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: «Рекомендаційна система вибору стратегій у Dota 2»

Виконав:
студент IV курсу, групи ІС-13
Малярчук Тарас Васильович _____

Керівник:
доцент, к.т.н., доцент
Жураковська Оксана Сергіївна

_____

Рецензент:
Доцент кафедри ОТ, к.т.н., доцент
Марковський Олександр Петрович _____

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
Малярчуку Тарасу Васильовичу

1. Тема проєкту «Рекомендаційна система вибору стратегій у Dota 2», керівник проєкту Жураковська Оксана Сергіївна, доцент, к.т.н., доцент, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

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

3. Вихідні дані до проєкту: набір даних OpenDota API, мови програмування Python, Typescript, бібліотеки React, numpy, torch, sklearn, середовища розробки WebStorm та PyCharm, numpy, база даних MongoDB

4. Зміст пояснювальної записки: опис процесу діяльності, огляд існуючих рішень, формування вимог до системи, вибір технологій розробки, розробка інформаційної системи, математичне забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): дві діаграми діяльності, діаграма потоку даних, діаграма класів

6. Дата видачі завдання: «5» березня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Опис процесу діяльності	16.04.2025	
2	Огляд існуючих рішень	21.04.2025	
3	Формування вимог до системи	25.04.2025	
4	Вибір технологій розробки	28.04.2025	
5	Розробка інформаційної системи	10.05.2025	
6	Математичне забезпечення	20.05.2025	
7	Подання ДП на попередній захист	25.05.2025	
8	Подання ДП на основний захист	09.06.2025	

Студент Тарас МАЛЯРЧУК

Керівник Оксана ЖУРАКОВСЬКА

АНОТАЦІЯ

Рекомендаційна система вибору стратегій у Dota 2.

Проєкт містить 65 с. тексту, 15 рисунків, 6 таблиць, посилання на 15 літературних джерел, 1 додаток та 4 конструкторські документи.

РЕКОМЕНДАЦІЙНА СИСТЕМА, ІНТЕЛЕКТУАЛЬНИЙ ІНТЕРФЕЙС, СТРАТЕГІЧНА ГРА, ПІДТРИМКА РІШЕНЬ, СИСТЕМА ДОПОМОГИ КОРИСТУВАЧУ.

Об’єктом розробки є процес вибору стратегії у грі Dota 2.

Мета розробки – покращення процесу прийняття рішень користувачем у грі Dota 2 на етапі вибору героїв та стратегії гри.

У дипломному проєкті розглянуто проблему прийняття рішень в умовах динамічного віртуального середовища стратегічної командної гри. Запропоновано концепцію інтелектуальної інформаційної системи, яка поєднує в собі рекомендації щодо вибору ігрових персонажів із визначенням відповідного стилю гри для кожного учасника.

У роботі проведено аналіз існуючих систем підтримки рішень, побудовано функціональну системи, розроблено математичне забезпечення та описано архітектуру програмного забезпечення. Особливу увагу приділено проєктуванню користувацького інтерфейсу, який забезпечує інтуїтивну взаємодію користувача із системою.

Результати дослідження можуть бути використані в ігрових рекомендаційних сервісах, освітніх тренажерах, а також у програмному забезпеченні, орієнтованому на вдосконалення прийняття рішень у середовищах з високою динамікою змін.

SUMMARY

Recommendation system for strategy selection in Dota 2.

The project consists of 65 pages of text, 15 figures, 6 tables, links to 15 literary sources, an appendix and 4 design documents.

Keywords: recommendation system, intelligent interface, strategic game, decision support, user assistance system.

The object of this development is a recommendation system for strategy selection in the game Dota 2.

The aim of the development is to improve the user's decision-making process in Dota 2 during the hero selection phase.

This thesis addresses the issue of decision-making in the context of a dynamic virtual environment of a strategic team-based game. A concept of an intelligent information system is proposed, which combines recommendations for hero selection with determining the appropriate playstyle for each team member.

The work includes an analysis of existing decision support systems, the construction of a functional model of the system, the development of mathematical support, and a description of the software architecture. Particular attention is paid to the design of the user interface, which enables intuitive interaction between the user and the system.

The results of this research can be applied in gaming recommendation services, educational simulators, and software aimed at improving decision-making in environments with high dynamics and variability.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС13.090БАК.005 ПЗ	Рекомендаційна система вибору	65		
6			стратегії у Dota 2. Пояснювальна			
7			записка			
8						
9	А3	ІС13.090БАК.005 Д1	Рекомендаційна система вибору	1		
10			стратегії у Dota 2. Діаграма			
11			діяльності. Робота системи в			
12			режимі відмови			
13						
14	А3	ІС13.090БАК.005 Д2	Рекомендаційна система вибору	1		
15			стратегії у Dota 2. Діаграма			
16			діяльності. Аналіз драфту			
17						
18	А3	ІС13.090БАК.005 Д3	Рекомендаційна система вибору	1		
19			стратегії у Dota 2. Діаграма			
20			потоку даних			
21						
22	А3	ІС13.090БАК.005 Д4	Рекомендаційна система вибору	1		
23			стратегії у Dota 2. Діаграма класів			
24						
25						
26						
27						
28						

Зм.	Лист	№ докум.	Підпис		ІС13.090БАК.005 ТП		
Розробив		Малярчук			Рекомендаційна система вибору стратегій у Dota 2. Відомість дипломного проєкту	Літ.	Арк. 1 / Аркушів 1
Перевірив		Жураковська				Т	
						КПІ ім. Ігоря Сікорського Група ІС-13	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Рекомендаційна система вибору стратегій у Dota 2»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 6
1 ОПИС ПРОЦЕСУ ДІЯЛЬНОСТІ 8
1.1 Опис процесу діяльності 8
1.1.1 Роль кіберспорту та Dota 2 у світі 8
1.1.2 Опис механіки гри та структури матчу 8
1.2 Постановка задачі 10
1.2.1 Призначення системи 10
1.2.2 Цілі та задачі розробки 11
Висновок до розділу 1 12
2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ 13
2.1 Опис Dota Plus 13
2.2. Обмеження Dota Plus 13
2.3 Обгрунтування доцільності власної розробки 14
Висновок до розділу 2 15
3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ 16
3.1 Вимоги до системи в цілому 16
3.1.1 Підсистема збору та обробки даних 16
3.1.2 Аналітична підсистема 16
3.1.3 Серверна частина 17
3.1.4 Клієнтська частина 17
3.2 Вимоги до функціональних характеристик 18
3.3 Вимоги до видів забезпечення 19
3.3.1 Математичне забезпечення 19
3.3.2 Інформаційне забезпечення 20
3.3.3 Програмне забезпечення 20

3.3.4 Технічне забезпечення 21
Висновок до розділу 3 21
4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ 22
Висновок до розділу 4 25
5 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 26
5.1 Структура системи 26
5.1.1 Клієнтський рівень 26
5.1.2 Серверний рівень 27
5.1.3 Аналітична підсистема 27
5.1.4 Рівень зберігання даних 28
5.1.5 Інфраструктура 28
5.2 Функціональна модель системи 29
5.2.1 Основні актори 29
5.2.2 Сценарії використання 30
5.2.2.1 Отримання рекомендацій під час драфту 30
5.2.2.2 Робота системи у режимі відмови 31
5.3 Модель бази даних 31
5.3.1 Колекція heroes 32
5.3.2 Колекція matches 34
5.3.3 Колекція matches_info 34
5.3.4 Об’єкт Player 35
5.4 Передавання та обробка даних 37
5.4.1 Загальна схема передачі даних 37
5.5 Архітектура програмного забезпечення 38
5.5.1 Доменний шар 38
5.5.2 Прикладний шар 39
5.5.3 Інфраструктурний шар 39
5.5.4 Шар представлення 40
5.5.5 Специфікація функцій 41
5.6 Розробка користувацького інтерфейсу 42

5.7 Інструкція користувача 51
Висновок до розділу 5 55
6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 56
6.1 Змістовна постановка задачі 56
6.2 Математична постановка задачі 56
6.2.1 Рекомендація вибору героїв 56
6.2.2 Рекомендація стилю гри 57
6.3 Обгрунтування методу розв’язання 58
6.3.1 Модель рекомендації вибору героїв 58
6.3.2 Число нейронів у прихованих шарах 59
6.3.3 Модель рекомендації стилю гри 60
6.4 Опис методу розв’язання 60
6.4.1 Опис методу розв’язання для моделі рекомендації героїв 60
6.4.2 Опис методу розв'язання для моделі рекомендації стилю гри 62
Висновок до розділу 6 62
ВИСНОВКИ 63
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Драфт – процес поетапного вибору героїв перед початком матчу у грі Dota 2.

Герой – ігрових персонаж, якого обирає гравець для участі в матчі.

Кріп – нейтральний або ворожий неігровий персонаж, який автоматично з’являється на лініях або в лісі з певним інтервалом часу. Використовується для отримання золота, досвіду або стратегічної переваги.

Вебклієнт (клієнт) – користувацький інтерфейс.

API (Application Programming Interface) – інтерфейс прикладного програмування, набір визначених правил для взаємодії між програмними компонентами.

KDA – показник ефективності гравця: співвідношення вбивств, вибувань з гри та допомоги.

GPM (Gold Per Minute) – показник кількості золота, яке гравець заробляє за хвилину гри.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту, що використовується для взаємодії вебклієнтів і серверів.

OpenDota API – відкрите інтерфейсне рішення для доступу до статистики матчів Dota 2 .

REST API (Representational State Transfer API) – архітектурний стиль створення вебінтерфейсів, що базується на протоколі HTTP.

ВСТУП

У сучасному цифровому світі дедалі більшої ваги набувають інтелектуальні інформаційні системи, які здатні підтримувати користувача під час прийняття рішень у складних умовах. Це стосується не лише критично важливих галузей, таких як охорона здоров’я, транспорт чи безпека, а й динамічних віртуальних середовищ, зокрема стратегічних комп’ютерних ігор. В умовах великої кількості варіантів дій, невизначеності щодо поведінки опонента та обмеженого часу на аналіз, користувачу стає дедалі важче приймати найкращі за наявних умов рішення без зовнішньої підтримки.

Особливого значення набувають системи, які здатні аналізувати ситуацію у реальному часі та пропонувати рекомендації. У багатокористувацьких іграх, де координація дій і правильний підбір ігрових персонажів мають суттєвий вплив на кінцевий результат, з’являється потреба в допоміжному інструменті, який би дозволяв гравцю ухвалювати стратегічно виважені рішення швидше й ефективніше.

Об’єктом дослідження в даній роботі є процес прийняття рішень у командних стратегічних іграх. Його складність зумовлена як великою кількістю змінних, що впливають на ситуацію, так і необхідністю хорошої командної взаємодії.

Предметом дослідження є інтелектуальна система підтримки рішень, яка дозволяє формувати рекомендації під час етапу вибору персонажів в ігровому процесі та визначати відповідний стиль гри для кожного з учасників команди.

Актуальність дослідження зумовлена тим, що в умовах зростаючої складності сучасних командних ігор виникає потреба у більш гнучких інструментах, здатних не лише пропонувати вибір персонажів, а й надавати підказки щодо їх подальшого застосування в ігровому процесі. Це обмежує ефективність таких рекомендацій і створює передумови для помилкових рішень, особливо у гравців без значного досвіду. Відтак, розробка системи, яка поєднує рекомендацію героїв із поясненням відповідного стилю гри, може суттєво

підвищити усвідомленість дій користувача й зробити процес формування команди більш прозорим і адаптивним.

Метою розробки є покращення процесу прийняття рішень користувачем під час формування команди за допомогою створення інтелектуального інтерфейсу, який дозволяє швидко й зручно отримувати рекомендації щодо вибору персонажів та подальшого стилю гри. Впровадження такого інструменту має на меті зробити ігрових досвід більш усвідомленим, сприяти зниженню порогу входу для нових гравців, а також надати можливість гнучкого стратегічного планування в динамічному середовищі.

Необхідність такого дослідження також спирається на стрімкий розвиток індустрії кіберспорту та збільшення кількості користувачів, які шукають шляхи для покращення власної продуктивності. За умов жорсткої конкуренції навіть незначні переваги, зумовлені кращою підготовкою або більш ефективним прийняттям рішень, можуть мати вирішальне значення. Це відкриває широкі перспективи для застосування систем інтелектуальної підтримки не лише в аматорському, а й у професійному ігровому середовищі.

Результати дослідження можуть знайти застосування в інтерактивних платформах підвищення ігрової майстерності, освітніх симуляторах, тренуваннях професійних гравців, а також у сервісах аналітики матчів. Запропонований підхід до поєднання динамічної рекомендаційної моделі та інтерфейсу користувача з високим рівнем інформативності може бути масштабований і на інші домени, де швидкість і точність рішень мають критичне значення.

1 ОПИС ПРОЦЕСУ ДІЯЛЬНОСТІ

1.1 Опис процесу діяльності

1.1.1 Роль кіберспорту та Dota 2 у світі

Кіберспорт за два останні десятиліття перетворився із хобі на глобальну індустрію зі стійкою економічною моделлю та багатомільйонною аудиторією. За даними статистичних агрегаторів сумарний призовий фонд офіційних турнірів Dota 2 вже перевищив 340 млн USD - це вдвічі більше, ніж у найближчого конкурента CS (~ 169 млн USD) [1]. Лише один турнір The International 2021 зібрав рекордні 40 млн USD призових, що перевищує стартовий фонд першого TI на 38,4 млн USD і свідчить про стрімку динаміку зростання галузі.

Порівняння з класичним спортом демонструє співставний масштаб: призовий фонд Чемпіонату світу з футболу 2022 (FIFA World Cup) становив 440 млн USD [2], але за підтримки таких партнерів як Adidas, Coca-Cola, Visa тощо. Натомість Valve принципово обмежує комерціалізацію своїх турнірів, залишаючись при цьому у верхньому сегменті спортивних виплат.

Україна входить до авангарду цієї галузі: перший The International виграла саме українська команда Natus Vincere, а у 2017 році Київ приймав мейджор-турнір із призовим фондом 3 млн USD. Держава офіційно визнала кіберспорт видом спорту, що відкриває шлях до залучення інвестицій у суміжні сфери, наприклад, до IT-інфраструктури.

Міжнародний олімпійський комітет, реагуючи на глобальний тренд, провів у 2023 році перший Олімпійський кіберспортивний тиждень (Olympic Esports Week). Захід став важливим кроком у визнанні кіберспорту як повноцінної спортивної дисципліни [3].

1.1.2 Опис механіки гри та структури матчу

Описуючи механіку гри, зосередимось на класичному режимі “All Pick”. В цьому режимі є дві команди по 5 гравців: Dire та Radiant. Команди грають на карті,

що зображена на рисунку 1.1. Головна мета команд – знищити основну будівлю суперника (Ancient), захищеною системою веж та казарм. Команда, яка це робить, виграє.

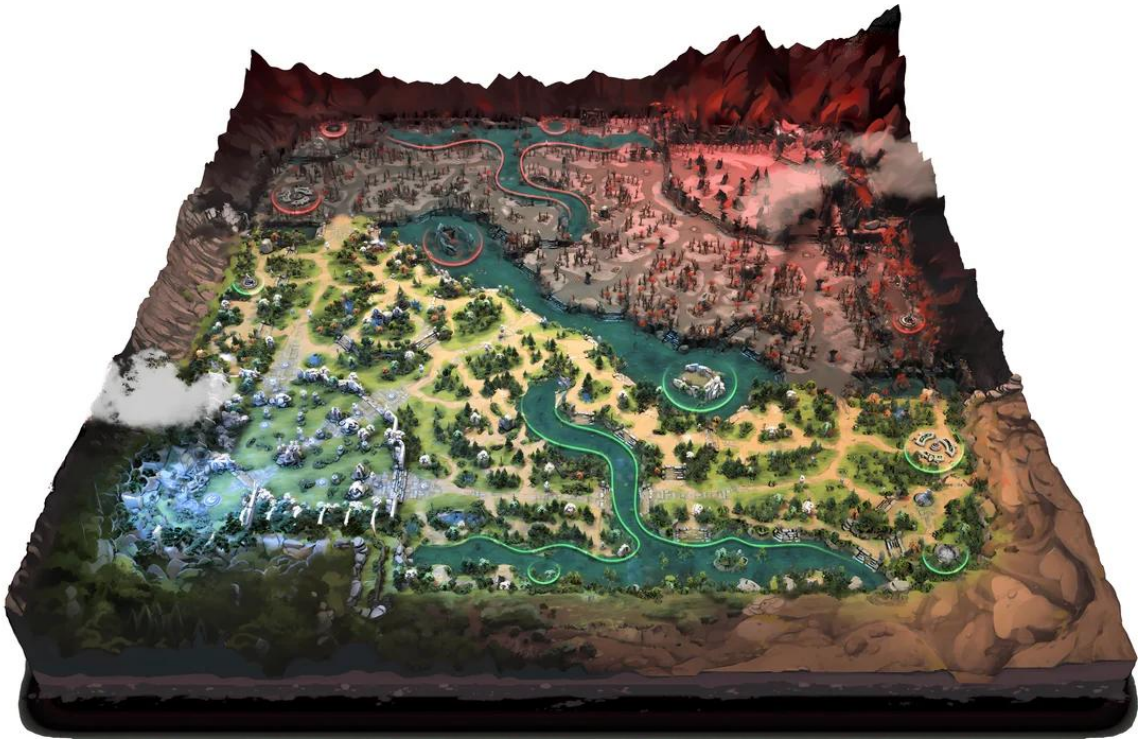

Рисунок 1.1 — Мапа Dota 2 в режимі “All pick”

Перед початком матчу проходить етап драфту, на якому команди почергово за визначеним шаблоном обирають героїв із загального пулу, де кожен з них має свої унікальні характеристики, здібності, роль у команді та стиль гри. Процес драфту поділяється на 3 послідовні фази:

– Radiant та Dire обирають по 2 герої;

– Radiant та Dire обирають по 2 герої;

– Radiant та Dire обирають по 1 герою.

Всього в доті є 126 героїв та понад 200 предметів, які гравець може придбати по ходу матчу. Ролі героїв мають свої позиції:

– carry (позиція 1) – гравець на цій ролі по ходу гри повинен здобувати перевагу в золоті та досвіді. Є особливо корисним на пізніх стадіях гри. З огляду на це, є ключовим в команді. Грає на нижній лінії;

– mid (позиція 2) – гравець на цій ролі грає на центральній лінії проти одного суперника. Роль вимагає високої індивідуальної майстерності та стратегічного мислення. Може бути корисним як на пізніх, так і на початкових стадіях гри. Здатен переломити гру в початкових та середніх фазах матчу, якщо його команда погано відіграла початкову стадію. Або ж навпаки – підсилити тиск на ворожу команду. Створює ігровий простір для команди;

– offlane (позиція 3) – гравець на цій ролі часто виступає ініціатором сутичок з командою суперника. На початкових стадіях ціль гравця не дати ворожому Carry здобувати перевагу в золоті та досвіді. Грає на верхній лінії;

– support (позиція 4) – гравець на цій ролі підтримує свого Offlane. Грає на верхній лінії. Як і Offlane може бути ініціатором командних сутичок, але його основна роль по ходу гри – допомагати 1, 2 та 3 позиціям і рятувати їх. Грає на верхній лінії або вільно пересувається по карті навіть на початковій стадії гри;

– hard Support (позиція 5) – гравець на цій ролі підтримує свого Carry. Його завданням є забезпечення хорошого старту для свого Carry, а також забезпечення команди розхідними ресурсами спрямованих на лікування, огляд карти тощо. Грає на нижній лінії.

Важливим аспектом є не лише індивідуальна сила героя, а й синергія з іншими героями в команді та ефективність проти суперників.

На етапі драфту команди не знають повного складу суперників до закінчення 3 фази. Тому драфт має характер послідовної інформації з неповною відкритістю – це ускладнює вибір і вимагає стратегічного мислення. Успішність гри значною мірою визначається саме на етапі драфту, адже невдалий вибір героїв часто призводить до значної переваги однієї з сторін ще до початку матчу.

1.2 Постановка задачі

1.2.1 Призначення системи

На етапі драфту гравці стикаються з необхідністю ухвалювати стратегічні рішення за умов обмеженої інформації: кого обрав супротивник, яку роль ще

потрібно закрити, чи добре герої в одній команді взаємодіють між собою тощо.

Система призначена для підтримки процесу прийняття стратегічних рішень під час драфту у грі Dota 2, що передбачає вибір героїв та стилю гри на основі аналізу поточної драфтової ситуації, включаючи вибір героїв обох команд та їх взаємодію.

Важливими елементами процесу прийняття рішень про стратегію гри є:

– вибір кращого за наявних умов та стадії драфту героя;

– аналіз синергії героя з уже обраними союзниками;

– аналіз ефективності героя проти складу супротивника;

– визначення рекомендованого стилю гри для кожного героя (агресивний, обережний тощо).

1.2.2 Цілі та задачі розробки

Система є веб сервіс, доступний для гравців різного рівня: як професійних гравців і тренерів, так і звичайних користувачів, які хочуть вдосконалити своє розуміння гри. Її рекомендації базуватимуться на даних із попередніх матчів і враховуватимуть драфт обох команд, внутрішню синергію героїв та ефективність проти супротивників.

Метою розробки є покращення якості прийняття рішень на етапі драфту в грі Dota 2 за рахунок надання рекомендацій щодо вибору героїв і стилю гри відповідно до конкретної ігрової ситуації.

Для досягнення цієї мети необхідно вирішити такі завдання:

– провести аналіз предметної області, зосередившись на особливостях драфту, типових помилках гравців та чинниках, які впливають на ефективність вибору героїв;

– зібрати та опрацювати дані з відкритих джерел (OpenDota API) для формування повноцінного датасету із матчами та характеристиками героїв і їх статистичних даних;

– побудувати математичну модель та розробити алгоритми формування

рекомендацій;

– розробити архітектуру системи, рекомендаційної підсистеми, серверної частини, бази даних та клієнтського інтерфейсу;

– навчити та протестувати модель, оцінивши точність і практичну корисність на реальних прикладах драфтів;

– розробити веб-інтерфейс, який дозволяє користувачам взаємодіяти із системою в зручний спосіб;

– реалізувати API для зв’язку між користувачем та рекомендаційними моделями;

– протестувати систему в різних сценаріях, оцінити її стабільність та зручність використання.

Висновок до розділу 1

У цьому розділі було розглянуто предметну область, в якій застосовуватиметься система, та описано процес діяльності, що автоматизується. Зокрема, було проаналізовано структуру матчу в Dota 2, особливості драфту та ролі героїв, які впливають на формування стратегії гри. Було визначено, що процес вибору героїв є критично важливим для результату гри та містить багато змінних, які складно оцінити без допоміжних інструментів.

Поставлено задачу розробки системи, яка надає аналітичні рекомендації щодо вибору героїв і стилю гри. Сформульовано мету створення такої системи та перелік завдань, реалізація яких дозволить досягти поставленої мети.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Перш ніж розробляти власну систему, необхідно проаналізувати вже наявні рішення, які частково або повністю реалізують подібний функціонал. Цей аналіз дозволяє зрозуміти поточний стан ринку інструментів, які надають рекомендації щодо вибору героїв та стилю гри у Dota 2, а також виявити їхні сильні сторони та обмеження.

Увагу буде приділено офіційному продукту від Valve – Dota Plus [4], який є інтегрований у безпосередньо в клієнт гри. Саме через Dota Plus багато гравців вперше взаємодіють з інструментами, які надають допомогу під час драфту.

2.1 Опис Dota Plus

Dota Plus – це офіційний розширений сервіс для гравців у Dota 2, розроблений компанією Valve. Інструмент доступний за платною підпискою.

Основні функції Dota Plus включають:

– розширену статистику для кожного гравця та героя;

– рекомендації щодо предметів під час гри;

– поради щодо талантів та вмінь;

– доступ до Plus Assistant – інструменту, який надає поради під час драфту та самої гри.

Plus Assistant автоматично аналізує склад команди супротивника, історичну статистику матчів і популярні комбінації, після чого пропонує список рекомендованих героїв. У самій грі цей асистент також надає підказки щодо черговості покупки предметів, вибору талантів, розстановки вардів тощо.

2.2. Обмеження Dota Plus

Хоча Dota Plus пропонує інтегрований у клієнт гри асистент з підказками під час драфту та самої гри, його функціональність має низку обмежень, які не дозволяють вважати його повноцінною аналітичною системою підтримки рішень.

Основні недоліки Plus Assistant:

– рекомендації базуються переважно на загальних показниках переможності та частоті вибору героїв;

– навіть якщо герой рекомендований правильно, система не надає порад щодо стилю гри: агресивно чи обережно грати, займати активну позицію на мапі чи зосередитись на фармі тощо. Користувач не має можливості зрозуміти, чому асистент радить саме цього героя;

– функції Dota Plus доступні лише за платною підпискою, що унеможливлює широке застосування інструменту серед усіх категорій гравців.

Ці обмеження створюють потребу у більш гнучкому, відкритому та інформативному рішенні, яке дозволяє враховувати контекст драфту та стиль гри, а також залишається безкоштовним, доступним і відкритим до модифікацій.

Крім того, варто зауважити, що серед самих гравців Dota 2 Plus Assistant не отримав широкого визнання. Його поради часто вважають надто загальними або навіть такими, що вводять в оману, особливо в складніших матчах. Plus Assistant у більшості випадків може бути корисним лише для новачків, і навіть у цьому випадку – далеко не завжди. У спільноті існує думка, що краще довіряти власному ігровому розумінню, ніж автоматичним порадам сервісу.

2.3 Обгрунтування доцільності власної розробки

На основі аналізу функціональності Plus Assistant та його обмежень можна зробити висновок, що в межах існуючих рішень відсутній інструмент, який би повноцінно підтримував гравця на етапі драфту – з урахуванням реального ігрового контексту та стилю гри.

Plus Assistant, попри свою інтеграцію в клієнт гри та використання статистики, не надає глибокого аналізу командних взаємодій та не доповнює рекомендацію корисною інформацією із врахуванням додаткових факторів. Крім того, як вже було сказано, сервіс є платним, що обмежує його доступність.

Запропонована система має на меті вирішити ці проблеми. Її основні

переваги полягають у наступному:

– врахування контексту драфту – як союзних, так і ворожих героїв;

– рекомендація стилю гри, що найкраще підходить для конкретного героя в поточному складі команд;

– безкоштовна доступність для широкого кола користувачів;

– прозорість логіки рекомендацій і можливість подальшої модифікації системи;

– реалізація у вигляді веб-сервісу, що забезпечує зручний інтерфейс та легке підключення до моделі через API [5]: Користувачам не обов’язково запускати клієнт гри, щоб отримати допомогу. Це може бути корисним для тренерів професійних команд.

Таким чином, розробка власної системи є доцільною як з точки зору розширення функціональності, так і з позиції доступності та зручності використання. Така система може стати корисним інструментом як для новачків, так і для досвідчених гравців і тренерів, які хочуть посилити стратегічну складову на етапі драфту.

Висновок до розділу 2

У цьому розділі було проаналізовано існуючий інструмент – Dota Plus, який частково реалізує функціонал підтримки прийняття рішень під час драфту в грі Dota 2. Було описано його основні можливості, а також виявлено обмеження, що не дозволяють вважати його повноцінною аналітичною системою. Зокрема, Plus Assistant не враховує стиль гри і контекст драфту, а також залишається закритим і платним сервісом.

Виявлені недоліки стали основою для постановки завдання розробити власну систему, яка зможе надавати гнучкі рекомендації щодо вибору героїв та стилю гри. Така система має бути прозорою, безкоштовною та доступною як для звичайних гравців, так і для тренерів професійних команд.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

3.1 Вимоги до системи в цілому

Система, яка розробляється в межах даного проєкту, повинна відповідати сучасним вимогам до веб-орієнтованих інформаційних систем, зокрема систем підтримки прийняття рішень у специфічному домені кіберспорту. Зважаючи на обмежений час для прийняття рішень у реальному матчі, а також потребу в аналізі великого обсягу даних у стислі терміни, система повинна забезпечити високу швидкодію, надійність і масштабованість.

Інформаційна система має бути побудована за модульною архітектурою [6], що передбачає чіткий розподіл функцій між окремими підсистемами. Такий підхід дозволяє досягнути гнучкості при оновленні та супроводі системи, а також забезпечити незалежність кожного модуля при розгортанні, тестуванні й оновленні.

3.1.1 Підсистема збору та обробки даних

Повинна реалізовувати процес отримання даних з відкритих джерел таких як OpenDota API [7], у форматі, придатному для подальшої обробки й аналізу. Повинна виконувати фільтрацію, попередню обробку та збереження даних у сховищі. Система має підтримувати виключення неактуальних або нерейтингових матчів відповідно до встановлених фільтрів. Має бути передбачено існування реплікації бази даних для забезпечення цілісності та відновлення інформації у випадку збоїв.

3.1.2 Аналітична підсистема

Повинна містити дві незалежні моделі машинного навчання, кожна з яких вирішує окрему підзадачу: вибір героя та визначення стилю гри. Обидві моделі повинні бути натреновані на реальних ігрових даних, з урахуванням специфіки ігрової механіки та актуального балансу. Повинна бути розгорнута як окремий

ізольований сервіс, доступний для зовнішніх звернень через HTTP [8] протокол. Моделі повинні повертати результат у форматі JSON зі структурованими полями. Повинна забезпечуватись можливість оновлення моделей без зупинки системи загалом. Моделі повинні працювати в умовах навантаження без деградації якості.

3.1.3 Серверна частина

Повинна бути реалізована у вигляді REST API [9] сервісу, який отримує запити від клієнтської частини, проводить базову валідацію даних та передає їх до аналітичної підсистеми для подальшої обробки. Після отримання результату від моделі, серверна частина повинна формувати відповідь для клієнта.

Сервер повинен реалізувати систему кешування, щоб у випадку повторного запиту з однаковими параметрами зменшити навантаження на аналітичну підсистему і прискорити відповідь.

Сервер повинен підтримувати масштабування у горизонтальному напрямі, щоб забезпечити стабільність навіть при зростанні кількості користувачів.

Сервер повинен здійснювати логування всіх запитів, помилок та часу обробки, щоб в майбутньому ефективніше виявляти неефективності та неоптимізовані частини системи, що допоможе зробити її більш відмовостійкою.

3.1.4 Клієнтська частина

Інтерфейс повинен бути розроблений як Single Page Application з використанням сучасних веб-технологій. Клієнтська частина повинна взаємодіяти із сервером через REST API, надсилаючи запити та отримуючи відповіді у форматі JSON.

Веб-застосунок повинен містити початкову сторінку із загальною інформацією про сервіс, сторінку із сіткою героїв для вибору та отримання рекомендацій. А також сторінку, де буде відображено рекомендацію щодо стилю гри. Рекомендації повинні формуватись в режимі реального часу.

Веб-застосунок повинен бути адаптивним для усіх комп’ютерних дисплеїв, повинен мати зручний UI/UX із мінімумом дій для отримання результату та стабільно працювати при високому навантаженні або тимчасових проблемах із сервером. А також застосунок повинен завантажуватись не перевищуючи час очікування в 1 секунду.

Система працює в трьох режимах: реального часу, підготовки до гри та режимі відмови.

У режимі реального часу користувач під час драфту може отримувати рекомендацію у відповідь на запит із мінімальною затримкою. Цей режим вимагає стабільного з’єднання з сервером і працездатності ML-моделей.

У режимі підготовки до гри користувач повинен мати змогу вводити різні склади команд, пробувати різні комбінації та отримувати рекомендації заздалегідь. В цьому режимі допускається більш тривала обробка.

У режимі відмови система працює в обмеженому режимі та повинна проінформувати користувача про це повідомленням.

Перспективи розвитку системи:

– автоматична адаптація до оновлень гри;

– персоналізація рекомендацій на основі історії гравця;

– реалізація системи інсайтів на основі історії запитів користувача, яка дозволить зберігати попередні драфти, виявляти повторювані патерни у виборі героїв та стилі гри;

– створення мобільного застосунку;

– створення кабінету для професійних команд із персоналізованими рекомендаціями для її гравців. Надання можливості емулювати матчі гравців цієї команди проти інших професійних команд, що дозволить гравцям обрати тактику і стратегію гри спираючись на результат емуляції та власний досвід.

3.2 Вимоги до функціональних характеристик

Розроблювана система призначена для автоматизації ряду задач, пов’язаних

із аналітичною підтримкою процесу вибору героїв і стилю гри в Dota 2. Вона виконує функції рекомендаційного сервісу та надає користувачу інформацію в зручній формі через веб-інтерфейс.

Перелік функцій, що підлягають автоматизації:

– формування рекомендації щодо вибору героя на основі обраних героїв союзників та супротивника;

– формування рекомендації щодо стилю гри для обраного героя в заданому драфтовому контексті.

Вимоги до якості реалізації:

– основна рекомендація відображається у вигляді ранжованого списку п’яти найкращих героїв. Час формування рекомендації не повинен перевищувати 1 секунди у реальному режимі використання при стабільному з’єднанні;

– стилістична рекомендація надається при повному складі команд після рекомендації героїв у вигляді спливаючого вікна, де вказано стиль гри для кожного героя та короткий текстовий опис. Стиль гри має відповідати загальному контексту драфту в не менше, ніж 80% випадків, згідно з внутрішнім тестуванням. Час формування стилістичної рекомендації не повинен перевищувати 1 секунди.

3.3 Вимоги до видів забезпечення

3.3.1 Математичне забезпечення

Система повинна використовувати математичне забезпечення, засноване на методах машинного навчання. Для реалізації рекомендаційного функціоналу необхідно передбачити:

– використання моделі ранжування героїв, яка має враховувати фактори драфту, частотність перемог та інші показники героїв;

– використання класифікаційної моделі для визначення стилю гри, з урахуванням ролі героя, контексту драфту та статистичних патернів;

– застосування алгоритмів попередньої обробки даних, таких як нормалізація, кодування категоріальних ознак, генерація додаткових ознак;

– підтримку алгоритмів валідації моделей, створення тестових та тренувальних датасетів.

3.3.2 Інформаційне забезпечення

Інформаційне забезпечення системи повинно охоплювати всі компоненти необхідні для зберігання, оновлення та використання ігрових даних. Зокрема:

– підтримку структури бази даних, яка включає колекції з інформацією про матчі, героїв, ролі, пік-бан послідовності, результати;

– інтеграцію із зовнішніми джерелами даних для отримання та оновлення інформації;

– підтримку реплікації бази для забезпечення надійності;

– організацію зберігання із використанням нереляційної СУБД.

3.3.3 Програмне забезпечення

Програмне забезпечення системи повинно складатися із модулів, реалізованих із використанням відкритих або ліцензійних компонентів:

– серверна частина повинна бути реалізована із використанням Nest js, Typescript;

– клієнтська частинна повинна бути реалізована із використанням React версії 18 та більше із підтримкою адаптивної верстки;

– моделі машинного навчання можуть бути реалізовані з використанням бібліотек scikit-learn, XGBoost, LightGBM або PyTorch. Сервіси, що надають доступ до моделей через API, повинні бути реалізовані за допомогою Python та Fast API;

– програмна архітектура має забезпечувати розділення відповідальностей між модулями, логування, обробку помилок, кешування;

– система повинна бути сумісною з Docker для спрощення розгортання;

– взаємодія між модулями повинна здійснюватися через чітко визначені інтерфейси із дотриманням принципів модульності та повторного використання коду;

3.3.4 Технічне забезпечення

Система повинна бути здатна працювати як на локальному сервері, так і в хмарному середовищі.

Для цього передбачаються такі вимоги:

– сервер з мінімально можливою конфігурацією повинен мати 2 CPU, 4 ГБ оперативної пам’яті, 10 ГБ дискового простору;

– підтримка автоматичного масштабування під навантаження;

– мінімальні вимоги до клієнтського пристрою передбачають; наявність у користувача сучасного браузера (Chrome, Firefox, Edge, Safari), доступ до інтернету та мінімальну роздільну здатність дисплею 1280 на 720 пікселів;

– підтримка технологій HTTPS, CORS.

Висновок до розділу 3

У цьому розділі було сформовано повний перелік вимог до розроблюваної системи, що виконує аналітичну підтримку драфту в грі Dota 2. Визначено основну структуру системи та склад її підсистем: збору і обробки даних, аналітичну, серверну та клієнтську частини. Наведено характеристики кожної підсистеми, описано режим функціонування системи в реальному часі, режимі підготовки до гри та при відмові.

Окреслено ключові функціональні можливості, вимоги до точності, швидкодії та якості представлення результатів. Окрему увагу приділено перспективам розвитку, серед яких – персоналізація рекомендацій, розширення інтерфейсу, інсайти на основі історії користувача та можливість емуляції матчів.

Також сформульовано вимоги до математичного, інформаційного, програмного та технічного забезпечення, необхідного для реалізації повноцінної, масштабованої та надійної системи.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

Для забезпечення гнучкого та продуктивного зберігання інформації, пов’язаної з матчами, героями, ролями та іншими елементами драфту, було прийнято рішення про використання MongoDB – документно-орієнтованої, нереляційної системи керування базами даних, яка ідеально підходить для роботи з напівструктурованими даними у форматі двійкового JSON.

MongoDB володіє низкою важливих характеристик, що зумовили її вибір:

– на відміну від реляційних СУБД, Mongo не вимагає жорстко заданої структури таблиць, що дозволяє динамічно змінювати формат документів у процесі розробки без потреби міграцій;

– вбудовані механізми шардування MongoDB дозволяють легко масштабувати систему при зростанні обсягів даних;

– має високу продуктивність читання і запису, що є критично важливою умовою, коли система повинна повертати рекомендації із мінімальною затримкою;

– природно інтегрується із JSON орієнтованими API, в тому числі з OpenDota API, який є джерелом вхідних даних.

Завдяки цим властивостям, Mongo забезпечує стабільну роботу системи, швидкий доступ до потрібної інформації та мінімізацію ризиків при оновленнях схеми даних, що робить її кращим за наявних умов вибором для аналітичної систем, побудованих на динамічних структурах.

Для реалізації аналітичної підсистеми, яка містить рекомендаційні моделі, було обрано мову програмування Python, яка є де факто стандартом для задач штучного інтелекту, машинного навчання та обробки даних.

Python володіє великою кількістю спеціалізованих бібліотек таких як PyTorch, NumPy, Pandas, Scikit-learn, XGBoost тощо – усе доступне через стандартний менеджер пакетів pip.

Завдяки динамічній типізації і високому рівню абстракції Python дозволяє зосередитись на логіці задачі, а не на деталях реалізації, а також володіє потужними засобами для обробки табличних та векторних даних, що критично важливо для

передобробки даних перед тренуванням моделей. В екосистемі Python є готові рішення для розгортання ML моделей, в тому числі FastAPI, який дозволяє швидко створювати REST API для сервісів машинного навчання.

Крім цього, Python може ефективно та зручно застосовуватись для побудови скриптів збору даних, їх парсингу, очищення та аугментації. Він виконує роль універсального інструменту у межах всієї аналітичної частини системи. Його використання дозволяє не тільки ефективно тренувати моделі, а й повністю автоматизувати цикл обробки даних.

Серверна логіка системи реалізується з використанням NodeJS, NestJS та Typescript. Цей стек є перевіреним та надійним, що забезпечить високу швидкодію та зручність побудови REST API.

NodeJS є подієво-орієнтованою, неблокуючою платформою для серверного виконання JavaScript коду, яка показує високу продуктивність при великій кількості паралельних запитів, завдяки однопотоковій обробці вводу та виводу даних.

NestJS у свою чергу є фреймворком над NodeJS, який надає потужну архітектуру із підтримкою модулів, контролерів, сервісів, ін’єкції залежностей, middleware, фільтрів та інтерцепторів. Це значно спрощує створення масштабованих, модульних REST сервісів, дотримуючись при цьому принципів SOLID.

Typescript – це надмножина JavaScript, яка додає строгість типів, покращує автодоповнення та підвищує надійність коду. Застосування Typescript дозволяє уникнути великої кількості помилок у ранній фазі розробки, покращує документацію і читання проєкту.

Цей стек надає збалансовану комбінацію продуктивності, зрозумілої архітектури та надійності, що робить його хорошим вибором для серверної частини системи.

Клієнтська частина реалізовуватиметься на базі таких технологій як React, Tailwind CSS і Typescript.

React дозволяє реалізувати односторінковий додаток, що забезпечує динамічну зміну вмісту сторінки без перезавантаження, підвищуючи тим самим швидкість і зручність взаємодії. Компонентна архітектура React сприяє повторному використанню елементів інтерфейсу, модульності та простоті тестування. Також забезпечується висока продуктивність завдяки використанню віртуального DOM.

Tailwind CSS – це фреймворк утилітарних стилів, який забезпечує швидку розробку інтерфейсу завдяки утилітарному підходу – стилі додаються безпосередньо до HTML елементів у вигляді класів, що дозволяє уникати перевантаження кастомними стилями. Це суттєво знижує технічний борг при масштабуванні інтерфейсу.

Typescript забезпечуватиме типізацію пропсів, API відповідей, стану компонента, що підвищує надійність та зручність розробки.

Завдяки цьому набору інструментів можна реалізувати швидкий і масштабований інтерфейс, який залишиться стабільним при активній взаємодії із сервером.

Для організації процесу розгортання було обрано Docker та Docker Compose – сучасні інструменти для контейнеризації програмного забезпечення.

Docker дозволяє упакувати кожну підсистему (клієнт, сервер, база даних, аналітичні моделі) у відокремлений контейнер із усіма залежностями. Це гарантує стабільну роботу незалежно від середовища виконання.

Docker Compose дозволяє керувати багатоконтейнерною архітектурою за допомогою одного конфігураційного файлу, що спрощує запуск системи, її масштабування та CI/CD процеси.

Застосування контейнеризації дозволяє досягти відтворюваності середовища розробки, легкого розгортання у хмарних системах або на локальному сервері, а також гнучкої інтеграції із системами моніторингу, балансування навантаження та оркестрації. Це сприяє підвищенню надійності та масштабованості системи, забезпечує спрощене оновлення компонентів та зменшує час на усунення збоїв. Контейнери дозволяють ізолювати окремі сервіси, що відкриває можливості для незалежної розробки, тестування та розгортання кожної підсистеми.

Висновок до розділу 4

У результаті досліджень було визначено та обгрунтовано використання наступних інструментів: MongoDB, Python, Pytorch, Numpy, Pandas, NodeJS, NestJS, Typescript, React, Tailwind CSS, Docker та Docker Compose.

Особливу увагу приділено технологіям, які сприяють ефективному опрацюванню даних та інтеграції моделей машинного навчання в архітектуру системи. Python як універсальний інструмент для розробки аналітичного ядра гармонійно поєднується із MongoDB, що здатна адаптуватися до змін у структурі даних без потреби у складних міграціях.

Окремо варто відзначити важливість використання контейнеризації. Вона забезпечує цілісність середовища, зменшує імовірність помилок при розгортанні та дозволяє швидко масштабувати систему відповідно до зростаючих вимог. Підхід, заснований на мікросервісній архітектурі, дає змогу кожному компоненту еволюціонувати незалежно, що є суттєвим у контексті подальшого розвитку проєкту.

Обрані технології є сучасними, активно підтримуваними та широко застосовуваними у професійній розробці. Вони дозволяють реалізувати проект, що відповідає усім функціональним і нефункціональним вимогам, а також забезпечують потенціал для подальшого розвитку системи.

5 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Структура системи

Розроблювана система є комплексним веборієнтованим програмним продуктом, що об’єднує низку взаємопов’язаних підсистем, кожна з яких виконує чітко визначену роль. Така модульна побудова сприяє спрощенню розробки, тестування, розгортання та масштабування окремих компонентів. Завдяки цьому система може бути розвинута, модифікована або перенесена в інше середовище без необхідності суттєвої перебудови архітектури.

Система реалізована у форматі багаторівневої клієнт-серверної архітектури, де чітко виокремлено клієнтський рівень, серверний рівень, аналітичну підсистему та сховище даних. Кожен з рівнів виконує власну функцію, але водночас взаємодіє з іншими через стандартизовані API та протоколи передачі даних.

5.1.1 Клієнтський рівень

Клієнтський інтерфейс є основною точкою взаємодії між користувачем і системою. Він реалізований як односторінковий застосунок на базі бібліотеки React. Сучасний інтерфейс забезпечує високу інтерактивність, мінімальну затримку при взаємодії та можливість швидкого оновлення окремих частин UI без повного перезавантаження сторінки. Компонентна архітектура React сприяє підтримуваності коду, повторному використанню елементів інтерфейсу та спрощенню процесу розробки. Для стилізації інтерфейсу обрано Tailwind CSS, що дозволяє швидко та гнучко створювати адаптивні компоненти без дублювання CSS коду.

Функції:

– введення складу команд для обробки моделями;

– отримання та відображення рекомендацій від моделей;

– інформування користувача про збої в системі.

5.1.2 Серверний рівень

Серверна частина виконує роль посередника між клієнтом та іншими підсистемами, а також реалізує основну логіку обробку запитів. Вона побудована на платформі NodeJS із використанням фреймворку NestJS, що дозволяє дотримуватись принципів модульності та розділення відповідальностей. Сервер реалізує REST API, до якого звертається клієнт, а також координує звернення до моделей машинного навчання.

Функції:

– валідація вхідних запитів;

– кешування результатів;

– логування;

– обробка помилок та fallback механізми у разі часткової або повної недоступності.

5.1.3 Аналітична підсистема

Це окремий модуль, реалізований мовою Python, який містить натреновані моделі машинного навчання для генерації рекомендацій. Він взаємодіє із сервером через HTTP, отримуючи дані про поточний стан драфту, і повертає у відповідь список рекомендованих героїв та відповідний стиль гри. Взаємодія реалізована у форматі REST API, що забезпечує простоту інтеграції та можливість масштабування

Цей модуль ізольований від решти системи, що дозволяє легко перевчати моделі, не впливаючи на клієнт чи API. Такий підхід відповідає принципам мікросервісної архітектури та підвищує гнучкість системи.

Функції:

– надання рекомендацій щодо драфту;

– надання рекомендацій щодо стилю гри.

5.1.4 Рівень зберігання даних

Для зберігання інформації використовується нереляційна база даних MongoDB, яка дозволяє працювати у форматі двійкового JSON. Це забезпечує зручну інтеграцію з OpenDota API, оскільки формат відповіді ззовні вже є JSON орієнтованим. MongoDB дозволяє легко масштабуватись горизонтально та динамічно змінювати структуру документів без необхідності складної міграції схеми.

Функції:

– зберігання історичних даних про матчі;

– зберігання даних про героїв та їх характеристики і глобальної статистики.

5.1.5 Інфраструктура

Усі компоненти системи працюють у Docker контейнерах, які забезпечують ізольоване середовище виконання для кожної підсистеми. Для одночасного запуску клієнта, API, моделей та бази даних використовується Docker Compose. Такий підхід спрощує процес розгортання та оновлення системи, а також дозволяє легко переносити її на інші сервери або в хмарне середовище.

На рисунку 5.1 показано структуру усієї системи.

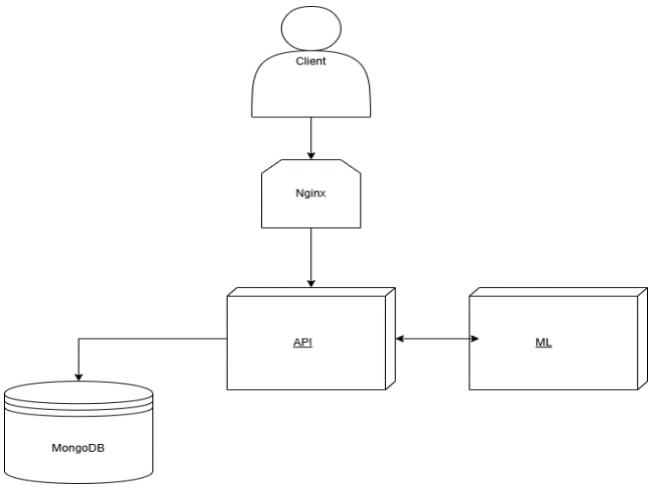

Рисунок 5.1 – Структура системи

Схема демонструє загальну логіку взаємодії між основними компонентами системи. Користувач надсилає запити через клієнтський інтерфейс, які перенаправляються на сервер через Nginx. Останній виконує роль API Gateway, а також, за необхідності, балансувальника навантаження між кількома екземплярами серверної частини.

5.2 Функціональна модель системи

Розроблювана система призначена для взаємодії із користувачем у режимі реального часу або підготовки до матчу, надаючи рекомендації щодо вибору героїв і відповідного стилю гри. Для опису логіки такої взаємодії доцільно побудувати функціональну модель, яка дозволяє структуровано подати ключові сценарії використання та визначити учасників, залучених до процесу.

У межах цієї моделі розглядаються різні категорії користувачів: гравці, які потребують швидких рішень у ході драфту; тренери, які працюють з системою на етапі підготовки, а також сама система як інструмент, що забезпечує обробку запитів і формування відповідей.

Функціональна модель системи фокусується не лише на можливостях програмного продукту, а передусім на способі їх використання. Вона описує, як саме ініціюються дії, які запити надсилаються, яким чином вони обробляються та як формується результат. Такий підхід дозволяє чітко визначити поведінку системи в типових ситуаціях і є важливою складовою при подальшому проєктуванні інтерфейсів, архітектури та логіки роботи.

5.2.1 Основні актори

Функціональна модель передбачає взаємодію між кількома типами користувачів, кожен з яких має власну роль у межах системи. Найважливішими з них є гравець, тренер та сама система, що виконує технічну обробку запитів.

Гравець є основним користувачем системи. Він звертається до неї під час драфту – етапу, коли потрібно приймати швидкі рішення щодо вибору героїв. Гравець вводить склад союзників і супротивників, після чого отримує рекомендації, що базуються на статистичних та аналітичних даних. Діаграма варіантів використання системи користувачем представлена на рисунку 5.2. У більшості випадків взаємодія із системою відбувається у режимі реального часу, коли затримка у відповіді повинна бути мінімальною.

Рисунок 5.2 – Діаграма варіантів використання

Система в межах цієї моделі розглядається як умовний актор, що забезпечує технічне виконання всіх запитів. Вона включає серверну частину, аналітичні моделі, а також логіку передачі даних і формування відповіді. Система не ініціює дії, однак забезпечує їх обробку та зворотний зв’язок, гарантуючи стабільність, надійність та швидкодію в усіх сценаріях використання.

5.2.2 Сценарії використання

5.2.2.1 Отримання рекомендацій під час драфту

Найтиповішим варіантом використання системи є взаємодії з боку гравця, який перебуває на етапі драфту. В умовах обмеженого часу й часткової інформації гравець вводить склад уже обраних героїв – як союзників, так і суперників, і очікує

швидкої рекомендації. Система формує відповідь на основі статистичних закономірностей та аналітики, враховуючи ролі, характеристики героїв, їх показники та інші фактори.

Після надсилання запиту клієнтська частина передає інформацію через сервер до аналітичної підсистеми, яка повертає ранжований список героїв із коефіцієнтами релевантності. Результат виводиться на інтерфейс користувача у зручному форматі, що дозволяє швидко приймати наступні рішення.

На кресленику ІС13.090БАК.005 Д2. зображено сценарій використання гравцем системи для отримання рекомендацій.

5.2.2.2 Робота системи у режимі відмови

Інформаційна система передбачає можливість тимчасової недоступності окремих компонентів – зокрема аналітичної моделі. У такому випадку ключовим завданням є підтримання базової функціональності без втрати загальної стабільності. Якщо система не може отримати актуальну відповідь від моделі, то вона перевіряє наявність кешованого результату з аналогічними параметрами. Якщо кеш відсутній – користувач інформується про технічну паузу або обмежену функціональність.

Цей сценарій забезпечує відмовостійкість системи та покращує загальний користувацький досвід, знижуючи ризики різкого переривання роботи. Сценарій зображено на кресленику ІС13.090БАК.005 Д1.

5.3 Модель бази даних

Для збереження, обробки та ефективного доступу до інформації, пов’язаної з матчами, героями та їх статистикою у розробленій системі використовується документно-орієнтована база даних MongoDB.

Модель бази даних є гнучкою, масштабованою та легко адаптується до змін у структурі даних без потреби у складних міграціях. Усі основні сутності системи

представлені у вигляді колекцій, які зберігають документи з відповідними полями. У цьому підрозділі розглянуто кожну з ключових колекцій бази даних: їхню структуру, призначення ключових полів тощо.

5.3.1 Колекція heroes

Зберігає інформацію про героя. Використовується як на клієнті, так і при аналізі даних та тренуванні моделі. Більш детальну інформацію з описом колекції надано в таблиці 5.1.

Таблиця 5.1 – Колекція heroes

Поле	Тип	Опис	Приклад
hero_id	Number	Ідентифікатор героя	1
name	String	Назва героя	«npc_dota_hero_shadow_fiend»
localized_name	String	Локалізована назва героя	«Shadow Fiend»
primary_attr	String	Основна характеристика героя	«agi»
attack_type	String	Тип атаки	«Ranged»
roles	Array	Масив ролей, які може виконувати героя	[«Carry», «Nuker»]
img	String	Шлях до зображення героя	«https://link_to_img»
phase_1_winrate	Number	Середній відсоток перемог героя при його виборі на першій фазі драфту	0.5122
phase_2_winrate	Number	Середній відсоток перемог героя при його виборі	0.5038

Поле	Тип	Опис	Приклад
		на другій фазі драфту	
pro_winrate	Number	Середній відсоток перемог героя у професійних матчах	0.4269
gold_per_min	Array<object>	Середній показник золота за хвилину у відповідному перцентилі	{percentile: 0.1, value: 398}
xp_per_min	Array<object>	Середній показник досвіду за хвилину у відповідному перцентилі	{percentile: 0.3, value: 510}
kills_per_min	Array<object>	Середній показник нейтралізацій за хвилину у відповідному перцентилі	{percentile: 0.3, value: 510}
last_hits_per_min	Array<object>	Середній показник кількості нейтралізованих крипів за хвилину	{percentile: 0.4, value: 410}
hero_damage_per_min	Array<object>	Середній показник шкоди по героях за хвилину у відповідному перцентилі	{percentile: 0.5, value: 321}
hero_healing_per_min	Array<object>	Середній показник лікування	{percentile: 0.2, value: 21}

Поле	Тип	Опис	Приклад
		союзників за хвилину у відповідному перцентилі	
tower_damage	Array<object>	Середній показник шкоди по будівлях у відповідному перцентилі	{percentile: 0.7, value: 2100}

5.3.2 Колекція matches

Містить загальну інформацію про матчі. Використовується при парсингу матчів. Детальну інформацію про опис колекції надано в таблиці 5.2.

Таблиця 5.2 – Колекція matches

Поле	Тип	Опис	Приклад
match_id	Number	Ідентифікатор матчу	8196584301
start_time	Number	Час початку матчу в Unix форматі	1740868366

5.3.3 Колекція matches_info

Містить детальну інформацію про матчі. Використовується для тренування моделей. Детальну інформацію про опис колекції надано в таблиці 5.3.

Таблиця 5.3 – Колекція matches_info

Поле	Тип	Опис	Приклад
match_id	Number	Ідентифікатор матчу	8196584301
Duration	Number	Тривалість матчу в секундах	2343
radiant_win	Boolean	Чи перемогла команда Radiant	true
players	Array<Player>	Масив з 10 об’єктів – по	Дивіться таблицю 5.4

Поле	Тип	Опис	Приклад
		одному для кожного гравця	
picks_bans	Array<object>	Послідовність пік/бан фаз під час драфту	{is_pick: true, hero_id: 74, team: 0, order: 0}

5.3.4 Об’єкт Player

Містить інформацію про гравця та його ігрову статистику в заданому матчі. Використовується для тренування моделей. Опис структури об’єкта наведено в таблиці 5.4.

Таблиця 5.4 – Структура об’єкта Player

Поле	Тип	Опис	Приклад
hero_id	Number	Ідентифікатор обраного героя	84
is_radiant	Boolean	Визначає за яку команду грає гравець	true
kills	Number	Кількість нейтралізацій здійснених гравцем	8
deaths	Number	Кількість елімінацій гравця	3
assists	Number	Кількість допомог гравця союзникам при здійсненні нейтралізацій	10
gold_per_min	Number	Дохід золота за хвилину	430
xp_per_min	Number	Дохід досвіду за хвилину	510
hero_damage	Number	Загальна шкода, завдана ворожим героям	18500
Tower_damage	Number	Шкода, завдана будівлям	2600

Поле	Тип	Опис	Приклад
last_hits	Number	Кількість знищених кріпів	8
level	Number	Кінцевий рівень героя в матчі	23
hero_healing	Number	Показник лікування союзників	0
kda	Number	Об’єднаний показник, який характеризує внесок гравця у командні бої	3.86
benchmarks	object	Містить оцінку ефективності гравця за основними статистичними показниками у матчі	Дивіться таблицю 5.5

У таблиці 5.5 описано структуру об’єкта benchmarks, який містить оцінку ефективності гравця за основними статистичними показниками гри за хвилину: золото, досвід, кількість нейтралізацій, шкода по героях, кількість знищених кріпів, лікування та шкода по будівлях. Такі показники дозволяють оцінити внесок кожного гравця у гру не лише за абсолютними значеннями, але й у порівнянні з іншими учасниками.

Таблиця 5.5 – Структура об’єкта Player

Поле	Тип	Опис	Приклад
raw	Number	Фактичне значення показника	537
pct	Number	Перцентиль у якому відсотку знаходиться показник гравця серед усіх інших.	0.794

5.4 Передавання та обробка даних

Розроблена система функціонує як багаторівневий програмний комплекс, що виконує обмін даними між кількома компонентами: веб-клієнтом, серверною частиною, аналітичною підсистемою та сховищем даних. Цей підрозділ описує канали передачі даних, структуру запитів і відповідей, процеси обробки інформації на всіх етапах, а також формати вхідних і вихідних даних.

5.4.1 Загальна схема передачі даних

Система працює за класичною схемою «клієнт – сервер», доповненою аналітичною підсистемою як окремим сервісом. Основні потоки даних зображено на кресленику ІС13.090БАК.005 ДЗ.

На початковому етапі взаємодії користувач, перебуваючи на інтерфейсі клієнтського додатку, формує запит – обрані герої обох команд на відповідній фазі драфту. Ці дані передаються у форматі JSON через HTTP протокол на серверну частину системи, яка виступає центральним посередником між клієнтом та аналітичною підсистемою.

Сервер виконує первинну валідацію отриманих даних, після чого здійснює перевірку на наявність відповідного результату у внутрішньому кеші. Якщо результат уже було згенеровано раніше для аналогічного складу героїв, він миттєво повертається користувачу, що значно скорочує час очікування та знижує навантаження на аналітичну підсистему.

У випадку, якщо кешованого результату не знайдено, запит передається до аналітичної підсистеми. Вихідні дані знову передаються у вигляді структурованого JSON об’єкта назад до серверної частини, де вони кешуються для майбутніх звернень і надсилаються користувачу.

Уся інформація, пов’язана зі статистикою матчів, характеристиками героїв зберігається у базі даних MongoDB.

5.5 Архітектура програмного забезпечення

Архітектура програмної системи реалізована на основі багаторівневого підходу, що передбачає розмежування обов'язків між різними логічними шарами. Кожен з цих шарів виконує строго визначену роль у загальній структурі додатку й оперує власними об'єктами. Усі взаємодії між шарами відбуваються виключно через явно визначені інтерфейси, що забезпечує ізольованість, слабкий зв'язок між компонентами, підвищену тестованість та можливість масштабування. Система розроблена за допомогою фреймворку NestJS, який завдяки підтримці модульної структури та інверсії залежності ідеально підходить для впровадження подібної архітектурної моделі. Діаграму класів системи зображено на кресленику ІС13.090БАК.005 Д4.

5.5.1 Доменний шар

Основною функцією цього шару є опис предметної області та формалізація її понять у вигляді типів, інтерфейсів та узгоджених контрактів взаємодії. В межах цього шару не реалізовується жодної логіки, яка стосується збереження даних або взаємодії з мережею. Натомість він слугує фундаментом, на якому базуються всі інші частини застосунку. Тут визначено структуру таких сутностей, як герой, пік, команда, структура відповіді на рекомендаційні запити та інші основні елементи. Дані у доменному шарі є виключно декларативними: описуються лише їхня форма, зв'язки та допустимі значення, але не спосіб їх отримання чи обробки. Це дозволяє зберігати доменний рівень стабільним і нечутливим до змін у нижніх технологічних прошарках. Завдяки використання інтерфейсів, таких як IHeroRepository, IDraftRecommendationService та IPlaystyleRecommendationService, домен також встановлює жорсткі рамки щодо очікуваної поведінки реалізацій, які будуть розташовані в інших шарах, але водночас залишається повністю незалежним від конкретного способу досягнення цієї поведінки.

5.5.2 Прикладний шар

Над доменом розташовано прикладний шар, який у цій архітектурі виступає посередником між абстрактними правилами предметної області та конкретними сценаріями використання системи. Основним завданням цього шару є координація бізнес логіки, обробка запитів, розподіл ролей між підлеглими службами й контроль за дотриманням умов переходу між станами системи. Усі сервіси, що розміщені на цьому рівні, реалізують лише одну відповідальність, що відповідає принципу єдиної відповідальності. Наприклад, HeroService відповідає винятково за взаємодію з інформацією про героїв. Він отримує їх список або лиш одного за ідентифікатором, делегуючи роботу з джерелом даних через інтерфейс репозиторію. Натомість DraftService виконує завдання передачі поточного стану драфту до зовнішнього рекомендаційного модуля, який повертає ранжований список доцільних для вибору героїв. Він не аналізує структуру даних, не зберігає їх і не робить жодних припущень про те, яка модель стоїть за рекомендаціями.

Важливо підкреслити, що логіка, пов’язана з оцінкою стилю гри, винесена у інший сервіс PlaystyleService, який працює незалежно, має власний інтерфейс взаємодії та не дублює поведінку інших служб. Такий поділ забезпечує чистоту структури, знижує ризик дублювання коду й полегшує тестування.

5.5.3 Інфраструктурний шар

Реальні дії, що стосуються взаємодії із зовнішніми сервісами, базами даних або будь-якими іншими сторонніми технологіями, делегуються інфраструктурному шару. Тут розміщуються конкретні класи, які реалізують контракти з домену, використовуючи реальні інструменти – зокрема бібліотеки для HTTP запитів, засоби для роботи з базами даних, сервіси конфігурації та логування. Наприклад HeroRepository є реалізацією інтерфейсу IHeroRepository, що використовує бібліотеку Mongoose для взаємодії з базою даних. Цей репозиторій відповідає за отримання всіх героїв або конкретного героя за ID, при цьому обробляє всі технічні

аспекти, включно з побудовою запитів, десеріалізацією результатів та перетворенням їх у доменні об’єкти. Реалізація сервісу рекомендації героїв здійснюється в окремому класі, що надсилає запити на зовнішню адресу, отримує відповідь і мапить її в структуру, яка відповідає визначеному типу DraftResponse. Аналогічно працює реалізація сервісу стилю гри StyleRecommendationService, який використовує той самий підхід, але з іншим кінцевим адресатом і типом відповіді.

5.5.4 Шар представлення

Останнім у цій структурі виступає рівень представлення, який відповідає за безпосередній контакт із користувачем або зовнішнім клієнтським інтерфейсом. Тут розташовані HTTP контролери, що обробляють запити REST API. Кожен контролер прив’язаний до відповідного сервісу на прикладному шарі й фактично слугує маршрутизатором, який приймає запит, викликає відповідний метод сервісу та повертає відповідь клієнту. Контролери не містять бізнес логіки, не зберігають стану та не взаємодіють із базами даних. Вони лише формують обгортку навколо вхідного та вихідного трафіку. HeroController працює із запитами на отримання героїв, DraftController – з рекомендацією піків, а StyleController – з визначенням стилю гри після завершення драфту. Така розподіленість дозволяє незалежно обслуговувати окремі маршрути не створюючи надлишкової складності в одному місці.

Згідно з принципами NestJS, кожна функціональна одиниця згрупована в окремий модуль, який поєднує інтерфейси, реалізації, сервіси та контролери, обмежуючи область видимості та створюючи замкнене середовище. Такі модулі, як HeoModule, DraftRecommendationModule, StyleRecommendationModule, а також LogginModule, під’єднуються до головного AppModule, який виконує початкове збирання всієї системи. У ньому також відбувається ініціалізація змінних середовища, налаштування бази даних та запуск HTTP сервера. Завдяки такій модульності стає можливою гнучка перебудова проєкту в майбутньому, наприклад

шляхом підключення нових моделей, додавання нових джерел даних або винесення окремих частин у мікросервіси.

Ця архітектура забезпечує структурну чіткість, відповідність принципам чистої архітектури та можливість нарощування функціоналу без повторного проєктування. Всі частини системи взаємодіють виключно через контракти, кожна відповідальність відокремлена, а код легко читати, тестувати та масштабувати.

5.5.5 Специфікація функцій

Таблиця 5.6 – Специфікація функцій

Назва функції	Вхідні дані	Вихідні дані	Опис функції
HeroController.getAll	-	Hero[]	Отримує список усіх героїв, доступних у системі
HeroController.getById	id: String	Hero | null	Отримує інформацію про героя за його ідентифікатором
HeroService.getAll	-	Hero[]	Викликає IHeroRepository для отримання списку героїв з бази даних
HeroService.getById	id: String	Hero | null	Повертає героя з бази по його ідентифікатору або null, якщо його не знайдено
HeroRepository.findAll	-	Hero[]	Зчитує всі записи героїв із бази
HeroRepository.findById	id: Number	Hero | null	Зчитує конкретного героя з бази за ID
DraftController.recommend	Draft	DraftResponse	Рекомендація героя

Назва функції	Вхідні дані	Вихідні дані	Опис функції
DraftService.recommend	Draft	DraftResponse	Здійснює звернення до IDraftRecommendationService
RecommendationService.recommend	Draft	DraftResponse	Надсилає запит до моделі рекомендацій.
StyleController.advise	Draft	PlaystyleResponse	Обробляє запит на отримання рекомендацій стилю гри
StyleRecommendationService.advise	Draft	PlaystyleResponse	Надсилає запит до моделі рекомендацій стилю гри та повертає відповідь

5.6 Розробка користувацького інтерфейсу

У процесі розробки програмного забезпечення важливу роль відіграє створення зручного та інтуїтивно зрозумілого інтерфейсу користувача. В даній системі, що реалізує рекомендаційний механізм для Dota 2, інтерфейс розроблявся відповідно до сучасних принципів адаптивного дизайну та орієнтованості на користувача. Проект побудований на базі бібліотеки React з використанням Typescript, що дозволяє поєднати гнучкість функціонального програмування із строгою типізацією. Також використано TailwindCSS для централізованої стилізації та швидкого компонування сторінок. Загалом інтерфейс поділений на кілька логічних етапів. Спочатку користувач ознайомлюється із системою на вітальній сторінці, де переходить до вибору команди, формування драфту та отримання рекомендацій стилю гри.

Головна сторінка є першою точкою взаємодії із системою. Вона поєднує в собі презентаційну функцію та функцію маршрутизації. Інтерфейс сторінки складається з блоків вітального тексту та коротких описів можливостей системи та

кнопки, яка переадресовує користувача на сторінку рекомендації драфту. Кожен блок з додатковою інформацією має візуальний супровід та відображається із поступовою анімацією при скролі сторінки. Це особливо важливо для користувачів, які вперше взаємодіють із застосунком. Зображення сторінки з привітанням користувача показано на рисунку 5.3.

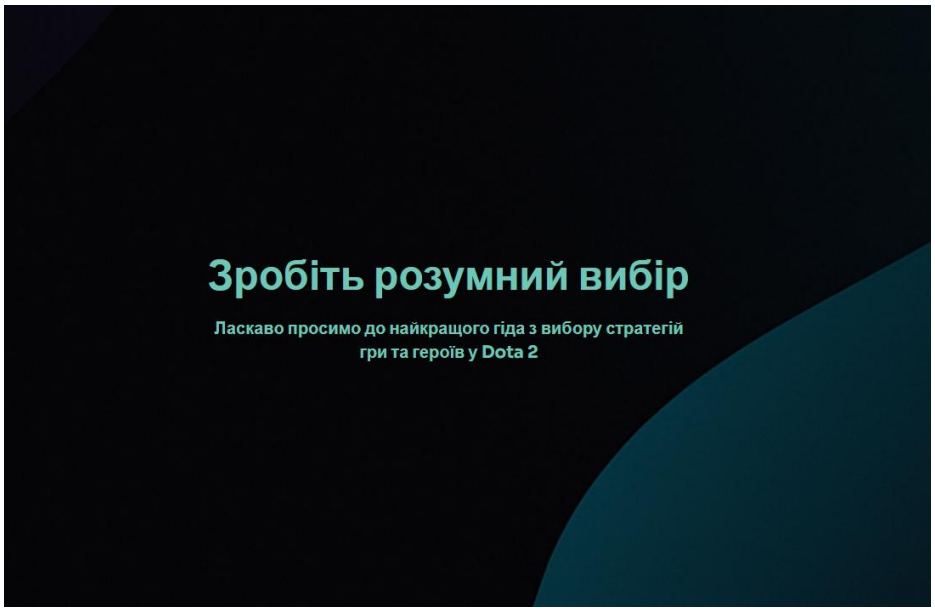

Рисунок 5.3 – Привітання користувача

На рисунках 5.4, 5.5 та 5.6 зображено блоки інформації, що коротко ознайомлять користувача із функціоналом застосунку.

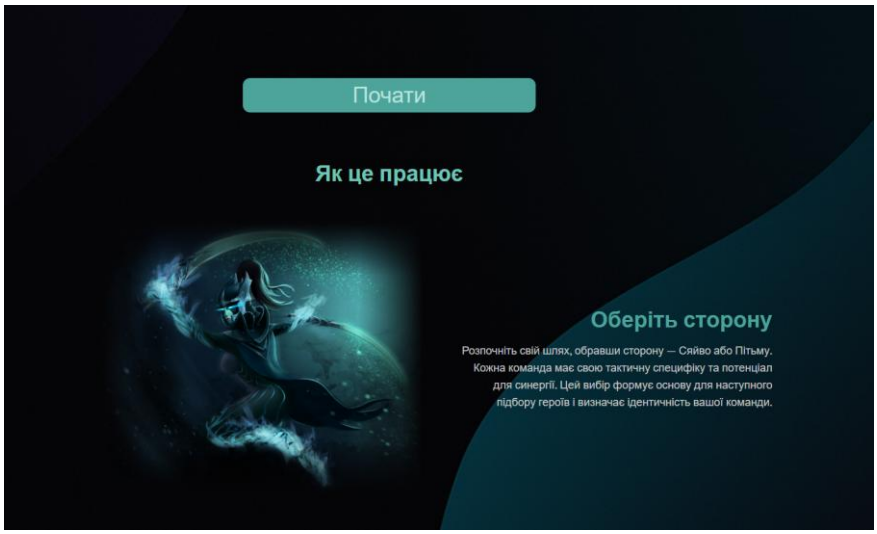

Рисунок 5.4 – Блок «Як це працює» та кнопка «Почати»

Даний блок також веде користувача до основного функціоналу системи з рекомендаціями.

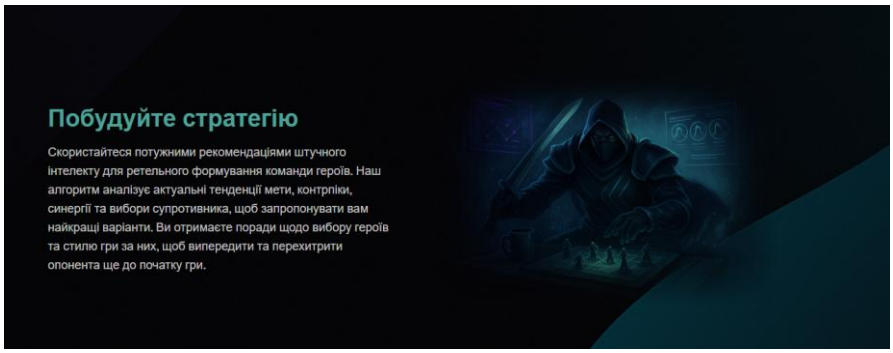

Рисунок 5.5 – Блок «Побудуйте стратегії»

Цей блок розповідає користувачеві, що система працює на основі рекомендацій штучного інтелекту.

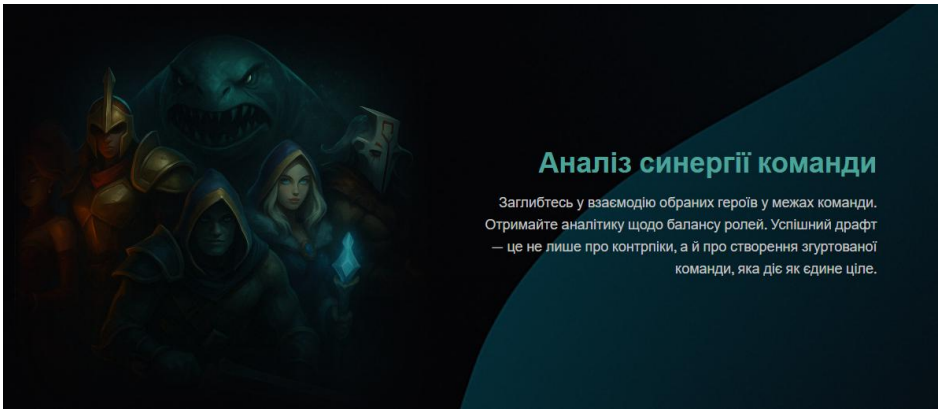

Рисунок 5.6 – Блок «Аналіз синергії команди»

Останній інформаційний блок містить більш детальну інформацію про функціонал системи.

Наступним етапом є сторінка вибору сторони. Вона реалізує логіку, яка відповідає за початку ініціалізацію параметрів драфту: вибір команди Сяйва або Пітьми. Структурно сторінка поділена на дві рівновеликі області, кожна з яких представляє одну із сторін у грі. В центрі початково з’являється напівпрозорий напис, що натякає користувачу про призначення сторінки: він повинен обрати сторону, за яку грає. Натискання на будь-яку з частин спричиняє збереження

вибору в контексті застосунку та перенаправлення користувача на сторінку драфту. Важливо, що сторінка не має надлишкових елементів, що дозволяє користувачу сконцентруватися саме на прийнятті рішення. Вигляд сторінки зображено на рисунку 5.7.

Рисунок 5.7 – Сторінка вибору сторони

Ключовим етапом взаємодії є драфт, який реалізований на сторінці DraftPage. Саме тут відбувається основна взаємодія користувача із системою, що передбачає вибір героїв, побудову складу команди та отримання рекомендацій щодо вибору героя на будь-якому етапі драфту. Візуальна структура сторінки побудована як поєднання компонент верхньої панелі з командами та нижньої секції зі списком доступних героїв. Вгорі виведено по п’ять клітинок на героя для кожної команди. Порожні клітинки заповнюються в міру вибору героїв. Нижня частина поділена на чотири секції відповідно до базових атрибутів героїв, що спрощує орієнтування серед сітки, що містить 126 варіантів вибору. Кожен герой відображається у вигляді окремої картки з ілюстрацією, назвою та додатковими індикаторами. Завдяки використанню глобального контексту, який обгортає сторінку та всі її компоненти, усі зміни стану централізовано обробляються редюсером, що забезпечує прозору зміну стану без дублювання логіки у кожному компоненті.

Дані про всіх доступних героїв завантажуються при монтуванні сторінки в DOM за допомогою асинхронного запиту до API системи. Результати кешуються,

що дозволяє уникнути надмірного навантаження на сервер та забезпечити швидкий доступ до списку героїв на будь-якому етапі взаємодії.

Сторінку драфту з сіткою усіх героїв, класифікованих по їх основному атрибуту та клітинками для заповнення складу команд зображено на рисунку 5.8.

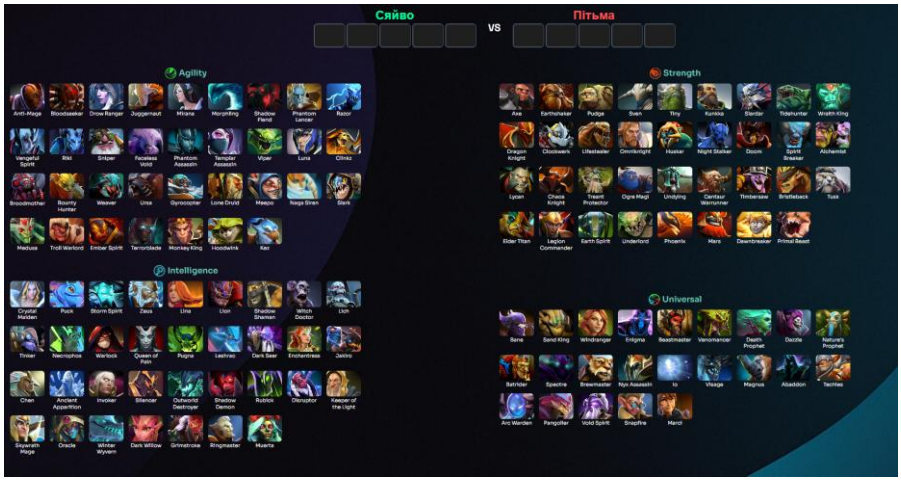

Рисунок 5.8 – Сторінка драфту

Після того, як користувач повністю сформував склади команд, йому надається можливість отримати рекомендації стилю гри для кожного героя в його команді при натисканні відповідної кнопки, що з’явиться на кінцевому етапі формування команд.

Сторінка StyleRecommendationPage реалізує фінальний етап взаємодії, де користувач отримує інформацію про найкращий за наявних умов стиль гри для кожного з п’яти обраних союзних героїв. Інтерфейс побудований у вигляді горизонтальної каруселі, де кожен слайд відповідає одному герою. В центрі кожного слайду відображено зображення героя, його назву та піктограму із стилем гри.

Система може рекомендувати три варіанти стилю гри: агресивний, збалансований та підтримуючий. Користувач може перегортати слайди за допомогою стрілок або натискати на навігаційні точки під каруселлю. Це дозволяє переглядати результат рекомендації у зручному темпі. Приклад рекомендації стилю гри показано на рисунку 5.9.

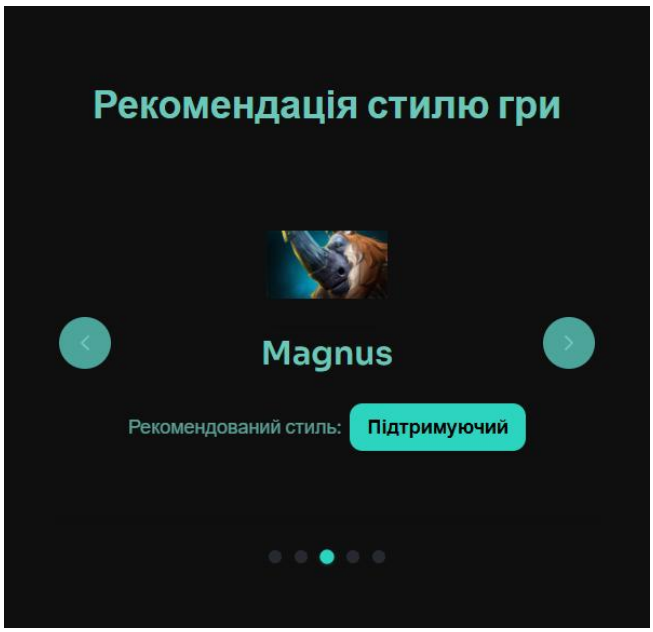

Рисунок 5.9 – Приклад рекомендації стилю гри на сторінці з рекомендаціями

У реалізації користувацького інтерфейсу ключову роль відіграє контекстна модель управління станом. За допомогою React Context API створено глобальний контекст DraftContext, який охоплює всі компоненти сторінки драфту. Це дозволяє передавати дані про поточний стан обраних героїв у кожній команді, а також функції для взаємодії з драфтом до дочірніх компонент DOM дерева, не передаючи ці параметри вручну через пропси компонент. Контекст забезпечує централізовану точку входу для зміни стану, що значно спрощує обслуговування коду та зменшує кількість потенційних помилок.

Функціонування контексту базується на редюсері draftReducer, який обробляє дискретні події, що називаються actions, які змінюють стан застосунку. Наприклад, при виборі героя до редюсера відправляється дія PICK_HERO, яка перевіряє до якої сторони належить герой і додає його до відповідної команди. Скасування вибору героя також централізовано реалізовано за допомогою того ж редюсера та дії UNPICK_HERO.

Список доступних героїв, що відображені у сітці, формується на основі запиту до API серверу, здійсненого за допомогою бібліотеки react tanstack query, яка забезпечує автоматичне кешування, повторні спроби запиту даних у разі

помилок та індикацію завантаження даних. Результати запиту використовуються для візуалізації повного списку героїв у вигляді карток HeroCard. Кожна картка відображає зображення героя, назву, а також стан вибору та рекомендації. Якщо герой вже обраний у будь-яку з команд, його картка стає неактивною. Якщо герой рекомендований системою, то рамка його зображення підсвічується спеціальним кольором. Візуальне оновлення картки відбувається динамічно при зміні складу команд, яку ініціює сам користувач.

На етапі завершення драфту, коли обидві команди укомплектовано, користувач отримує можливість перейти до сторінки рекомендацій стилю гри. Комунікація з серверною частиною для отримання стилістичних рекомендацій також реалізована через react tanstack query із використанням окремого хука useStyleRecommendation. У відповідь система повертає список героїв користувача разом із відповідними їм стилями гри. На основі цих даних формується карусель, де кожен слайд містить назву героя, його зображення та піктограму, що візуально ілюструє рекомендований стиль гри для кожного героя.

Користувач може зручно переглядати рекомендації за допомогою кнопок навігації або точок індикатора. Перемикання між героями відбувається без перезавантаження сторінки, що досягається завдяки локальному стану, керованому хуком з бібліотеки React useState. Усі компоненти побудовані таким чином, щоб залишатися максимально ізольованими, проте водночас інтегруватися в єдину систему.

На загальному рівні архітектура інтерфейсу дозволяє не лише підтримувати наявний функціонал, а й розширювати його в майбутньому без істотної зміни існуючої структури.

Завдяки суворій типізації на основі TypeScript, велика частина логіки захищена від типових помилок на етапі компіляції. Типи Hero, Draft, RecommendationResponse, PlayStyleAdvice тощо використовуються у взаємодії між усіма рівнями клієнтського застосунку – від API запитів до рендерингу в UI компонентах. Це суттєво полегшує розробку та тестування застосунку.

Дизайн клієнтського інтерфейсу застосунку базується на концепції feature based структурування. Такий підхід дозволяє об'єднувати логіку, компоненти, сервіси, хуки та стилі в межах однієї функціональної одиниці. Це значно покращує модульність і масштабованість системи. Замість того, щоб розділяти код за технічними ролями, наприклад, як «components», «services» тощо, у межах feature based організації усі пов'язані сутності, як от компоненти драфту чи вибору сторони, зберігаються спільно.

Керування станом користувацького інтерфейсу реалізовано з урахуванням складності взаємодії між компонентами. У тих випадках, де логіка проста, як от перемикання слайдів у каруселі, використано базовий механізм React useState, який дозволяє зберігати і оновлювати локальний стан у рамках одного компонента. Однак для більш комплексних задач, як керування станом драфту героїв, обрано підхід з використанням useReducer [10] у поєднанні з Context API [11]. Це пояснюється тим, що сторінка драфту має значну кількість взаємозалежних змінних стану таких як список героїв у кожній команді, перевірка на повтори вибору, умовна логіка переходу між фазами вибору. Використання useState [12] у такому випадку призвело б до надмірної кількості окремих станів, ускладненння їх синхронізації та збільшення імовірності помилок.

Редюсер дозволяє централізовано обробляти всі дії, які змінюють стан застосунку, через єдину функцію. Такі дії як «обрати героя», «відмінити вибір героя» не реалізуються окремо в кожному компоненті, а натомість обробляються узгоджено в редюсері. Завдяки цьому будь-яка зміна стану має єдину точку входу та підлягає суворій перевірці умов. А контекст забезпечує доступ до цих дій і поточного стану для будь-якого дочірнього компонента.

Для реалізації асинхронної взаємодії з сервером системи застосовано сучасний підхід на базі React Query [13]. Цей інструмент автоматизує управління життєвим циклом HTTP запитів, кешує відповіді сервера та обробляє повторні спроби у випадку помилок. У межах кожного функціонального модуля створені окремі хуки, що інкапсулюють відповідні запити. Наприклад, запит на отримання списку героїв або рекомендацій щодо вибору героїв, відбувається через окремі,

ізольвані хуки, які також надають інформацію про стан запиту, включно з індикацією завантаження даних або помилкою.

Анімації та візуальні ефекти реалізовано на базі бібліотеки Framer Motion [14], що забезпечує плавну появу елементів при скролі, наведені чи кліках, що покращує візуальне сприйняття та створює відчуття інтерактивності.

Важливим аспектом реалізації користувацького інтерфейсу є правильна організація навігації між окремими сторінками застосунку.

Маршрутизацію побудовано за допомогою бібліотеки React Router DOM [15], яка є однією з найпоширеніших бібліотек для управління маршрутами у React застосунках. Навігація реалізована у вигляді окремого конфігураційного файлу, в якому зафіксовано шляхи до основних сторінок. Завдяки цьому забезпечено просто розширення маршрутизатора у випадку появи нових етапів взаємодії користувача з системою або функціоналу.

Кожна сторінка має власний маршрут, а рендеринг компонента здійснюється через конфігураційне дерево маршрутизатора. Крім того, маршрути підтримують роботу з параметрами URL. Наприклад, при виборі сторони в адресному рядку зберігається інформація про те, яку сторону обрав користувач. Це дозволяє зберігати контекст вибору навіть при перезавантаженні сторінки або при навігації через браузерну історію.

Структура директорій вибудувана відповідно до концепції feature based архітектури. Кожна функціональність застосунку міститься в папці features. Зокрема, ChooseYourSide містить компоненти, що відповідають за логіку вибору сторони, Draft охоплює повний цикл драфту – від відображення героїв до взаємодії з системою рекомендацій, а StyleRecommendation відповідає за кінцеву візуалізацію стилів гри для обраних героїв.

Усередині кожної функціональної папки передбачено окремі підкаталоги або логічні групування, що охоплюють UI компоненти, локальні сервіси, специфічні хуки, а також типи. Такий підхід дозволяє уникнути перемішування загального коду та специфічної логіки для окремої функціональності. Наприклад, у папці Draft зосереджено як компоненти, відповідальні за візуальне відображення сітки героїв,

так і контекст управління станом, сервіси для фільтрації героїв та інтерфейси для обміну даними з бекендом.

Окремо винесено папку shared, яка виконує роль централізованого сховища загальнодоступних елементів, що використовуються в декількох частинах застосунку. Серед таких є, до прикладу, кнопки, індикатори завантаження тощо. Компоненти, що не належать до жодної конкретної сторінки, але при цьому забезпечують уніфікований вигляд та поведінку інтерфейсу, зберігаються саме тут. Це дозволяє уникати дублювання та забезпечує одноманітність у візуальному оформленні усіх частин застосунку.

Окремої уваги заслуговує структура API запитів. Кожен API представлено у вигляді об’єкта з асинхронними функціями, які звертаються до відповідних маршрутів серверної частини системи. Така інкапсуляція дозволяє чітко розмежовувати логіку клієнтської та серверної частин, а також централізовано керувати адресами, параметрами запитів та обробкою відповіді. Всі функції є типізованими, що дозволяє виявляти помилки ще на етапі розробки.

Взаємодія з API інтерфейсом реалізована за допомогою спеціалізованих хуків. Вони побудовані з використанням React Query, що забезпечує ефективне кешування відповідей та автоматичне оновлення даних у разі зміни параметрів запиту. Це дозволяє уникати зайвих запитів до сервера та забезпечує плавну роботу інтерфейсу навіть при нестабільному з’єднанні у користувача.

5.7 Інструкція користувача

З метою забезпечення повноцінної та безперешкодної взаємодії з програмним забезпеченням, даний підрозділ містить покрокову інструкцію щодо користування клієнтською частиною вебзастосунку. Інтерфейс програми спроектовано таким чином, щоб навіть користувач без спеціальної технічної підготовки мав можливість безпечно, інтуїтивно та ефективно виконувати необхідні дії у межах запропонованого функціоналу.

Перед початком роботи з системою користувачу необхідно переконатися у наявності стабільного з’єднання з Інтернетом та актуальної версії браузера, що підтримує сучасні вебтехнології. У разі використання локального запуску, користувач повинен встановити NodeJS, виконати команду npm install для завантаження залежностей проєкту та запустити його за допомогою команди npm run dev. Після цього система буде доступна за локальною адресою на порті 5173.

Після відкриття застосунку, користувач автоматично потрапляє на головну сторінку, яка є вхідною точкою до всієї системи. На цій сторінці користувач має можливість ознайомитися з основним призначенням програми та можливостями, які вона надає. Користувач повинен звернути увагу на вітальний текст та супровідні блоки з описом функціоналу, що розміщені на сторінці з плавною анімацією, яка активується під час прокручування вмісту. У нижній частині екрана розміщується кнопка, натискання на яку здійснює перехід до наступного етапу – вибору команди.

На сторінці вибору команди, яка відкривається після переходу з головної сторінки, користувачу потрібно обрати команду – Сяйво або Пітьму. Вся площина екрану поділена на дві симетричні частини, кожна з яких ілюструє одну зі сторін. У центрі екрана короткочасно з’являється напівпрозоре повідомлення з інструкцією. Для здійснення вибору користувач повинен натиснути на відповідну половину екрану. Цей вибір зберігається у внутрішньому контексті програми. Користувач повинен усвідомлювати, що обрана сторона визначає, за яку команду буде здійснюватися вибір героїв, і впливає на подальші рекомендації.

На сторінці формування драфту зосереджена основна функціональність застосунку: користувачеві потрібно поступово обирати героїв для своєї та ворожої команди, натискаючи на інтерактивні картки героїв у сітках. Щоб обрати героя для команди Сяйва, йому потрібно натиснути ліву кнопку миші, для Пітьми – праву. Щоб відмінити вибір будь-якого героя, достатньо клацнути по його портрету у верхній частині екрану, де розміщено склади команд Сяйва та Пітьми. Одного і того самого героя неможливо обрати двічі незалежно від команди, до складу якої він вже входить (рисунок 5.10).

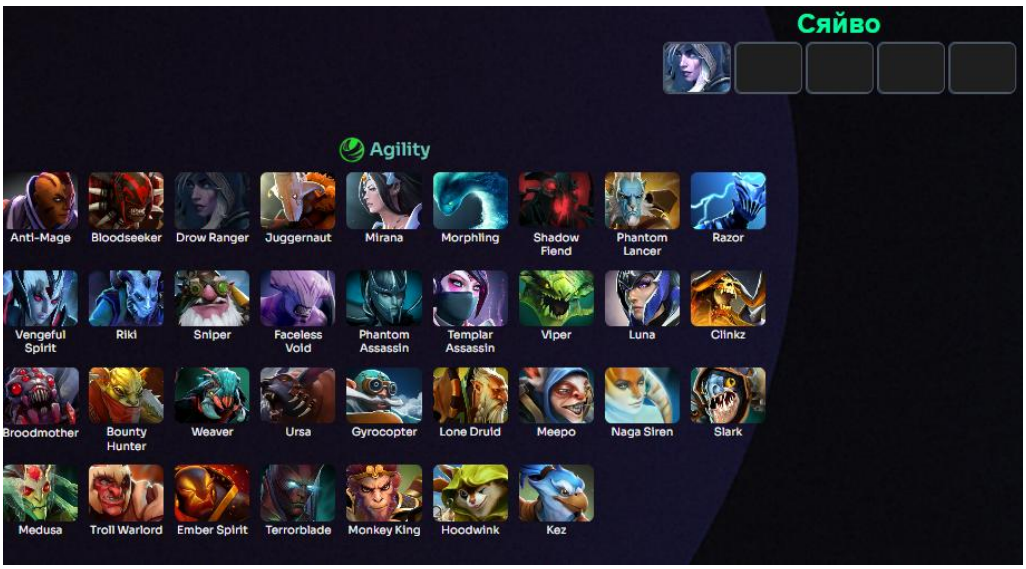

Рисунок 5.10 – Обрання героя блокує його до повторного вибору

Кожного разу, коли користувач обиратиме героя для будь-якої з команд, система в той же час даватиме йому рекомендацію у вигляді ранжованого списку рекомендованих героїв. Рамки портретів героїв, які були рекомендовані системою, будуть підсвічуватися яскравим зеленим кольором. Це зроблено з метою допомогти користувачу швидше зорієнтуватися серед великої кількості доступних героїв та зробити вибір, якщо він бажає. Користувач, орієнтуючись на рекомендації, може адаптувати свою стратегію під конкретну ігрову ситуацію, зберігаючи гнучкість та контроль над формуванням команди. Приклад того, як виглядає рекомендація показано на рисунку 5.11.

Рисунок 5.11 – Приклад рекомендації вибору героїв

Після того, як склад команд був сформований, користувач має опціональну змогу отримати рекомендації стилю гри для кожного героя в його команді, натиснувши кнопку внизу сторінки. Це дає йому змогу бачити, як повинна грати його команда, якщо він тренер або підказувати своїм випадковим союзникам, як вони повинні діяти. Можливість переходу до фази рекомендації стилів гри показано на рисунку 5.12.

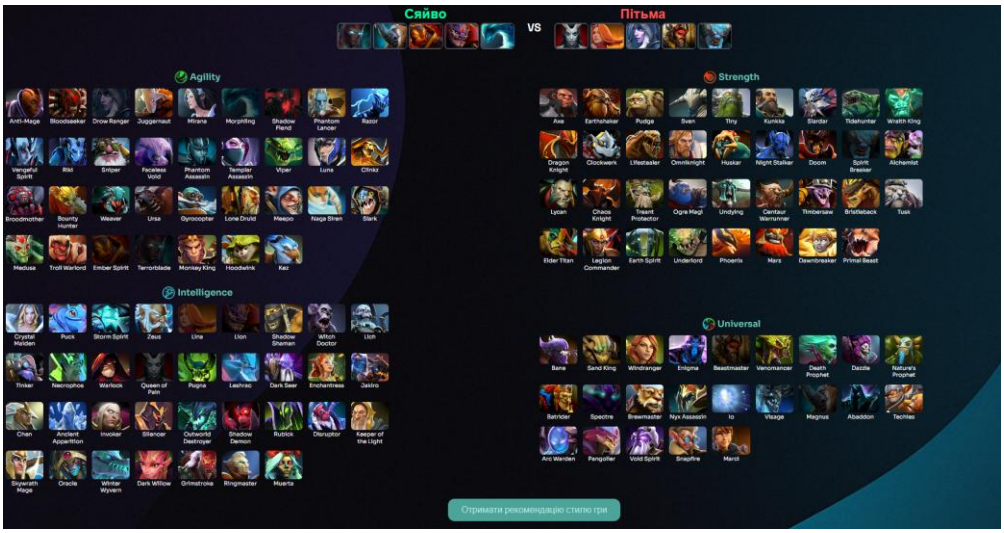

Рисунок 5.12 – Можливість переходу на сторінку рекомендацій стилів гри

Якщо користувач натисне кнопку «Отримати рекомендації стилю гри», то буде спрямований на сторінку з рекомендаціями, де для кожного героя буде показано бажаний для нього стиль гри. Це показано на рисунку 5.13.

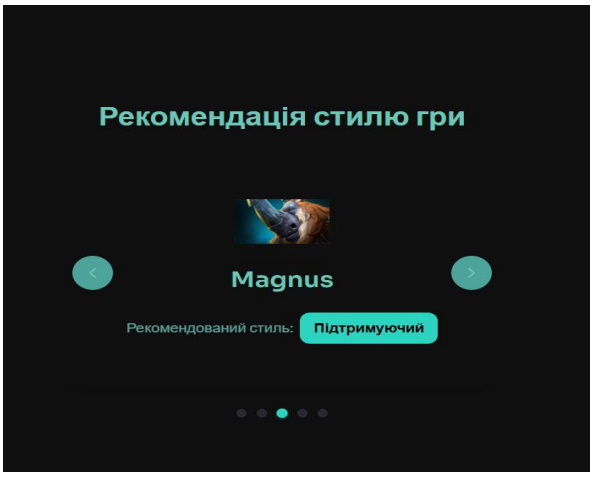

Рисунок 5.13 – Рекомендація стилю гри для героя

Щоб переглянути рекомендації для усіх героїв, користувач може тиснути на стрілки «Вліво» та «Вправо».

Висновок до розділу 5

У даному розділі було проведено комплексний аналіз та опис архітектурних, функціональних і технічних рішень, що лягли в основу створення програмного забезпечення. Представлено функціональну модель системи, побудовані діаграми класів, варіантів використання, діяльності тощо. Описано специфікації функцій, що реалізують ключову бізнес логіку та структуру даних. Значну увагу приділено розробці користувацького інтерфейсу. Створено інструкцію користувача.

Розроблена система базується на сучасних підходах до побудови веборієнтованих додатків із чітким розподіленням рівнів клієнта, сервера та сховища даних. Реалізована структура дозволяє ефективно організовувати взаємодію між компонентами, мінімізуючи ризики виникнення помилок та спрощуючи процес розгортання в різних середовищах.

Окрему увагу приділено взаємодії із системою на рівні інтерфейсу користувача, що реалізований згідно з принципами адаптивного дизайну та з урахуванням вимог до швидкодії і зручності.

6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Змістовна постановка задачі

У грі Dota 2 під час драфту обидві команди по черзі обирають по одному герою, формуючи свій склад із п’яти унікальних персонажів. Кожен герой виконує певну роль у грі та має унікальні характеристики, які по-різному взаємодіють з героями союзників та суперників.

Успішність команди часто залежить від якості драфту – того, наскільки грамотно підібрані герої в команді, та наскільки ефективно вони протидіють складу противника. Саме на цьому етапі виникає потреба в інструменті підтримки прийняття рішень, який міг би рекомендувати гравцю, якого героя краще обрати в тій чи іншій ситуації, виходячи з історичних даних.

Задача полягає в тому, щоб створити модель, яка здатна враховувати поточний стан драфту (вибраних героїв, етап вибору, наявні ролі) та на його основі визначати, наскільки перспективним є вибір того чи іншого героя-кандидата для перемоги в матчі.

6.2 Математична постановка задачі

6.2.1 Рекомендація вибору героїв

Нехай H – множина усіх героїв, $\mathrm{A_t}$ – множина героїв, які вже обрані союзною командою на кроці t, E_t – множина ворожих героїв, обраних ворожою командою на кроці t, M_t – множина вже обраних героїв. $\mathbb{R}^{276}$ – простір вхідних ознак драфту, $\mathbb{R}^{126}$ – простір вихідних логіт значень, x_t – вектор вхідних ознак потужністю 276 на кожній фазі t драфту, z_t – логіт вектор на кроці t, який містить 126 чисел, кожне з яких є числовою оцінкою моделі для відповідного героя з множини H.

$$H = \{h_1, h_2, \dots, h_{126}\},$$

$$\mathrm{A_t} \subseteq \mathrm{H},$$

$$E_\mathrm{t} \subseteq \mathrm{H},$$

$$M_t = A_t \cup E_t,$$
$$1 \leq t \leq 10.$$

Необхідно побудувати функцію $f_\theta : \mathbb{R}^{276} \Rightarrow \mathbb{R}^{126}$, що вектору вхідних ознак на кожній фазі драфту $x_t \in \mathbb{R}^{276}$ ставить у відповідність логіт вектор $z_t \in \mathbb{R}^{126}$.

6.2.2 Рекомендація стилю гри

Після завершення матчу для кожного героя доступні числові показники, які характеризують ефективність його ігрової активності. Нижче перелічено ці показники:

– співвідношення результативних дій до кількості вибувань з гри;

– середня кількість результативних дій за хвилину;

– загальна кількість допоміжних дій союзникам;

– середня кількість отриманого золота за хвилину;

– середній досвід за хвилину;

– обсяг завданої шкоди іншим героям;

– сумарна шкода, завдана будівлям;

– кількість відновленого здоров’я;

– кількість нейтралізованих кріпів.

Потрібно для кожного героя віднести його стиль гри до одного з трьох класів (6.1).

$$S = \{Aggresive, Balanced, Supportive\}. \quad (6.1)$$

На основі схожості його ігрових характеристик до інших гравців.

Задача формалізується як задача класифікації у 9 вимірному просторі і представлена формулою (6.2).

$$c: \mathbb{R}^9 \Rightarrow \{0, 1, 2\}. \quad (6.2)$$

Де кожному вектору ознак героя s приписується кластер, а кожному кластеру надається інтерпретація стилю гри на основі аналізу центроїдів.

6.3 Обгрунтування методу розв’язання

6.3.1 Модель рекомендації вибору героїв

На початковому етапі дослідження було сформовано два потенційні набори ознак. Перший містив інформацію, яка доступна під час драфту, а саме двійкові індикатори вже обраних героїв, їхні ролі та фазові показники виграшу. Другий включав післяматчеві статистики, зокрема середній приріст золота та досвіду за хвилину, завдану шкоду по героях і вежах тощо. Результат кореляційного аналізу, що зображений на рисунку 6.1, засвідчив істотний зв’язок згаданих післяматчевих метрик із кінцевим результатом зустрічі. Наприклад, відносна шкода по ворожих будівлях демонструє коефіцієнт кореляції приблизно 0.7, а аналогічний показник супротивника – близько 0.71. Проте жодна з таких величин не є відомою на етапі вибору героя, отже, їхнє використання призвело б до data leakage, коли під час навчання алгоритм підглядає інформацію з майбутнього, і таким чином отримує перевагу, що є недоступна в реальному використанні. Щоб уникнути некоректної оцінки якості та помилок узгодженості, усі післяігрові метрики було вилучено зі схеми ознак, а подальший аналіз відбувався виключно на основі тих даних, які були доступні у момент драфту.

Після такого очищення даних вектор ознак набуває розмірності 276. Дві третини вектору становлять one hot індикатори героїв. Коли підібрано лише десять персонажів, то активними залишаються не більше десяти координат. Решта координат містить значення у межах від 0 до 1, які відображають середній відсоток перемог по фазах вибору героя та його бінарну приналежність до ролей. Вони доповнені лічильниками вже обраних ядрових та підтримуючих героїв-союзників. Така гетерогенна структура простору накладає низку вимог на модель. По-перше, вона повинна коректно працювати з високою спарністю, не втрачаючи інформації, закладеної в рідкісних одиничних бітових компонентах.

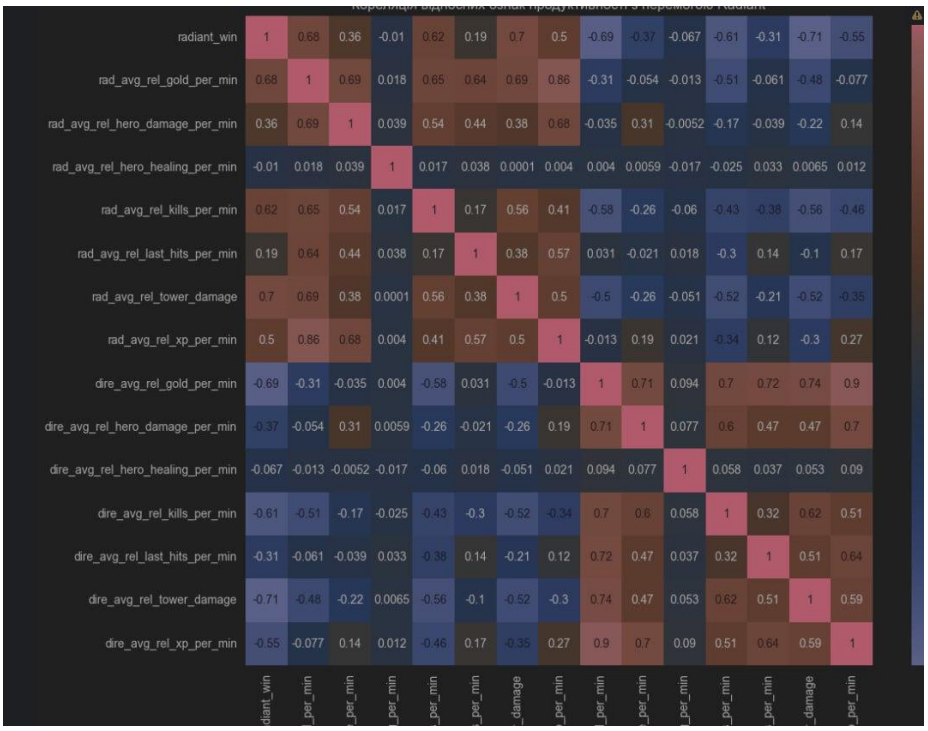

Рисунок 6.1 – Результат кореляційного аналізу відносних ігрових метрик із перемогою команди Сяйва

По-друге, їй необхідно відтворювати нелінійні залежності, адже ефект синергії або контр піку проявляється лише у взаємодії кількох героїв.

6.3.2 Число нейронів у прихованих шарах

Орієнтовну ширину першого шару було визначено, виходячи з емпіричного судження: перший прихований шар доцільно брати в 1-2 рази ширшим за розмірність вхідного вектора. Вибір 512, а не 552 нейронів, зроблений через апаратні міркування. Число 512 є точним степенем двійки і виконання множення матриць розміром кратним степеням двійки проходить без додаткових витрат на довирівнювання блоків.

Другий шар звужено до 256 нейронів через те, що таке звуження змушує інформацію, що пройшла крізь перший шар, ущільнюватись у вужчому просторі. Такий підхід дозволяє нейронній мережі зосереджуватись на найбільш значущих характеристиках вхідних даних, відкидаючи другорядні або надлишкові компоненти, що покращує узагальнюючу здатність моделі.

6.3.3 Модель рекомендації стилю гри

Заздалегідь відсутні ручні лейбли стилю гри для кожного героя, тому класифікацію неможливо виконати супервізовано. Проте ігрові показники є зручними ознаками для порівняння й агрегації поведінки гравців. Серед численних алгоритмів групування обрано алгоритм k-means, оскільки він ефективно масштабується до десятків тисяч матчів, забезпечує розбиття простору на неперетинні кластери з мінімізацією внутрішньокластерної дисперсії та його центроїди легко інтерпретуються. Якщо герой зазвичай має високі значення KDA, шкоди й фарму, то йому властивий агресивний стиль, а високі показники лікування та підмог – підтримуючому.

З технічної точки зору k-means є оптимальним, тому що його обчислювальна складність зростає лінійно з кількістю героїв та ознак.

6.4 Опис методу розв’язання

6.4.1 Опис методу розв’язання для моделі рекомендації героїв

З урахуванням маскування вже вибраних героїв, імовірність вибору кожного кандидата обчислюється згідно з формулою (6.3). Для кожного героя, який ще не обраний, модель визначає логіт, після чого ці значення перетворюються на імовірності так, що недоступні герої автоматично отримують нульову імовірність. Остаточний розподіл імовірностей формується лише серед тих героїв, яких ще можна обрати на даному кроці драфту.

$$\Pr(h|x_t) = \frac{exp_{z_t,h}}{\sum_{j \notin M_t} exp_{z_t,h}}, \tag{6.3}$$

де $z_{t,h}$ – компонент логіт вектора (6.4).

$$z_{t,i} = f_\theta(x_t)_i m_{t,i}, \tag{6.4}$$

де $m_{t,i}$ – бінарний індикатор, який визначає, чи герой вже був обраний на кроці t (6.5).

$$m_{t,i} = \begin{cases} 0, & i \in M_{\mathrm{t}} \\ 1, & i \notin M_{\mathrm{t}} \end{cases}. \quad (6.5)$$

Навчання моделі здійснюється шляхом знаходження параметрів, які мінімізують усереднену крос ентропію (6.6).

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum logPr(\, h_{true,t} \mid x_t). \quad (6.6)$$

Вектор вхідних ознак має вигляд як у формулі (6.7).

$$x = \left[team_{ally}, team_{enemy}, \bar{\phi}_{ally}, \bar{\phi}_{enemy}, c, s\right]^T \in \mathbb{R}^{276}, \quad (6.7)$$

де $team_{ally}$ і $team_{enemy}$ – бінарні one hot вектори потужністю 126, які кодують обраних союзників та ворогів. У кожному з цих векторів одиниця стоїть на позиції, що відповідає певному герою, якщо він уже обраний відповідною командою, а решта компонент вектора рівні нулю.

$\bar{\phi}_{ally}$ (6.9) і $\bar{\phi}_{enemy}$ (6.10) – усереднені статистичні профілі союзної та ворожої команд, кожен з яких є вектором потужністю 11 (6.8).

$$\bar{\phi} = \left[w^{(1)}, w^{(2)}, w^{(pro)}, r_{Carry}, r_{Support}\right], \quad (6.8)$$

де $w^{(1)}$, $w^{(2)}$ та $w^{(pro)}$ – середній відсоток перемог на першій фазі драфту, середній відсоток перемог на другій фазі драфту та середній відсоток перемог у професійних матчах відповідно.

c та s – скалярні ознаки, які визначають кількість героїв із ролями Carry та Support у складі союзної команди на поточній фазі драфту.

$$\bar{\phi}_{ally} = \frac{1}{|A_t|} \sum_{h \in A_t} \phi(h), \tag{6.9}$$

$$\bar{\phi}_{ally} = \frac{1}{|E_t|} \sum_{h \in E_t} \phi(h), \tag{6.10}$$

Якщо жодного героя в команді не обрано, то повертається нуль вектор.

6.4.2 Опис методу розв'язання для моделі рекомендації стилю гри

Всі ознаки вектора s нормалізуються, щоб запобігти домінуванню показників із великою дисперсією та до вже нормалізованих даних застосовується алгоритм k-means із такими параметрами: k = 3 та 10 рестартів стійкості, а також ініціалізацією k-means++ для уникнення потрапляння в локальні мінімуми.

Герою у конкретному матчі надається стиль відповідно до найближчого центроїда

Висновок до розділу 6

У даному розділі було розглянуто математичний апарат для побудови системи рекомендацій. Поетапно проведено постановку задачі як для вибору героя під час драфту, так і для визначення стилю гри на основі ігрових показників.

Показано, що підхід, заснований на використанні багатошарового перцептрону, є найкращим за наявних умов. Вхідні ознаки даних було детально проаналізовано й підібрано таким чином, щоб максимально відобразити як індивідуальні характеристики вибраних героїв, так і специфіку командних поєднань, запобігаючи витоку даних. У модулі визначення стилю гри застосовано кластеризацію на основі ігрових метрик, що дозволяє автоматично виділяти основні моделі поведінки гравців без ручної розмітки. Вибір саме алгоритму k-means забезпечує як достатню інтерпретованість результату, так і можливість обробки великих масивів даних із прийнятною обчислювальною складністю.

ВИСНОВКИ

У дипломному проєкті було розглянуто та реалізовано інтелектуальну систему підтримки прийняття рішень, орієнтовану на стратегічний етап формування команди в багатокористувацькому ігровому середовищі. Проведено аналіз предметної області, а також запропоновано підхід до формування рекомендацій на основі поєднання історичних даних та контексту гри.

У результаті виконаного дослідження було досягнуто поставленої мети – покращення процесу прийняття рішень користувачем під час вибору ігрових персонажів шляхом розробки системи, яка формує як рекомендації щодо доцільного вибору героїв, так і припущення щодо очікуваного стилю гри кожного з учасників команди.

Оцінка отриманих результатів засвідчує відповідність проєкту поставленій меті. Реалізована система демонструє стабільну роботу, має зручний користувацький інтерфейс, підтримує гнучку логіку рекомендацій та дозволяє масштабування для подальших досліджень.

У процесі розробки також було виявлено важливу проблему, пов’язану з формуванням вектора вхідних ознак. На одному з етапів до моделі було включено ігрові метрики, пов’язані безпосередньо із самим матчем, що призвело до витоку даних. Як наслідок, система почала непропорційно часто рекомендувати окремих героїв, таких як Jakiro та Lion, навіть у випадках, де це було стратегічно недоцільно. Після виявлення цієї помилки було проведено аналіз джерела відхилення, переформульовано структуру ознак та скориговано логіку побудови тренувального набору. Це дозволило усунути проблему та забезпечити більшу точність рекомендацій.

Результати роботи можуть бути використані в низці галузей, зокрема у створенні рекомендаційних ігрових сервісів. Завдяки модульній структурі система може бути розширена для аналізу інших ігор, що потребують стратегічного планування на основі вхідних даних у реальному часі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Esports Charts. URL: https://escharts.com/top-games?year=all-time (дата звернення: 15.04.2025).
2. Fifa Publications. URL: https://publications.fifa.com/en/annual-report-2022/finances/2019-2022-cycle-in-review/2019-2022-investments-expenses/ (дата звернення: 15.04.2025).
3. The News Market. URL: https://www.thenewsmarket.com/news/ioc-s-first-olympic-esports-week-brings-thrilling-live-competition--historic-firsts-and-communities-/s/8ae712be-c691-44c6-8065-6d41b82a9ee1 (дата звернення: 15.04.2025).
4. Dota Plus. URL: https://www.dota2.com/plus (дата звернення: 15.04.2025).
5. AWS. URL: https://aws.amazon.com/what-is/api/ (дата звернення: 15.04.2025).
6. JavaRush. URL: https://javarush.com/ua/quests/lectures/ua.questservlets.levellecture05 (дата звернення: 20.04.2025).
7. OpenDota API. URL: https://docs.opendota.com/ (дата звернення: 20.04.2025).
8. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview (дата звернення: 20.04.2025).
9. Postman Blog. URL: https://blog.postman.com/rest-api-examples/ (дата звернення: 20.04.2025).
10. React Dev. URL: https://react.dev/reference/react/useReducer (дата звернення: 15.05.2025).
11. React Dev. URL: https://react.dev/reference/react/useContext (дата звернення: 15.05.2025).
12. React Dev. URL: https://react.dev/reference/react/useState (дата звернення: 15.05.2025).

13. TanStack Query. URL: https://tanstack.com/query/latest/docs/framework/react/overview (дата звернення: 15.05.2025).
14. Motion. URL: https://motion.dev/ (дата звернення: 15.05.2025).
15. React Router. URL: https://reactrouter.com/ (дата звернення: 15.05.2025).