів. Її головною перевагою є компонентний підхід до розробки, що дає змогу створювати компоненти та використовувати їх декілька разів. Також компоненти можуть передавати властивості та дані один одному, але тільки в одному напрямку – від батьківських до дочірніх. Це допомагає реалізувати чітку ієрархію та полегшує налагодження. Однонаправлений потік даних полегшує розуміння, звідки саме до елементу надійшли дані. Ще однією особливістю React є те, що він створює та зберігає в кеші віртуальне DOM-дерево – копію DOM, яка змінюється швидше, ніж реальна структура. Це потрібно, щоб швидко оновлювати сторінки. Якщо користувач виконає дію або настане будь-яка подія, DOM повинна змінитись, оскільки зміняться об'єкти на сторінці. Але реальна об'єктна модель може бути величезною, її оновлення є повільним процесом. Тому React працює не з нею, а з віртуальною копією у кеші, яка важить менше [8].

Завдяки цій бібліотеці буде забезпечуватись швидке оновлення лише тих частин інтерфейсу, які змінюються, що помітно підвищує продуктивність розробленої системи і саме це нам буде потрібно, для того щоб реалізовувати SPA, включаючи сторінки профілю гравця, компоненти аналізу матчів та оптимізації вибору героїв.

4.5 GraphQL

GraphQL – це мова формування запитів та маніпулювання даними з відкрити кодом для API. Вона була розроблена компанією Facebook, та надає можливість користувачам дані, які їм необхідні, без надлишкової інформації. Ця мова стала швидко популярною завдяки своїй концепції запиту. Користувач отримує саме те, що йому потрібно. Під час надсилання запитів до API GraphQL повертає дуже передбачуваний результат без будь-якої надмірної або недостатньої вибірки, що

гарантує швидкість, стабільність та масштабованість додатків.

На відмінну від REST, для кожного ресурсу визначаються окремі маршрути, GraphQL дає змогу користувачу самостійно обирати потрібні поля та об’єкти в одному запиті. Це особливо корисно для систем із великою кількістю пов’язаних даних, де REST може призводити до надлишкових або повторних запитів [9]. Різницю в запитах між технологіями REST та GraphQL представлено на рисунку 4.1. Також GraphQL дозволяє використовувати інструменти для автоматичної генерації документації, перевірки запитів на відповідність схемі, що знову спрощує розробку і тестування системи.

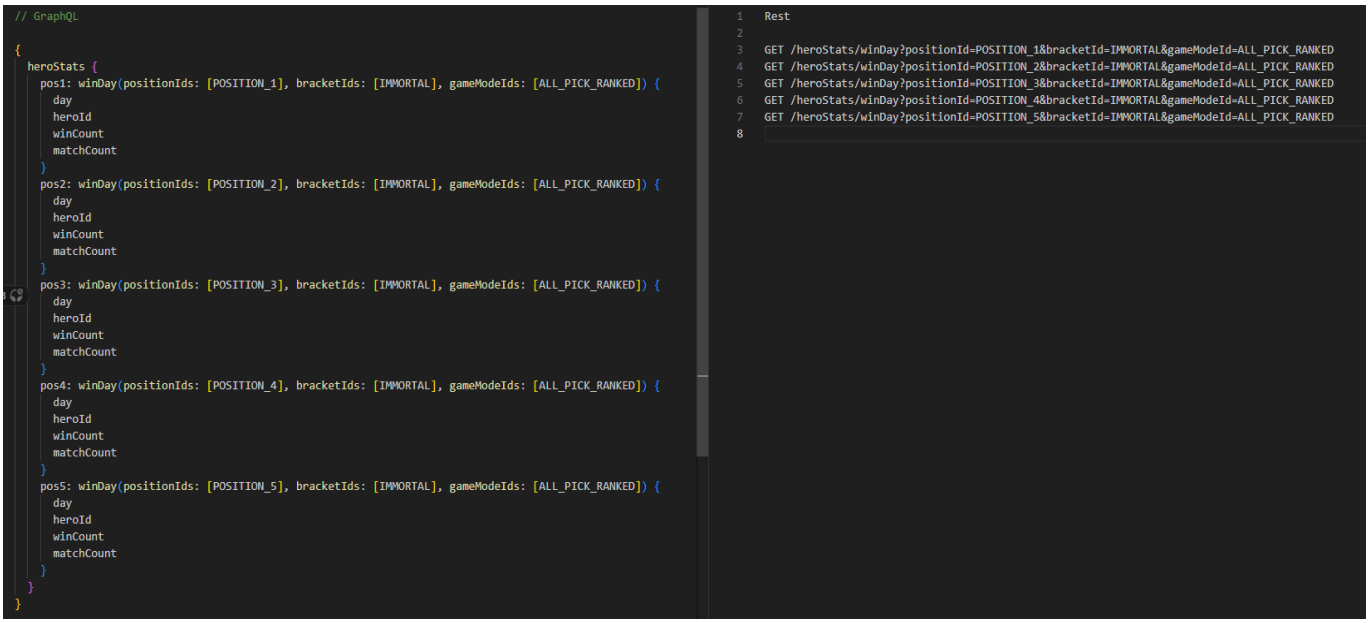

Рисунок 4.1 – Структура клієнтської частини

У системі аналізу та перегляду матчевої статистики GraphQL виступає основним способом взаємодії між клієнтською частиною та зовнішнім API. Саме GraphQL дає можливість уникати передачі зайвих даних, скоротити кількість звернень до сервера та полегшує обробку складних структур даних. Завдяки цьому можна будувати запити з урахуванням потреб системи, що потрібні для аналізу та виведення статистики для користувача.

4.6 PostgreSQL

PostgreSQL – це передова система управління базами даних корпоративного класу з відкритим вихідним кодом. Вона має розвинені механізми індексації, реплікації, забезпечує продуктивність навіть при значних обсягах інформації та підтримує як реляційні(SQL) так і нереляційні(JSON) запити. Це дуже стабільна система управління базами даних, що підтримується більше ніж 20-річним досвідом спільноти розробників. Цей ретельний та спільний підхід сприяє її високому рівню стійкості та коректності. PostgreSQL використовується як головне сховище даних для багатьох вебдодатків [10].

В межах проєкту PostgreSQL використовується для зберігання інформації про героїв, гравців та матчів, а завдяки кешуванню й оптимізованим запитам забечпечується стабільна та швидка робота системи.

4.7 Інші технології

Варто також зазначити, що в застосунку використовується бібліотека pdfmake, що дає можливість генерувати PDF-файли безпосередньо у браузері. Саме це забезпечує користувачу створення звітів, в яких зберігається аналітика та статистика матчів, з структурованим виглядом, що можна використовувати під час змагань або у навчальних цілях [11].

Суттєву роль в системі має механізм автентифікації через додаток Steam, платформи, де знаходиться сама дисципліна. Цей додаток надає відкритий протокол автентифікації – Steam OpenID, що дозволяє користувачам без перешкод входити на сторонні ресурси, використовуючи вже існуючий обліковий запис [12]. У нашій системі це забезпечує інтеграцію з екосистемою Steam, що дозволяє безпечно та оперативно ідентифікувати користувача без потреби створювати новий обліковий запис у додатку. Цей механізм гарантує достовірність користувача через цифровий підпис та дозволяє отримувати профільні дані без прямого доступу до облікових записів. Варто також зазначити, що навіть після того, як ви отримаєте

автентифікацію користувача в Steam, якщо видимість його профілю встановлена як приватна, отримати жодні його дані неможливо.

Головна перевага цього підходу не лише в спрощенні процесу входу, а й у підвищенні безпеки, тому що усі операції з паролями та іншими важливими даними виконує сам Steam, що є перевіреною та авторитетною платформою. Це дає можливість не витрачати часу на реалізацію власної інфраструктури управління користувачами.

Висновок до розділу 4

У цьому розділі було детально обґрунтовано вибір технологій, що стали основою для створення інформаційної системи аналізу та перегляду матчевої статистики матчів кіберспортивної дисципліни Dota 2. Було обрано концепцію односторінкового застосунку (SPA), що забезпечує швидкість, динамічність та приємну взаємодію з користувачем. Основною перевагою такої архітектури є відокремлення фронтенд та бекенд частини, що надає гнучкості системі та дозволяє легше підтримувати код у майбутньому.

Обрані технології формують цілісну та збалансовану екосистему для розробки сучасного вебзастосунку. Використання JavaScript як єдиної мови програмування для клієнтської та серверної частини спрощує процес розробки та підтримки системи. React.js забезпечує створення динамічного користувацького інтерфейсу з компонентним підходом, Node.js надає ефективне середовище для серверної логіки, GraphQL оптимізує взаємодію з API, а PostgreSQL гарантує надійне зберігання даних.

Загалом, запропонований технологічний стек повністю відповідає вимогам проєкту, забезпечує масштабованість системи та створює міцну основу для реалізації всіх функціональних можливостей інформаційної системи аналізу ігрових даних.

5 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Структура системи

Структура інформаційної системи для аналізу та перегляду матчевої статистики кіберспортивної дисципліни Dota 2 розроблялась, як готовий вебзастосунок з клієнтської та серверної частиною, що використовує технології SPA(Single Page Application) та архітектуру завдяки якій можна легко масштабуватися, додаючи нові аналітичні блоки. Саму систему можна поділити на таку структуру:

– клієнтська частина реалізована за допомогою мови програмування JavaScript та бібліотеки React.js; ця бібліотека займається побудовою усіх сторінок, відображенням інтерфейсу, зв’язком з серверною частиною та генерацією PDF-протоколів; завдяки цьому користувач може зареєструватися на сторінці, бачити свій профіль, історію матчів та переглядати інформацію по обраному матчем;

– серверна частина реалізована за допомогою використання платформи Node.js для побудови та обробки запитів з клієнтської частини, взаємодії з усіма таблицями в використаній базі даних та з API; з останнім допомагає мова написання запитів GraphQL, завдяки якій, було написано усі зручні та компактні запити до відкритого API, серверна частина отримувала лише потрібну інформацію; також в цьому модулі реалізовується використання генетичного алгоритму, який обраховує та намагається знайти найкращий вибір героїв для кожної команди;

– за зберігання усієї інформації відповідала система управління базами даних PostgeSQL, яка має задану структуру таблиць з користувачами та героями; крім цього в базі даних кешуються сесії користувачів, для того щоб відвідувач був зареєстрований навіть після того, як перезапускається вебзастосунок; вона зберігає структуровану інформацію, що отримує з клієнтської частини; головним є також те, що вона зберігає в нашому випадку як реляційні моделі дані так і не реляційні, що набагато спрощує розробку;

– уся зовнішня інформація, що зберігається та оброблюється в системі, отримується з відкритих API; перша із них це Stratz API, що надає безкоштовно повний перелік усіх можливих даних із дисципліни, для якої була створена наша система, а саме: дані про героїв, матчів, статистику, турніри, відсоток перемог, тощо [13]; і другий сервіс Steam OpenId, завдяки якому була легко реалізована безпечна система реєстрації користувачів, що також полегшує користування застосунку, тому що у кожного гравця є вже створений обліковий запис.

5.2 Функціональна модель системи

У розробленій системі всі користувачі є рівноправними та мають доступ до повного функціоналу. Авторизація не є обов'язковою умовою для використання основних можливостей системи, однак зареєстровані користувачі отримують додаткові функції. Функціональні можливості не зареєстрованого користувача включають:

– перегляд актуальних героїв з найвищого рейтингу;

– пошук матчу по його ідентифікаційному номеру;

– перегляд профілю будь-якого користувача;

– обрати матч із певного профілю;

– переглянути сторінку матчу, в якій наведено його результат статистику та аналіз;

– використання функції оптимізування набору героїв, для того щоб знайти найкращий вибір з покращеною ймовірністю перемоги;

– генерація PDF-протоколу матчу.

Перелік додаткових функціональних можливостей зареєстрованого користувача:

– додавання матчу до улюблених, з цілю зберегти певний матч та потім його легко знайти в наступний раз;

– написання невеликого нотатку до матчу, що зрозуміти з легкістю розуміти, що це був за матч або можливо яка подія там була.

Діаграма варіантів використання представлена на кресленику ІС13.020БАК.005 Д1.

5.3 Модель бази даних

У даній системі було застосовано система управління базами даних PostgeSQL, що надала можливості зберігати, оновлювати дані в створених таблицях, що використовувалися вже в застосунку для перегляду та аналізу відповідної статистики. Саму структуру бази даних для проєкту було описано нижче і вона складається з таких таблиць:

– таблиця Users містить основну інформацію, що потрібна в застосунку про користувачів, які реєструються через додаток Steam; вона є однією з головних таблиць, тому що зв’язується з іншими через зовнішній ключ, завдяки чому реалізовується функціонал зберігання обраних користувачем матчів; поля таблиці Users наведено в таблиці 5.1;

Таблиця 5.1 – Структура таблиці Users

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор користувача системи
steam_id_64	bigint	Унікальний Steam ID користувача у форматі 64 біт
steam_id_32	integer	Унікальний Steam ID користувача у форматі 32 біт
display_name	text	Ім’я користувача, що він поставив собі у додатку
avatar	text	Посилання на зображення, що користувач завантажив
created_at	timestamp	Дата створення цього запису
updated_at	timestamp	Дата оновлення цього запису

– таблиця Session містить інформацію про сесії користувача, що використовуються для авторизації користувача; таке рішення дає можливість

користувачу завжди бути зареєстрованим в системі навіть після перезапуску додатка; поля таблиці Session наведено в таблиці 5.2;

Таблиця 5.2 – Структура таблиці Session

Назва поля	Тип даних	Опис
sid	varchar	Ідентифікатор сесії
sess	json	Об’єкт сесії у форматі JSON
expire	timestamp	Час завершення сесії

– таблиця Favorite_matches містить у собі список матчів, які були відзначені як улюблені, користувача, що був зареєстрований у додатку; це надає можливості зберігати певні матчі та потім їх окремо переглядати через певний час; поля таблиці Favorite_matches наведено в таблиці 5.3;

Таблиця 5.3 – Структура таблиці Favorite_matches

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор
user_id	bigint	Зовнішній ключ на користувача
match_id	bigint	ID матчу
created_at	timestamp	Дата додавання в обране

– таблиця Match_notes містить у собі інформацію про нотатки для матчів; зареєстрвоаний користувач, має можливість написати коротенький текст, що може бути як якась подія, що відбулась або певні досягнення; поля таблиці Match_notes представлено в таблиці 5.4;

Таблиця 5.4 – Структура таблиці Match_notes

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор нотатки

Продовження таблиці 5.4

user_id	integer	Ідентифікатор користувача (steam_id_32), що залишив нотатку
match_id	bigint	ID матчу, до якого залишено нотатку
note	text	Текст нотатки користувача
created_at	timestamp	Дата та час створення нотатки
updated_at	timestamp	Дата та час останнього оновлення нотатки

– таблиця Heroes містить у собі усю інформацію про героїв у кібеспортивній дисципліні Dota 2; це може бути як характеристики, назва, головний атрибут, тощо; поля таблиці Heroes представлено в таблиці 5.5;

Таблиця 5.5 – Структура таблиці Heroes

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор героя
display_name	text	Назва героя
short_name	text	Коротке ім’я для зображень
attack_type	text	Тип атаки
primary_attribute	text	Основний атрибут
strength_base	integer	Базова сила
agility_base	integer	Базова спритність
intelligence_base	integer	Базовий інтелект
strength_gain	double	Приріст сили
agility_gain	double	Приріст спритності
intelligence_gain	double	Приріст інтелекту
attack_range	integer	Дальність атаки

Продовження таблиці 5.5

attack_rate	double	Швидкість атаки
starting_armor	double	Початкова броня
move_speed	integer	Швидкість переміщення
hp_regen	double	Відновлення HP
mp_regen	double	Відновлення MP
lore	text	Лор героя
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення

– таблиця Hero_roles містить у собі роль, яку зберігає у собі кожний герой, наприклад підтримка, ініціатор тощо; і також відображається степінь від одиниці до трьох на скільки цей герой належить цій ролі; поля таблиці Hero_roles представлено в таблиці 5.6;

Таблиця 5.6 – Структура таблиці Hero_roles

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор
hero_id	integer	Зовнішній ключ на heroes
role_id	text	Роль героя
level	integer	Степінь причетності героя до цієї ролі

– таблиця Hero_meta_stats містить у собі дані сьогоднішньої популярності героїв, що дозволяє визначити найефективнішого героя на конкретній позиції; ця функція буде використовуватися для того щоб користувачі мали можливість переглядати найактуальніших гравців за певний період часу; крім цього ці дані використовуються для генетичного алгоритму у застосунку; поля таблиці Hero_meta_stats представлено в таблиці 5.7.

Таблиця 5.7 – Структура таблиці Hero_meta_stats

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор
hero_id	integer	Зовнішній ключ на heroes
win_count	integer	Кількість виграних ігор
match_count	integer	Загальна кількість матчів
position_id	integer	Позиція (1–5)
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення

Усі таблиці таблиці формують функціонуючу структуру бази даних, яка зберігає великий масив даних, що забезпечують роботу системи. На основі цих даних, працюють усі функції та користувачі можуть бачити статистику та аналітику матчів в вебсторінці. Діаграма структури бази даних представлена на кресленику ІС13.020БАК.005 Д4.

5.4 Передавання та обробка даних

Передавання даних у системі реалізовано за допомогою окремих компонентів. У кожному компоненті реалізовуються запити, що відповідають обробнику. Це можуть бути запити на перевірку авторизації, створення звітів, формування аналітики тощо. Кожен запит обробляється та при необхідності логується. Сам процес передавання даних можна описати в декілька етапів, кожна з яких відповідає за різні передачі даних. Для опису потрібно почати з вхідних даних, що можуть надходити або в HTTP або GraphQL запиті з фронтенд частини, після певних дій користувача в додатку. Такими діями можуть бути:

– запит на створення протоколу;

– запит на перегляд облікового запису користувача;

– запит на оновлення актуальних даних героїв;

– запит на оптимізацію вибору героїв;

– запит на перегляд статистики або аналітики матчу.

Після кожної такої дії, запит іде до бекенд частини, яка виконує попередню обробку, перевірку запиту та прав доступу. Сам же сервер може відповісти помилкою. Якщо все добре то сервер звертається до зовнішніх компонентів, наприклад відкрите API або база даних, від них він може отримати певну інформацію та почати її додатково обробляти, рахувати або перетворювати у різні формати. Після цих дій він ці дані повертає до клієнтської частини. В певних випадках дії від користувача можуть спровокувати вже зміну певних таблиць в базі даних, таких як збереження улюблених матчів користувача, записи нотатків або додавання користувача. Це значно зменшує завантаженість на зовнішні API, що дає можливість підвищити працездатність системи. Усі дії відбуваються з урахуванням усіх типів даних та відповідних зв'язків. Самі дані, що відправляються до клієнтської частини, можуть надсилатися у різних форматах, наприклад JSON або PDF файл. В будь якому випадку усі вихідні дані користувач бачить у клієнтському інтерфейсі додатка. Діаграма взаємодії компонентів представлена на кресленику ІС13.020БАК.005 Д2.

5.5 Архітектура програмного забезпечення

Архітектура цього програмного забезпечення використовує сучасний підхід з використанням клієнт-серверної частини (React.js та Node.js) та системою управління базами даних PostgreSQL для збереження даних у таблиці. Такий підхід архітектури можна умовно назвати спрощеним шаблоном MVC [14]. Діаграма розгортання представлена на кресленику ІС13.020БАК.005 Д3. Тут React.js відповідає за роль View, який відображає інформацію користувачам, що використовує компонентний підхід. Клієнтська частина побудована з використанням сучасних інструментів розробки, включаючи Vite як збирач проекту. В цьому модулі всі компоненти структуровані та розділені на логічні

блоки, що відповідають за певний функціонал. Архітектура передбачає використання різних UI-компонентів та сторінок додатку. Усі статичні ресурси, а саме: зображення та іконки – організовані в окремій папці для забезпечення ефективного завантаження. Структура клієнтської частини наведена на рисунку 5.1.

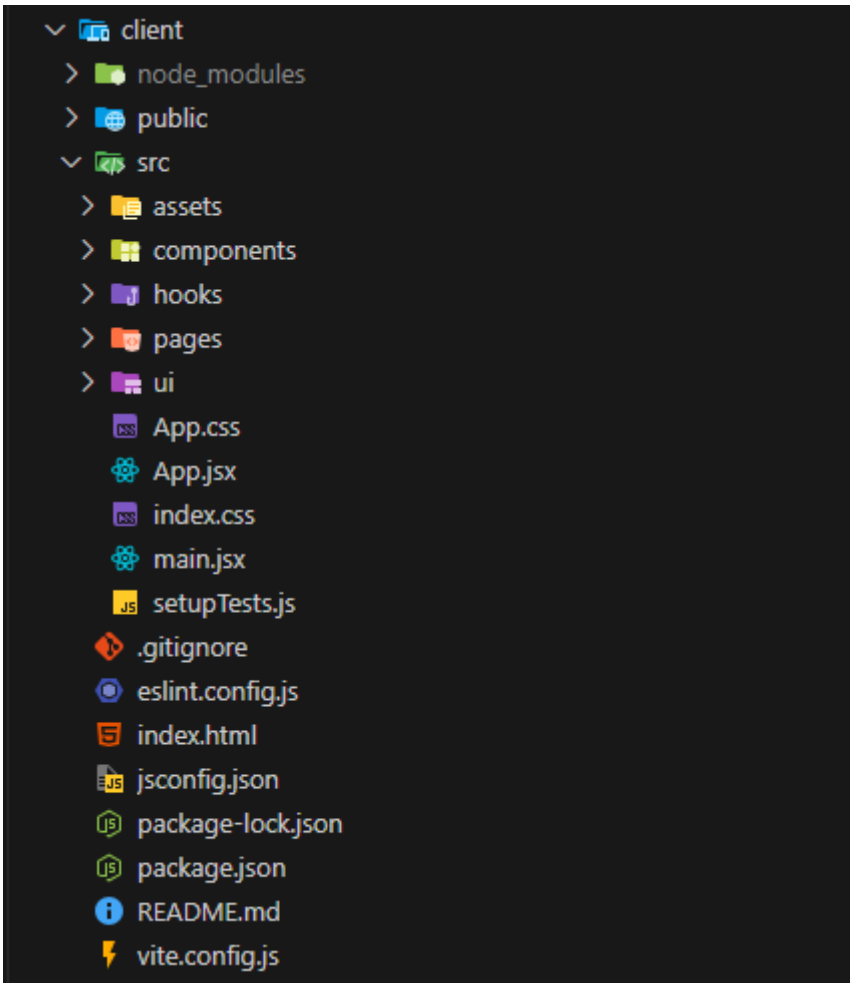

Рисунок 5.1 – Структура клієнтської частини

В ролі Controller виступає серверна частина Node.js, що обробляє запити користувачів, відповідає за певні обчислення та підключається до бази даних PostgreSQL, що виконує роль Model, яка зберігає усю інформацію додатку. Архітектура серверної частини організована за модульним принципом, де кожен компонент має свою чітко визначену функцію. Основні маршрути програмного застосунку розділені на окремі файли, що забезпечує зручність підтримки та масштабування системи. До ключових модулів належать компоненти для роботи з

героями, матчами, оптимізацією та автентифікацією. Саму структуру серверної частини наведено на рисунку 5.2.

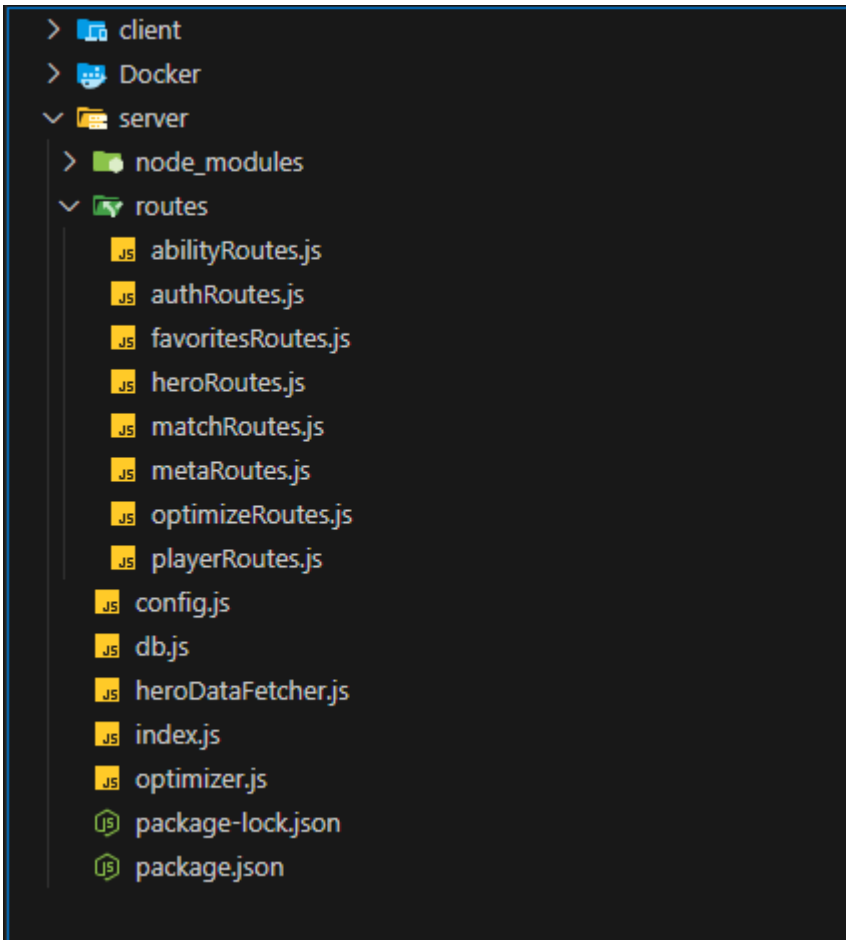

Рисунок 5.2 – Структура серверної частини

Висновок до розділу 5

У цьому розділі детально представлено та охарактеризовано повну логічну структуру розробленої інформаційної системи для перегляду та аналізу статистики кіберспортивної дисципліни Dota 2. Детально описано всі її основні функціональні компоненти та чітко визначені ролі акторів та їх дії з системою. Також було надано опис моделі даних, який включає в себе опис усіх таблиць, що були застосовані під час розробки проєкту та реалізовані за допомогою системи керування базами даних PostgreSQL. В розділі розглянуто процес обробки даних та користувацьких запитів. Проаналізовано вхідні та вихідні дані та обмін даними за зовнішніми API

сервісами. Увагу також було приділено оптимізації цих процесів для того щоб забезпечити високу працездатність системи. Було досліджено архітектуру програмного забезпечення, яка була спроєктована за умовним шаблоном MVC(Model-View-Controller), що є актуальною та поєднує в собі якісну обробку даних і зрозумілу візуалізацію інтерфейсу та даних. Крім цього представлено структуру серверної та клієнтської частини. Такий підхід до проєктування архітектури, забезпечив структуроване розмежування між компонентами, що виконують кожний свою задачу та мають власну відповідальність та може спростити процес масштабування та подальшого вдосконалення системи

6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Змістовна постановка задачі

Мета – розробка системи для аналізу ймовірності перемоги команди та надання рекомендацій щодо оптимального вибору героїв у грі Dota 2 з використанням генетичного алгоритму. Тобто знайти такий склад з 5 героїв для команди (Dire), який забезпечить максимальну ймовірність перемоги проти відомого складу супротивника (Radiant) і навпаки.

6.1.1 Вхідні дані

Наведено перелік вхідних даних, на основі яких здійснюється формування оптимального вибору героїв:

– історичні дані матчів з API Stratz (статистичні дані про ефективність кожного героя та взаємодію між ними, тобто коли вони в одній команді або в протилежних);

– склад супротивника (5 героїв);

– список усіх доступних героїв.

6.1.2 Обмеження

Під час підбору оптимального складу героїв необхідно враховувати такі обмеження, що випливають із правил гри:

– кожен гравець може обрати лише одного героя;

– кожен герой під час матчу унікальний.

6.1.3 Очікуваний результат

У результаті роботи системи мають бути отримані такі результати, що відображають ефективність підібраної команди:

– оцінка ймовірності перемоги команди з поточним вибором героїв;

– рекомендації щодо оптимального вибору героїв, які підвищують ймовірність перемоги.

6.2 Математична постановка задачі

Завдання оптимального підбору команди героїв у грі Dota 2 можна сформулювати як комбінаторну задачу оптимізації з використанням критеріїв синергії між героями та ймовірності перемоги команди.

Метою є знайти таку п’ятірку героїв для команди Radiant або Dire, яка має найбільшу ймовірність перемоги проти заданого складу супротивника, з урахуванням історичних даних взаємодії героїв за даними з Stratz API.

6.2.1 Вхідні дані

Позначення основних змінних, які використовуються в алгоритмі:

– H = {h1, h2, …, hn} - множина всіх доступних героїв у грі;

– T_{enemy} = {e1, e2, …, em}⊂ H - фіксована команда супротивника;

– T_{ally} = {a1, a2, …, am}⊂ H - кандидатна команда союзників (підбирається оптимально);

– S(hi, hj) - коефіцієнт синергії,що вираховується на основі відсотків перемог між героями hi та hj (чим більше, тим краще взаємодіють);

– W(hi, ej) - коефіцієнт протидії героя hi проти ej, що базується на ймовірності перемоги в очних зустрічах.

6.2.2 Обмеження

У математична модель має такі обмеження:

– усі герої в команді мають бути унікальними: ai ≠ aj для i ≠j;

– жоден герой з T_{enemy} не може бути у T_{ally}: $T_{ally} \cap T_{enemy} = \emptyset$.

6.2.3 Цільова функція

Необхідно максимізувати фітнес-функцію, яка оцінює якість кандидатної команди T_{ally} проти заданого складу T_{enemy}. Фітнес функція буде визначатися як

$$Fitness(T_{ally}) = \alpha * Synergy(T_{ally}) + \beta * Counter(T_{ally}, T_{enemy}), \quad (6.1)$$

де α, β – вагові коефіцієнти, що регулюють баланс між внутрішньою синергією та контрпіками (наприклад, α = 0.4, β = 0.6);

$Synergy(T_{ally})$ – середній коефіцієнт синергії всередині команди;

$Counter(T_{ally}, T_{enemy})$ – середній відсоток перемог союзників проти ворогів.

Середній коефіцієнт синергії всередині команди буде визначатися як

$$Synergy(T_{ally}) = \frac{1}{n * (n-1)} \sum_{i=1}^{5} \sum_{j=i+1}^{5} S(h_i, h_j), \quad (6.2)$$

де n – кількість героїв у союзній команді;

$S(h_i, h_j)$ – коефіцієнт синергії.

Середній відсоток перемог союзників против ворогів буде визначатися як

$$Counter(T_{ally}, T_{enemy}) = n * m \sum_{i=1}^{5} \sum_{j=1}^{5} W(h_i, e_j), \quad (6.3)$$

де n – кількість героїв у союзній команді;

m – кількість героїв у ворожій команді;

$W(h_i, e_j)$ – коефіцієнт протидії.

6.3 Обґрунтування методу розв’язання

У задачах оптимального добору елементів (у нашому випадку героїв) серед великої кількості комбінацій, які не піддаються повному перебору через експоненційне зростання кількості варіантів, традиційні методи (наприклад, повний перебір або динамічне програмування) неефективні через високу обчислювальну складність. На практиці для таких задач використовуються евристичні та мета евристичні методи, серед яких виділяють:

– генетичні алгоритми;

– бджолиний алгоритм;

– евристичний пошук A*;

– мурашиний алгоритм;

– алгоритм рою часток;

– алгоритм зозулі.

В нашому випадку ми обрали генетичний алгоритм з огляду на такі переваги, як: ефективність у вирішенні комбінаторних задач, гнучкість в адаптації під різні типи цільових функцій, дозволяють використовувати евристичну оцінку рішень через функцію пристосованості. Вище приведені алгоритми нам би менше допомогли, тому що наприклад мурашиний алгоритм краще працює в задачах з безперервним або маршрутним простором рішень, а не для дискретних задач. А с бджолиний більше орієнтований на локальний пошук навколо перспективних областей, що могло б призвести до втрати глобальних оптимальних рішень. Ну і вони обидва би мали складніше реалізацію ніж генетичний.

Для побудови моделі використовується GraphQL API платформи Stratz, яка надає статистику ігрових матчів у Dota 2. Враховуючи специфіку задачі, пов'язаної з вибором п'яти героїв із набору понад 120 можливих, використання ГА дозволяє уникнути повного перебору понад 10^9 варіантів, при цьому отримати якісне рішення за рахунок комбінування сильних індивідуально та синергетично взаємодіючих героїв за прийнятний для нас час.

Таким чином, генетичний алгоритм є доцільним та ефективним методом вирішення поставленої задачі інформаційної системи аналізу ігрових даних та матчевої статистики обраної кіберспортивної дисципліни в умовах великої кількості варіантів і нечіткої цільової функції, яка ґрунтується на емпіричних статистичних даних.

6.4 Опис методу розв’язання

Генетичний алгоритм імітує природний процес еволюції, де кожне можливе рішення (у нашому випадку набір із 5 героїв) розглядається як "особина". Алгоритм поступово покращує якість рішень шляхом відбору найкращих особин, їхнього схрещування та внесення випадкових змін [15]. Для опису методу розв’язання найкраще підійде приклад розв’язання.

Перший етап – це ініціалізація популяції. Спочатку ми створюємо набір випадкових героїв (популяцію). Наприклад, якщо команда противника має набір героїв: [Nature's Prophet, Razor, Storm Spirit, Winter Wyvern, Pudge], то ми генеруємо декілька випадкових наборів для нашої команди:

– особина 1: [Treant Protector, Dazzle, Abaddon, Omniknight, Warlock];

– особина 2: [Juggernaut, Shadow Shaman, Pugna, Earthshaker, Lich];

– особина 3: [Anti-Mage, Rubick, Invoker, Tidehunter, Witch Doctor].

Наступним етапом ми обчислюємо для кожної особини його “якість”. Це залежить від того, наскільки добре герої взаємодіють один з одним (синергія) та наскільки ефективно вони протидіють героям із протилежної команди.

Перед схрещуванням нам потрібно обрати двох батьків. Вони обираються з популяції, як найкращі особини, тобто мають найбільше значення нашої цільової функції. Наприклад, якщо найкращими виявилися Особина 1 і Особина 3, то вони стануть батьками.

Під час схрещування створюються нові особини, шляхом об’єднання генів(героїв) із двох батьків. У нас використовується одно точковий кросинговер, для пояснення наведемо такий приклад. Маємо таких батьків:

– батько 1: [Treant Protector, Dazzle, Abaddon, Omniknight, Warlock];

– батько 2: [Anti-Mage, Rubick, Invoker, Tidehunter, Witch Doctor].

Генерація нащадків буде відбуватися таким чином, Нащадок 1 бере перших трьох героїв від Батька 1 і двох останніх від Батька 2, а Нащадок 2 бере перших трьох героїв від Батька 2 і двох останніх від Батька 1. Таким чином виходять два нащадка:

– нащадок 1: [Treant Protector, Dazzle, Abaddon, Tidehunter, Witch Doctor];

– нащадок 2: [Anti-Mage, Rubick, Invoker, Omniknight, Warlock].

Після схрещування відбувається процес мутації, тобто за певною ймовірністю(в нашому випадку 20%), змінюється ген(герой) у сгенерованого нащадка для забезпечення різноманітності. Наприклад, у Нащадка 1 замість героя Abaddon з'являється герой-танк, наприклад, Lina. Нащадок 1: [Treant Protector, Dazzle, Lina, Tidehunter, Witch Doctor].

Нове покоління створюється на основі старого та нових нащадків. Кількість особин в популяції у нас завжди однакове, тому ми обираємо найгірших особин серед старого покоління та нових нащадків. Наприклад Особина 2 із старого покоління та Нащадок 1 виявились найгіршими, тому нове покоління буде виглядати так:

– особина 1: [Treant Protector, Dazzle, Abaddon, Omniknight, Warlock];

– особина 2: [Anti-Mage, Rubick, Invoker, Tidehunter, Witch Doctor];

– особина 3 (Нащадок 2): [Anti-Mage, Rubick, Invoker, Omniknight, Warlock].

Усі ці етапи повторюються до тих пір поки не буде досягнуто заданої кількості поколінь або поки якість рішень не перестане покращуватись.

Результати, представлені у таблиці 6.1, демонструють поступове зростання ймовірності перемоги команди зі збільшенням кількості поколінь у генетичному алгоритмі. На перший погляд, може здатися, що приріст у кілька відсотків не є значним. Проте варто враховувати, що в контексті кіберпортивної дисципліни Dota 2, де результат матчу залежить від багатьох факторів, навіть невелике покращення може мати вирішальне значення. Також варто розуміти, що чим більша кількість генерацій, тим довше працює сам алгоритм.

Таблиця 6.1 – Результати виконання генетичного алгоритму

Кількість поколінь	Значення ймовірності перемоги
10	51.01%
100	52.38%
1000	54.31%
5000	55.11%

Генетичний алгоритм враховує всі аспекти (синергія між героями в команді, протидія з героями суперника) через цільову функцію, яка оцінює синергію та ефективність проти супротивника. Оскільки ці фактори взаємодіють між собою складним чином, досягнення значного приросту ймовірності перемоги (наприклад, на 10% і більше) є дуже складним завданням. Проте навіть приріст у 1-2% може бути критичним для професійних команд або в умовах високого рівня гри.

Таким чином, результати підтверджують ефективність генетичного алгоритму для задачі оптимізації набору героїв у Dota 2. Він дозволяє врахувати складну взаємодію багатьох факторів і знайти набір героїв, який забезпечує максимальну можливу перевагу в конкретних умовах.

Висновок до розділу 6

У цьому розділі розроблено і детально розглянуто систему для оптимізації набору героїв у кіберспортивній дисципліні Dota 2 з використанням генетичного алгоритму. Задача полягала в створенні інструменту, що дозволяє підвищити ймовірність перемоги команди шляхом надання рекомендацій щодо вибору героїв.

Для досягнення поставленої мети використано генетичний алгоритм, що імітує процес природного відбору. Алгоритм генерує популяцію можливих наборів героїв, оцінює їхню пристосованість на основі даних про ймовірності перемоги героїв разом або один проти одного, а потім покращує популяцію за допомогою операторів схрещування та мутації.

Результати роботи показали, що генетичний алгоритм здатний знаходити набір героїв, які мають вищу ймовірність перемоги порівняно з вибором гравців під час гри. Зокрема, було досягнуто збільшення фітнес-функції, що свідчить про покращення результату. Збільшення відбувається на пару відсотків, тому що багато факторів взаємодіють між собою складним чином і досягти вибору героїв з ймовірністю перемоги 65% або більше просто неможливо, але навіть 2 або 3 відсотка може суттєво вплинути на результат гри. Це дозволяє зробити висновок, що розроблена система є перспективним інструментом для підтримки прийняття рішень при виборі героїв у кіберспортивній дисципліні Dota 2 та має потенціал бути корисним інструментом для усіх користувачів, а саме: новачків, що тільки поринають у цю дисципліну та досвідчених спортсменів та тренерів, що завжди прагнуть покращуватись. Крім того, цей функціонал дає змогу набути нові ідеї для певних тактичних рішень та може легко слугувати, як навальний ресурс для шкіл, академій, або команд.

7 ТЕСТУВАННЯ СИСТЕМИ

7.1 Мета випробувань

Мета випробувань полягає у перевірці програмного забезпечення функціональним та нефункціональним вимогам, які були докладно описані та сформовані у минулих розділах. Основні завдання цього тестування включають в себе перевірку роботи різних підсистем, а саме: підсистема авторизації, генерації звітів, збереження матчів та написання нотатків, оптимізації вибору героїв, відповідність даних облікового запису та останніх матчів.

7.2 Загальні положення

Тестування проводилось шляхом перевірки всіх важливих функцій розробленої системи. При перевірці було перевірено працездатність усіх модулів: серверної частини, що написана за допомогою Node.js, клієнтської частини, що написана за допомогою React js та реляційної системи управління базами даних PostgreSQL. Взаємодія між сервером та API виконувалась за допомогою мови запитів GraphQL. Під час тестування здійснювалася не лише перевірка окремих модулів програмного забезпечення, а і їх взаємодію один з одним, для виявлення можливих конфліктів. Крім того, було проаналізовано зручність використання інтерфейсу, його адаптацію для інших пристроїв та різних браузерів, а також швидкодію системи.

7.3 Результати випробувань

У таблиці 7.1 наведено результати тестування оптимізації вибору героїв за допомогою генетичного алгоритму. Також на рисунку 7.1 представлено вигляд результату для користувачів. Для зручності наведена ймовірність перемоги до оптимізації та після.

Таблиця 7.1 – Тестування оптимізації підбору героїв

Назва	Оптимізація героїв
Передумови	Завантажені дані героїв
Вхідні дані	Список героїв противника
Послідовність дій	Натиснути кнопку для запуску оптимізації
Очікуваний результат	Отриманий список героїв для кожної команди з підвищеної ймовірністю перемоги

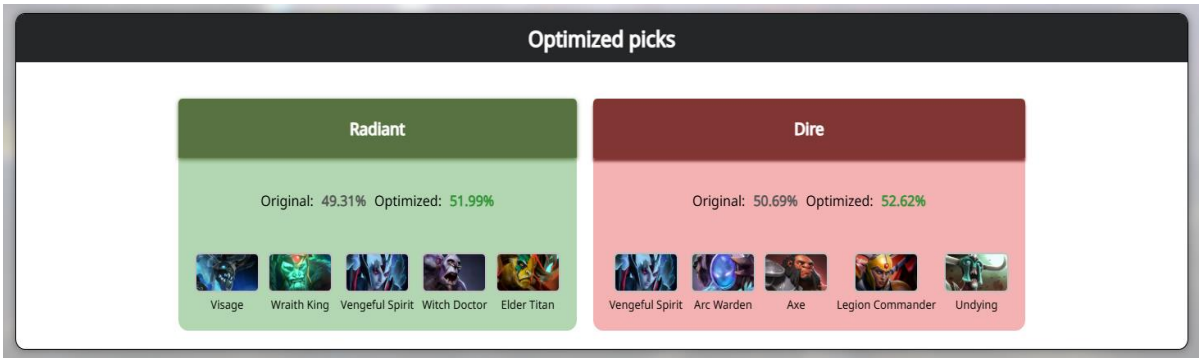

Рисунок 7.1 – Демонстрація результату оптимізації

У таблиці 7.2 наведено результати тестування оптимізації вибору героїв за допомогою генетичного алгоритму. Також на рисунках 7.2 та 7.3 представлено реєстрацію для користувачів та відображення облікового запису користувача в додатку.

Таблиця 7.2 – Тестування реєстрації користувача

Назва	Авторизація користувача
Передумови	Наявність акаунту Steam
Вхідні дані	Дані користувача
Послідовність дій	Натиснути на кнопку реєстрації
Очікуваний результат	Успішна авторизація, коректне відображення профілю та відповідних даних

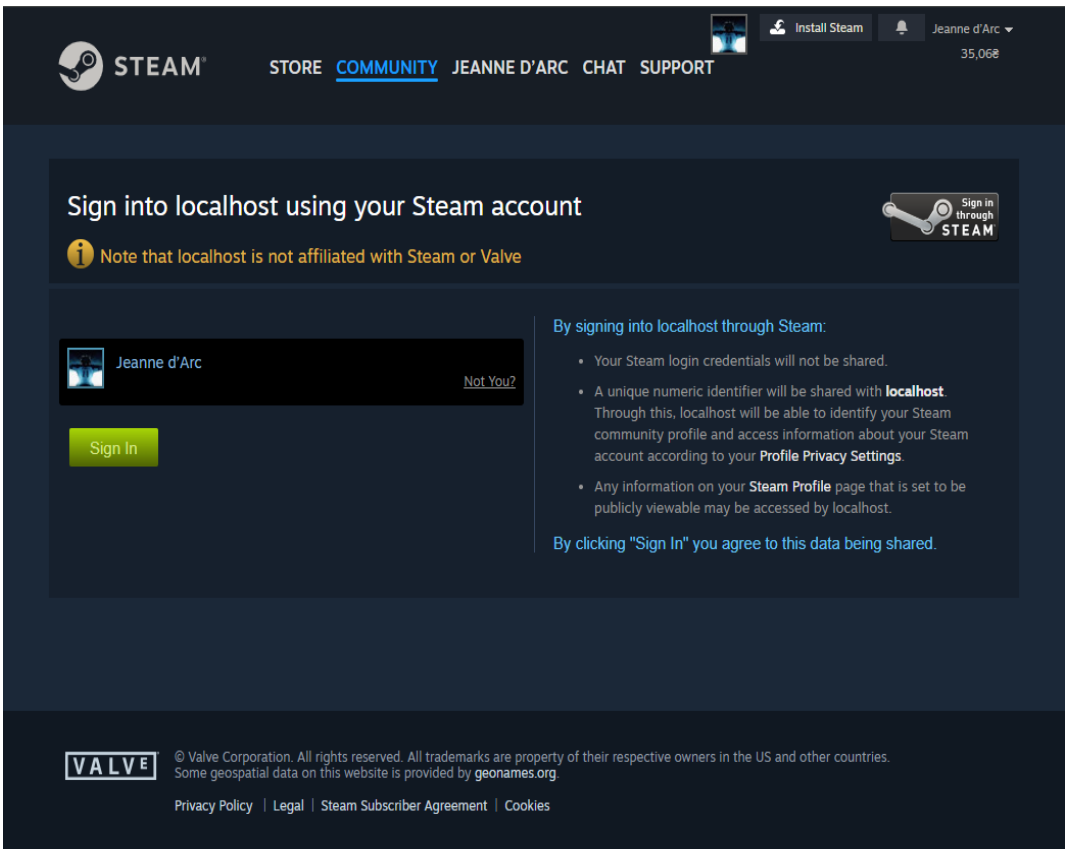

Рисунок 7.2 – Інтерфейс реєстрації

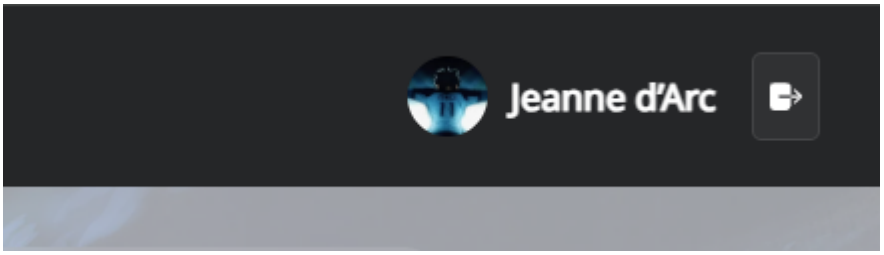

Рисунок 7.3 – Демонстрація успішної реєстрації

У таблиці 7.3 наведено результати тестування формування та генерації PDF протоколів післяматчевої інформації. Також на рисунку 7.4 представлено загальний вигляд такого протоколу, який можна завантажити в будь-який момент.

Таблиця 7.3 – Тестування генерації PDF протоколу

Назва	Генерація PDF протоколів
Передумови	Завершено аналіз матчу
Вхідні дані	Дані аналізу, статистики та оптимізації

Продовження таблиці 7.3

Послідовність дій	Натискання кнопки для генерації протоколу
Очікуваний результат	Згенерований PDF звіт, що містить усі матчі відповідного матчу і не має помилок

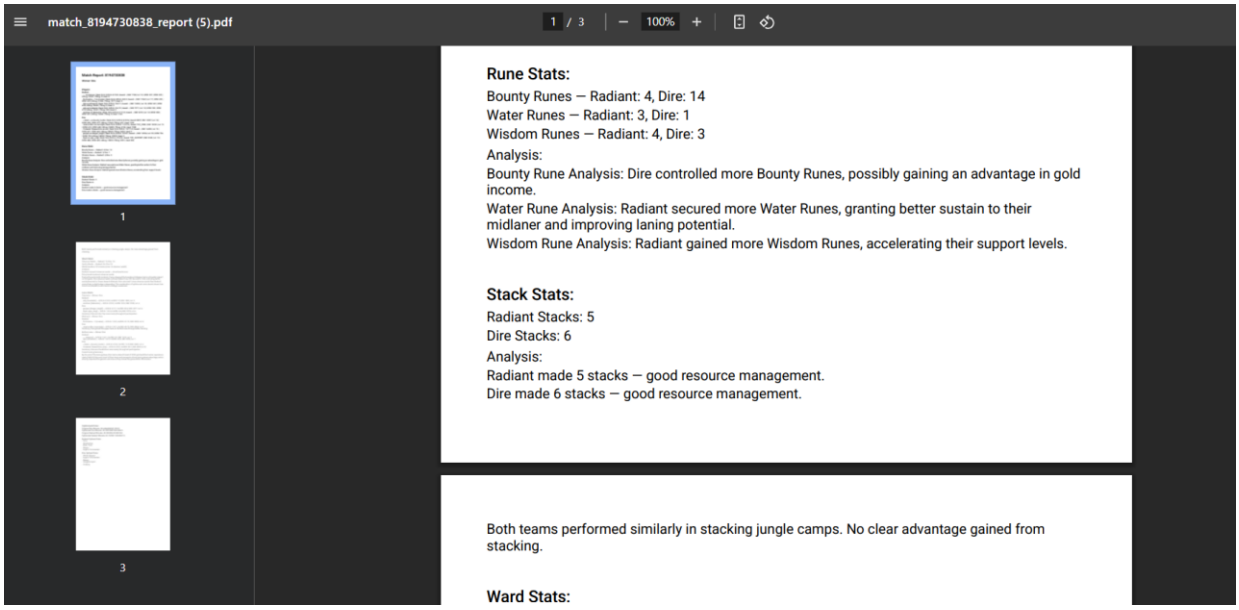

Рисунок 7.4 – Демонстрація інтерфейсу реєстрації

У таблиці 7.4 наведено результати тестування перегляду героїв з найвищим відсотком перемоги. Таких героїв можна назвати найсильнішими в певному періоді часу, така інформація оновлюється кожний день. Також на рисунку 7.5 представлено загальний вигляд блоку, де користувач може передивитися таких героїв.

Таблиця 7.4 – Тестування генерації PDF протоколу

Назва	Перевірка функції відображення героїв мети
Передумови	Працюючий інтерфейс і доступ до API
Вхідні дані	Запит на перегляд мети
Послідовність дій	Відкрити головну сторінку
Очікуваний результат	Список героїв з відсотком та кількістю матчів

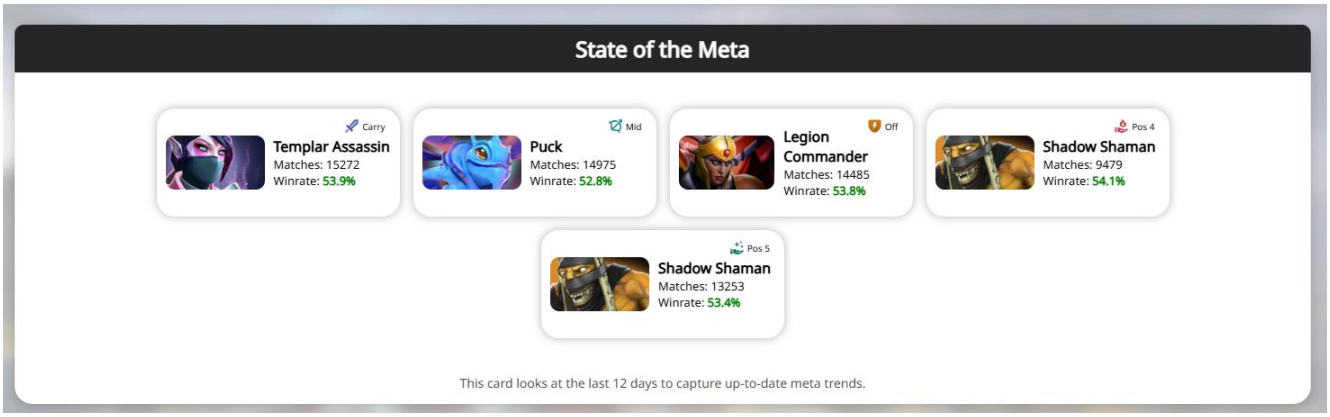

Рисунок 7.5 – Демонстрація мети героїв

У таблиці 7.5 наведено результати тестування функції пошуку матчу за допомогою використання його id. Також на рисунках 7.6 і 7.7 наведено вигляд пошуку та яку сторінку отримує користувач після успішного пошуку.

Таблиця 7.5 – Тестування пошуку матчу по ID

Назва	Пошук матчу за введеним ID
Передумови	Відомий ID матчу
Вхідні дані	Введений Match ID
Послідовність дій	Ввести ID у відповідней поле та натиснути Enter
Очікуваний результат	Перехід на сторінку матчу та відображення даних

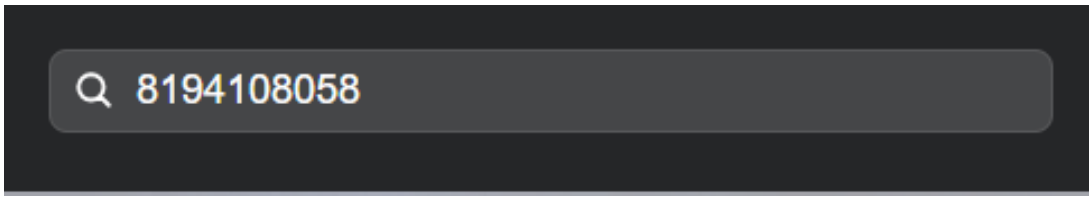

Рисунок 7.6 – Поле для пошуку

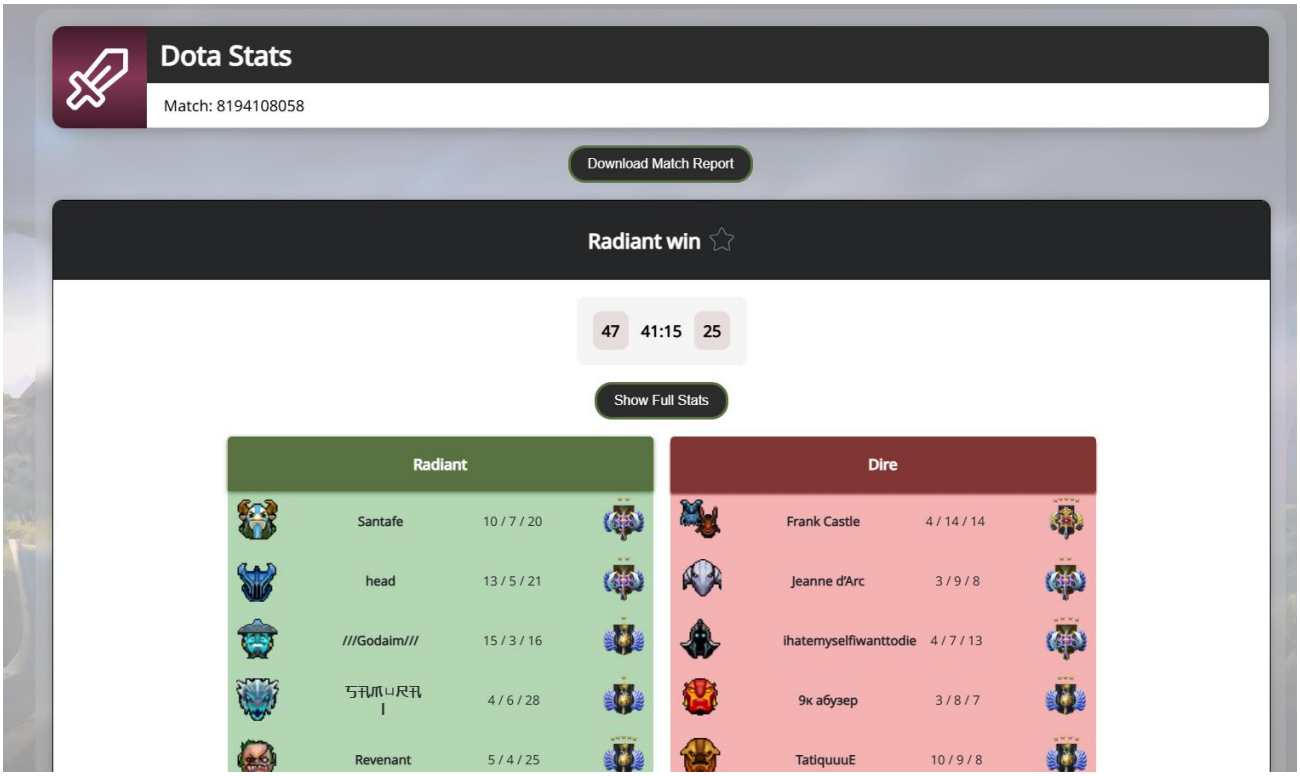

Рисунок 7.7 – Інтерфейс обраного матчу

У таблиці 7.6 наведено результати тестування додавання нотаток до матчу. Також на рисунках 7.8 представлено, як користувач буде бачити нотатки.

Таблиця 7.6 – Тестування додавання нотаток до матчу

Назва	Додавання нотаток до матчу
Передумови	Користувач авторизований, на сторінці профілю
Вхідні дані	Текст нотатки
Послідовність дій	Ввести текст у поле, втратити фокус
Очікуваний результат	Нотатка зберігається і відображається, навіть після перезупуску сторінок.

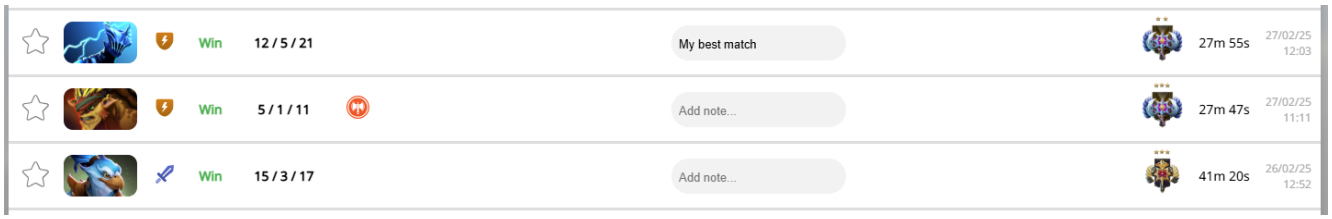

Рисунок 7.8 – Вигляд написаних нотаток

У таблиці 7.7 наведено результати тестування функції додавання матчу до улюблених. Також на рисунках 7.9 та 7.10 наведено відображення того, що матч доданий та функція перегляду тільки улюблених матчів в профілі гравця.

Таблиця 7.7 – Тестування пошуку матчу по ID

Назва	Додавання матчу до обраного
Передумови	Користувач авторизований, переглядає матч або на сторінці профілю
Вхідні дані	Булеві дані(доданий матч або ні)
Послідовність дій	Натиснути на відповідну кнопку
Очікуваний результат	Матч збережено в списку обраного у профілі

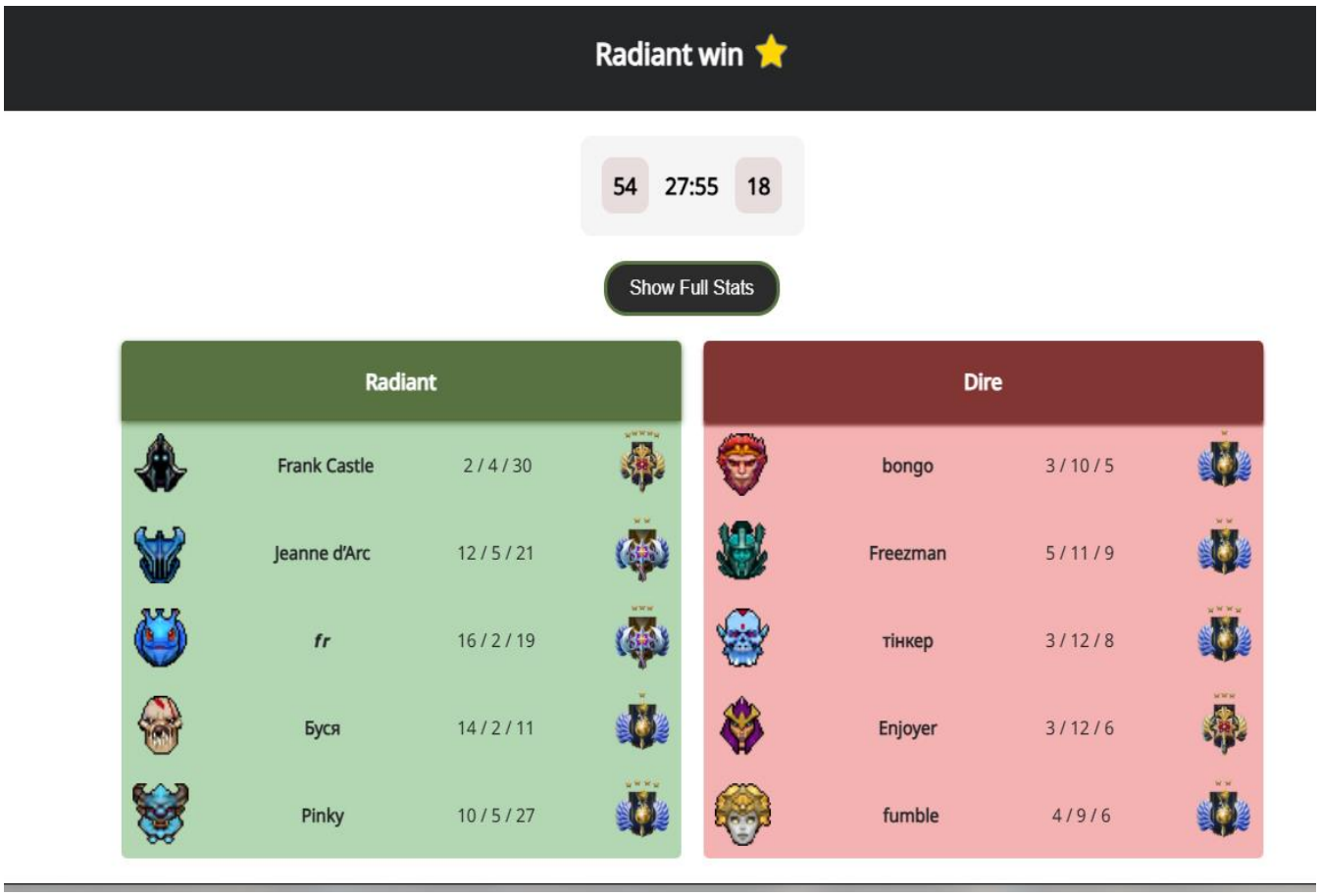

Рисунок 7.9 – Відображення доданого матчу

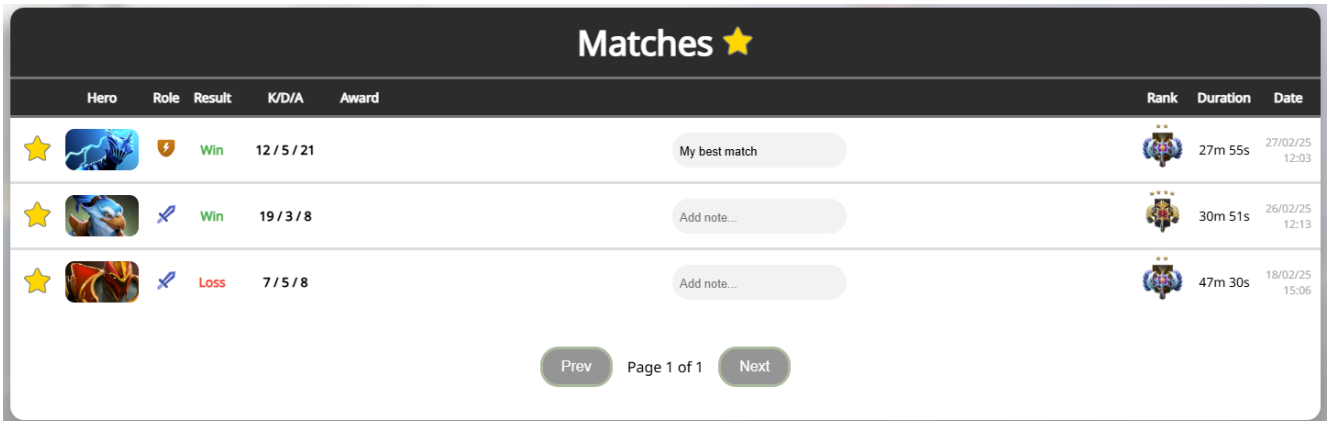

Рисунок 7.10 – Перегляд улюблених матчів

На рисунках 7.11 та 7.12 представлено інтерфейс відкритої сторінки користувача та сторінки користувача з анонімним доступом відповідно. У таблиці 7.8 наведено результати тестування перегляду профілю користувача.

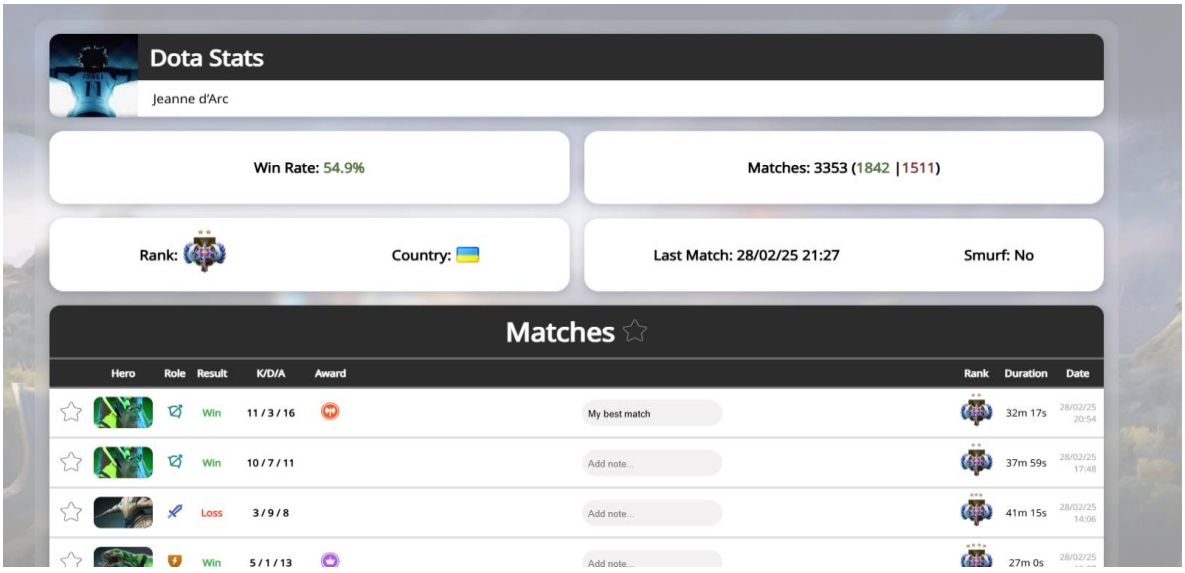

Рисунок 7.11 – Інтерфейс відкритого профілю користувача

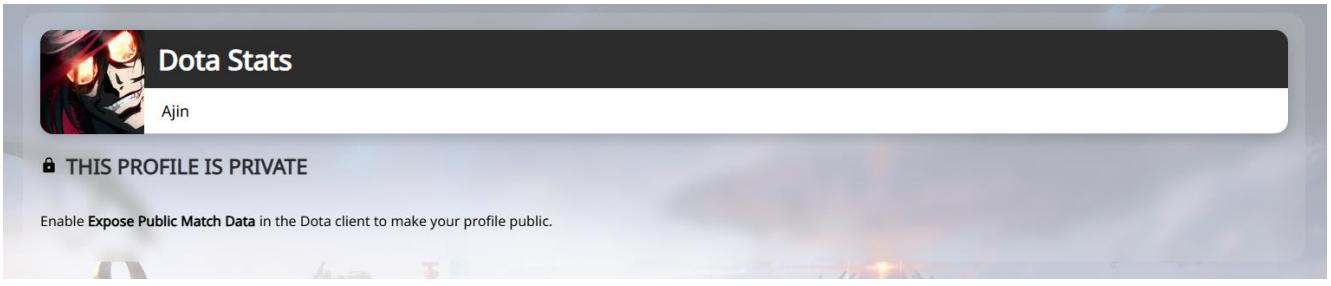

Рисунок 7.12 – Інтерфейс анонімного профіля користувача

Таблиця 7.8 – Тестування перегляду профілю

Назва	Перевірка відображення профілю користувача
Передумови	Користувач має відкритий профіль
Вхідні дані	Запит на перегляд профілю
Послідовність дій	Перейти за посиланням до профілю або натиснути на аватар
Очікуваний результат	Відображається профіль користувача з останніми матчами та статистикою
Фактичний результат	Профіль відкривається коректно; якщо користувач зробив профіль анонімним — не відображається

Висновок до розділу 7

У цьому розділі проведено тестування розробленої системи відповідно до встановлених в попередніх пунктах функціональних вимог. Перевірено коректність роботи авторизації, швидкість та ефективність функції оптимізації вибору героїв, генерування PDF-протоколів, пошуку матчу по його ID, перегляду мети героїв, додавання нотаток і додавання матчів до улюблених, після перезапуску сторінок, всі ці дані відображаються також і не зникають, за допомогою використання бази даних. Усі перевірені функції працюють згідно очікувань, що підтверджує відповідність реалізації технічного завдання та якість створеного додатку. Також для демонстрації зручності та сучасності інтерфейсу, були наведены рисунки певних блоків з програмного забезпечення. Програмний вебзастосунок запускається на різних браузерах та підтримується адаптивність для різних пристроїв.

ВИСНОВКИ

У ході виконання дипломного проєкту розроблена та спроєктована інформаційна система аналізу ігрових даних та матчевої статистики Dota 2. Система дає можливість користувачам постійно покращувати свої вміння, бачити аналіз матчів і та також за допомогою алгоритмів отримувати рекомендації щодо вибору героїв.

На початку дипломного проєктування проведено детальний аналіз предметної області та вже існуючих рішень на ринку. Цей аналіз дає розуміння, що є найголовнішим для користувача та що вже пропонують існуючі додатки. Знайдені переваги та недоліки формують висновок, що жоден із проєктів не поєднує в собі усього потрібного функціоналу в одному додатку.

У розділі «Формування вимог до системи» наведено більш комплексне та детальне формування вимог до інформаційної системи таких як: функціональні характеристики надійності та зберігання інформації в таблицях бази даних. Також чітко розділено системи на певні підсистеми, кожна з яких відповідає за свої функції. Таким чином визначено, що потрібно проводити аналітику матчів, використати генетичний алгоритм для оптимізації виборів героїв із найвищим відсотком перемоги, авторизація користувача та створення протоколів. І головним було також розробити зручний користувацький інтерфейс, який би не заважав, а навпаки мотивував користуватися цим додатком.

У розділі «Вибір технологій розробки» для реалізації вже створених вимог проведено аналіз сучасних технологій, для того щоб знайти найкращі для розробки системи. В результаті пошуків, обрано односторінкову архітектуру SPA з використанням мови програмування JavaScript. За допомогою цієї мови підібрана бібліотека React.js для реалізації фронтенд частини та Node.js для бекенд частини. Також обрана мова написання запитів GraphQL, для того щоб зручно отримувати саме ті дані, які були потрібно і ніяких зайвих. Отримані дані вирішено зберігати в системі управління бази даних PostgreSQL, завдяки високої надійності, гнучкості та здатністю обробляти складні запити. Ця система була також вже перевірена

часом, тому не залишилось ніяких сумнівів в її використанні. Для реєстрування користувачів проведено інтеграцію з платформою Steam та її технологіє OpenId, що забезпечила простоту автентифікації та безпеку і впевненість для користувачів. Це дало можливість легко отримувати відкриті дані.

Для математичного забезпечення інформаційної системи реалізовано генетичний алгоритм, що дає можливість ефективно знаходити оптимальні рішення для набору героїв, при умовах великої кількості даних та обмежень. Цей алгоритм у результаті враховує синергію між героями та протидію проти конкретних суперників, що також дало можливість реалізувати потрібну функцію в системі. На практиці було протестовано цей алгоритм і результати показали збільшення ймовірності перемоги, якщо порівнювати з оригінальною ймовірністю. Це робить цю розробку дуже ціним інструментом для усіх охочих гравців.

Отже, розроблена інформаційна система є ефективним рішенням, яке надає можливість користувачу отримувати корисні аналітичні поради, покращувати свої навички та вчитися на помилках. Така система має високий потенціал до масштабування за допомогою додавання ще більше аналітичних блоків або ж інтегрування інших кіберспортивних дисциплін, що надає перспективи для подальшого розвитку цієї системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Medium. DotA 2 – the game which defined & broke the rules of E-sports. URL: https://niharpachpande.medium.com/dota-2-the-game-which-defined-broke-the-rules-of-e-sports-8c8812c597b4 (дата звернення: 03.04.2025).

2. OpenDota. Open source Dota 2 data platform. URL: https://www.opendota.com/ (дата звернення: 10.04.2025).

3. Dotabuff. DOTABUFF - Dota 2 Statistics. URL: https://www.dotabuff.com/ (дата звернення: 04.04.2025).

4. Stratz. STRATZ - Dota 2 Stats. URL: https://stratz.com/ (дата звернення: 04.04.2025).

5. Overwolf. The guild of in-game creators. URL: https://www.overwolf.com/ (дата звернення: 03.04.2025).

6. Javascript Info. Сучасний підручник з JavaScript. URL: https://uk.javascript.info/ (дата звернення: 14.04.2025).

7. Node.js. Introduction to Node.js. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 03.05.2025).

8. React. React documentation for quick start. URL: https://react.dev/learn (дата звернення: 30.04.2025).

9. Blog Postman. GraphQL vs. REST. URL: https://blog.postman.com/graphql-vs-rest/ (дата звернення: 17.04.2025).

10. Neon. PostgreSQL Tutorial. URL: https://neon.com/postgresql/tutorial (дата звернення: 06.05.2025).

11. Pdfmake. PDF document generation library for server-side and client-side usage in pure JavaScript.. URL: https://pdfmake.github.io/docs/0.1/ (дата звернення: 16.05.2025).

12. Steam. Steam Web API Documentation. URL: https://steamcommunity.com/dev (дата звернення: 10.05.2025).

13. Stratz. What is the STRATZ API?. URL: https://stratz.com/api (дата звернення: 01.04.2025).

14. Bezkoder. React + Node.js + Express + PostgreSQL example: Build a CRUD App. URL: https://www.bezkoder.com/react-node-express-postgresql/ (дата звернення: 14.04.2025).

15. Habr. Генетический алгоритм. URL: https://habr.com/articles/861334/ (дата звернення: 11.04.2025).