Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Довгополюк В. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н., доцент каф. АІІТ
Богач І. В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«05» травня 2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Довгополюку Владиславу Олександровичу

1. Тема роботи: Розробка онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.
керівник роботи: Крилик Людмила Вікторівна, к.т.н., доц.
затверджені наказом вищого навчального закладу «20» березня 2025 року № 97
2. Термін подання студентом роботи 30 травня 2025
3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована; кількість підтримуваних платформ не менше 2; інтерфейс користувача має бути інтуїтивно зрозумілий, масштабована та продуктивна інфраструктура, темплейти для розгортання інфраструктури в хмарі.
4. Зміст текстової частини (перелік питань, які потрібно розробити): вступ; обґрунтування доцільності розробки онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; проектування онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; програмна реалізація онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; висновки; список використаних джерел; додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): загальна структурна схема функціонування системи; UML-діаграма класів бекенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; UML-діаграма класів фронтенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; ER-модель для бази даних; схема алгоритму роботи онлайн-платформи; архітектурна діаграма хмарного середовища онлайн-платформи для вибору стратегії гонки Формула-1; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Крилик Л. В., к.т.н., доц. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вступ. Обґрунтування доцільності розробки онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень	05.05.25	07.05.25	
2	Проектування онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень	08.05.25	11.05.25	
3	Програмна реалізація онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень	12.05.25	15.05.25	
4	Висновки та аналіз отриманих результатів	16.05.25	26.05.25	
5	Оформлення додатків	27.05.25	29.05.25	
6	Розробка інструкції користувача	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ **Довгополюк В. О.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Крилик Л. В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 100 сторінок формату А4, які включають 21 рисунок і 7 таблиць. У списку використаних джерел зазначено 33 найменування.

Метою роботи є розширення функціональних можливостей онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

У загальній частині на основі аналізу сучасних технологій хмарних обчислень та огляду існуючих платформ обґрунтовано доцільність розробки з модульною архітектурою, інтегрованою з AWS. Розроблено схеми алгоритмів, ER-модель бази даних та UML-діаграми класів клієнтської та серверної частин.

Онлайн-платформу реалізовано з використанням сучасних WEB-технологій: фронтенд – TypeScript з React, бекенд – Python, FastAPI, база даних – PostgreSQL, інфраструктура розгорнута через Terraform в AWS.

Результати тестування підтвердили працездатність, стабільність, високу продуктивність. Результати тестування розробки вказують на те, що основні функції онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень реалізовано.

Ключові слова: онлайн-платформа, Формула-1, хмарні обчислення, модульна архітектура, AWS.

ABSTRACT

The bachelor's thesis consists of 100 A4 pages, including 21 pictures and 7 tables. The list of references includes 33 titles.

The purpose of the work is to expand the functionality of an online platform for choosing a Formula 1 race strategy based on cloud computing.

In the general part, based on the analysis of modern cloud computing technologies and a review of existing platforms, the expediency of development with a modular architecture integrated with AWS is substantiated. Algorithmic schemes, an ER model of the database and UML diagrams of the client and server parts are developed.

The online platform was implemented using modern WEB technologies: frontend – TypeScript with React, backend – Python, FastAPI, database – PostgreSQL, infrastructure deployed via Terraform in AWS.

The test results confirmed the operability, stability, and high performance. The results of the development testing indicate that the main functions of the online platform for choosing a Formula 1 race strategy based on cloud computing have been implemented.

Keywords: online platform, Formula 1, cloud computing, modular architecture, AWS.

ЗМІСТ

ВСТУП 4
1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ 7
ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ.....7
1.1 Огляд технологій хмарних обчислень та їх застосування в автоспорті..... 7
1.2 Огляд платформ-аналогів 12
1.3 Формулювання вимог та постановка задачі 15
1.4 Висновок 17
2 ПРОЕКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ.....18
2.1 Розробка структури онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень..... 18
2.2 Розробка UML-діаграми класів..... 23
2.3 Розробка ER-моделі бази даних..... 30
2.4 Розробка схеми алгоритму роботи онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень 33
2.5 Обґрунтування вибору хмарного провайдера для реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень 38
2.6 Розробка структури хмарної інфраструктури..... 39
2.7 Висновок 41
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ..... 43
3.1 Обґрунтування вибору мови реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень 43
3.2 Обґрунтування вибору мови керування базою даних 45

3.3 Обґрунтування вибору середовища розробки 48
3.4 Розробка клієнтської частин 49
3.5 Розробка серверної частин 51
3.6 Розробка темплейтів хмарної інфраструктури 56
3.7 Тестування роботи програми та аналіз результатів 59
3.8 Висновок 67
ВИСНОВКИ 69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71
ДОДАТКИ 75
ДОДАТОК А (обов’язковий) Протокол перевірки кваліфікаційної роботи 76
ДОДАТОК Б (обов’язковий) Лістинг програми 77
ДОДАТОК В (обов’язковий) Графічна частина 83
ДОДАТОК Г (довідниковий) Інструкція користувача 94

ВСТУП

Актуальність досліджень.

Формула-1 давно є однією з найпрестижніших та інноваційних сфер автоспорту, де технологічний прогрес і стратегічні рішення безпосередньо впливають на результат гонки. Зі зростанням обсягів доступних даних, вдосконаленням телеметричних систем та появою потужних хмарних обчислювальних ресурсів, оптимізація стратегій для перегонів набуває нових можливостей. Актуальність розробки онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень зумовлена кількома ключовими аспектами:

- Збільшення обсягів даних та складності аналізу: сучасні боліди генерують величезну кількість інформації: дані телеметрії, стан двигуна, шин і гальм, витрату пального та інші характеристики. Крім цього, команди отримують додаткові дані з різних джерел, таких як метеорологічні сервіси для прогнозу погоди, стан траси та позицій суперників. Без ефективних інструментів обробки та аналізу цих даних стратегам команд стає складно правильно і швидко ухвалювати оптимальні тактичні рішення.

- Індивідуалізація стратегій: кожна команда та пілот мають унікальні характеристики – стиль пілотування, особливості боліда, уподобання щодо шин, а також власний підхід до управління ризиками. Онлайн-платформа, інтегрована з хмарними обчисленнями, допоможе формувати індивідуальні рекомендації з урахуванням реальних умов гонки.

- Оперативність прийняття рішень: Формула-1 – спорт, де рішення приймають за лічені секунди. Застосування хмарних обчислень дає змогу виконувати складні розрахунки в режимі реального часу, миттєво реагувати на зміни ситуації на трасі та випереджати суперників.

- Конкурентна перевага: команди, які здатні ефективно застосовувати сучасні технології для обробки та інтерпретації даних, отримують суттєву перевагу перед суперниками. Такі рішення допомагають не лише оптимізувати стратегії піт-стопів чи вибір шин, але й сприяти сталому покращенню результатів.

Таким чином, розробка онлайн-платформи для вибору стратегії гонки Формули-1 на основі хмарних обчислень» є надзвичайно актуальною, оскільки вона сприятиме гнучкому та оперативному прийняттю рішень, підвищенню конкурентоспроможності команд та більш глибокому розумінню складних процесів, що відбуваються під час перегонів. Її реалізація може зробити вагомий внесок у розвиток спортивної аналітики та забезпечення більшої конкуренції.

Метою дослідження є розширення функціональних можливостей онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

Об’єктом дослідження процес прийняття стратегічних рішень у перегонах.

Предметом дослідження є методи та програмні засоби для реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

Задачі дослідження:

1. Обґрунтувати доцільність розробки онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.
2. Спроектувати архітектуру онлайн-платформи для обробки великих обсягів даних та виконання складних обчислень.
3. Розробити програмний модуль для аналізу та моделювання стратегій гонки з використанням технологій хмарних обчислень.
4. Виконати тестування та провести аналіз отриманих результатів.
5. Розробити інструкцію користувача з експлуатації платформи.

Апробація роботи.

Результати досліджень було апробовано на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ) м. Вінниці у 2025 р. [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1] та подано заявку на реєстрацію авторського права на твір у формі комп'ютерної програми – «Онлайн-платформа для вибору стратегії гонки Формула-1 на основі хмарних обчислень», номер заявки С202503799 від 28.04.2025 р.

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ

1.1 Огляд технологій хмарних обчислень та їх застосування в автоспорті

Хмарні обчислення стали основною складовою цифрової епохи, що радикально змінила підхід до управління даними та обчислювальними ресурсами. Їхня основна особливість полягає в можливості обробляти дані в інтернеті, це дозволяє користувачам отримувати доступ до необхідної інформації та потужності у будь-який час і з будь-якої точки світу. Це знімає необхідність у створенні та підтримці дорогої власної інфраструктури, такої як сервери чи сховища даних. Завдяки цьому можна зосередитися на своїх основних завданнях, не витрачаючи значних ресурсів на технічне забезпечення.

Саме це дозволяє забезпечити високий рівень гнучкості, що дозволяє швидко адаптуватися до змін у робочих процесах та швидко масштабувати ресурси залежно від поточних потреб. Наприклад, можна реалізувати автоматичне масштабування ресурсів, коли під час пікових навантажень обчислювальна потужність збільшується для стабільної роботи системи, або зменшується при меншій потребі у використанні для уникнення зайвих витрат. Крім того, використання хмарних сервісів значно знижує капітальні витрати, адже замість придбання та обслуговування фізичного обладнання, оплата здійснюється за принципом «pay-as-you-go», що означає «плати лише за використані ресурси» [2].

Формула-1 є найвищою категорією автоперегонів на відкритих колесах, що організовується Міжнародною автомобільною федерацією (FIA). Цей вид спорту відомий своїми високими швидкостями, передовими технологіями та складними інженерними рішеннями. Кожна гонка, або «Гран-прі», проходить на спеціально

підготовлених трасах або міських вулицях у різних країнах світу, збираючи мільйони глядачів як на трибунах, так і перед екранами. Формула-1 є не лише змаганням між гонщиками, яким доводиться під час кожної гонки балансувати на межі життя і смерті, розвиваючи швидкість до 350 км/год, але й полем битви для інженерів, технологій та стратегій, де кожна деталь може визначити перемогу або поразку.

Кожен болід оснащений десятками датчиків, які збирають інформацію про швидкість, витрату палива, температуру гуми, стан двигуна та аеродинамічні показники. Протягом гонки ці дані надходять у реальному часі до інженерів команди, які аналізують їх для прийняття стратегічних рішень. Саме тут на допомогу приходять хмарні обчислення, які дозволяють обробляти ці дані швидко та ефективно. Хмарні технології в Формулі-1 застосовуються для кількох ключових завдань. По-перше, це аналіз телеметричних даних у реальному часі. Дані, отримані з болідів, передаються на сервери в хмарі, де вони обробляються потужними алгоритмами. Це дозволяє командам прогнозувати зношування гуми, планувати піт-стопи та швидко реагувати на зміни погодних умов. Наприклад, компанія AWS, яка є офіційним партнером Формули-1, надає інструменти для аналізу даних, які дозволяють створювати графіки продуктивності болідів та передбачати оптимальні стратегії.

По-друге, хмарні обчислення використовуються для моделювання та симуляцій [3]. Команди Формули-1 активно використовують суперкомп'ютери і хмарні платформи для створення цифрових моделей болідів та трас. Це допомагає випробовувати різні аеродинамічні рішення, моделювати поведінку автомобіля на різних типах покриття та розробляти оптимальні налаштування для кожної траси.

Крім технічних аспектів, хмарні технології також сприяють взаємодії з уболівальниками. Формула-1 активно використовує хмарні сервіси для трансляції перегонів, створення інтерактивного контенту та аналізу аудиторії.

Серед провідних постачальників хмарних послуг виділяються Amazon Web Services, AWS, Microsoft Azure та Google Cloud Platform, GCP. Кожний з них має свої особливості, переваги та недоліки, але виконують свою роботу найкраще.

AWS є однією з провідних хмарних платформ [4]. Він пропонує понад 200 послуг, включаючи обчислювальну потужність, зберігання даних, бази даних та інструменти машинного навчання. AWS має розгалужену глобальну інфраструктуру з 105 центрами обробки даних у 33 регіонах. Перевагами AWS є висока надійність, масштабованість та різноманітність сервісів. Однак складність деяких послуг та ціноутворення можуть стати проблемою для новачків (рис. 1.1).

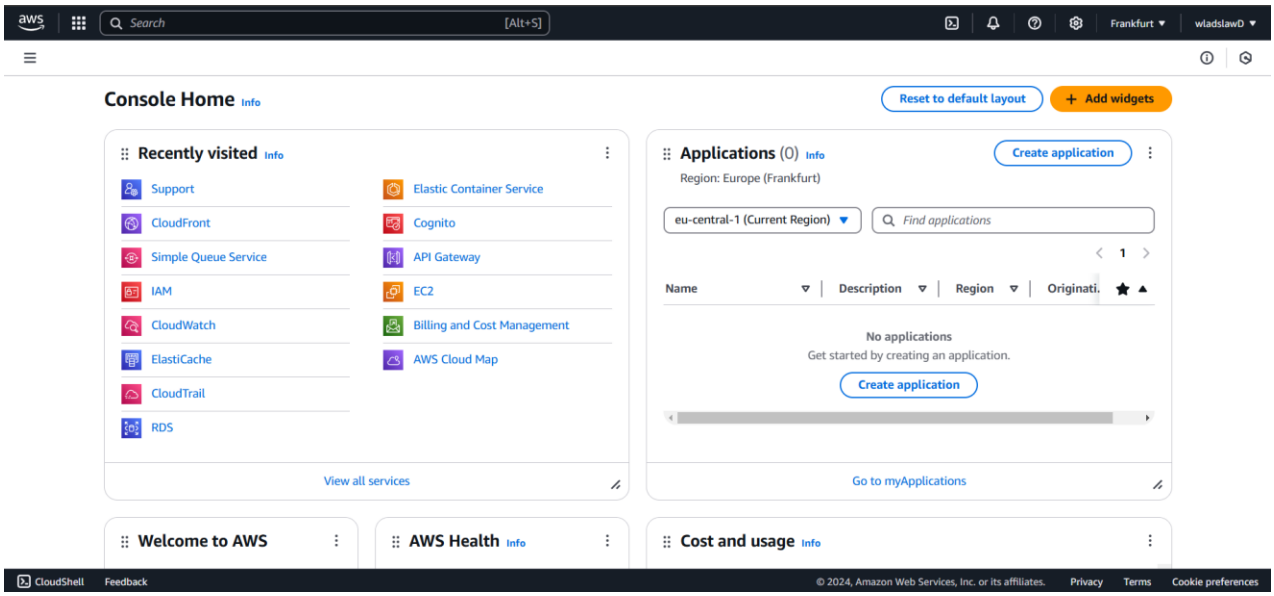


Рисунок 1.1 – Загальний вигляд інтерфейсного вікна платформи AWS

Запущений в 2010 році, Azure швидко став однією з провідних хмарних платформ, особливо популярною серед підприємств, які вже використовують продукти Microsoft. Azure пропонує також понад 200 сервісів, включаючи віртуальні машини, бази даних, аналітику та інструменти для розробки. Інтеграція з іншими продуктами Microsoft, такими як Windows Server, Active Directory та Office 365, є значною перевагою [4]. Azure має понад 60 регіонів по всьому світу,

що забезпечує низьку затримку та високу доступність. Проте, як і AWS, Azure може бути складним для використання новачків (рис. 1.2).

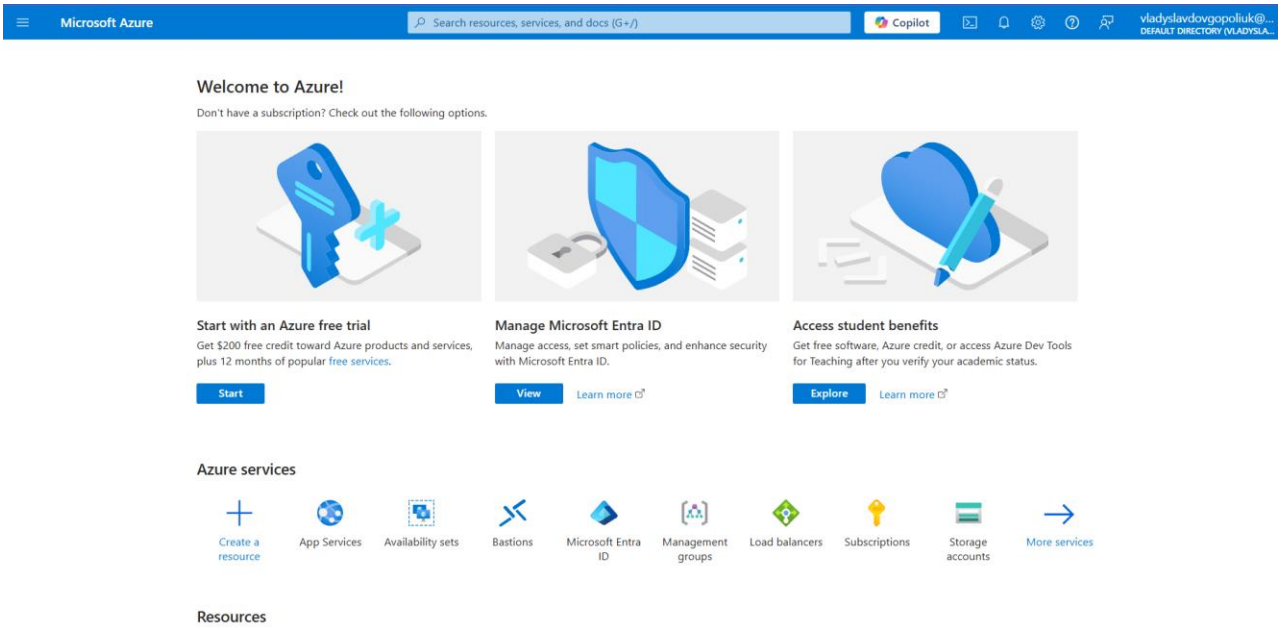


Рисунок 1.2 – Загальний вигляд інтерфейсного вікна платформи Microsoft Azure

У 2008 році була запущена Google Cloud Platform, яка відома своїми передовими рішеннями в галузі великих даних, машинного навчання та штучного інтелекту [4]. GCP пропонує велику кількість сервісів, включаючи обчислювальні потужності, зберігання даних, бази даних та інструменти для аналітики. Її глобальна мережа забезпечує високу продуктивність та низьку затримку. Перевагами є інноваційні сервіси, конкурентоспроможні ціни та глибока інтеграція з іншими продуктами Google. Однак GCP має меншу кількість регіонів порівняно з AWS та Azure, що може впливати на доступність у деяких регіонах (рис. 1.3). В таблиці 1.1 подано порівняльна характеристика популярних хмарних платформ.

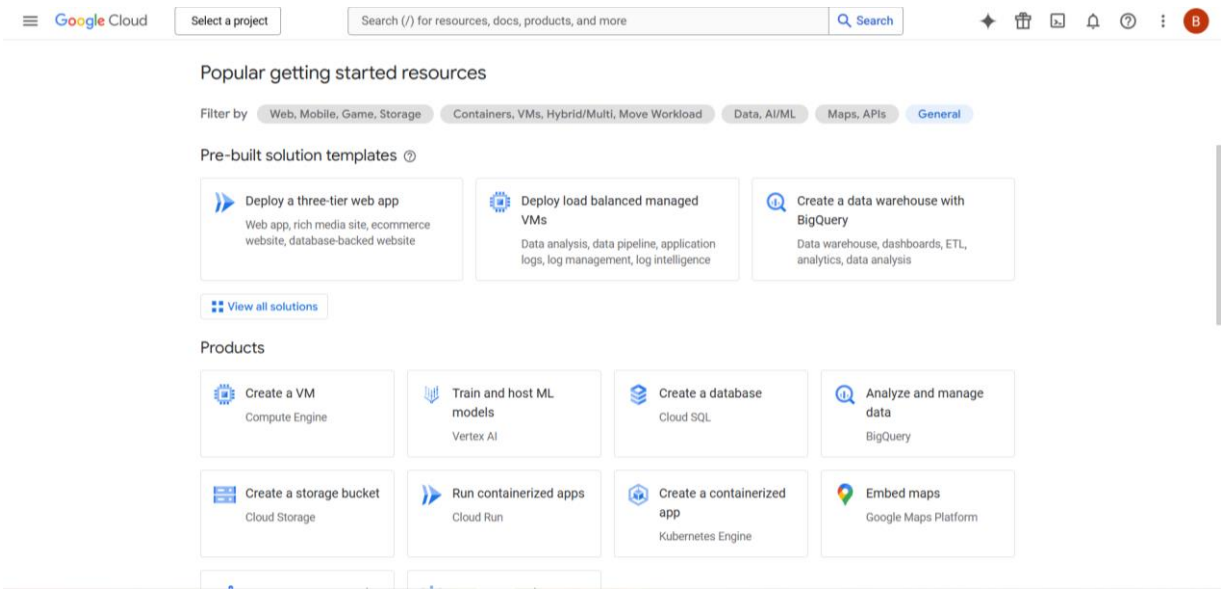


Рисунок 1.3 – Загальний вигляд інтерфейсного вікна платформи Google Cloud Platform

Таблиця 1.1 – Порівняльна характеристика популярних хмарних платформ

Платформа	Переваги	Недоліки
Amazon Web Services	Широка глобальна інфраструктура, різноманітність сервісів, висока надійність і масштабованість	Складність у налаштуванні для новачків, заплутане ціноутворення
Microsoft Azure	Інтеграція з продуктами Microsoft, велика кількість регіонів, підтримка підприємств	Заплутане ціноутворення, складність використання для нових користувачів
Google Cloud Platform	Інноваційні інструменти для аналітики та машинного навчання, конкурентоспроможні ціни	Менша кількість регіонів порівняно з конкурентами, деякі сервіси мають обмежений функціонал

Таким чином, хмарні обчислення відкривають нові можливості для команд автоспорту, дозволяючи ефективно обробляти великі обсяги даних, приймати обґрунтовані рішення та підвищувати конкурентоспроможність на трасі.

1.2 Огляд платформ-аналогів

У світі сучасного автоспорту, стратегічне планування та оптимізація процесів під час гонок є критично важливими аспектами. Команди Формули-1, інженери та навіть сімрейсери використовують різноманітні онлайн-інструменти для збору, обробки й аналізу даних. Такі платформи допомагають покращувати продуктивність болідів, прогнозувати результати та адаптувати стратегії до умов гонки. Розглянемо деякі популярні платформи, які вже використовуються для аналізу та стратегічного планування: офіційний сайт Формула-1, Pitwall та Speed-Wiz [5–7]. Кожна з них має унікальні функціональні можливості, які слугують цілям аналізу даних і розробки гоночних стратегій.

Сайт Формула-1 є офіційним джерелом інформації про свесвіт Формула-1. Сайт надає широкий спектр матеріалів, включаючи новини, результати гонок, розклад змагань, інформацію про команди та пілотів. Однією з ключових функцій є розділ Live Timing, який дозволяє стежити за перебігом гонки в режимі реального часу, отримуючи дані про позиції пілотів, проміжні результати та іншу телеметрію (рис. 1.4) [5].

Переваги:

- Офіційне та надійне джерело інформації;
- Доступ до ексклюзивного контенту, включаючи інтерв'ю та аналітику;
- Інтуїтивно зрозумілий інтерфейс та регулярні оновлення.

Недоліки:

- Деякі розширені функції, такі як повний доступ до Live Timing, можуть вимагати підписки на F1 TV;
- Обмежені можливості для глибокого аналізу даних та розробки власних стратегій.

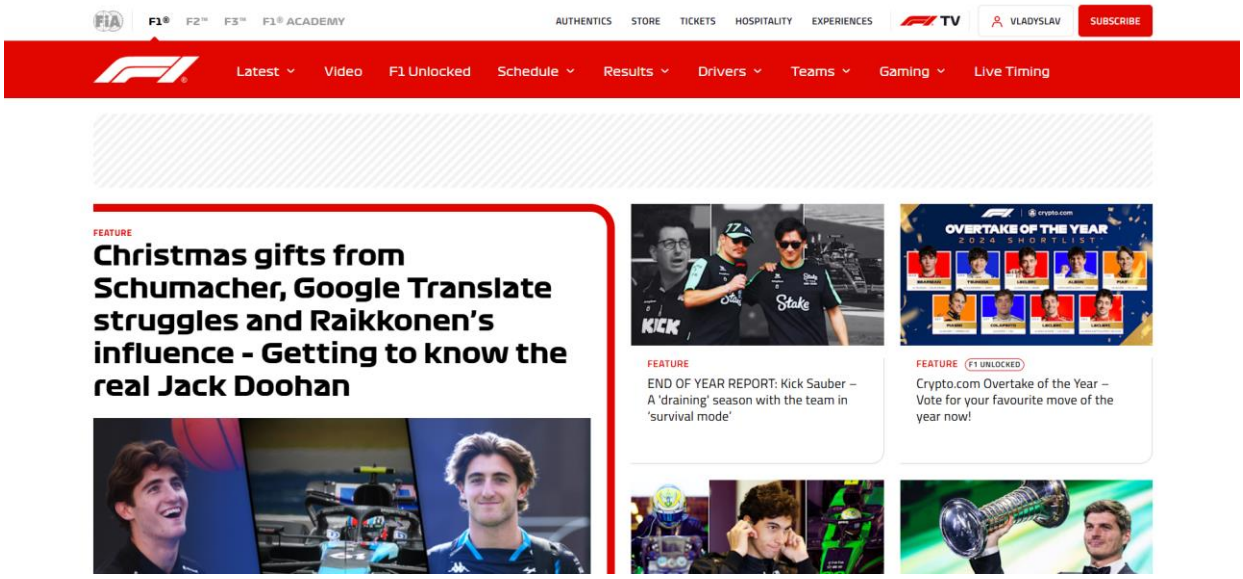


Рисунок 1.4 – Загальний вигляд інтерфейсного вікна головної сторінки офіційного сайту Формула-1

Pitwall – це онлайн-платформа, розроблена для надання розширених можливостей живого таймінгу та стратегічного аналізу під час перегонів. Спочатку онлайн-платформа була створена для сімрейсингу, вона здобула популярність і в реальному автоспорті, надаючи інструменти для аналізу даних, відстеження позицій автомобілів на трасі та планування стратегій піт-стопів (рис. 1.5) [6].

Переваги:

- Детальний живий таймінг з візуалізацією даних;
- Можливість інтеграції з різними платформами для стрімінгу, такими як YouTube та Twitch;
- Доступність через WEB-браузер без необхідності встановлення додаткового програмного забезпечення.

Недоліки:

- Основна аудиторія – сімрейсери, що може обмежувати застосування в професійному автоспорті;

- Деякі функції можуть вимагати платної підписки для повного доступу.

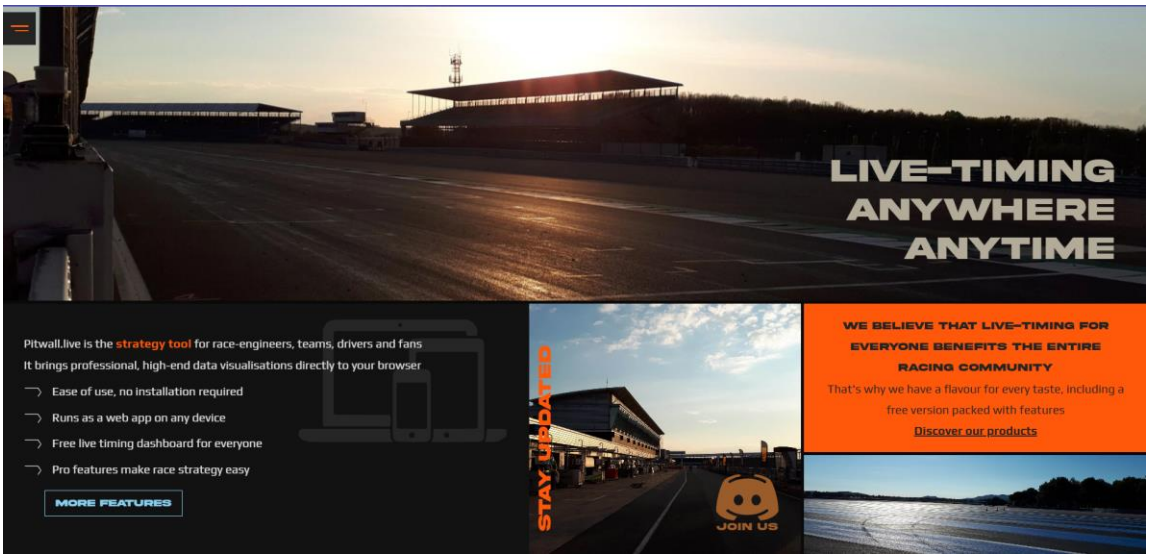


Рисунок 1.5 – Загальний вигляд інтерфейсного вікна головної сторінки платформи Pitwall

Speed-Wiz – це онлайн-інструмент, призначений для розрахунку та оптимізації гоночних стратегій. Платформа дозволяє користувачам вводити різні параметри гонки, такі як кількість піт-стопів, час на дозаправку та заміну шин, і на основі цих даних розраховувати оптимальну стратегію для досягнення найкращого результату (рис. 1.6) [7].

Переваги:

- Простий та зручний інтерфейс для швидких розрахунків;
- Можливість експериментувати з різними стратегіями та оцінювати їх ефективність;
- Корисний для аматорських команд та інженерів, які прагнуть покращити свої результати.

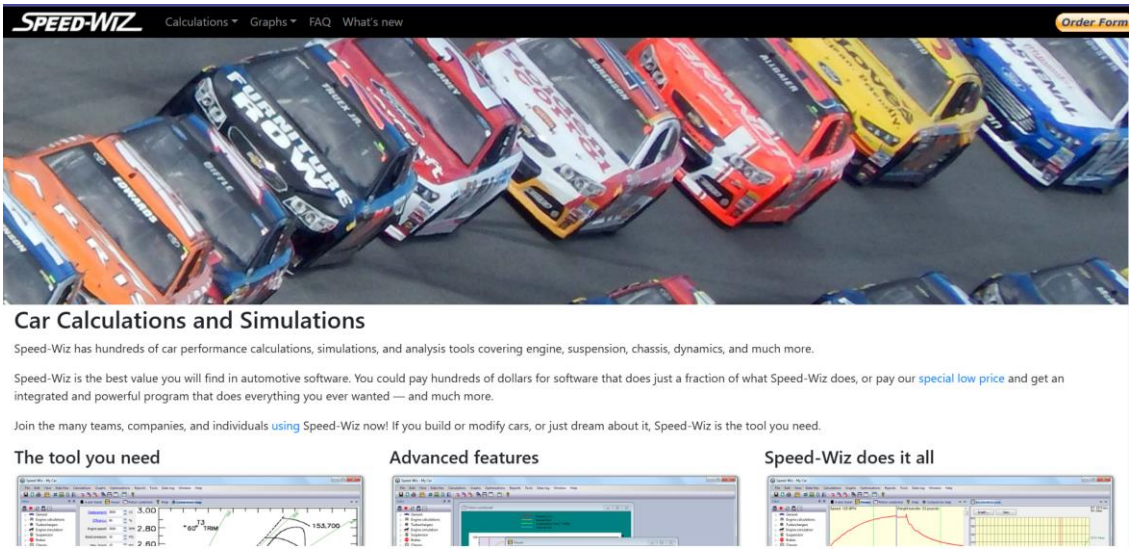


Рисунок 1.6 – Загальний вигляд інтерфейсного вікна головної сторінки платформи Speed-Wiz

Недоліки:

- Обмежена функціональність порівняно з професійними інструментами аналізу даних;
- Відсутність інтеграції з реальними телеметричними даними та живим таймінгом.

1.3 Формулювання вимог та постановка задачі

Кожного року Формула-1 залучає мільйони нових вболівальників по всьому світу, демонструючи не лише швидкість і майстерність пілотів, але й технічну досконалість команд. Проте, ключовими факторами успіху стає не тільки видатний гонщик і швидкий болід, а й максимально точне стратегічне планування. Грамотна організація піт-стопів, вибір гуми, прогнозування погодних умов та аналіз продуктивності болідів є важливими для досягнення високих результатів. Усе це вимагає обробки великих обсягів даних у реальному часі, що неможливо без сучасних технологій.

У Формулі-1 вже використовують різноманітні програмні рішення для аналізу та оптимізації стратегій [5]. Наприклад, офіційний сайт Формула-1 надає вболівальникам доступ до базових статистичних даних, таких як швидкість болідів, стан траси та прогноз погоди. Однак ця платформа орієнтована здебільшого на фанатів і не надає спеціалізованих інструментів для професійного аналізу.

Онлайн-платформа Pitwall є прикладом рішення, що дозволяє аналізувати гоночні дані в реальному часі. Її основна перевага полягає в інтерактивному відображенні позицій автомобілів на трасі та плануванні піт-стопів. Ця платформа здобула популярність серед сімрейсерів, однак її функціонал обмежений для використання в реальному автоспорті [6].

Інша платформа, Speed-Wiz, пропонує можливість моделювання гоночних стратегій на основі введених параметрів, таких як час заміни гуми або дозаправки. Її перевагою є простота використання та доступність для широкого кола користувачів. Однак, як й інші аналоги, вона не інтегрується з реальними телеметричними даними та має обмежений функціонал для складного аналізу [7].

Розробка онлайн-платформи, орієнтованої на професійні команди Формула-1, дозволить поєднати всі ключові аспекти: збір даних у реальному часі, прогнозування результатів та планування стратегій. У цій платформі мають бути передбачені такі функції:

- автоматизований збір даних про боліди, погоду та стан траси;
- алгоритми прогнозування, що допоможуть ухвалювати оптимальні рішення;
- інтуїтивно зрозумілий інтерфейс для взаємодії користувачів з платформою.

У роботі планується створити онлайн-платформу, яка стане інструментом для комплексного аналізу та підтримки прийняття рішень під час гонок. Вона має бути масштабованою, інтегрованою з хмарними технологіями та забезпечувати високу швидкість обробки даних. Інтерфейс платформи розроблятиметься з

урахуванням принципів UX, щоб зробити її максимально зручною для користувачів.

Для реалізації такої платформи, потрібно вирішити такі завдання:

- Реалізувати модуль збору та аналізу даних.
- Створити алгоритми для прогнозування ефективності стратегій.
- Розробити функціонал симуляції сценаріїв гонки.
- Забезпечити високу продуктивність та безпеку платформи.
- Провести тестування та оптимізацію системи.

1.4 Висновок

В цьому розділі було проаналізовано існуючі платформи для оптимізації стратегій. Проведений аналіз показав, що наявні рішення не повністю задовольняють потреби професійних команд Формули-1, оскільки більшість з них обмежується базовою телеметрією або спрощеними функціями моделювання стратегій без інтеграції з реальними даними в режимі реального часу.

Розробка онлайн-платформи для вибору стратегії гонки Формули-1 є актуальною, оскільки сучасний автоспорт вимагає використання потужних інструментів аналізу для підвищення ефективності прийняття рішень. Така платформа дозволить командам обробляти великі обсяги даних, моделювати сценарії гонок та швидко реагувати на змінні умови, що значно покращить їх конкурентоспроможність.

В розділі також було сформульовано основні завдання для подальшого дослідження та розробки, включаючи створення модулів для збору, аналізу та симуляції даних, забезпечення інтеграції з хмарними технологіями та реалізацію інтуїтивного користувацького інтерфейсу. Це закладає основу для розробки платформи, яка стане важливим інструментом для команд Формули-1.

2 ПРОЕКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ

2.1 Розробка структури онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

У розробці онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень принципово важливо обрати відповідну архітектуру. Існують різні підходи до побудови програмних систем: традиційна монолітна архітектура, розподілена мікросервісна архітектура та проміжний варіант – модульний моноліт. Кожен із цих підходів має свої особливості, переваги та обмеження, які безпосередньо впливають на швидкість розробки, масштабованість, надійність та вартість експлуатації рішення [8].

Монолітна архітектура передбачає реалізацію всієї функціональності в межах єдиного кодового базису й одного середовища виконання. Здебільшого це один процес або один виконуваний файл, що звертається до єдиної бази даних і розгортається як єдине ціле. Така модель забезпечує простоту початкової розробки та розгортання, адже всі компоненти тісно інтегровані між собою і залежать від єдиного середовища [9].

Серед головних переваг моноліту – швидкість старту проєкту: одна кодова база, єдині залежності та єдине середовище. Моноліт дозволяє здійснювати інтегроване тестування швидше, а продуктивність може бути вищою через відсутність мережевих викликів між компонентами.

Проте з ростом проєкту моноліт стикається з суттєвими недоліками: масштабування лише всієї програми одразу, складність внесення локальних змін, тобто потреба перезбірки й повторного розгортання всієї системи, висока

ймовірність порушення роботи інших модулів при найменших змінах, а також бар’єри для імплементації нових технологій.

Мікросервісна архітектура полягає в декомпозиції системи на незалежні сервіси, кожен із яких реалізує окрему бізнес-функцію і має власну кодову базу, середовище виконання та, за потреби, базу даних. Сервіси взаємодіють через API або повідомлення й розгортаються автономно.

До ключових переваг мікросервісів належать незалежність розробки та розгортання сервісів, гнучкість масштабування кожного сервісу окремо, можливість використання різних технологій під конкретні завдання, підвищена надійність, помилка в одному сервісі не обов’язково паралізує всю систему і пришвидшення випуску нових версій завдяки автономності команд [10].

Мікросервісна модель значно підвищує складність системи: із кожним новим сервісом множаться точки взаємодії, зростає кількість мережевих викликів та залежностей, що в результаті може привести до змішування функціональних зон і втрати чітких кордонів між компонентами, збільшення інфраструктурних витрат на кожен сервіс, ускладнення налагодження й моніторингу розподіленої системи, а також потребу в жорсткій організаційній координації та стандартизації протоколів обміну даними.

Проміжним варіантом між монолітом і мікросервісами є модульний підхід. Це однопроцесна система з єдиним розгортанням, але внутрішньо розділена на чітко виокремлені модулі. Кожен модуль відповідає за окремий набір функціональних можливостей, має власні інтерфейси та чіткі межі відповідальності, але всі вони працюють у межах одного середовища.

Головні переваги модулів – збереження простої схеми розгортання, зменшення оперативних витрат порівняно з мікросервісами, можливість незалежної розробки та тестування модулів, а також гнучкість у підтримці дисципліни меж між модулями. Завдяки модульній побудові можливо поступово

виділяти будь-який компонент у самостійний сервіс: коли навантаження або вимоги до масштабування зростають, достатньо відокремити відповідний модуль, перенести його в окремий деплоймент і під'єднати до існуючої системи, що дозволяє мінімізувати технічні ризики та звести до мінімуму витрати на міграцію.

Серед викликів, що можуть з'явитися з використанням модулів – ризик поганого визначення та підтримки кордонів між модулями, єдиний контур розгортання не дає змоги дослівно масштабувати або оновлювати тільки один модуль, а центральна база даних може приводити до високого ступеня зв'язності, що ускладнює поділ на пізніші мікросервіси [11].

Обраний модульний підхід дозволяє легко розпочати проєкт без зайвої складності мікросервісів, водночас зберігаючи можливість переходу до розподіленої архітектури в разі збільшення навантаження. Така модель оптимально поєднує економію ресурсів хмарної інфраструктури та швидкість розробки з достатньою гнучкістю масштабування окремих компонентів у перспективі.

Усі модулі отримують індивідуальні імена та можуть мати власну файлову і папкову ієрархію. Завдяки такій організації розробка стає більш структурованою та зручною.

У контексті онлайн-платформи підбору стратегії на основі хмарних технологій, можна розглянути такі модулі:

- Модуль головної сторінки;
- Модуль перегляду гонщиків;
- Модуль перегляду команд;
- Модуль перегляду наявних трас;
- Модуль перегляду легенд спорту;
- Модуль таймеру наступних гран-прі;
- Модуль вибору стратегії.

На рисунку 2.1 подано діаграму загальної структури онлайн-платформи підбору стратегії на основі хмарних технологій.

Розглянемо кожну сторінку та відповідний модуль:

- Коли користувач заходить на головну сторінку, інтерфейс завантажує відповідний модуль сторінки. Цей модуль взаємодіє з модулем календарем гран-прі, що повертає найближче гран-прі, яке має відбутися та скільки часу ще залишилося до нього. Сама ж головна сторінка надає коротку інформацію про цю платформу.

- Користувач також може взаємодіяти із сторінкою гонщиків. Модуль відображає всіх наявних гонщиків, які на даний момент знаходяться у пелетоні. Надає таку інформацію: ім’я та прізвище, за яку команду виступає гонщик, кількість подіумів, перемог, титулів чемпіона.

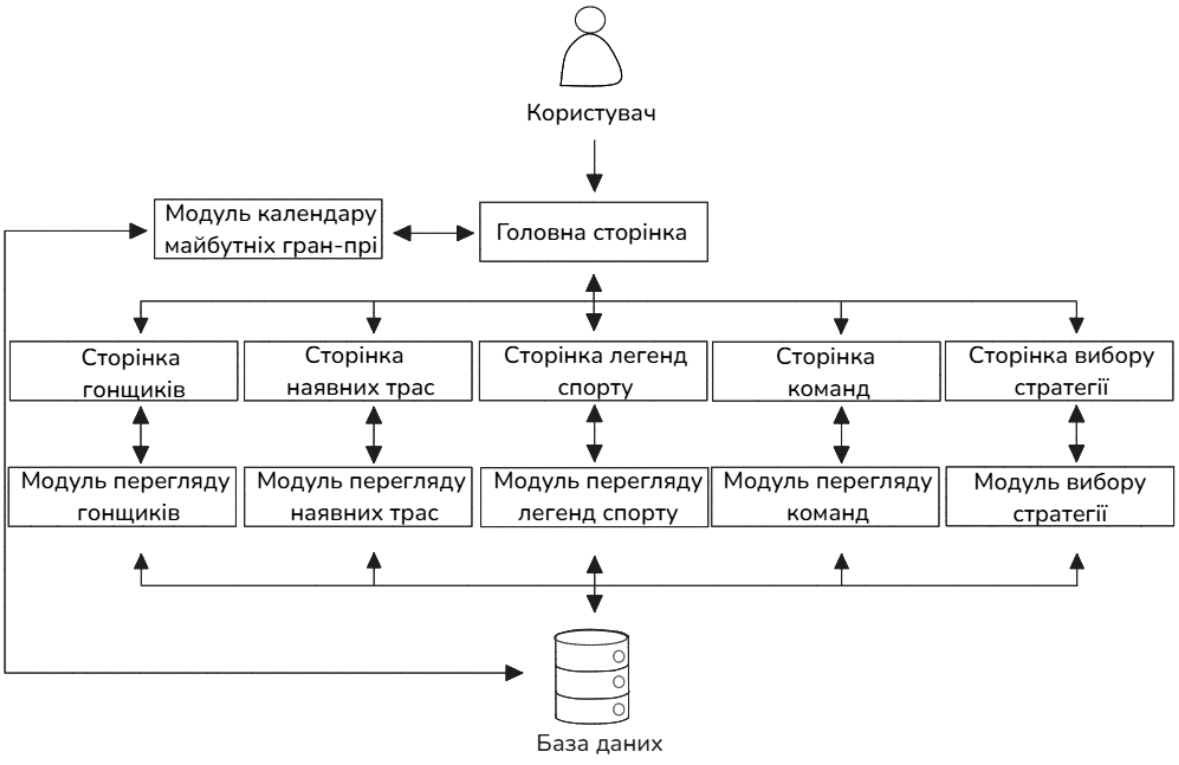


Рисунок 2.1 − Загальна структурна схема функціонування системи

- Наступна сторінка наявних трас, відноситься до інформативних сторінок як і сторінка гонщиків. Її модуль відповідає за інформування про траси, які зараз присутні у Формулі-1, а саме їхні повні назви, довжину, кількість кіл, коли трек приєднався до автоспорту та час і власника, який поставив рекорд кола.

- Сторінка легенд спорту має модуль, що виконує запит до бази даних, для того щоб надати інформацію про найкращих гонщиків Формули-1. Це включає інформацію про їхні досягнення за кар'єру.

- Сторінка команд відповідає за представлення команд, які виступають у пелетоні. Модуль звертається до бази даних, щоб надати інформацію про назву команди, кількість командних титулів, керівника та двох гонщиків що її представляють.

- Однією з найважливіших сторінок, які присутні у цій системі – це сторінка підбору стратегії. Її модуль містить у собі дуже важливий функціонал. Перший – це генерація практик, вони потрібні аби в подальшому наступний функціонал міг на основі них коректно підбирати стратегію відповідно до показників, які видав би болід. Наступним є генерація кваліфікації, вона дозволяє дізнатися з якої позиції на стартовій решітці займе гонщик, що також дуже допоможе при генерації стратегії. Ну й основний функціонал, невелика нейромережа, яка на основі даних з практики та кваліфікації, підбирає найоптимальнішу стратегію для досягнення максимального рівня.

Всі внутрішні компоненти платформи спілкуються через єдиний серверний шар із REST-інтерфейсом, який відповідає за прийом HTTP-запитів, їхню обробку та взаємодію з базою даних. Запити від користувача спочатку надходять на балансувальник навантаження, що рівномірно розподіляє трафік, забезпечуючи масштабованість і високу доступність системи [12].

Серверний шар виконує кілька важливих завдань: маршрутизує запити від різних модулів наприклад, для перегляду гонщиків, відображення календаря гран-

прі або розрахунку стратегії шин, до відповідних обробників; через проміжний рівень доступу виконує CRUD-операції над реляційною БД, де зберігаються відомості про гонки, пілотів, команди, траси, результати симуляцій і дані для машинного навчання.

Реляційна база даних виступає центральним сховищем усієї інформації – від детальних записів про практичні сесії, кваліфікації та гонки до навчальних наборів для ML-модуля. Вона гарантує швидкий пошук, надійне оновлення й підтримку зв'язків між таблицями, що дозволяє зберігати узгодженість даних і забезпечувати їх доступність усім компонентам платформи [13].

2.2 Розробка UML-діаграми класів

Для успішної побудови будь-якого програмного забезпечення потрібно побудувати його UML-діаграму класів та їхніх залежностей [14]. Діаграма класів є візуальним поданням структури програми на рівні об'єктно-орієнтованого дизайну. Вона допомагає зрозуміти, які класи існують у системі, які атрибути та методи вони мають, а також яким чином ці класи взаємодіють між собою. Завдяки чіткому рисунку можна змоделювати основні бізнес-об'єкти, забезпечити узгодженість архітектурних рішень та своєчасно виявити можливі помилки проєктування ще до початку кодування.

У контексті цього проєкту, серверна частина використовує класи та принципи об'єктно-орієнтованого програмування [15]:

- Інкапсуляція. Визначає чіткі межі між внутрішнім станом класу та зовнішнім кодом, забезпечуючи доступ до даних тільки через продумані методи.
- Наслідування. Дає змогу визначати нові класи на основі вже існуючих, успадковуючи їхню поведінку та властивості. Це сприяє повторному використанню коду та скорочує кількість дубльованих фрагментів.

- Поліморфізм. Дозволяє об’єктам різних типів відповідати єдиному інтерфейсу або базовому класу, що спрощує заміну й розширення компонентів без зміни клієнтської логіки.

- Абстракція. Зосереджується на найважливішому спрощенні моделі, відкидаючи зайві деталі, що допомагає формулювати загальні концепції системи та пришвидшує розуміння архітектури.

Для реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень передбачено такі класи серверної частини:

Клас Database відповідає за підключення до бази даних, зберігає параметри підключення, такі як користувач, хост, назва бази, пароль і порт. Він має методи connect() для встановлення пулу з’єднань з базою даних та disconnect() для його закриття. Цей клас є основою для роботи з даними у всіх API-класах, забезпечуючи централізоване керування підключенням до бази даних.

Клас DriversAPI реалізує логіку для роботи з інформацією про гонщиків. Він містить посилання на підключення бази даних. Основний метод get_drivers() дозволяє отримати список гонщиків, з можливістю фільтрації за командою. Метод виконує SQL-запит до бази даних, обробляє параметри фільтрації та повертає результат у вигляді списку.

Клас TeamsAPI призначений для роботи з командами. Він також містить посилання на об’єк Database. Основний метод get_teams() повертає список команд разом із переліком гонщиків у кожній команді. Метод формує SQL-запит із об’єднанням таблиць команд і гонщиків, групує результати та повертає їх у зручному для фронтенду вигляді.

Клас TracksAPI відповідає за отримання інформації про траси. Він містить метод get_tracks(), який повертає всі траси з бази даних. Клас також використовує Database для виконання SQL-запитів.

Клас LegendsAPI реалізує логіку для роботи з легендами автоспорту. Його метод get_legends() повертає список легенд, включаючи інформацію про відповідних гонщиків і команди. Клас використовує складний SQL-запит із об’єднанням кількох таблиць для отримання повної інформації.

Клас GrandprixAPI призначений для роботи з інформацією про гран-прі. Основний метод get_grandprix() повертає всі гран-прі, відсортовані за датою проведення. Клас також використовує Database для доступу до даних. Цей клас потрібний для того, аби в подальшому реалізувати таймер, до найближчого гран-прі.

Клас PracticeGenerator відповідає за генерацію, зберігання та отримання даних практичних сесій. Він містить атрибут db_url для підключення до бази даних та список імен таблиць для різних практик. Серед основних методів: create_tables() створює необхідні таблиці для практик, format_time() форматує час у зручний вигляд, get_practice_data() отримує дані з таблиць практик, get_track_data() повертає інформацію про конкретну трасу, generate_tire_sequence() генерує послідовність використання шин для сесії, generate_practice_rows() створює дані для кожного кола практики, clear_table() очищає таблицю для подальшої її використання, а generate_practice_data() генерує повний набір даних для практики на основі параметрів траси та погоди.

Клас QualificationGenerator реалізує логіку для генерації та отримання даних кваліфікаційних сесій, а також ініціює підключення до бази даних, як попередні класи. Метод create_table() створює таблицю кваліфікації, get_drivers_data() отримує інформацію про гонщиків і їхні команди, format_time() форматує час для відображення в секундах, generate_qualification_data() генерує результати кваліфікації з урахуванням різних характеристик, наприклад рейтингу гонщиків, команд, погоди та траси, а get_qualification_data() повертає результати кваліфікації.

Клас TireStrategyGenerator відповідає за генерацію та отримання стратегій використання шин під час гонки. Як і всі класи має підключення до бази даних, а також використовує принцип композиції щодо класів TireWearPredictor та StrategyOptimizer. У процесі генерації стратегії створюються екземпляри цих класів, які виконують відповідно прогнозування зносу шин на основі даних практик та оптимізацію стратегії шин з урахуванням параметрів траси, погоди та результатів кваліфікації. Метод create_table() створює відповідну таблицю для зберігання даних, fetch_practice_data() отримує дані згенерованих практик для подальшого аналізу, generate_strategy() генерує оптимальну стратегію шин, використовуючи можливості машинного навчання та оптимізації, а get_strategy() повертає поточну стратегію з бази даних. Завдяки композиції з ML-класами TireWearPredictor і StrategyOptimizer, основний їхній клас TireStrategyGenerator не містить усієї складної логіки всередині себе, а делегує її відповідним об’єктам і робить архітектуру більш чистою та підтримуваною.

Клас TireWearPredictor реалізує машинне навчання для прогнозування зносу шин на основі даних практик. Він містить нейронну мережу, яка навчається на історичних даних про проходження кіл, параметри шин, температури, зчеплення та інші характеристики. Основні методи класу – prepare_sequences(), формує послідовності даних для навчання, train(), що здійснює навчання моделі на підготовлених даних та predict_wear(), який дозволяє отримати прогноз зносу шин для поточного стану боліда.

Клас StrategyOptimizer відповідає за побудову та оптимізацію стратегії шин на гонку, використовуючи TireWearPredictor для оцінки зносу різних типів шин у різних умовах. Він враховує кількість кіл, погодні умови, стартову позицію, характеристики траси та результати кваліфікації. Генерує різні варіанти стратегії, оцінює їхню ефективність і повертає найкращі варіанти для користувача.

На рисунку 2.2 зображена UML-діаграма класів бекенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

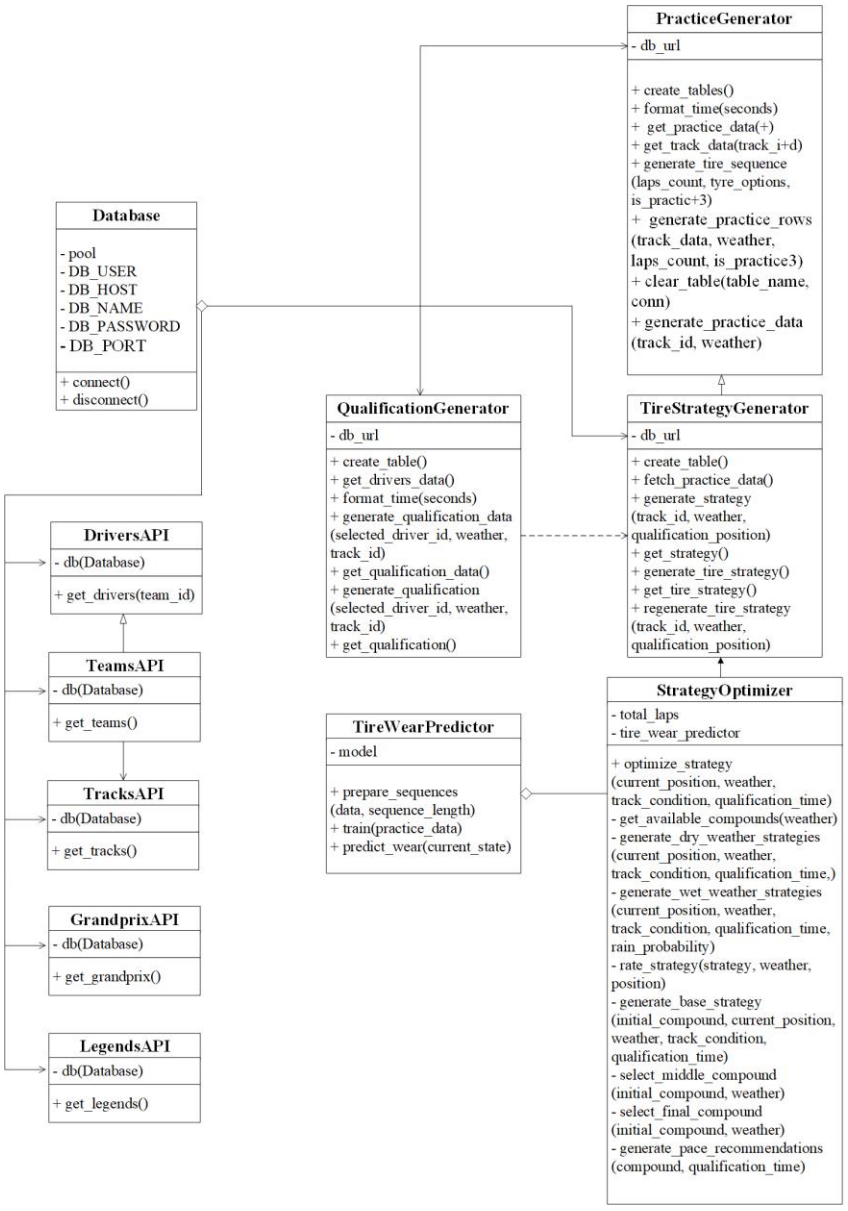


Рисунок 2.2 – UML-діаграма класів бекенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Для клієнтської частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень будуть використані такі класи:

Клас App є кореневим компонентом усього застосунку, який відповідає за загальне компонування сторінок, підключення глобальних стилів, провайдерів і розміщення основних структурних елементів, таких як Header. Він містить атрибут layout, що визначає структуру розміщення дочірніх компонентів.

Клас Header відповідає за реалізацію верхньої панелі навігації у застосунку. Його основна функція – забезпечити швидкий доступ користувача до основних інформаційних сторінок платформи, таких як сторінки гонщиків, команд, легенд, треків та загальної інформації про Формулу-1. Для цього клас містить атрибут menuItems, який визначає список пунктів меню. Метод renderMenu() відповідає за побудову та відображення цих пунктів меню у панелі навігації, дозволяючи користувачу легко переміщатися між різними розділами застосунку. Крім цього, клас має метод renderNextRaceCountdown(), який відповідає за відображення таймера до наступної гонки, що підвищує інформативність та інтерактивність інтерфейсу.

Клас HomePage є головною сторінкою застосунку, з якої починається взаємодія користувача з платформою. Він містить метод start(), який ініціює перехід до наступного етапу – вибору треку для симуляції або підбору стратегії. TrackSelectionPage відповідає за відображення списку доступних треків і дозволяє користувачу обрати потрібний трек для подальшої роботи. Основний метод selectTrack() приймає ідентифікатор треку та зберігає вибір для наступного кроку. Аналогічно, TeamSelectionPage() дозволяє обрати команду з доступного списку, а метод selectTeam() приймає ідентифікатор команди та передає його далі. DriverSelectionPage() реалізує вибір гонщика, використовуючи метод selectDriver(), який приймає ідентифікатор гонщика та зберігає його для подальшої симуляції. Клас SimulationPage є центральною сторінкою для підбору стратегії гонки. Він містить кілька ключових методів: handleWeatherChange() дозволяє змінювати погодні умови під час підбору стратегії, handleGeneratePractice() ініціює

генерацію даних практики, handleGenerateQualification() відповідає за генерацію кваліфікаційних даних, handleGenerateTireStrategy() запускає підбір оптимальної стратегії шин, а renderStrategyCards() відображає результати симуляції у вигляді карток стратегій. Цей клас взаємодіє з іншими допоміжними компонентами для відображення процесу та результатів симуляції.

LoadingStrategy – це допоміжний клас-компонент, який відповідає за відображення індикатора завантаження під час виконання асинхронних операцій, таких як генерація. Його основний атрибут isLoading відповідає за анімацію завантаження.

Клас TireStrategyVisualize – це компонент для візуалізації стратегії шин, який приймає об’єкт стратегії та загальну кількість кіл. Він відображає графічне представлення сегментів шин, що використовуються протягом гонки, дозволяючи користувачу швидко оцінити структуру стратегії.

Клас PaceRecommendations – це компонент, який відображає рекомендації щодо темпу для різних типів шин. Його атрибут paceData містить дані про рекомендовані режими їзди наприклад, кваліфікаційний, гоночний, атакуючий темп для кожного типу гуми, що допомагає користувачу краще зрозуміти, як потрібно використовувати шини у різних умовах.

Окрему групу складають інформаційні сторінки: InfoLegendsPage, InfoTracksPage, InfoDriversPage, InfoTeamsPage та InfoF1Page. Кожна з них відповідає за відображення відповідної інформації – про легенд Формули-1, траси, гонщиків, команди та загальні відомості про Формулу-1. Вони містять методи renderLegends(), renderTracks(), renderDrivers(), renderTeams() та renderF1Info() відповідно, які відповідають за отримання та відображення даних у зручному для користувача вигляді.

На рисунку 2.3 зображена UML-діаграма класів фронтенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

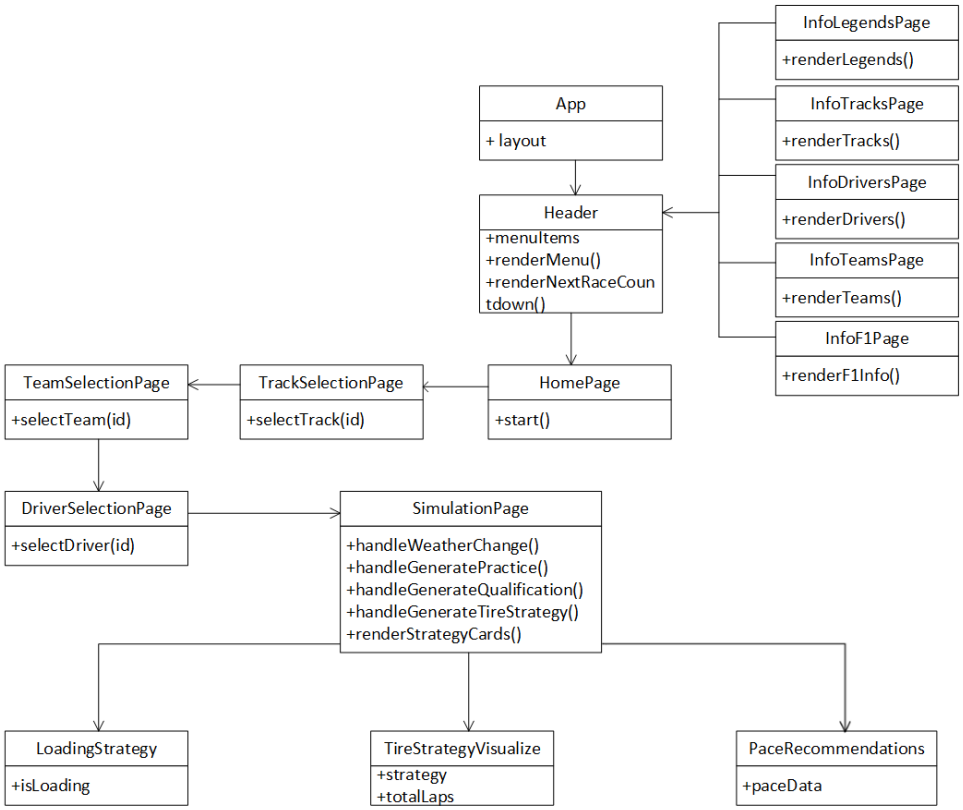


Рисунок 2.3 – UML-діаграма класів фронтенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Отже, розроблена UML-діаграма класів дозволяє наочно відобразити структуру системи, визначити її основні сутності, їхні атрибути та взаємозв’язки, що слугує чіткою основою для подальшої реалізації, тестування та підтримки коду.

2.3 Розробка ER-моделі бази даних

ER-модель – це концептуальне подання структури бази даних через сутності, їх атрибути та зв’язки між ними. Вона існує для узгодженого проєктування й перевірки коректності структури даних перед реалізацією, що є дуже важливим етапом перед розробкою.

В універсальне відношення потрібно включити атрибути, що описують такі сутності: команди, гонщики, зв’язки команда-гонщик, легенди, гран-прі, сесії практики та кваліфікацію. Універсальне відношення для реалізації бази даних системи автогонок буде мати такий вигляд: R(код команди, назва команди, логотип команди, керівник команди, опис команди, чемпіонства команди, рейтинг команди, код гонщика, ім’я гонщика, зображення гонщика, опис гонщика, номер гонщика, чемпіонства гонщика, перемоги гонщика, поули гонщика, подіуми гонщика, найшвидші кола гонщика, рейтинг гонщика, код команди гонщика, код легенди, ім’я легенди, зображення легенди, опис легенди, чемпіонства легенди, перемоги легенди, поули легенди, подіуми легенди, код гран-прі, назва гран-прі, локація гран-прі, дата гонки, код практики, кола, поточний час кола, час сектору 1 практики, час сектору 2 практики, час сектору 3 практики, швидкість повільних поворотів, швидкість середніх поворотів, швидкість швидких поворотів, паливне навантаження, деградація шин, температура треку, температура повітря, рівень щеплення, тип гуми, позиція, код гонщика кваліфікації, код команди кваліфікації, час кола, час сектору 1 кваліфікації, час сектору 2 кваліфікації, час сектору 3 кваліфікації). Ступінь універсального відношення – 53.

У поточній схемі бази даних онлайн-платформи для вибору стратегії гонки Формули-1 визначено такі пари сутностей та їхні типи зв’язків: зв’язок ОДИН-ДО-ОДНОГО (1:1) реалізовано між сутностями «Гонщик» – «Кваліфікації» та «Гран-прі» – «Кваліфікації»; зв’язок ОДИН-ДО-БАГАТЬОХ (1:Б) використовуються між сутностями «Команда»–«Гонщик», «Команда»–«Легенда», «Гран-прі»–«Практики», «Гонщик»–«Практики», «Команда»–«Кваліфікації» та «Команда»–«Кваліфікації». Зв’язків типу БАГАТО-ДО-БАГАТЬОХ (Б:Б) у цій моделі немає. Характеристики відношень між сутностями представлено у таблиці 2.1.

Таблиця 2.1 – Характеристики відношень між сутностями

Ім’я сутності 1	Ім’я сутності 2	Тип зв’язку	Клас належності
Гонщик	Кваліфікації	1 : 1	Обов’язковий
Гран-прі	Кваліфікації	1 : 1	Обов’язковий
Команда	Гонщик	1 : Б	Обов’язковий
Команда	Легенда	1 : Б	Необов’язковий
Гран-прі	Практики	1 : Б	Обов’язковий
Гонщик	Практики	1 : Б	Обов’язковий
Команда	Кваліфікації	1 : Б	Обов’язковий
Команда	Практики	1 : Б	Обов’язковий

На рисунку 2.4 представлено ER-модель для бази онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень.

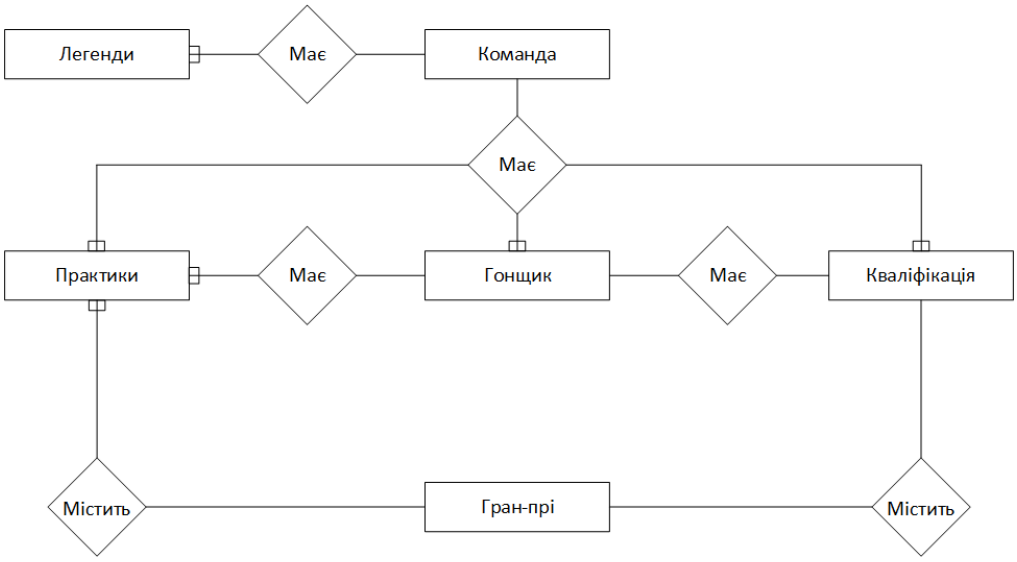


Рисунок 2.4 – ER-модель для бази даних онлайн-платформи для вибору стратегії гонки Формули-1 на основі хмарних обчислень

Отже, розроблена ER-модель бази даних дозволяє чітко відобразити ключові сутності, їх атрибути та зв’язки, забезпечуючи грамотно сплановану структуру даних. Це сприяє узгодженню вимог, виявленню логічних помилок на ранній стадії та полегшує подальшу реалізацію та підтримку системи.

2.4 Розробка схеми алгоритму роботи онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Розробка схеми алгоритму роботи онлайн-платформи для вибору стратегії гонки Формули-1 на основі хмарних обчислень – це ключовий етап проєктування системи, який дозволяє візуально і логічно відобразити послідовність дій від першого відкриття платформи до формування рекомендацій [16].

Розглянемо декілька основних алгоритмів, які будуть найчастіше використовуватися на онлайн-платформи для вибору стратегії гонки Формула-1. Для кращого розуміння алгоритмів буде представлено покрокові дії.

І випадок

Алгоритм перегляду основної інформації про Формулу-1 (рис. 2.5.):

Крок 1. Користувач відвідує головну сторінку онлайн-платформи.

Крок 2. Користувач переходить на сторінку «Гонщики». Сервер робить запит до бази даних та повертає інформацію про гонщиків.

Крок 3. Користувач переходить на сторінку «Команди». Сервер робить запит до бази даних та повертає інформацію про команди.

Крок 4. Користувач переходить на сторінку «Треки». Сервер робить запит до бази даних та повертає інформацію про треки.

Крок 5. Користувач переходить на сторінку «Легенди». Сервер робить запит до бази даних та повертає інформацію про легенд.

Крок 6. Користувач переходить на сторінку «Інфо».

Рисунок 2.5 – Схема алгоритму роботи онлайн-платформи

II випадок

Алгоритм вибору налаштувань підбору стратегії (рис. 2.5, аркуш 2):

Крок 1. Користувач відвідує головну сторінку онлайн-платформи.

Крок 7. Користувач натискає на кнопку «Почати».

Крок 8. Користувач вибирає трек.

Крок 9. Користувач вибирає команду.

Крок 10. Користувач вибирає гонщика.

Крок 11. Користувач переходить на сторінку «Підбору стратегії».

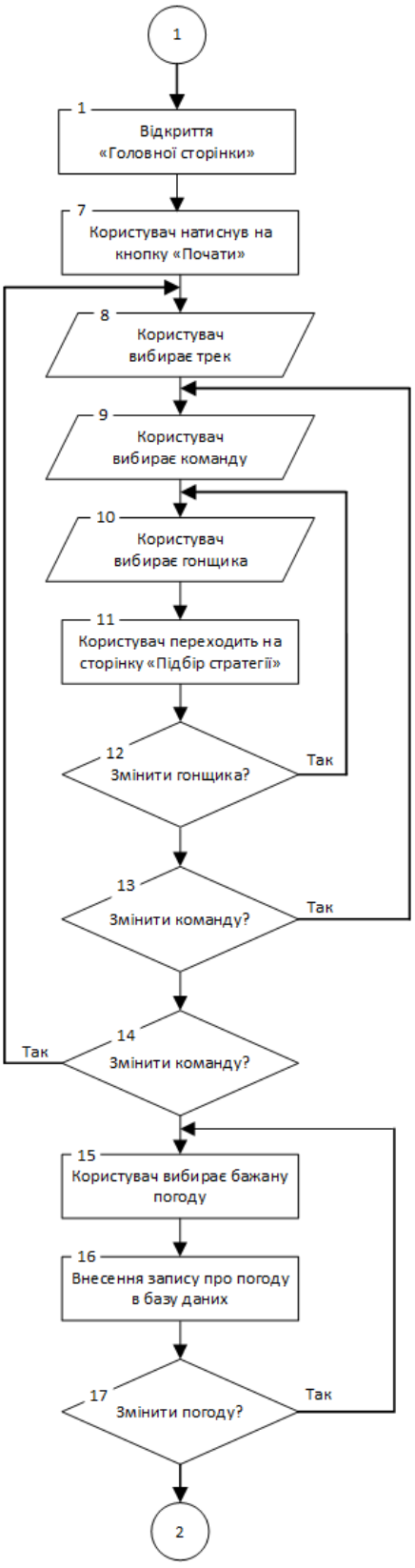


Рисунок 2.5, аркуш 2

Крок 12. Якщо користувач бажає змінити гонщика, то натискає на значок «переобрати» біля «Обраний гонщик».

Крок 13. Якщо користувач бажає змінити команду, натискає на значок «переобрати» біля «Обрана команда».

Крок 14. Якщо користувач бажає змінити трек, натискає на значок «переобрати» біля «Обрана траса».

Крок 15. Користувач вибирає бажану погоду.

Крок 16. Сервер записує значення погоди у базу даних.

Крок 17. Якщо користувач бажає змінити погоду, то натискає на інший варіант погоди.

III випадок

Алгоритм підбору стратегії (рис. 2.5, аркуш 3):

Крок 18. Для генерації даних про практичні сесії, користувач натискає на кнопку «Генерація практики» у блоці «Практика».

Крок 19. Сервер генерує дані практики.

Крок 20. Якщо користувач бажає перегенерувати дані з практик, то натискає на кнопку «Генерація» у блоці «Практика».

Крок 21. Для генерації даних кваліфікаційної сесії, користувач натискає на кнопку «Генерація кваліфікації» у блоці «Кваліфікація».

Крок 22. Сервер генерує дані кваліфікації.

Крок 23. Якщо користувач бажає перегенерувати дані кваліфікації, то натискає на кнопку «Генерація кваліфікації» у блоці «Кваліфікація».

Крок 24. Для підбору стратегію на гонку, користувач натискає на кнопку «Підбір стратегії гуми» у блоці «Стратегія гуми»

Крок 25. Сервер аналізує отримані дані з бази даних та виводить користувачу відповідні стратегії.

Крок 26. Якщо користувач бажає перегенерувати варіанти стратегії, то натискає на кнопку «Підбір стратегії гуми» у блоці «Стратегія гуми».

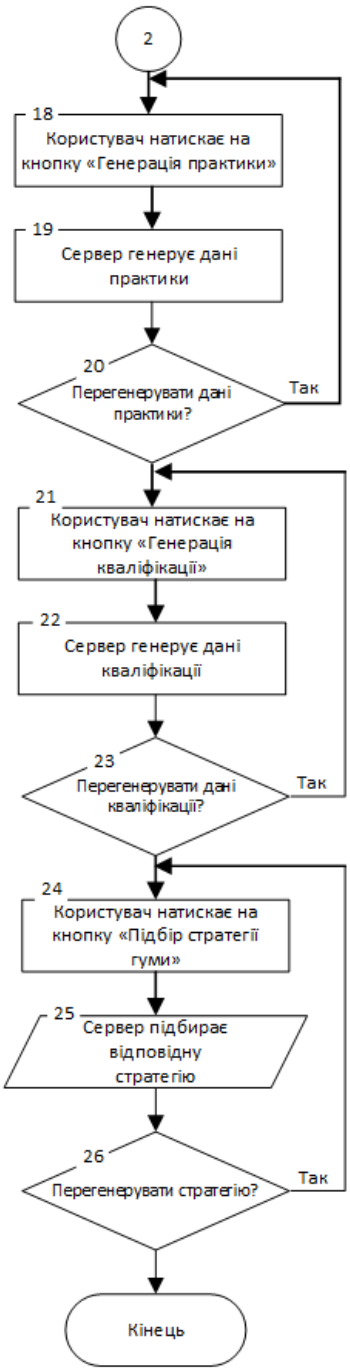


Рисунок 2.5, аркуш 3

2.5 Обґрунтування вибору хмарного провайдера для реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Під час вибору хмарного провайдера для реалізації онлайн-платформи підбору стратегії гонки Формула-1 важливо оцінити не лише технічні можливості та вартість, але й екосистему сервісів, простоту інтеграції з інструментами розробки, рівень підтримки та глобальне покриття дата-центрів. Нині на ринку лідирують три великі гравці – Google Cloud Platform, Microsoft Azure та AWS [17–19].

Google Cloud Platform вирізняється глибокою інтеграцією з передовими рішеннями у сфері штучного інтелекту та машинного навчання [17]. Microsoft Azure надає велику кількість інструментів для інтеграції з корпоративними рішеннями, а також міцно інтегрована з екосистемою Windows та продуктами Office 365, що полегшує автоматизацію бізнес-процесів та управління даними [18]. AWS, у свою чергу, є лідером за глибиною та широтою спектра послуг, що охоплюють обчислювальні потужності, зберігання даних, аналітику, контейнерні та безсерверні архітектури, а також передбачає розвинуті інструменти для моніторингу та безпеки [19].

Обираючи між цими трьома провайдерами, особливу увагу необхідно звернути на цілі проєкту: високі вимоги до швидкості обробки, масштабованість під час пікових навантажень, а також інтеграція з інструментами аналітики та машинного навчання. Хоча GCP ефективний у машинному навчанні, його екосистема менш розвинена в контексті глобальних дата-центрів для мінімізації затримок. Azure прийнятний для організацій, що вже працюють на платформах Microsoft.

AWS пропонує найбільшу кількість регіонів та зон доступності, що дозволяє забезпечити низькі затримки передачі даних. Різні сервіси AWS дозволяють реалізувати безсерверні та контейнерні рішення з автоскейлінгом, що критично для коливань навантаження. Для зберігання великих обсягів даних, оптимально використовувати Amazon S3 з можливістю налаштовування різних рівнів доступу.

Тому для розгортання інфраструктури онлайн-платформи буде використовуватися AWS, як хмарний провайдер.

2.6 Розробка структури хмарної інфраструктури

У сучасних умовах розробка чіткої та продуманої структури хмарної інфраструктури є ключовим чинником забезпечення безперебійної роботи, масштабованості та безпеки. Чіткий розподіл ресурсів та зон відповідальності дозволяє мінімізувати точки відмови, оптимізувати витрати на обчислювальні потужності й зберігання даних, а також легко інтегрувати нові сервіси в міру зростання навантаження [20].

Структуроване середовище гарантує високу відмовостійкість: розподілені між різними зонами доступності ресурси продовжать функціонувати навіть у разі виходу з ладу окремих вузлів або дата-центрів. Автоматичне масштабування дозволяє реагувати на пікові навантаження або сезонні коливання, знижуючи ризик перевантаження та уповільнення роботи платформи.

На рисунку 2.6 зображено розроблену діаграму хмарного середовища онлайн-платформи для вибору стратегії гонки Формула-1.

Першим рівнем маршрутизації користувацьких запитів виступає сервіс Amazon Route 53, який виконує DNS-розв’язування доменних імен, і спрямовує трафік по архітектурі. Завдяки вбудованим перевіркам стану Route 53 автоматично

фіксує недоступні ресурси та перенаправляє запити до здорових інстансів, це мінімізує час простою та підвищує відмовостійкість сервісу.

У межах виділеної VPC мережі створено два типи підмереж: публічну для ресурсів, які безпосередньо взаємодіють із зовнішнім світом та приватну для внутрішніх компонентів системи. Такий поділ із суворими правилами маршрутизації дозволяє ізолювати внутрішні сервіси від зовнішніх загроз, залишаючи доступними лише ті інтерфейси, які необхідні для коректної роботи платформи.

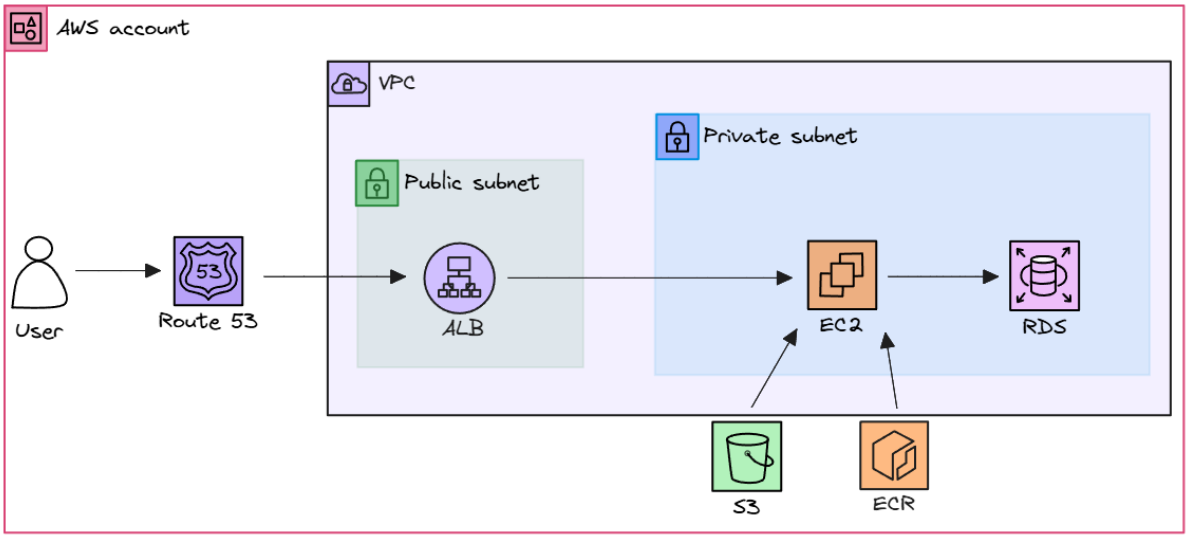


Рисунок 2.6 – Архітектурна діаграма хмарного середовища онлайн-платформи для вибору стратегії гонки Формула-1

У публічній підмережі розміщений Application Load Balancer, який приймає HTTP/HTTPS-запити від користувачів, шифрує незахищений трафік за допомогою SSL сертифікатів та розподіляє навантаження між інстансами з фронтенд-частиною додатку. ALB здійснює перевірки здоров’я таргет-груп, автоматично виводячи з обігу недоступні вузли та забезпечуючи високу доступність і можливість масштабування у відповідь на зміну трафіку.

У приватній підмережі працює EC2-інстанс із контейнеризованими фронтендом та бекендом, розгорнутими за допомогою Docker. Контейнерні образи зберігаються в Amazon Elastic Container Registry, що гарантує безпечне та швидке завантаження образів під час розгортання, підтримку їх.

Для зберігання статичних даних – фотографій гонщиків і легенд Формули-1, логотипів команд та треків – використовується Amazon S3. Це об’єктне сховище забезпечує високу надійність, масштабованість під будь-які обсяги.

Центральним сховищем даних виступає Amazon RDS, що пропонує керовану реляційну базу даних із автоматичними резервними копіями, шифруванням даних, можливістю масштабування обсягу сховища та вибору режиму розміщення Multi-AZ для підвищеної відмовостійкості. Вбудовані моніторингові метрики й налаштування параметрів продуктивності дозволяють оптимізувати часові затримки запитів і оперативно підлаштовуватися під змінні навантаження.

Такий підхід до побудови хмарної інфраструктури із виділенням чітких зон публічного й приватного доступу, балансувальника навантаження та керованих сервісів для зберігання й обробки даних відповідає найкращим практикам AWS і дозволяє забезпечити високі показники відмовостійкості, безпеки та продуктивності платформи [21].

2.7 Висновок

В цьому розділі було розроблено комплексне проектування онлайн-платформи для підбору стратегії гонки Формула-1 на основі хмарних обчислень. Розглянуто три ключові архітектурні підходи – моноліт, мікросервіси та модульну – з аналізом їхніх переваг і недоліків для розробки. Обґрунтовано вибір модульного підходу як оптимального компромісу між простотою розгортання та

гнучкістю масштабування, що дозволяє в майбутньому окремо виділяти компоненти у самостійні сервіси.

Було розроблено UML-діаграми класів як для серверної так і для клієнтської частини платформи. Для серверного шару описано ключові API-класи, а для клієнтської частини спроектовано кореневі компоненти, спеціалізовані сторінки та візуальні компоненти, що забезпечують інтуїтивний інтерфейс.

Розроблена ER-модель бази даних чітко відображає ключові сутності платформи, їхні атрибути та взаємозв’язки. Вона слугує основою для коректної реалізації сховища інформації.

Представлено схеми алгоритмів, які відображають послідовність дій починаючи від перегляду інформаційних розділів, вибір треку, команди та пілота, а також кроки генерації даних практик, кваліфікації та відповідної стратегії шин на гонку. Це створює чітку логіку взаємодії користувача з платформою та закладає основу для програмної реалізації.

Особлива увага приділена обґрунтуванню вибору хмарного провайдера. З огляду на глобальну інфраструктуру, різноманіття сервісів для контейнеризації, безсерверних функцій і керованих баз даних, обрано AWS як оптимальне рішення для розгортання та забезпечення високої доступності платформи.

Також, розроблено архітектурну діаграму хмарного середовища. Розроблена структура гарантує безпеку й можливість динамічного масштабування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ

3.1 Обґрунтування вибору мови реалізації онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Однією з мов програмування для реалізації фронтенду є JavaScript – динамічна мова програмування, яка виконується в браузері та на сервері й відповідає за інтерактивність і асинхронну логіку WEB-застосунків. Вона дозволяє маніпулювати DOM, обробляти події, виконувати HTTP-запити та використовувати велику екосистему пакетів.

Для реалізації клієнтської частини проєкту було обрано мову програмування TypeScript, оскільки вона забезпечує статичну типізацію, дозволяє виявляти помилки на етапі компіляції та сприяє підтримці масштабованого коду завдяки чіткому визначенню інтерфейсів і типів [22]. Платформа Next.js виступає основою фреймворку, що спрощує налаштування серверного рендерингу і генерування готових сторінок заздалегідь, що робить застосунок швидшим, а також надає вбудовані механізми маршрутизації та кешування.

Бібліотека React використовується для побудови декларативного і компонентного інтерфейсу, вона гарно інтегрується з TypeScript [23]. Для стилізації обрано Tailwind CSS, оскільки її підхід дозволяє швидко створювати адаптивні макети без написання громіздких CSS-класів. Radix UI слугує джерелом готових, доступних та консистентних компонентів інтерфейсу, що пришвидшує розробку.

У таблиці 3.1 представлено порівняння мов програмування для фронтенд частини.

Таблиця 3.1 – Порівняння мов програмування фронтенду

	TypeScript	JavaScript
Статична типізація	+	-
Підтримка ООП	+	+
Виявлення помилок під час компіляції	+	-
Простота синтаксису	-	+
Інтеграція з IDE	+	+

З таблиці 3.1 видно, що TypeScript поєднує в собі переваги JavaScript з додатковою безпекою типізації та розвиненою підтримкою IDE.

Для серверної частини обрано Python як основну мову програмування завдяки її лаконічному та зрозумілому синтаксису, який суттєво скорочує час розробки і підвищує читабельність коду [24]. Python володіє потужною стандартною бібліотекою й розвиненою екосистемою сторонніх пакетів, що робить його ідеальним вибором для побудови як простих REST-сервісів, так і складних систем із компонентами машинного навчання. В якості WEB-фреймворку використовується FastAPI [25]. Його асинхронна модель дозволяє обробляти велику кількість одночасних запитів з низькими затримками.

Також розглядалися мови програмування C# та Go. Перша забезпечує потужну об’єктно-орієнтовану модель, високу продуктивність і глибоку інтеграцію з екосистемою .NET, що спрощує розробку корпоративних сервісів. А Go вирізняється простим синтаксисом, вбудованою підтримкою конкурентності та швидким часом виконання, що робить його ідеальним для високонавантажених розподілених систем.

Для реалізації компонентів машинного навчання застосовуються бібліотеки TensorFlow та scikit-learn. TensorFlow надає можливості побудови, тренування й розгортання глибоких нейронних мереж. Scikit-learn доповнює цю екосистему класичними алгоритмами машинного навчання, дозволяючи швидко прототипувати моделі й проводити їхню оцінку без зайвих зусиль. Для підготовки

даних і проведення аналітики використовуються Pandas та NumPy. NumPy є базовим інструментом для ефективних чисельних обчислень на масивах, а Pandas забезпечує зручні структури даних із багатим набором функцій для фільтрації, агрегування та трансформації таблиць.

У таблиці 3.2 представлено порівняння мов програмування для бекенд частини.

Таблиця 3.2 – Порівняння мов програмування бекенду

	Python	C#	Go
Типізація	-	+	+
Підтримка ООП	+	+	+
Простота синтаксису	+	-	+
Швидкодія	-	+	+
Інтеграція з IDE	+	+	+
Екосистема для машинного навчання	+	-	-

З таблиці 3.2 видно, що Python поєднує в собі простоту розробки з потужними бібліотеками для роботи з даними та машинного навчання, тоді як C# і Go, хоча і забезпечують статичну типізацію, вищу продуктивність та оптимальні для вирішення конкурентних задач, але все ж поступаються Python у можливості швидко і просто працювати з машинним навчанням.

3.2 Обґрунтування вибору мови керування базою даних

Є багато різних мов керування базами даних, але у контексті цього проекту розглянемо такі: Object Query Language, Structured Query Language та Not Only SQL.

Structured Query Language, SQL – це декларативна мова запитів, яка використовується для роботи з реляційними базами даних. У SQL дані організовані

в таблиці з рядками та стовпцями, де стовпці відповідають атрибутам сутностей. Основні операції SQL – це SELECT, INSERT, UPDATE, DELETE, що дозволяють отримувати, додавати, змінювати та видаляти дані. SQL підтримує складні JOIN-операції для роботи з пов'язаними таблицями, забезпечує транзакції з ACID-властивостями та має розвинений механізм прав доступу [26].

Object Query Language, OQL – це мова запитів, розроблена для об'єктно-орієнтованих СУБД. Вона підкреслює роботу з об'єктами та їхніми методами, а не з табличними структурами. Запити в OQL є виразами, які можна вкладати один в одного без обмежень, що спрощує композицію складних запитів. OQL прагне приховати деталі схеми даних і забезпечити безпечний доступ через методи об'єктів, але при цьому може бути менш передбачуваною через декларативність і вкладеність виразів, що іноді ускладнює оптимізацію запитів [27].

Not Only SQL, NoSQL – це сукупність підходів і технологій для роботи з нереляційними базами даних, які не використовують традиційну табличну структуру. Замість таблиць, зберігають дані у вигляді ключ-значення, документів типу JSON, стовпців або графів. Такі бази даних забезпечують гнучкість структури даних, горизонтальне масштабування та високу швидкодію при обробці великих обсягів даних [28].

Основна відмінність між ними полягає в тому, що SQL орієнтований на реляційну модель з чітко визначеними таблицями та зв'язками, OQL – на об'єктну модель з роботою через атрибути й методи об'єктів, а NoSQL – на альтернативні моделі збереження даних без жорсткої схеми. SQL-сервери мають гарно оптимізовані алгоритми виконання запитів, що дозволяє ефективно працювати з великими обсягами даних і гарантувати передбачувані плани виконання. Натомість OQL забезпечує гнучкість у побудові виразів, а NoSQL системи пропонують масштабованість і простоту роботи з великими розподіленими наборами даних.

У таблиці 3.3 наведено порівняння основних характеристик SQL, OQL та NoSQL.

Таблиця 3.3 – Порівняння основних характеристик OQL та SQL мов програмування для управління організованою базою даних

	SQL	OQL	NoSQL
Модель даних	Таблиці, рядки/стовпці	Класи, об’єкти, методи	Ключ-значення, документи, стовпці, графи
Основний синтаксис	SELECT, INSERT, UPDATE, DELETE	Вирази SELECT як об’єднувальні вирази	API, запити мовою драйвера
Композиція запитів	Підтримка CTE, вкладених підзапитів, JOIN	Вирази повністю вкладені без обмежень	Обмежена підтримка складних JOIN
Оптимізація продуктивності	Потужна система планування запитів, індекси, паралельність	Менш прогнозована оптимізація через складну вкладеність	Горизонтальне масштабування, кешування
Безпека та права доступу	Гранулярні права на рівні схем, таблиць, стовпців	Механізми через методи класів, залежить від реалізації	Прості моделі доступу
Сфера застосування	Узагальнені реляційні завдання, аналітика, OLTP	Об’єктно-орієнтовані системи, специфічні бізнес-логіки	Високонавантажені додатки, розподілені системи

У підсумку, для проєкту обирається SQL у вигляді PostgreSQL. Це рішення обумовлене широкою підтримкою реляційної моделі, потужним механізмом оптимізації запитів, багатим набором вбудованих розширень і стабільністю роботи в масштабованих середовищах.

3.3 Обґрунтування вибору середовища розробки

Існує багато середовищ розробки, але в контексті цього проєкту розглянемо найпопулярніші: Visual Studio Code, WebStorm та Sublime Text.

Visual Studio Code – безкоштовний та відкритий редактор коду від Microsoft. Відрізняється високою швидкістю запуску, інтегрованою системою розширень, вбудованим терміналом і відлагоджувачем для різних мов. VS Code підтримує інтеграцію з Git, має велику спільноту та регулярні оновлення. Завдяки широкому набору плагінів можна налаштувати середовище під розробку на JavaScript, TypeScript, Python, Docker, Kubernetes та багато інших технологій [29].

JetBrains WebStorm – комерційна IDE, орієнтована на розробку на JavaScript та TypeScript. Забезпечує потужну підтримку фреймворків, автодоповнення коду, рефакторинг, аналіз якості та покриття тестами. Має вбудовані інструменти для роботи з Git, налаштуваннями запуску й відлагодження, інтеграцію з популярними серверами та базами даних. Однак WebStorm потребує більше ресурсів і вимагає купівлі ліцензії [30].

Sublime Text – легкий та швидкий текстовий редактор з мінімалістичним інтерфейсом. Підтримує плагіни, дозволяє працювати з великими файлами та проектами без зниження продуктивності. Однак для роботи з контейнерами доводиться додатково встановлювати і налаштовувати плагіни, а деякі можливості можуть бути обмежені порівняно з повноцінними IDE. Sublime Text також є платним після випробувального періоду [31].

У таблиці 3.4 наведено порівняння основних характеристик Visual Studio Code, WebStorm та Sublime Text.

Таблиця 3.4 – Порівняння середовищ розробки Visual Studio Code, WebStorm та Sublime Text.

Характеристика	VS Code	WebStorm	Sublime Text
Запуск і продуктивність	Помірне споживання ресурсів	Високе навантаження на систему	Мінімальне навантаження
Розширюваність	Marketplace з тисячами розширень	Плагінами обмежені JetBrains	Доступні плагіни через Package Control
Відлагодження	Інтегрований дебаггер для багатьох середовищ	Потужний відлагоджувач із глибоким аналізом	Обмежені можливості без додаткових плагінів
Вартість	Безкоштовно	Платно	Платно після пробного періоду

На основі аналізу даних таблиці 3.4 для розробки програмного продутку обирається Visual Studio Code. Це середовище забезпечує оптимальний баланс між швидкістю роботи, масштабованою екосистемою розширень.

3.4 Розробка клієнтської частин

Фронтенд частина платформи реалізована на базі Next.js з використанням TypeScript та спеціальних UI-компонентів. Кожна сторінка відповідає окремому етапу взаємодії користувача з платформою та організована у вигляді окремих маршрутів у папці app/.

Початковий файл – app/page.tsx. Відображає вітальний екран платформи RaceMind, коротко описує її призначення та містить кнопку для старту роботи, яка веде до вибору треку, зокрема:

```
export default function HomePage() {
  return (
    <div className="min-h-screen flex flex-col bg-background text-foreground">
```

```
<main className="flex-grow flex flex-col items-center justify-center bg-background"><div
className="text-center mb-8">
<h1 className="text-5xl font-bold mb-4">
<span className="text-primary text-6xl">RaceMind</span> - онлайн платформа</h1>
<p className="text-2xl">для оптимізації стратегій на гонку Формули-1</p></div>
<Link href="/track-selection" className="mt-8">
<Button size="lg" className="bg-primary text-primary-foreground hover:bg-
primary/90">Почати</Button>
</Link></main></div>)}
```

Сторінки вибору треку, команди та гонщика мають подібну структуру коду. Сторінка вибору треку дозволяє користувачу обрати трасу для симуляції. Дані про треки отримуються з API, кожен трек відображається з картинкою, назвою та кнопкою для переходу до вибору команди. Після вибору треку користувач обирає команду Формули-1. Відображає логотипи, назви та керівників команд, отриманих з API. Після вибору команди користувач переходить до вибору гонщика. Така сторінка дозволяє обрати гонщика з обраної команди. Відображає фото, ім’я та кнопку для переходу до симуляції стратегії. Сторінки вибору треку, команди та гонщика мають подібну структуру коду, тому для прикладу було обрано представлення сторінки вибору гонщика:

```
{drivers.map((driver) => (
<div key={driver.id} className="border rounded-lg p-4 flex flex-col items-center bg-card text-card-
foreground">
<Image src={driver.image || "/placeholder.svg"} alt={driver.name} width={200} height={200}
className="mb-4 rounded-full" />
<h2 className="text-xl font-semibold mb-4">{driver.name}</h2>
<Link
href={`/simulation?track=${selectedTrackId}&team=${selectedTeamId}&driver=${driver.id}`}>
<Button className="bg-primary text-primary-foreground hover:bg-primary/90">Вибрати
гонщика</Button><Link></div>))}
```

Найголовніше сторінка платформи – «Сторінка підбору стратегії», дозволяє моделювати стратегії гонки з урахуванням обраного треку, команди, гонщика та

погодних умов. Відображає рекомендації щодо шин, темпу, піт-стопів, а також візуалізує стратегію на графіку, зокрема:

```
<TireStrategyVisualizer strategy={strategy} totalLaps={totalLaps} />
<PaceRecommendations paceData={strategy.pace_recommendations} />
<Button onClick={handleGenerateTireStrategy}>Згенерувати стратегію</Button>
```

3.5 Розробка серверної частин

Серверна частина платформи розроблена з використанням фреймворку FastAPI, який забезпечує високу продуктивність та зручний розвиток REST API.

Початковим файлом для роботи онлайн-платформи є main.py, який ініціалізує FastAPI додаток та налаштовує необхідні middleware для обробки CORS запитів. Додаток організований за модульним принципом, де кожен компонент відповідає за свою частину функціональності:

```
app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["https://racemind.vladops.click/"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
@app.get("/api/health", include_in_schema=False)
async def health_check():
    return {"status": "ok"}

drivers_api = DriversAPI(db)
grandprix_api = GrandprixAPI(db)
legends_api = LegendsAPI(db)
teams_api = TeamsAPI(db)
tracks_api = TracksAPI(db)
app.include_router(drivers_api.router, prefix="/api")
```

```
# інші роути прописані за схожим сценарієм
```

Всі логічні блоки API винесені в окремі модулі, кожен з яких підключається до основного FastAPI-додатку через include_router(). Це дозволяє зберігати код чистим, масштабованим та організованим за доменами. Також визначено простий ендпоїнт /api/health, який використовується для перевірки стану сервісу.

Клас Database відповідає за створення та керування пулом асинхронних з'єднань до бази даних PostgreSQL з використанням бібліотеки asyncpg. Конфігурація завантажується із змінних оточення, що дозволяє легко змінювати параметри в різних середовищах без змін у коді.

Реалізація класу Database має такий зміст:

```
def __init__(self):
    self.pool = None
    self.DB_USER = os.environ.get("DB_USER")
    self.DB_HOST = os.environ.get("DB_HOST")
    self.DB_NAME = os.environ.get("DB_NAME")
    self.DB_PASSWORD = os.environ.get("DB_PASSWORD")
    self.DB_PORT = int(os.environ.get("DB_PORT", 5432))
    if not (self.DB_USER and self.DB_HOST and self.DB_NAME and self.DB_PASSWORD):
        print("Відсутні змінні оточення! Переконайтеся, що вони задані.")
        exit(1)
async def connect(self):
    self.pool = await asyncpg.create_pool(
        user=self.DB_USER,
        password=self.DB_PASSWORD,
        database=self.DB_NAME,
        host=self.DB_HOST,
        port=self.DB_PORT,
    )
    print("Підключення до БД встановлено.")
async def disconnect(self):
    await self.pool.close()
    print("Підключення до БД закрито.")
```

Під час запуску застосунку викликається метод connect(), який створює пул підключень. Метод disconnect() відповідає за коректне завершення сесії. Цей підхід дозволяє ефективно обробляти паралельні запити та знижує навантаження на СУБД.

Кожен із drivers.py, teams.py, legends.py, tracks.py, grandprix.py відповідає за окремий аспект даних Формули-1 і реалізує власний клас з маршрутизатором FastAPI. DriversAPI дозволяє отримати список гонщиків, з можливістю фільтрації за ідентифікатором команди. Реалізовано динамічне складання SQL-запиту з умовою WHERE, якщо параметр team_id передано у запиті. TeamsAPI дозволяє отримати список команд разом з усіма гонщиками, які представляли ці команди. Для цього в SQL-запиті використовується агрегатна функція ARRAY_AGG, яка формує масив імен гонщиків. LegendsAPI відповідає за виведення таблиці легенд Формули-1, об’єднуючи дані про гонщиків та команди. TracksAPI і GrandprixAPI надають базові списки трас і гонок відповідно, без додаткових фільтрів або параметрів. Обидва використовують прості SQL-запити SELECT *, які можна масштабувати в майбутньому, додаючи нову бізнес-логіку.

Для прикладу представлено один з варіантів такого класу запитів LegendsAPI.

Реалізація класу LegendsAPI має такий зміст:

```
class LegendsAPI:
    def __init__(self, db: Database):
        self.db = db
        self.router = APIRouter(prefix="/legends", tags=["legends"])
        self.router.add_api_route("", self.get_legends, methods=["GET"])

    async def get_legends(self):
        # Запит до таблиці legends
        try:
            rows = await self.db.pool.fetch(query)
            return [dict(row) for row in rows]
```

```
    except Exception as error:
        print("Помилка отримання легенд:", error)
        raise HTTPException(status_code=500, detail="Не вдалося отримати дані легенд")
```

Модуль practice.py реалізує клас PracticeGenerator, який відповідає за створення таблиць практик та заповнення їх даними. У методі create_tables() виконується SQL-інструкція CREATE TABLE IF NOT EXISTS, яка гарантує, що потрібні таблиці існують.

Реалізація класу PracticeGenerator має такий зміст:

```
def __init__(self, db_url: str = DATABASE_URL):
    self.db_url = db_url
    self.table_names = ["practice1", "practice2", "practice3"]

def format_time(self, seconds):
    total = float(seconds)
    minutes = int(total // 60)
    secs = int(total % 60)
    ms = int((total * 1000) % 1000)
    return f"{minutes}:{secs:02d}:{ms:03d}"
```

Дані з таблиць обробляються методом get_practice_data(), де особливу увагу приділено форматуванню часу кола – перетворення з секунд у формат Хв:Сек:мс. Це дозволяє краще візуалізувати результати для подальшої аналітики.

Реалізація метода def get_practice_data(self) має такий зміст:

```
def get_practice_data(self):
    results = {}
    conn = connect(self.db_url)
    cur = conn.cursor()
    for table in self.table_names:
        cur.execute(sql.SQL("SELECT* FROM {}").format(sql.Identifier(table)))
        raw = cur.fetchall()
        formatted = []
        for row in raw:
            row = list(row)
            row[2] = self.format_time(row[2])
```

```
            formatted.append(row)
        results[table] = formatted
    cur.close()
    conn.close()
    return results
```

У qualification.py реалізовано генератор кваліфікаційних даних. Його логіка побудована на симуляції змагання між гонщиками на основі їхнього рейтингу, рейтингу команди, погодних умов та вибраної траси. Метод generate_qualification_data() виконує обчислення часу кола для кожного гонщика та зберігає результати в таблицю qualification.

Реалізація класу QualificationGenerator має такий зміст:

```
def generate_qualification_data(self, selected_driver_id, weather, track_id):
    drivers = self.get_drivers_data()
    execute_values(cur, insert_query, values)
    conn.commit()
    cur.close()
    conn.close()
    return {}


def get_qualification_data(self):
    conn = connect(self.db_url)
    cur = conn.cursor()
    # SELECT із об'єднанням із таблицями drivers та teams
    cur.close()
    conn.close()
    return results
```

Модуль tire_strategy.py є одним з найбільш інноваційних у серверній частині. Він поєднує обробку SQL-запитів із застосуванням машинного навчання для генерації оптимальної стратегії гонки.

Реалізація класу TireStrategyGenerator має такий зміст:

```
def fetch_practice_data(self):
async def generate_strategy(self, track_id, weather, qualification_position):
```

```
practice_df = self.fetch_practice_data()
predictor = TireWearPredictor(); predictor.train(practice_df)
optimizer = StrategyOptimizer(total_laps=total_laps, tire_wear_predictor=predictor)
strategies = optimizer.optimize_strategy(
   current_position=qualification_position,
   weather=weather,
   track_condition={"temperature": practice_df['trackTempC'].mean(),
                   "grip": practice_df['gripLevel'].mean()},
   qualification_time=float(qualification_time[0]))
return strategies
```

Це дозволяє реалізувати адаптивні стратегії гонки, які змінюються відповідно до контексту – погодних умов, зносу шин та стартової позиції гонщика.

Модуль ml_models.py містить два важливих компоненти: TireWearPredictor і StrategyOptimizer. TireWearPredictor – це нейронна мережа на базі LSTM, яка прогнозує знос шин на основі історичних даних практик. У методі prepare_sequences() дані перетворюються у формат, прийнятний для навчання моделі. Модель має два LSTM-шари, Dropout та два Dense-шари, що дозволяє ефективно захоплювати часові залежності. А StrategyOptimizer – використовує навчену модель для оцінки різних сценаріїв стратегії гонки: скільки разів змінювати шини, які типи шин використовувати, коли робити піт-стопи. Стратегії ранжуються за внутрішньою метрикою і користувачу повертаються найкращі з них.

3.6 Розробка темплейтів хмарної інфраструктури

Для забезпечення надійної, масштабованої та керованої інфраструктури онлайн-платформи використовується інструмент інфраструктури як код – Terraform [32]. Це дозволяє автоматизувати процес розгортання, налаштування та супроводу всіх необхідних хмарних ресурсів у середовищі AWS. Кожен компонент інфраструктури описується у вигляді окремих конфігураційних файлів,

що забезпечує прозорість, повторюваність і легкість внесення змін. Нижче наведено короткий опис основних файлів, які відповідають за створення ключових ресурсів платформи.

Темплейт з назвою ec2 відповідає за автоматизоване створення віртуальної машини в AWS. Він визначає основні параметри інстансу, такі як AMI, тип сервера, підмережа, ключ доступу, а також підключає скрипт ініціалізації для автоматичного налаштування середовища під час запуску, зокрема:

```
resource "aws_instance" "app_instance" {
  ami                  = var.ami
  instance_type        = var.instance_type
  subnet_id            = module.vpc.public_subnets[0]
  key_name             = var.key_pair_name
  user_data            = base64encode(file("user_data.sh"))
  vpc_security_group_ids = [aws_security_group.sg_for_instance.id]
  iam_instance_profile   = aws_iam_instance_profile.ec2_instance_profile.name
  tags = {Name = "${var.project}-app"}}
```

Крім цього, EC2-інстанс асоціюється з відповідною security group та IAM-профілем, що забезпечує необхідний рівень безпеки та доступу до AWS-сервісів.

Шаблон створення балансувальника навантаження реалізує створення та налаштування Application Load Balancer, який забезпечує розподіл вхідного трафіку між різними сервісами платформи, а саме:

```
listeners = {
  https = {
    port           = 443
    protocol       = "HTTPS"
    certificate_arn = module.acm.acm_certificate_arn
    forward = {target_group_key = "frontend"}
    rules = {
      api = {
        priority =  1
        conditions = [{path_pattern = { values = ["/api/*"] }}]
        actions = [{
```

```
      type             = "forward"
      target_group_key = "backend"}]}}}}
```

Особливістю є використання HTTPS слухача з підключенням SSL-сертифіката та гнучке маршрутизаційне правило, що дозволяє розділяти трафік між фронтендом і бекендом на основі шляху запиту.

Основною всієї хмарної інфраструктури є створення ізольованої віртуальної мережі із публічними та приватними підмережами, що дозволяє розділяти ресурси за рівнями доступу та підвищувати безпеку архітектури:

Реалізація модуля "vpc" має такий зміст:

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  name = "Racemind-vpc"
  cidr = "10.0.0.0/16"
  public_subnets   = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]
  private_subnets  = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  map_public_ip_on_launch = true}
```

Для того, щоб ec2 сервер міг коректно взаємодіяти з іншими сервісами AWS, використовується темплейт, який створює IAM роль і прив’язує до відповідного інстансу, а саме:

```
resource "aws_iam_role" "ec2_role" {
  name = "${var.project}-ec2-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action = "sts:AssumeRole"
      Effect = "Allow"
      Principal = { Service = "ec2.amazonaws.com" }
    }]})}
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "${var.project}-ec2-profile"
  role = aws_iam_role.ec2_role.name}
```

3.7 Тестування роботи програми та аналіз результатів

Тестування програмного забезпечення є важливою складовою процесу його розробки, яка спрямована на забезпечення високої якості, стабільності, безпеки та ефективності роботи продукту. Воно дозволяє знизити витрати на усунення дефектів, підвищити рівень задоволеності користувачів і забезпечити відповідність встановленим вимогам та технічній документації.

Було проведено детальне тестування онлайн-платформи (понад 100 тестових запусків). При вході на онлайн-платформу користувач одразу бачить головну сторінку. На рисунку 3.1 представлено загальний вигляд інтерфейсу головної сторінки онлайн-платформи.

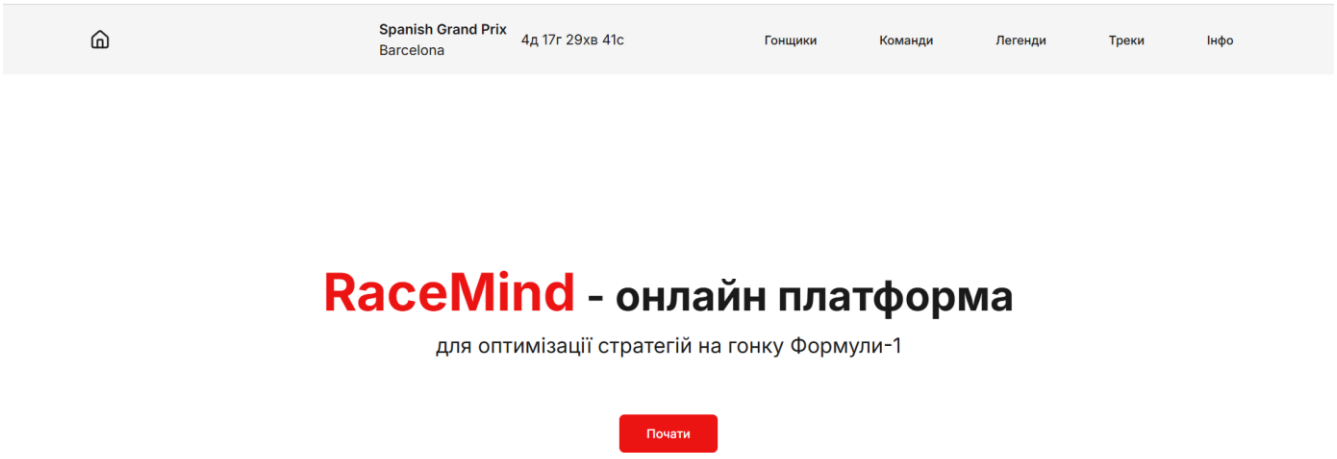


Рисунок 3.1 – Загальний вигляд інтерфейсу головної сторінки онлайн-платформи

На цій сторінці можна дізнатися про платформу або перейти відразу до підбору стратегії натиснувши на кнопку «Почати». Натиснувши, користувач

перейде на сторінку вибору треку, на цій сторінці в одному рядку буде 4 треки, щоб бачити інші, користувачу потрібно рухати повзунок донизу (рис. 3.2).

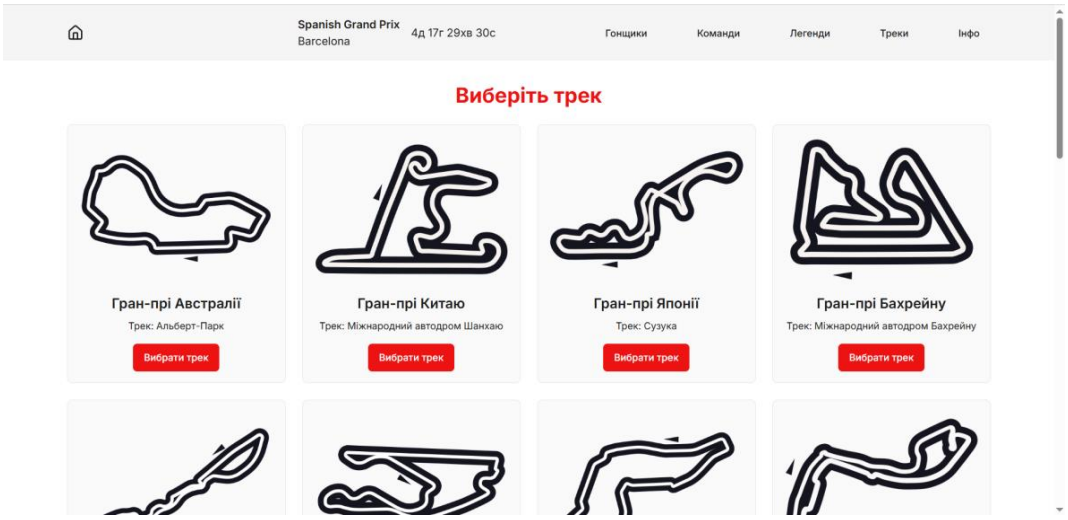


Рисунок 3.2 – Загальний вигляд інтерфейсу сторінки вибору треку

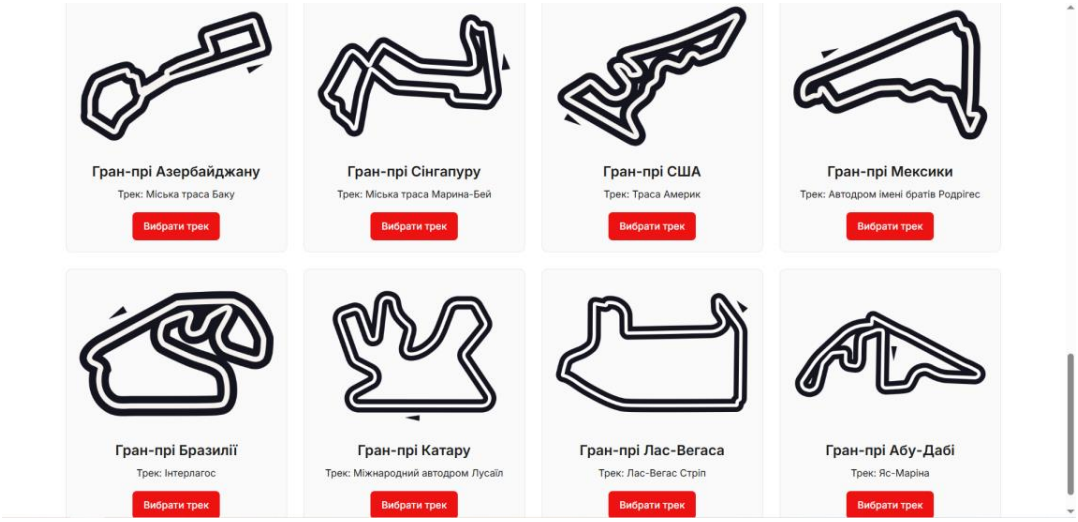


Рисунок 3.2, аркуш 2

Після вибору треку, користувача перекидає на сторінку вибору команди. Принцип дії аналогічний, як і на попередній сторінці. Щоб побачити більше команд, потрібно рухати повзунок донизу. На рисунку 3.3 представлено сторінку вибору команд.

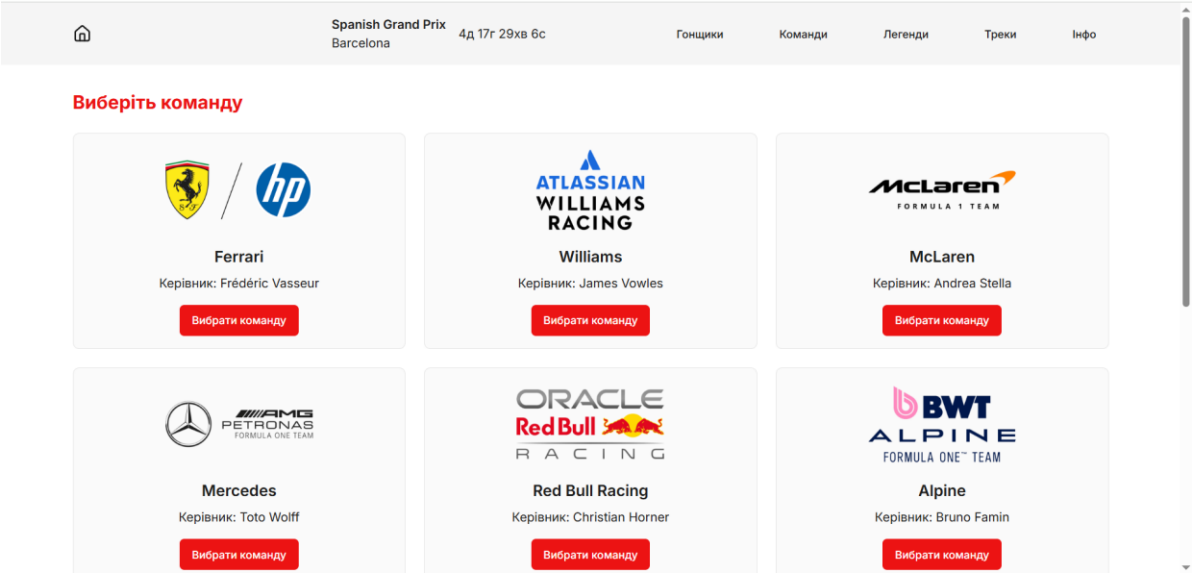


Рисунок 3.3 – Загальний вигляд інтерфейсу сторінки вибору команди

Потім користувачу буде потрібно вибрати гонщика серед двох, які виступають за обрану команду. На рисунку 3.4 представлено сторінку вибору гонщика.

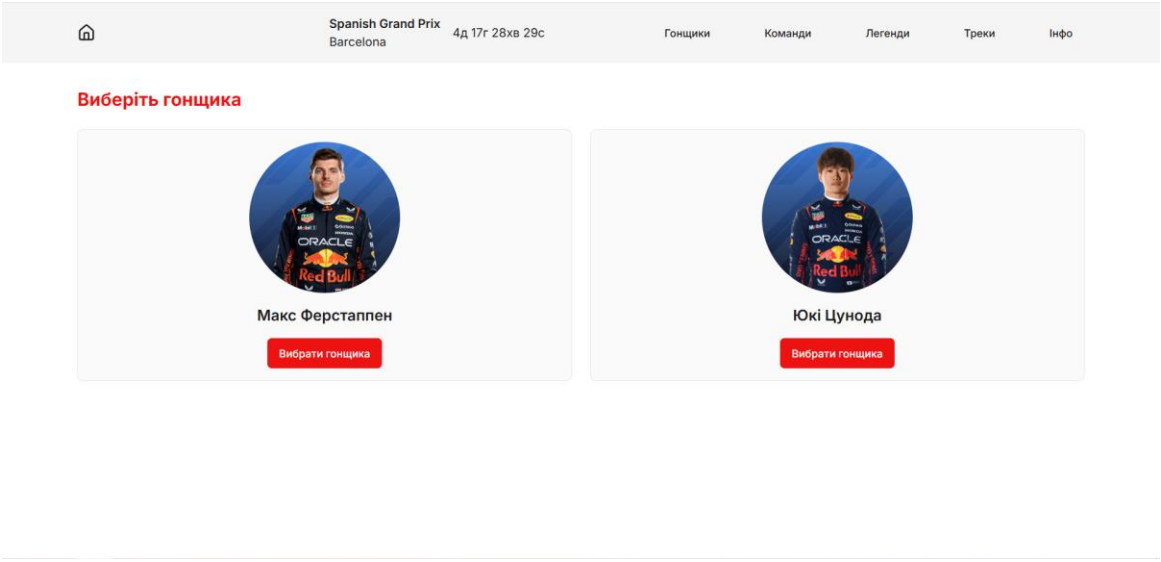


Рисунок 3.4 – Загальний вигляд інтерфейсу сторінки вибору гонщика

Після того як користувач вибере бажаного гонщика його переадресує на сторінку підбору стратегії. На ній користувач може переобрати трасу, команду та гонщика, відповідно як це було на етапі вибору. Ще однією важливою функцією є вибір погоди на конкретний тип сесії: чи практика, чи кваліфікація, чи для вибору стратегії на гонку. На рисунку 3.5 показана сторінка підбору стратегії на гонку з врахуванням різної погоди.

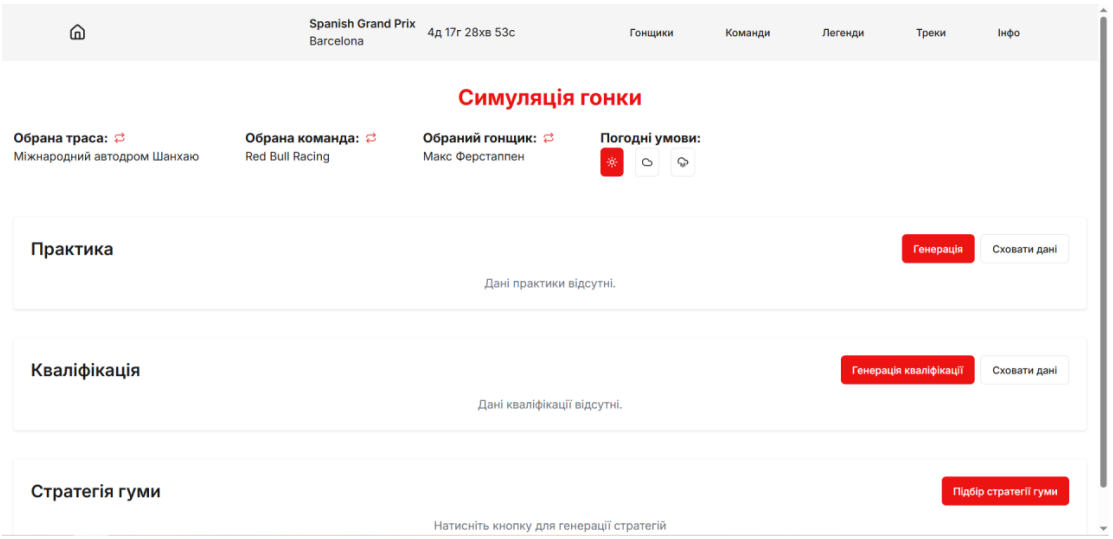


Рисунок 3.5 – Загальний вигляд інтерфейсу сторінки підбору стратегії з врахуванням різної погоди

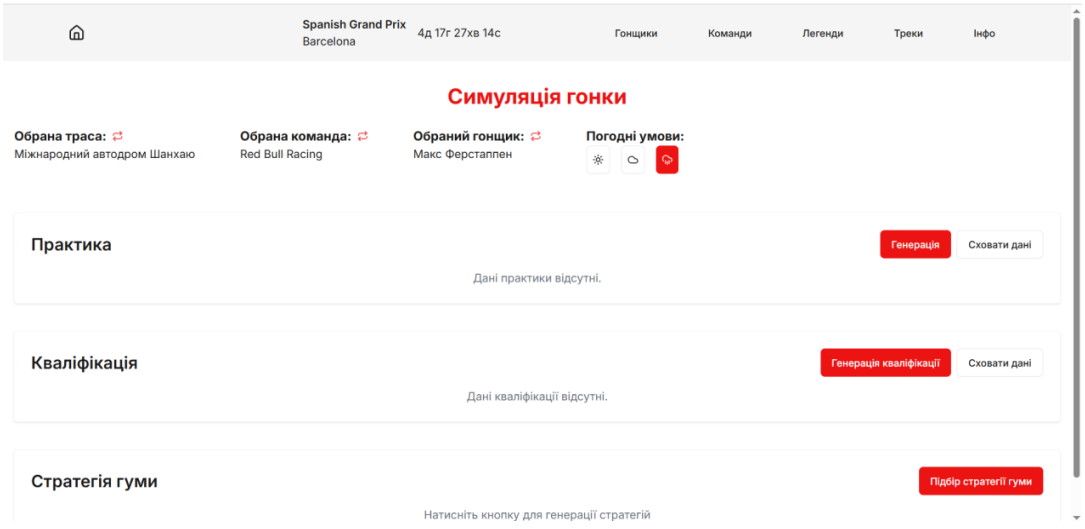


Рисунок 3.5, аркуш 2

У разі якщо вибрано трек на якому сильні дощі не відбуваються, то платформа повідомить, що тут можливі варіанти лише сухі або хмарні погодні умови. На рисунку 3.6 подано варіант сторінки, коли не можна використовувати дощову погоду.

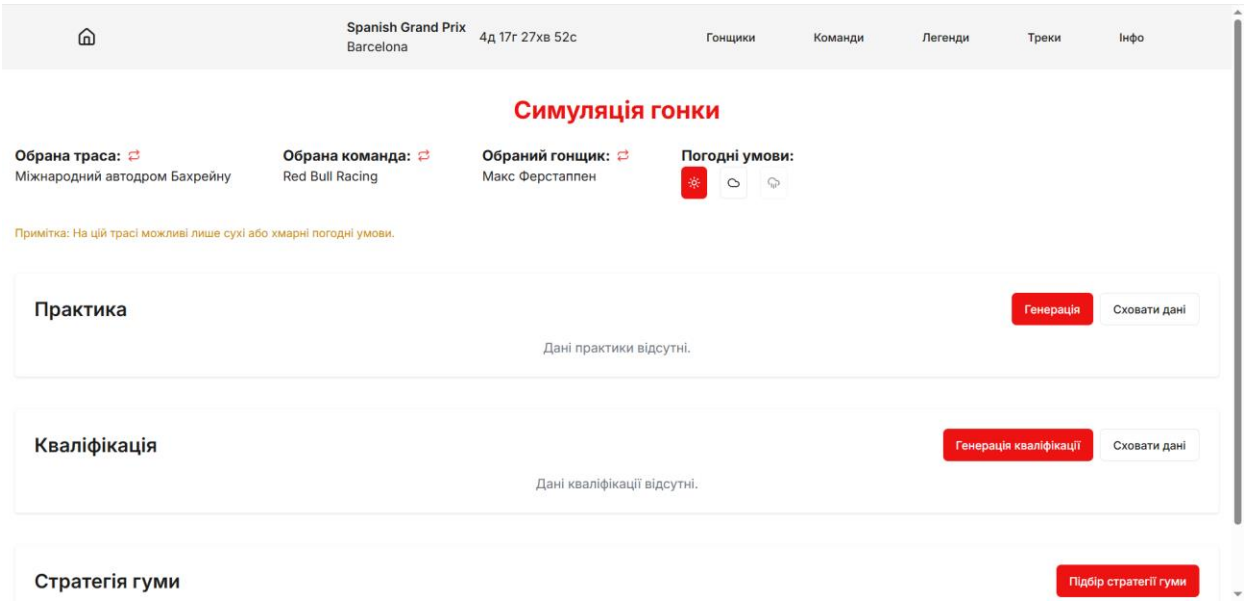
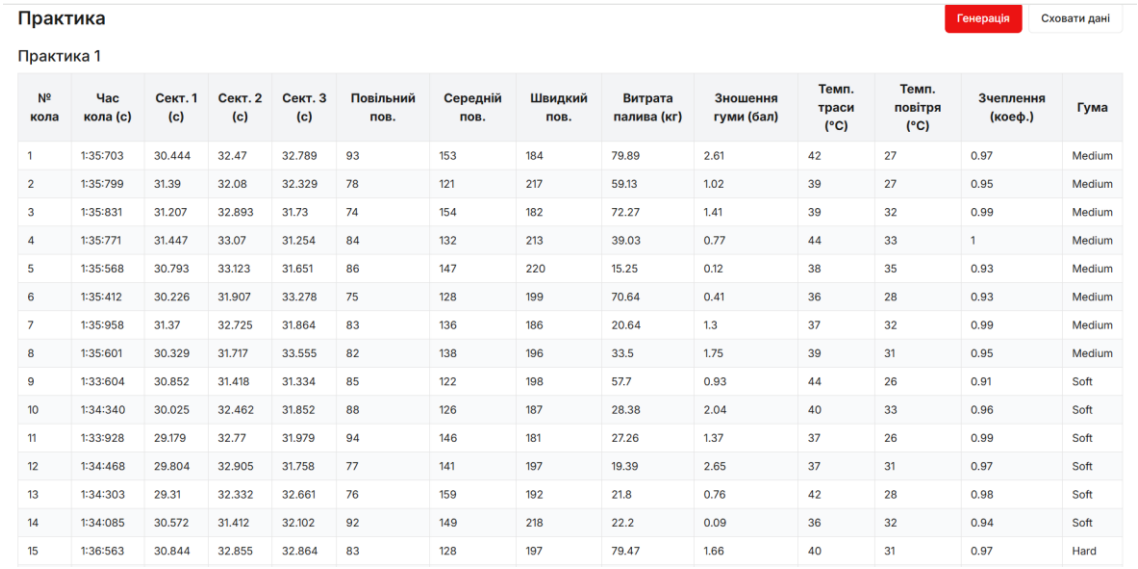


Рисунок 3.6 – Варіант загального вигляду інтерфейсу сторінки, коли не можна використовувати дощову погоду

Користувач може згенерувати дані про 1 – 3 практики, які потім будуть використовуватися для підбору стратегії. Після чого їх можна буде сховати натиснувши кнопку «Сховати дані». Важливим моментом є те, що результати практик можна перегенерувати, натиснувши на «Генерація». Результати генерації можна побачити на рисунку 3.7.



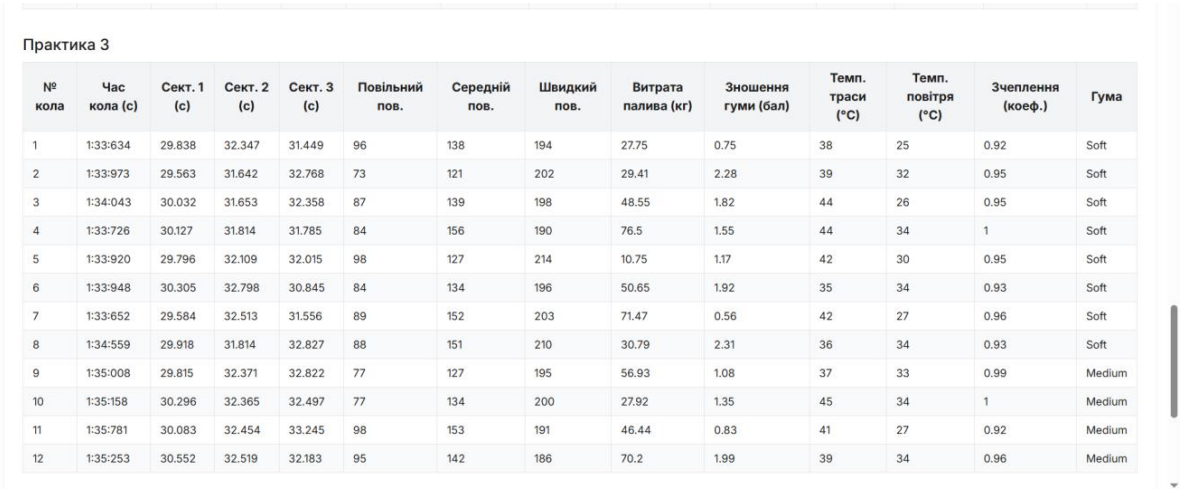
Практика

Генерація | Сховати дані

Практика 1

№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:35:703	30.444	32.47	32.789	93	153	184	79.89	2.61	42	27	0.97	Medium
2	1:35:799	31.39	32.08	32.329	78	121	217	59.13	1.02	39	27	0.95	Medium
3	1:35:831	31.207	32.893	31.73	74	154	182	72.27	1.41	39	32	0.99	Medium
4	1:35:771	31.447	33.07	31.254	84	132	213	39.03	0.77	44	33	1	Medium
5	1:35:568	30.793	33.123	31.651	86	147	220	15.25	0.12	38	35	0.93	Medium
6	1:35:412	30.226	31.907	33.278	75	128	199	70.64	0.41	36	28	0.93	Medium
7	1:35:958	31.37	32.725	31.864	83	136	186	20.64	1.3	37	32	0.99	Medium
8	1:35:601	30.329	31.717	33.555	82	138	196	33.5	1.75	39	31	0.95	Medium
9	1:33:604	30.852	31.418	31.334	85	122	198	57.7	0.93	44	26	0.91	Soft
10	1:34:340	30.025	32.462	31.852	88	126	187	28.38	2.04	40	33	0.96	Soft
11	1:33:928	29.179	32.77	31.979	94	146	181	27.26	1.37	37	26	0.99	Soft
12	1:34:468	29.804	32.905	31.758	77	141	197	19.39	2.65	37	31	0.97	Soft
13	1:34:303	29.31	32.332	32.661	76	159	192	21.8	0.76	42	28	0.98	Soft
14	1:34:085	30.572	31.412	32.102	92	149	218	22.2	0.09	36	32	0.94	Soft
15	1:36:563	30.844	32.855	32.864	83	128	197	79.47	1.66	40	31	0.97	Hard

Рисунок 3.7 – Загальний вигляд інтерфейсу результату генерації практик 1 – 3

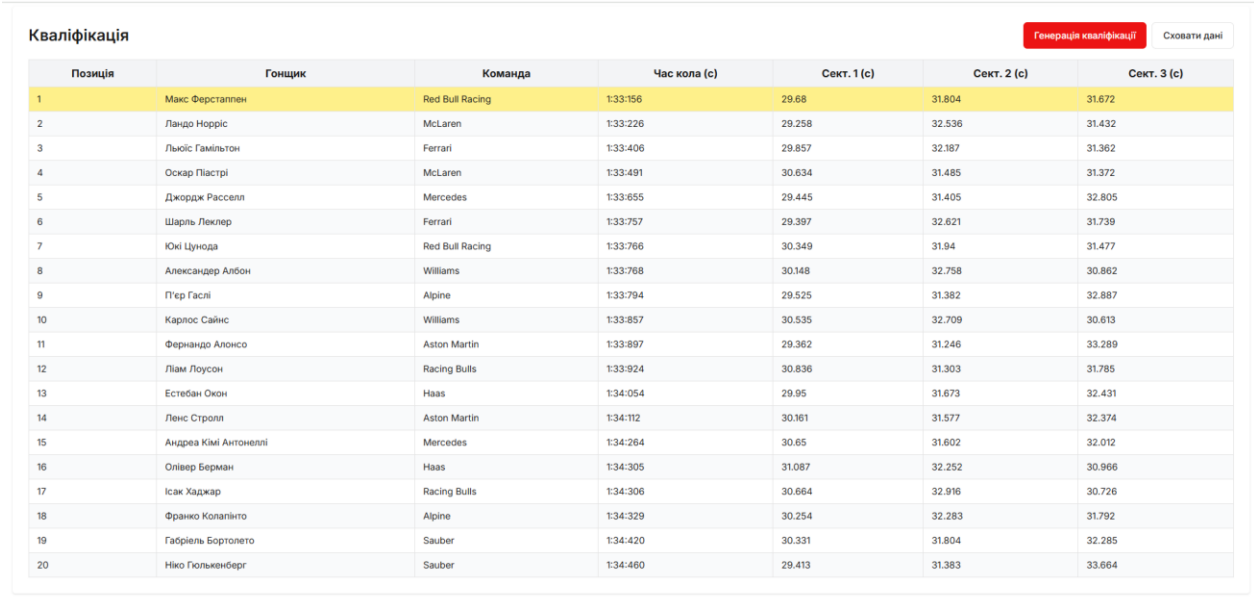


Практика 3

№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:33:634	29.838	32.347	31.449	96	138	194	27.75	0.75	38	25	0.92	Soft
2	1:33:973	29.563	31.642	32.768	73	121	202	29.41	2.28	39	32	0.95	Soft
3	1:34:043	30.032	31.653	32.358	87	139	198	48.55	1.82	44	26	0.95	Soft
4	1:33:726	30.127	31.814	31.785	84	156	190	76.5	1.55	44	34	1	Soft
5	1:33:920	29.796	32.109	32.015	98	127	214	10.75	1.17	42	30	0.95	Soft
6	1:33:948	30.305	32.798	30.845	84	134	196	50.65	1.92	35	34	0.93	Soft
7	1:33:652	29.584	32.513	31.556	89	152	203	71.47	0.56	42	27	0.96	Soft
8	1:34:559	29.918	31.814	32.827	88	151	210	30.79	2.31	36	34	0.93	Soft
9	1:35:008	29.815	32.371	32.822	77	127	195	56.93	1.08	37	33	0.99	Medium
10	1:35:158	30.296	32.365	32.497	77	134	200	27.92	1.35	45	34	1	Medium
11	1:35:781	30.083	32.454	33.245	98	153	191	46.44	0.83	41	27	0.92	Medium
12	1:35:253	30.552	32.519	32.183	95	142	186	70.2	1.99	39	34	0.96	Medium

Рисунок 3.7, аркуш 2

Згодом користувачу потрібно згенерувати дані кваліфікації. Їх як і практики можна сховати або перегенерувати. Гонщика, якого обрав користувач буде підсвічено жовтим, щоб легше зрозуміти позицію гонщика. На рисунку 3.8 представлені результати генерації кваліфікації.



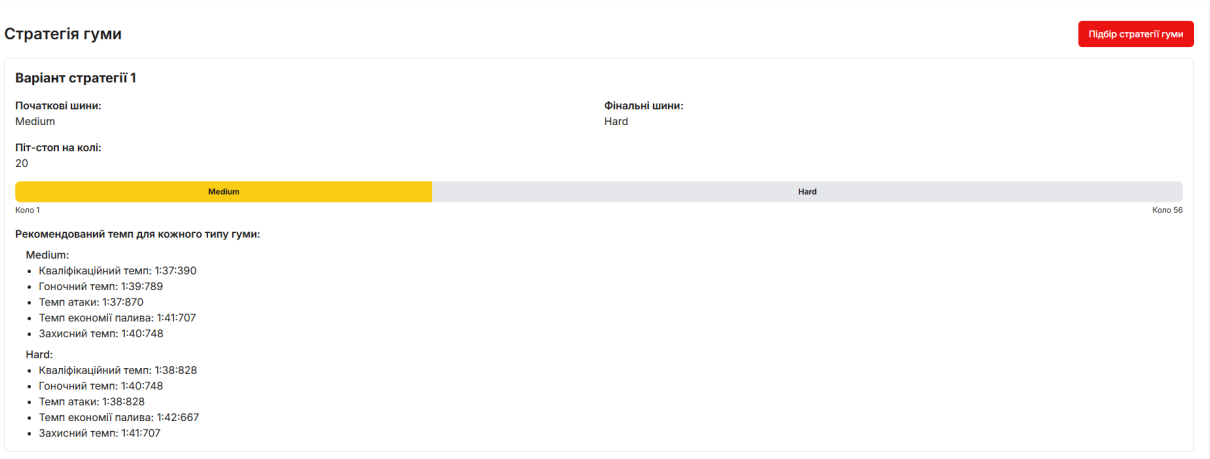
Кваліфікація

Генерація кваліфікації | Сховати дані

Позиція	Гонщик	Команда	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)
1	Макс Ферстаппен	Red Bull Racing	1:33:156	29.68	31.804	31.672
2	Ландо Норріс	McLaren	1:33:226	29.258	32.536	31.432
3	Льюіс Гамільтон	Ferrari	1:33:406	29.857	32.187	31.362
4	Оскар Піастрі	McLaren	1:33:491	30.634	31.485	31.372
5	Джордж Расселл	Mercedes	1:33:655	29.445	31.405	32.805
6	Шарль Леклер	Ferrari	1:33:757	29.397	32.621	31.739
7	Юкі Цунода	Red Bull Racing	1:33:766	30.349	31.94	31.477
8	Александер Албон	Williams	1:33:768	30.148	32.758	30.862
9	П'єр Гаслі	Alpine	1:33:794	29.525	31.382	32.887
10	Карлос Сайнс	Williams	1:33:857	30.535	32.709	30.613
11	Фернандо Алонсо	Aston Martin	1:33:897	29.362	31.246	33.289
12	Ліам Лоусон	Racing Bulls	1:33:924	30.836	31.303	31.785
13	Естебан Окон	Haas	1:34:054	29.95	31.673	32.431
14	Ленс Стролл	Aston Martin	1:34:112	30.161	31.577	32.374
15	Андреа Кімі Антонеллі	Mercedes	1:34:264	30.65	31.602	32.012
16	Олівер Берман	Haas	1:34:305	31.087	32.252	30.966
17	Ісак Хаджар	Racing Bulls	1:34:306	30.664	32.916	30.726
18	Франко Колапінто	Alpine	1:34:329	30.254	32.283	31.792
19	Габріель Бортолето	Sauber	1:34:420	30.331	31.804	32.285
20	Ніко Гюлькенберг	Sauber	1:34:460	29.413	31.383	33.664

Рисунок 3.8 – Загальний вигляд інтерфейсу результату генерації кваліфікації

Потім користувачу потрібно натиснути кнопку «Підбір стратегії гуми», система виведе на сторінку найкращий варіант, відповідно згенерованих раніше даних. Також виведе рекомендований оптимальний час їзди на кожному комплекту гуми для різних ситуацій на треку. Результат підбору стратегії на гонку, можна побачити на рисунку 3.9.



Стратегія гуми

Підбір стратегії гуми

Варіант стратегії 1

Початкові шини:
Medium

Фінальні шини:
Hard

Піт-стоп на колі:
20

Medium | Hard

Коло 1 | Коло 56

Рекомендований темп для кожного типу гуми:

Medium:
- Кваліфікаційний темп: 1:37:390
- Гоночний темп: 1:39:789
- Темп атаки: 1:37:870
- Темп економії палива: 1:41:707
- Захисний темп: 1:40:748

Hard:
- Кваліфікаційний темп: 1:38:828
- Гоночний темп: 1:40:748
- Темп атаки: 1:38:828
- Темп економії палива: 1:42:667
- Захисний темп: 1:41:707

Рисунок 3.9 – Загальний вигляд інтерфейсу результату підбору стратегії на гонку

Отже, проаналізувавши та протестувавши онлайн платформу, можна дійти висновку, що поставлених цілей було досягнуто.

В таблиці 3.5 можна побачити результати тестування розробленої онлайн-платформи та аналогів. В результаті розробки було реалізовано функціонал, який був відсутній в подібних платформах:

- надає можливість ознайомитися з основними аспектами Формули-1.
- розроблена онлайн-платформа має безкоштовний доступ до всього функціоналу.
- дозволяє аналізувати та підбирати найкращу стратегію, яка відповідає потребам користувача.
- використання надійної та гарно масштабованої хмарної архітектури, яка має високу продуктивність та доступність у різних куточках планети.

Таблиця 3.5 – Результати тестування розробленої онлайн-платформи та аналогів.

	Розроблена онлайн-платформа	Офіційний сайт Формули-1	Pitwall	Speed-Wiz
Ознайомлення з Формулою-1	+	+	-	-
Наявність підписки	-	+	+	-
Підбір стратегії на гонку	+	-	-	+
Висока доступність	+	+	+	-
Висока продуктивність	+	+	-	-

Отже, проаналізувавши та протестувавши розроблену платформу підбору стратегії гонки Формула-1, можна стверджувати, що всі поставлені цілі успішно реалізовані. Зі змісту таблиці 3.5 видно, що онлайн-платформа для вибору стратегії

гонки Формула-1 перевершує наявні рішення щонайменше за 4 ключовими параметрами: надає можливість ознайомитися з основними аспектами Формули-1, має безкоштовний доступ до всього функціоналу; дозволяє підбирати стратегію, яка відповідає потребам користувача; використання надійної та гарно масштабованої хмарної архітектури, яка має високу продуктивність та доступність у різних куточках планети.

3.8 Висновок

У третьому розділі було детально описано повну програмну реалізацію онлайн-платформи для підбору стратегії гонки Формула-1, починаючи від обґрунтування вибору технологій і завершуючи всебічним тестуванням отриманих результатів. Для розробки клієнтської частини обрано TypeScript та Next.js із Tailwind CSS, що забезпечує швидкий рендеринг, статичну типізацію та компонентний підхід. Використано можливості Next.js для статичної генерації сторінок і серверного рендерингу, що значно знижує час першого завантаження. Для уніфікації інтерфейсу застосовано Radix UI-примітиви, які разом із системою адаптивної верстки Tailwind CSS гарантують коректне відображення.

На серверній частині застосовано Python із FastAPI, що забезпечують високу пропускну здатність і можливість обробляти тисячі запитів на секунду. Для роботи з даними обрано asyncpg для асинхронної взаємодії з PostgreSQL, а модулі прогнозування зносу шин і оптимізації стратегії гонки реалізовано на основі TensorFlow і scikit-learn.

Інфраструктуру розгорнуто за допомогою Terraform в AWS. Декларативно описано створення VPC, підмереж, RDS-бази даних, EC2-інстансів із авто-масштабуванням та Application Load Balancer для рівномірного розподілу трафіку.

Використано S3 для зберігання статичних ресурсів, що гарантує відмовостійкість, масштабованість і безпечний доступ до всіх сервісів.

- Проведене тестування підтвердило, що платформа відповідає функціональним вимогам: від вибору траси, команди й пілота до генерації детальних даних практик, кваліфікації та побудови оптимальної стратегії шин. Розробка демонструє покращений функціонал порівняно з аналогами на 3 можливості, а саме, має безкоштовний доступ до всього функціоналу; дозволяє аналізувати та підбирати найкращу стратегію, яка відповідає потребам користувача; використання надійної та гарно масштабованої хмарної архітектури, яка має високу продуктивність та доступність у різних куточках планети.

Результати демонструють високу продуктивність, доступність і зручність використання в порівнянні з існуючими Таким чином, програмна реалізація повністю досягла поставлених цілей.

ВИСНОВКИ

Всі задачі, поставлені для реалізації онлайн-платформи підбору стратегії гонок Формула-1 на основі хмарних обчислень, були виконані в повному обсязі, а саме: обґрунтовано доцільність розробки онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень; спроектовано архітектуру онлайн-платформи для обробки великих обсягів даних та виконання складних обчислень; розроблено програмний модуль для аналізу та моделювання стратегій гонки з використанням технологій хмарних обчислень; виконано тестування та проведено аналіз отриманих результатів; розроблено інструкцію користувача з експлуатації платформи.

Для розробки клієнтської частини використано TypeScript із Next.js і Tailwind CSS, що забезпечує сучасний та адаптивний інтерфейс. Серверну логіку реалізовано на Python із застосуванням FastAPI та бібліотек для машинного навчання, для управління базою даних використано PostgreSQL. Інфраструктуру організовано через AWS за допомогою Terraform. Це сприятиме значному скороченню часу розробки завдяки уніфікованому стеку технологій, що зменшує кількість контекстних перемикань між різними мовами та фреймворками.

Для реалізації було розроблено архітектуру платформи з виділеними модулями, моделювання стратегій та візуалізації результатів. Побудовано UML-діаграми класів для клієнтської та серверної частин, сформовано ER-модель бази даних, а також розроблено схему алгоритму взаємодії. Це дозволить масштабувати систему з мінімальними зусиллями, оскільки модульна архітектура підтримує незалежне розгортання й оновлення компонентів без впливу на інші сервіси.

Розроблений продукт успішно пройшов тестування коректності роботи, а порівняльний аналіз із відомими платформами показав не менше трьох додаткових можливостей.

Мета роботи, яка полягала в розширенні функціональних можливостей онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень досягнута за рахунок введення таких 4 додаткових можливостей: можливість ознайомитися з Формулою-1, безкоштовний доступ до всього функціоналу; дозволяє аналізувати та підбирати стратегію, яка відповідає потребам користувача; використання надійної та гарно масштабованої хмарної архітектури, яка має високу продуктивність та доступність у різних куточках планети.

Робота виконана у повному обсязі відповідно до поставлених цілей, вимог та методичних рекомендацій щодо підготовки бакалаврських кваліфікаційних робіт. Усі етапи були реалізовані належним чином, що підтверджує якість, завершеність і практичну цінність розробленого рішення [33].

Отже, створена онлайн-платформа підбору стратегії гонки Формули-1 повністю відповідає поставленим вимогам та готова до експлуатації професійними командами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Довгополюк В. О., Крилик Л. В. Аналіз передумов розробки онлайн-платформи для вибору стратегії гонки Формули-1 на основі хмарних обчислень. *LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації.* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23133/19144 (дата звернення: 05.05.2025).

2. Що таке хмарні обчислення? URL: https://www.guru99.com/uk/what-is-cloud-computing-with-example.html (дата звернення: 05.05.2025).

3. HOW Formula 1 Is Using The Power Of AI & Cloud. URL: https://medium.com/globant/how-formula-1-is-using-the-power-of-ai-cloud-7a24c3245d94 (дата звернення: 05.05.2025).

4. AWS vs Azure vs Google Cloud. URL: https://www.cloudwards.net/aws-vs-azure-vs-google (дата звернення: 05.05.2025).

5. Офіційний сайт Формули-1. URL: https://www.formula1.com (дата звернення: 06.05.2025).

6. Офіційний сайт Pitwall. URL: https://pitwall.live (дата звернення: 06.05.2025).

7. Офіційний сайт Speed-wiz. URL: https://speed-wiz.com (дата звернення: 06.05.2025).

8. Основні типи архітектури програмного забезпечення. URL: https://artofba.com/uk/post/main-types-of-software-architecture (дата звернення: 08.05.2025).

9. Монолітна архітектура. URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura (дата звернення: 06.05.2025).

10. Мікросервісна архітектура. URL: https://medium.com/microservices-architecture-461687045b3d (дата звернення: 06.05.2025).

11. Архітектури сучасних фронтенд-додатків. URL: https://javascript.org.ua/arhitekturi-suchasnih-frontend-dodatkiv (дата звернення: 06.05.2025).

12. Повний огляд REST. URL: https://dou.ua/forums/topic/50364/ (дата звернення: 09.05.2025).

13. Типи баз даних. URL: https://dou.ua/lenta/articles/types-of-databases/ (дата звернення: 09.05.2025).

14. All You Need to Know About UML Diagrams. URL: https://medium.com/javarevisited/all-you-need-to-know-about-uml-diagrams-546c9f191a9d (дата звернення: 09.05.2025).

15. Що таке ООП (OOP). Основні принципи. URL: https://medium.com/madit-story/що-таке-ооп-оор-основні-принципи-8839c5b793ef (дата звернення: 14.05.2025).

16. Савчук Т. О., Ваховська Л. М. Основи інформатики та обчислювальної техніки. *Навчальні посібники Вінницького національного технічного університету. Мережні електронні видання*. URL: https://web.posibnyky.vntu.edu.ua/fitki/1savchuk_osnovy_inform_obchisl_tehn/Topic5. (дата звернення: 14.05.2025).

17. Official website of the cloud provider GCP. URL: https://cloud.google.com (дата звернення: 15.05.2025).

18. Official website of the cloud provider Azure. URL: https://azure.microsoft.com (дата звернення: 15.05.2025).

19. Official website of the cloud provider AWS. URL: https://aws.amazon.com (дата звернення: 15.05.2025).

20. AWS cloud architecture. URL: https://miro.com/diagramming/aws-cloud-architecture (дата звернення: 16.05.2025).

21. Exploring Best Practices in Software Architecture for Cloud Environments. URL: https://medium.com/@agilechris/exploring-best-practices-in-software-architecture-for-cloud-environments-9f5e84b59392 (дата звернення: 16.05.2025).

22. TypeScript – це JavaScript на стероїдах. URL: https://drukarnia.com.ua/articles/typescript-ce-javascript-na-steroyidakh-uAxV2 (дата звернення: 16.05.2025).

23. React Js – Complete beginners guide. URL: https://medium.com/@ajayss/react-js-complete-beginners-guide-e90cc09f23d2 (дата звернення: 16.05.2025).

24. Інтерактивний аналіз даних за допомогою Python. URL: https://medium.com/@vekhnyk/interactive-data-analysis-with-python-d8106b4f8422 (дата звернення: 16.05.2025).

25. FastAPI Documentation. URL: https://fastapi.tiangolo.com (дата звернення: 16.05.2025).

26. SQL Database. URL: https://medium.com/@danielbenhayoun/sql-database-a-brief-overview-792673e4fb0e (дата звернення: 20.05.2025).

27. Object Query Language (OQL). URL: https://iammanicse.medium.com/object-query-language-oql-35a526adef69 (дата звернення: 20.05.2025).

28. What are NoSQL databases? URL: https://medium.com/@interviewready/what-are-nosql-databases-94bdf1b7fa3c (дата звернення: 20.05.2025).

29. Visual Studio Code. URL: https://code.visualstudio.com/docs (дата звернення: 20.05.2025).

30. WebStorm. URL: https://www.jetbrains.com/help/webstorm/meet-webstorm.html (дата звернення: 20.05.2025).

31. Sublime Text. URL: https://www.sublimetext.com (дата звернення: 20.05.2025).

32. Terraform Documentation. URL: https://developer.hashicorp.com/terraform/docs (дата звернення: 23.05.2025).

33. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. Вінниця : ВНТУ, 2023. (PDF, 54 с.).

ДОДАТКИ

ДОДАТОК А (обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,06 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А .А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О. К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В. С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Крилик Л. В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Довгополюк В. О.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (обов'язковий)

Лістинг програми

```
# === acm.tf ===
module "acm" {
  source  = "terraform-aws-modules/acm/aws"
  version = "~> 4.0"
  domain_name = "vladops.click"
  zone_id     = "Z0330362MSH6KFB246EU"
  subject_alternative_names = ["*.vladops.click"]
  wait_for_validation = true
  validation_method   = "DNS"
  tags = {Name = "vladops.click"}}

# === alb.tf ===
module "alb" {
  source  = "terraform-aws-modules/alb/aws"
  version = "9.7.0"
  name   = "my-alb"
  vpc_id = module.vpc.vpc_id
  subnets = [
    module.vpc.public_subnets[0],
    module.vpc.public_subnets[1],]
  enable_deletion_protection = false
  security_group_ingress_rules = {
    all_http = {
      from_port   = 80
      to_port     = 80
      ip_protocol = "tcp"
      description = "HTTP web traffic"
      cidr_ipv4   = "0.0.0.0/0"}
    all_https = {
      from_port   = 443
      to_port     = 443
```

```
    ip_protocol = "tcp"
    description = "HTTPS web traffic"
    cidr_ipv4   = "0.0.0.0/0"}}
security_group_egress_rules = {
  all = {
    ip_protocol = "-1"
    cidr_ipv4   = "0.0.0.0/0"}}
target_groups = {
  frontend = {
    name_prefix = "front"
    protocol    = "HTTP"
    port        = 3000
    target_type = "instance"
    target_id   = aws_instance.app_instance.id
    health_check = {
      enabled             = true
      interval            = 30
      path                = "/"
      port                = "traffic-port"
      healthy_threshold   = 3
      unhealthy_threshold = 3
      timeout             = 6
      protocol            = "HTTP"
      matcher             = "200-399"}}
  backend = {
    name_prefix = "back"
    protocol    = "HTTP"
    port        = 3001
    target_type = "instance"
    target_id   = aws_instance.app_instance.id
    health_check = {
      enabled             = true
      interval            = 30
      path                = "/api/health"
      port                = "traffic-port"
```

```
      healthy_threshold   = 3
      unhealthy_threshold = 3
      timeout             = 6
      protocol            = "HTTP"
      matcher             = "200-399"}}}
  listeners = {
   http = {
    port     = 80
    protocol = "HTTP"
    redirect = {
     port        = "443"
     protocol    = "HTTPS"
     status_code = "HTTP_301"}}
   https = {
    port            = 443
    protocol        = "HTTPS"
    certificate_arn = module.acm.acm_certificate_arn
    forward = {
     target_group_key = "frontend"}
    rules = {
     api = {
      priority =  1
      conditions = [{
       path_pattern = { values = ["/api/*"] }
      }]
      actions = [{
       type             = "forward"
       target_group_key = "backend"
      }]}}}}
  tags = {
   Environment = "Development"
   Project     = "RaceMind"}}

resource "aws_route53_record" "frontend" {
  zone_id = "Z0330362MSH6KFB246EU"
```

```
  name    = "racemind.vladops.click"
  type    = "A"
  alias {
    name                   = module.alb.dns_name
    zone_id                = module.alb.zone_id
    evaluate_target_health = true}}
# === backend.tf ===
terraform {
  cloud {
    organization = "wladslaw-org"
    workspaces {
      name = "RaceMind"}}}

# === ec2.tf ===
resource "aws_instance" "app_instance" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = module.vpc.public_subnets[0]
  key_name               = var.key_pair_name
  user_data              = base64encode(file("user_data.sh"))
  vpc_security_group_ids = [aws_security_group.sg_for_instance.id]
  iam_instance_profile = aws_iam_instance_profile.ec2_instance_profile.name
  tags = {
    Name = "${var.project}-app"}}
resource "aws_security_group" "sg_for_instance" {
  name_prefix = "${var.project}-private-sg"
  description = "Security group for EC2 instance"
  vpc_id      = module.vpc.vpc_id
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]}
  ingress {
    from_port       = 3000
```

```
    to_port         = 3000
    protocol        = "tcp"
    security_groups = [module.alb.security_group_id]}
  ingress {
    from_port       = 3001
    to_port         = 3001
    protocol        = "tcp"
    security_groups = [module.alb.security_group_id]
    description     = "Backend access from ALB"}
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]}

  tags = {
    Name = "${var.project}-app-sg"
  }}
# === iam.tf ===
resource "aws_iam_role" "ec2_role" {
  name = "${var.project}-ec2-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "ec2.amazonaws.com"
        }}] })}
resource "aws_iam_role_policy_attachment" "ec2_role_policy" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"}
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "${var.project}-ec2-profile"
```

```
  role = aws_iam_role.ec2_role.name}

# === vpc.tf ===
data "aws_availability_zones" "available" {
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  name = "my-vpc"
  cidr = "10.0.0.0/16
  azs     =     [data.aws_availability_zones.available.names[0],     data.aws_availability_zones.available.names[1],
data.aws_availability_zones.available.names[2]]
  private_subnets  = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets   = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]
  map_public_ip_on_launch = true
  enable_nat_gateway = false
  public_subnet_tags = {
    Name = "Public-Subnet"}
  private_subnet_tags = {
    Name = "Private-Instance-Subnets"}
```

ДОДАТОК В (обов'язковий)

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ВИБОРУ СТРАТЕГІЇ ГОНКИ ФОРМУЛА-1 НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ»

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Довгополюк В.О.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Крилик Л.В.
(прізвище та ініціали)

«03» червня 2025 р.

Вінниця ВНТУ - 2025 рік

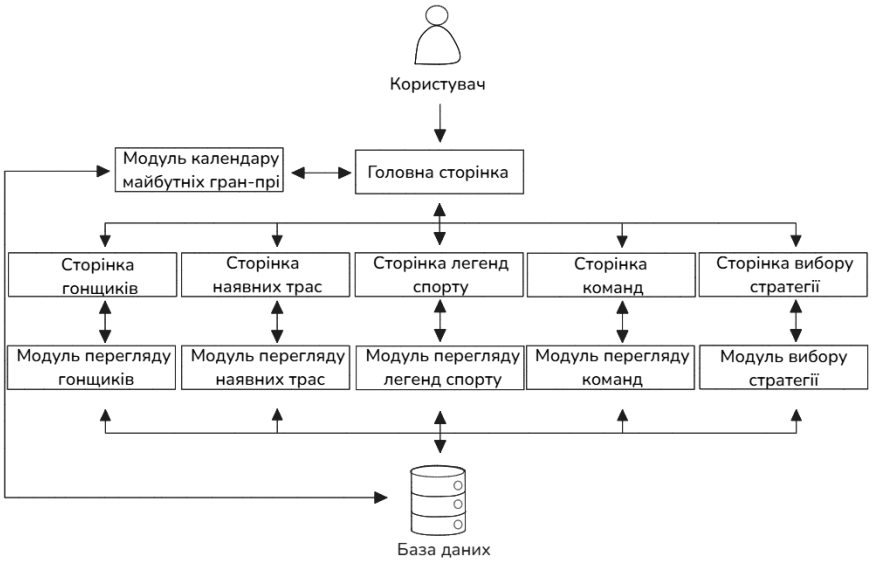


Рисунок В.1 − Загальна структурна схема функціонування системи

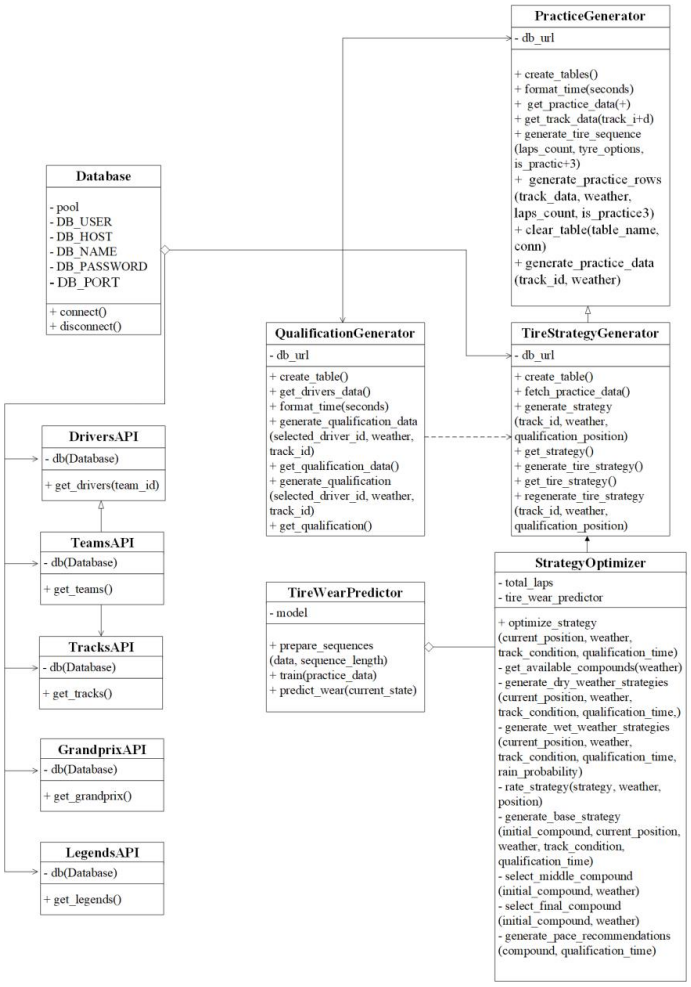


Рисунок В.2 – UML-діаграма класів бекенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

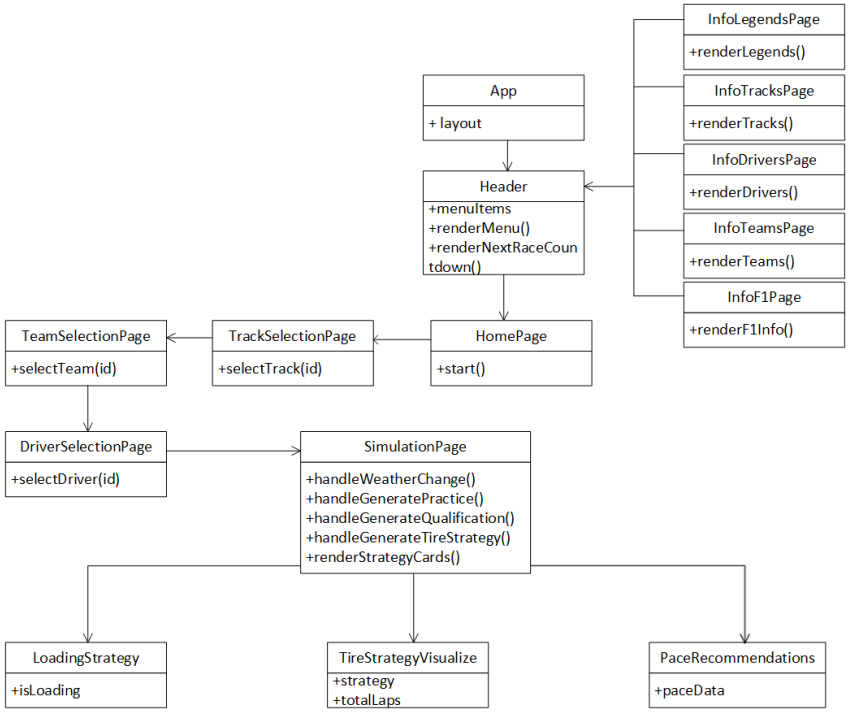


Рисунок В.3 – UML-діаграма класів фронтенд частини онлайн-платформи для вибору стратегії гонки Формула-1 на основі хмарних обчислень

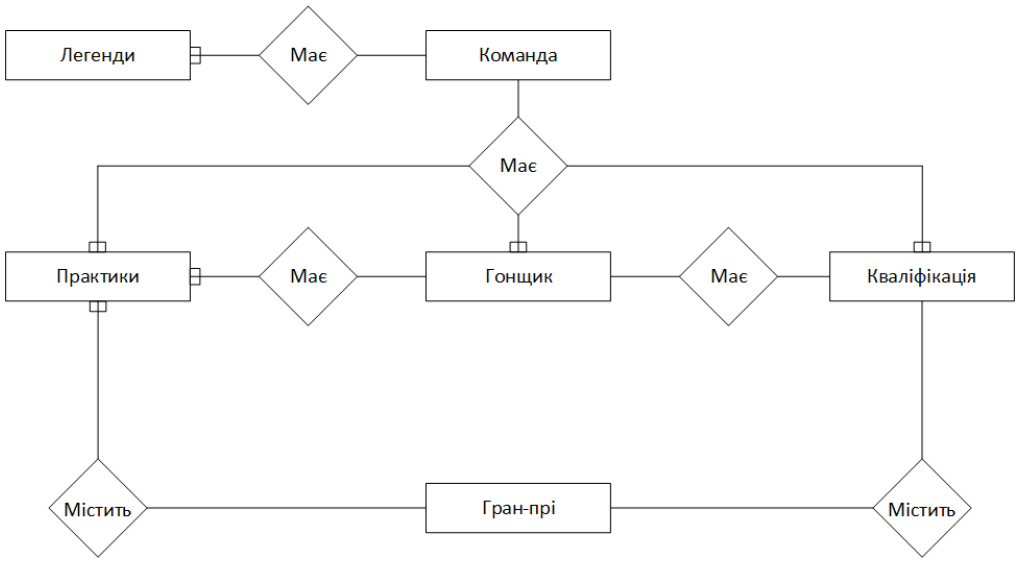


Рисунок В.4 – ER-модель для бази даних

Рисунок В.5 – Схема алгоритму роботи онлайн-платформи

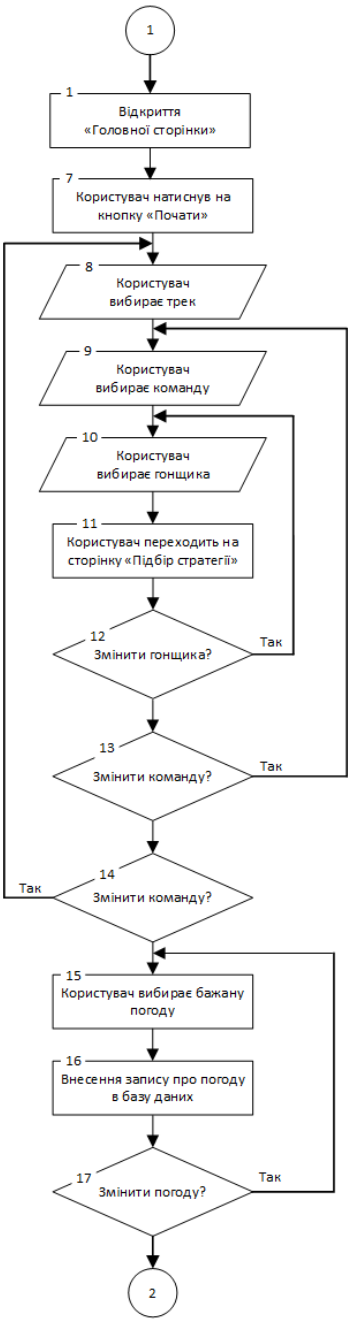


Рисунок В.5, аркуш 2

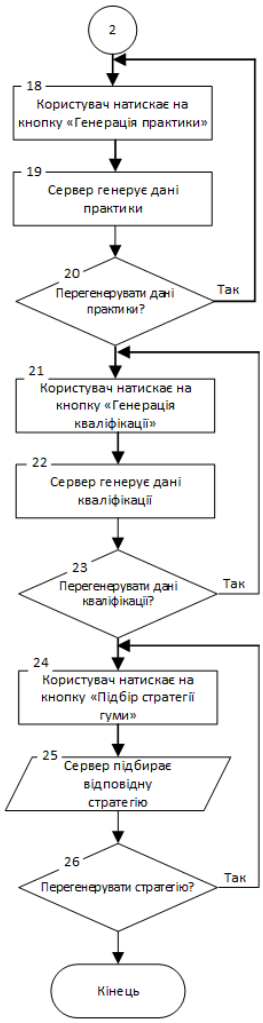


Рисунок В.5, аркуш 3

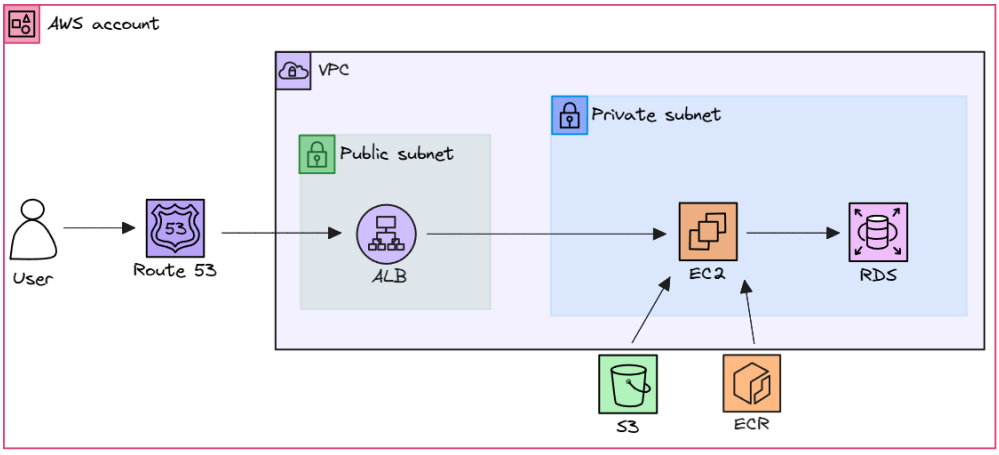


Рисунок В.6 – Архітектурна діаграма хмарного середовища онлайн-платформи для вибору стратегії гонки Формула-1

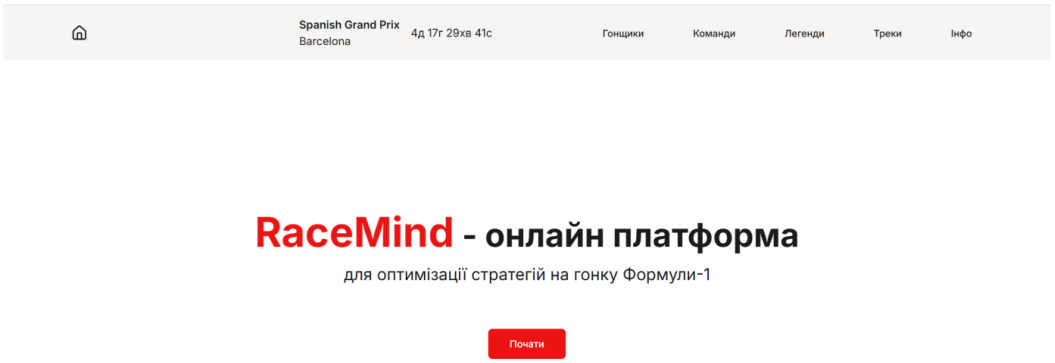


Рисунок В.7– Загальний вигляд інтерфейсу головної сторінки онлайн-платформи

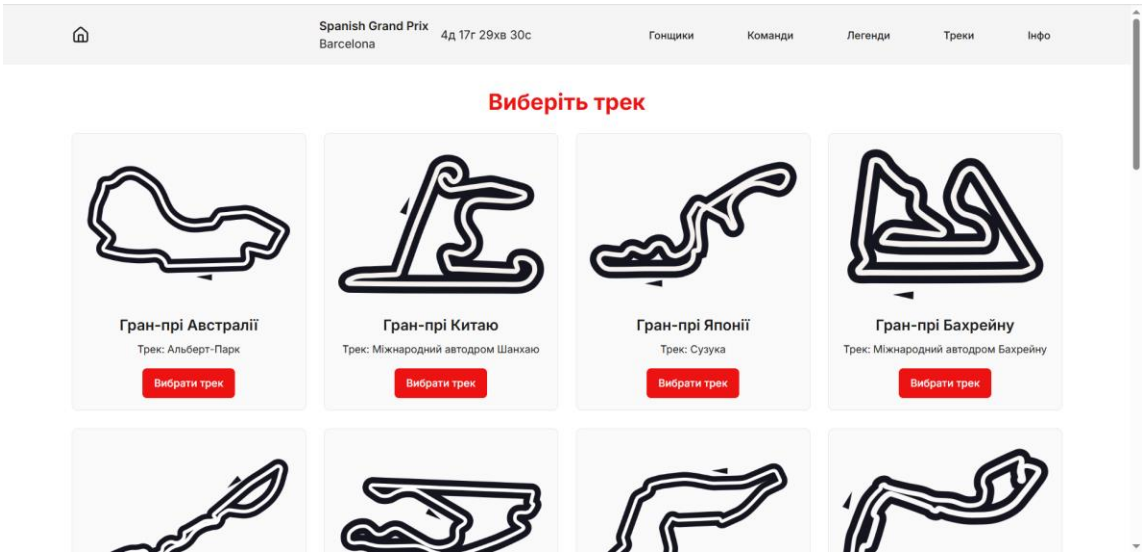


Рисунок В.8 – Загальний вигляд інтерфейсу сторінки вибору треку

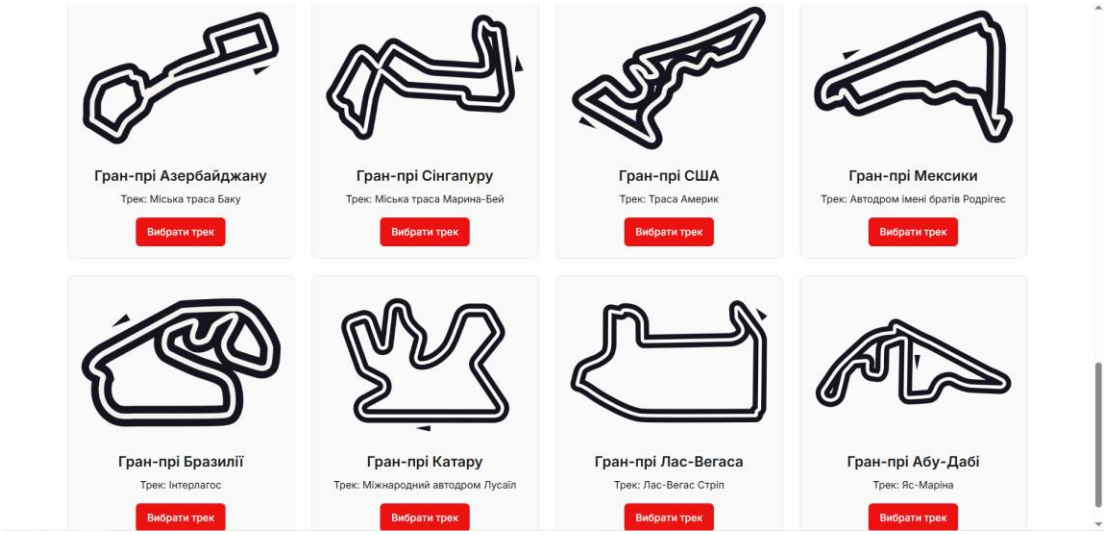


Рисунок В.8, аркуш 2

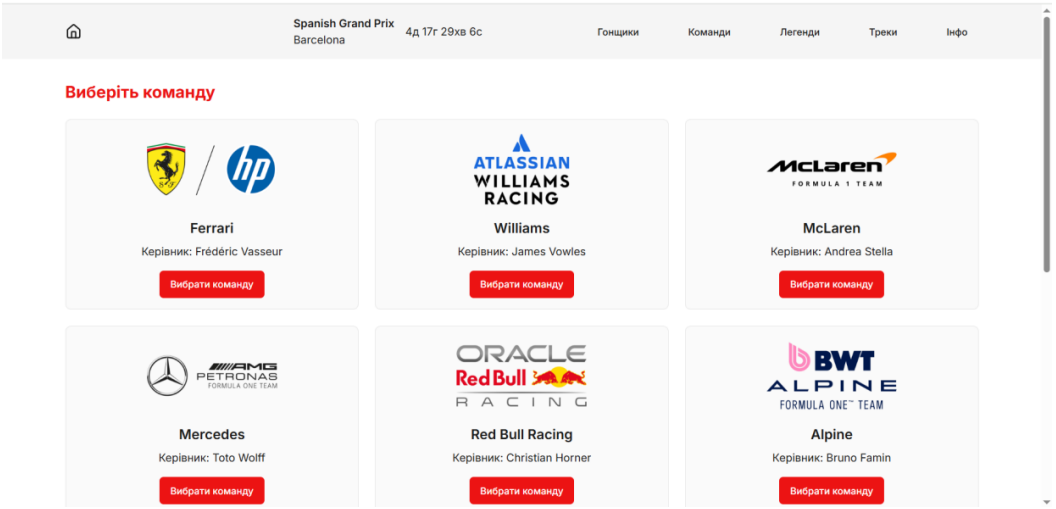


Рисунок В.9 – Загальний вигляд інтерфейсу сторінки вибору команди

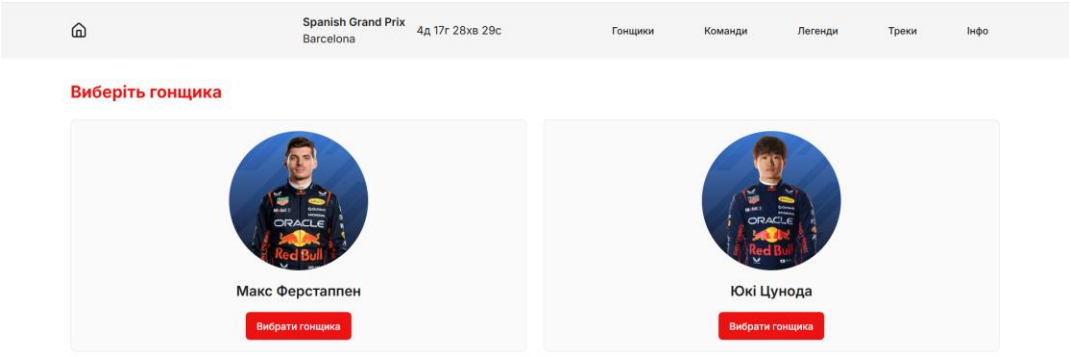


Рисунок В.10 – Загальний вигляд інтерфейсу сторінки вибору гонщика

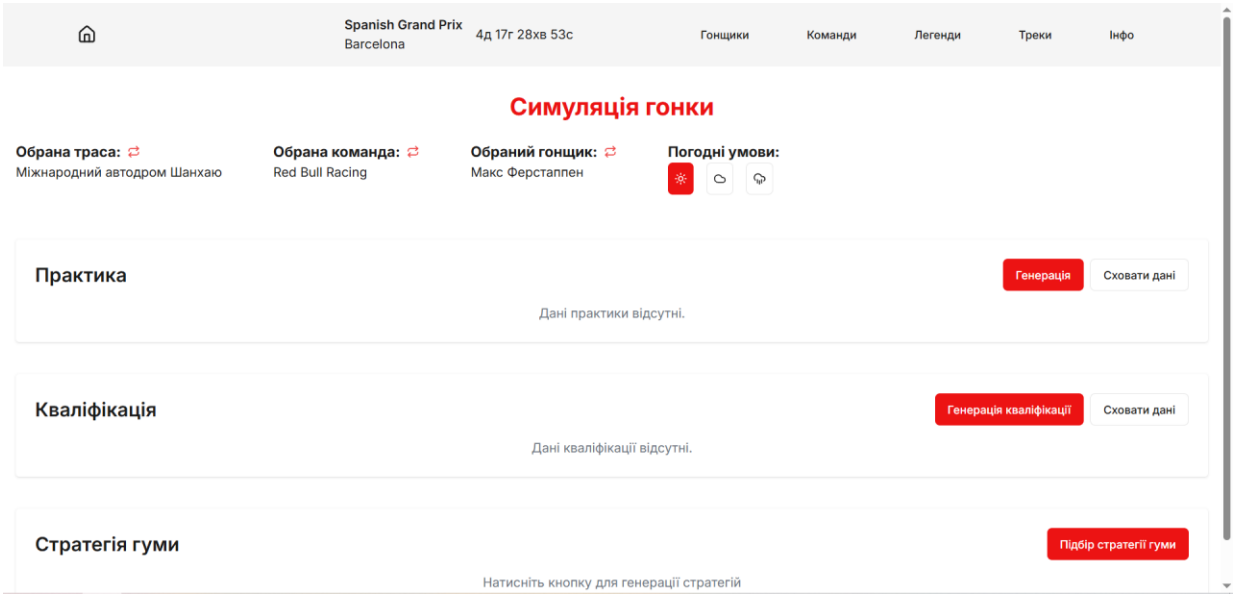


Рисунок В.11 – Загальний вигляд інтерфейсу сторінки підбору стратегії з врахуванням різної погоди

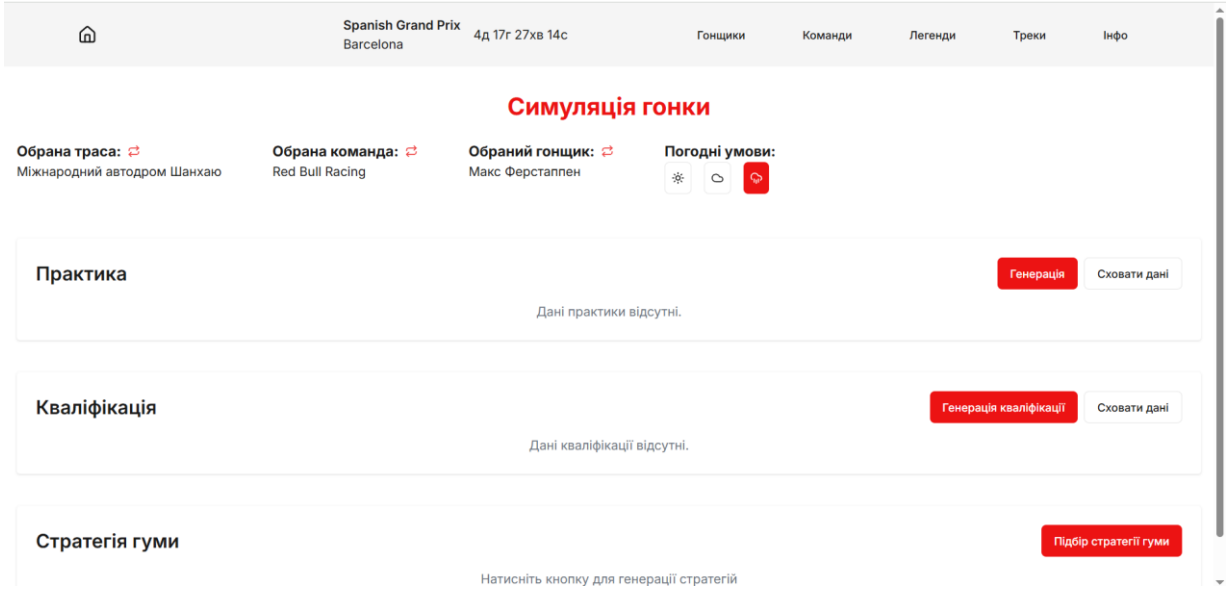


Рисунок В.11, аркуш 2

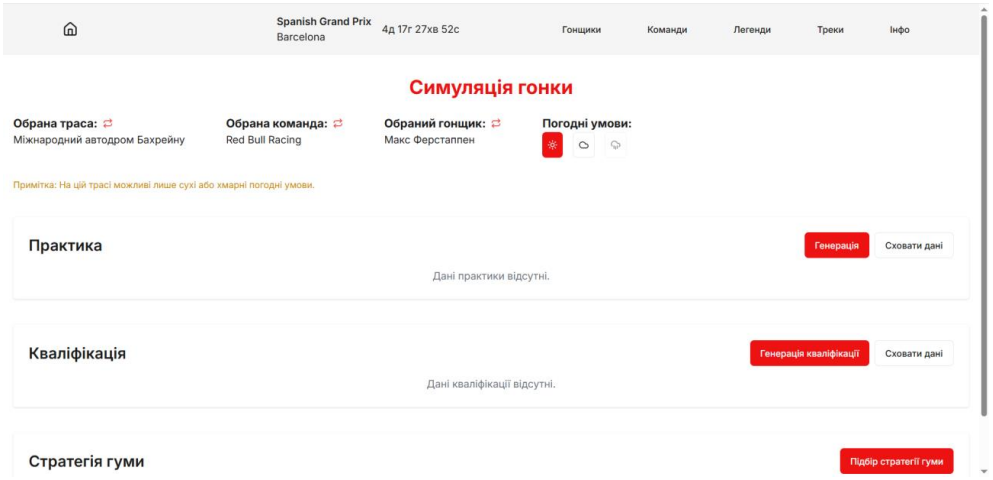
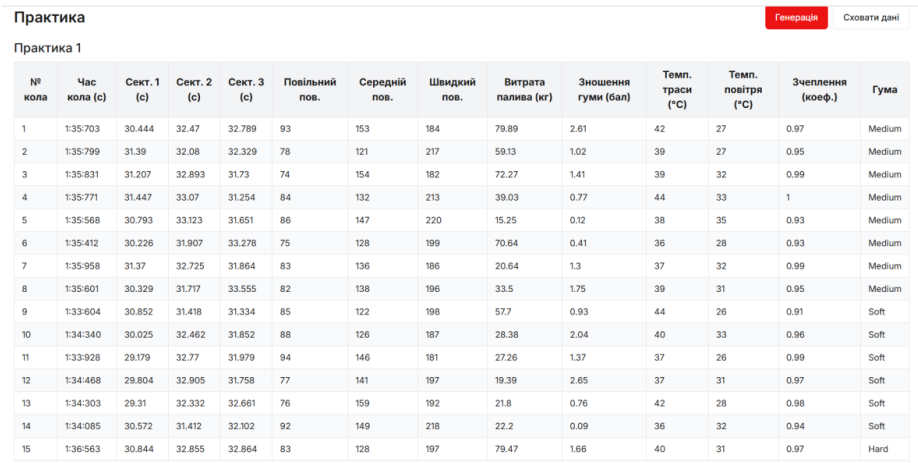


Рисунок В.12 – Варіант загального вигляд інтерфейсу сторінки, коли не можна використовувати дощову погоду



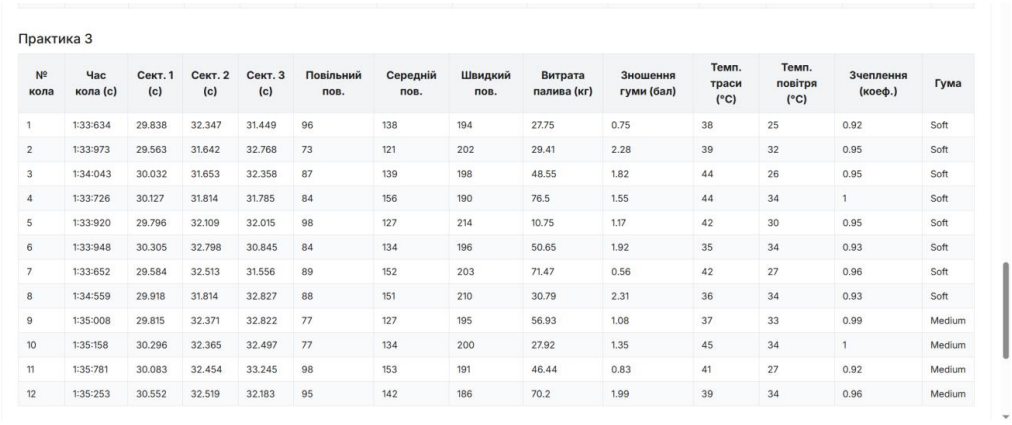
Практика

Генерація | Сховати дані

Практика 1

№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:35:703	30.444	32.47	32.789	93	153	184	79.89	2.61	42	27	0.97	Medium
2	1:35:799	31.39	32.08	32.329	78	121	217	59.13	1.02	39	27	0.95	Medium
3	1:35:831	31.207	32.893	31.73	74	154	182	72.27	1.41	39	32	0.99	Medium
4	1:35:771	31.447	33.07	31.254	84	132	213	39.03	0.77	44	33	1	Medium
5	1:35:568	30.793	33.123	31.651	86	147	220	15.25	0.12	38	35	0.93	Medium
6	1:35:412	30.226	31.907	33.278	75	128	199	70.64	0.41	36	28	0.93	Medium
7	1:35:958	31.37	32.725	31.864	83	136	186	20.64	1.3	37	32	0.99	Medium
8	1:35:601	30.329	31.717	33.555	82	138	196	33.5	1.75	39	31	0.95	Medium
9	1:33:604	30.852	31.418	31.334	85	122	198	57.7	0.93	44	26	0.91	Soft
10	1:34:340	30.025	32.462	31.852	88	126	187	28.38	2.04	40	33	0.96	Soft
11	1:33:928	29.179	32.77	31.979	94	146	181	27.26	1.37	37	26	0.99	Soft
12	1:34:468	29.804	32.905	31.758	77	141	197	19.39	2.65	37	31	0.97	Soft
13	1:34:303	29.31	32.332	32.661	76	159	192	21.8	0.76	42	28	0.98	Soft
14	1:34:085	30.572	31.412	32.102	92	149	218	22.2	0.09	36	32	0.94	Soft
15	1:36:563	30.844	32.855	32.864	83	128	197	79.47	1.66	40	31	0.97	Hard

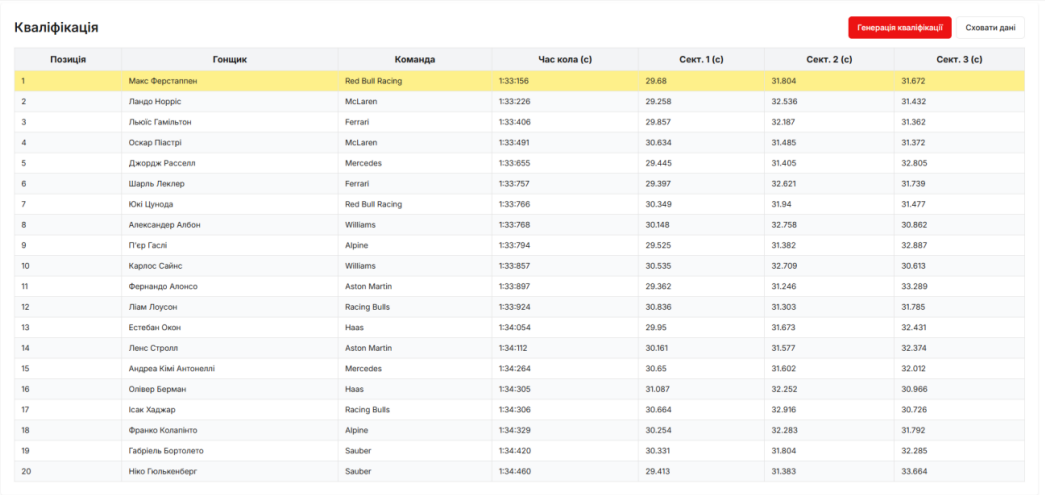
Рисунок В.13– Загальний вигляд інтерфейсу результату генерації практик 1 – 3



Практика 3

№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:33:634	29.838	32.347	31.449	96	138	194	27.75	0.75	38	25	0.92	Soft
2	1:33:973	29.563	31.642	32.768	73	121	202	29.41	2.28	39	32	0.95	Soft
3	1:34:043	30.032	31.653	32.358	87	139	198	48.55	1.82	44	26	0.95	Soft
4	1:33:726	30.127	31.814	31.785	84	156	190	76.5	1.55	44	34	1	Soft
5	1:33:920	29.796	32.109	32.015	98	127	214	10.75	1.17	42	30	0.95	Soft
6	1:33:948	30.305	32.798	30.845	84	134	196	50.65	1.92	35	34	0.93	Soft
7	1:33:652	29.584	32.513	31.556	89	152	203	71.47	0.56	42	27	0.96	Soft
8	1:34:559	29.918	31.814	32.827	88	151	210	30.79	2.31	36	34	0.93	Soft
9	1:35:008	29.815	32.371	32.822	77	127	195	56.93	1.08	37	33	0.99	Medium
10	1:35:158	30.296	32.365	32.497	77	134	200	27.92	1.35	45	34	1	Medium
11	1:35:781	30.083	32.454	33.245	98	153	191	46.44	0.83	41	27	0.92	Medium
12	1:35:253	30.552	32.519	32.183	95	142	186	70.2	1.99	39	34	0.96	Medium

Рисунок В.13, аркуш 2

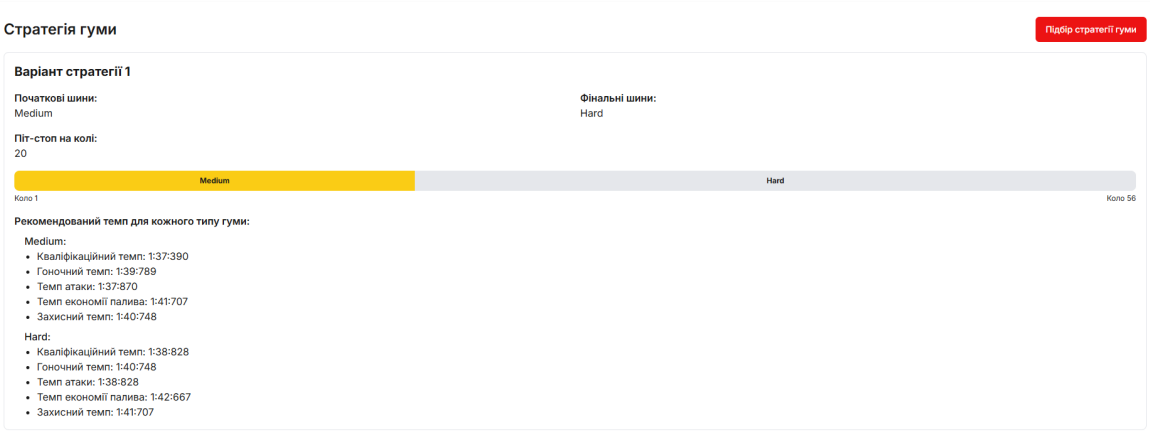


Кваліфікація

Генерація кваліфікації | Сховати дані

Позиція	Гонщик	Команда	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)
1	Макс Ферстаппен	Red Bull Racing	1:33:156	29.68	31.804	31.672
2	Ландо Норріс	McLaren	1:33:226	29.258	32.536	31.432
3	Льюїс Гамільтон	Ferrari	1:33:406	29.857	32.187	31.362
4	Оскар Піастрі	McLaren	1:33:491	30.634	31.485	31.372
5	Джордж Расселл	Mercedes	1:33:655	29.445	31.405	32.805
6	Шарль Леклер	Ferrari	1:33:757	29.397	32.621	31.739
7	Юкі Цунода	Red Bull Racing	1:33:766	30.349	31.94	31.477
8	Александер Албон	Williams	1:33:768	30.148	32.758	30.862
9	П'єр Гаслі	Alpine	1:33:794	29.525	31.382	32.887
10	Карлос Сайнс	Williams	1:33:857	30.535	32.709	30.613
11	Фернандо Алонсо	Aston Martin	1:33:897	29.362	31.246	33.289
12	Ліам Лоусон	Racing Bulls	1:33:924	30.836	31.303	31.785
13	Естебан Окон	Haas	1:34:054	29.95	31.673	32.431
14	Ленс Стролл	Aston Martin	1:34:112	30.161	31.577	32.374
15	Андреа Кімі Антонеллі	Mercedes	1:34:264	30.65	31.602	32.012
16	Олівер Берман	Haas	1:34:305	31.087	32.252	30.966
17	Ісак Хаджар	Racing Bulls	1:34:306	30.664	32.916	30.726
18	Франко Колапінто	Alpine	1:34:329	30.254	32.283	31.792
19	Габріель Бортолето	Sauber	1:34:420	30.331	31.804	32.285
20	Ніко Гюлькенберг	Sauber	1:34:460	29.413	31.383	33.664

Рисунок В.14 – Загальний вигляд інтерфейсу результату генерації кваліфікації



Стратегія гуми

Підбір стратегії гуми

Варіант стратегії 1

Початкові шини:
Medium

Фінальні шини:
Hard

Піт-стоп на колі:
20

Medium | Hard

Коло 1 | Коло 56

Рекомендований темп для кожного типу гуми:

Medium:
- Кваліфікаційний темп: 1:37:390
- Гоночний темп: 1:39:789
- Темп атаки: 1:37:870
- Темп економії палива: 1:41:707
- Захисний темп: 1:40:748

Hard:
- Кваліфікаційний темп: 1:38:828
- Гоночний темп: 1:40:748
- Темп атаки: 1:38:828
- Темп економії палива: 1:42:667
- Захисний темп: 1:41:707

Рисунок В.15 – Загальний вигляд інтерфейсу результату підбору стратегії на гонку

ДОДАТОК Г (довідниковий)

Інструкція користувача

При вході на онлайн-платформу користувач одразу бачить головну сторінку. На рисунку Г.1 представлено загальний вигляд інтерфейсу головної сторінки онлайн-платформи

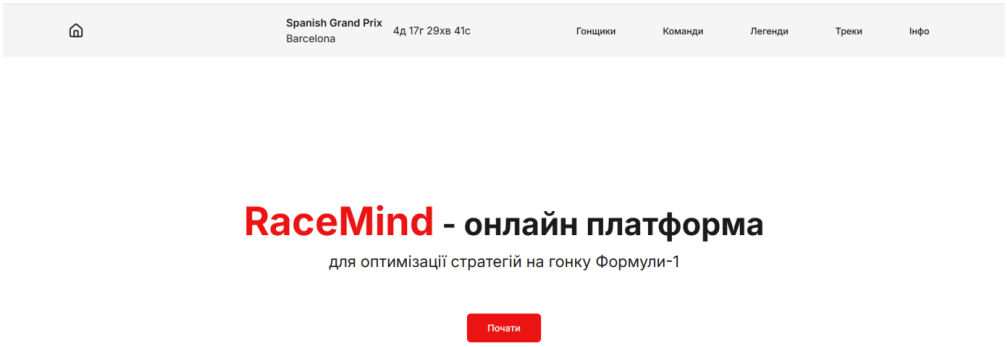


Рисунок Г.1 – Загальний вигляд інтерфейсу головної сторінки онлайн-платформи

На цій сторінці можна дізнатися про платформу або перейти відразу до підбору стратегії натиснувши на кнопку «Почати». Натиснувши, користувач перейде на сторінку вибору треку, на цій сторінці в одному рядку буде 4 треки, щоб бачити інші, користувачу потрібно рухати повзунок донизу (рис. Г.2).

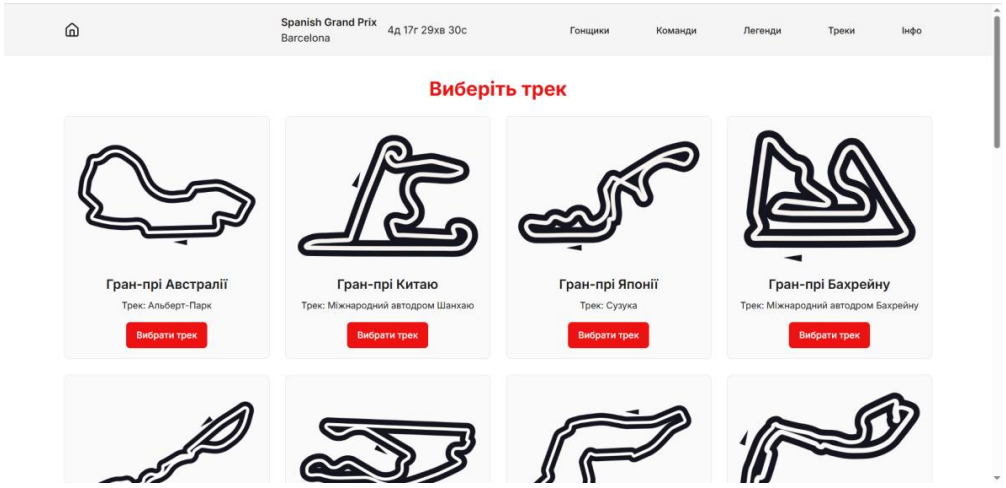


Рисунок Г.2 – Загальний вигляд інтерфейсу сторінки вибору треку

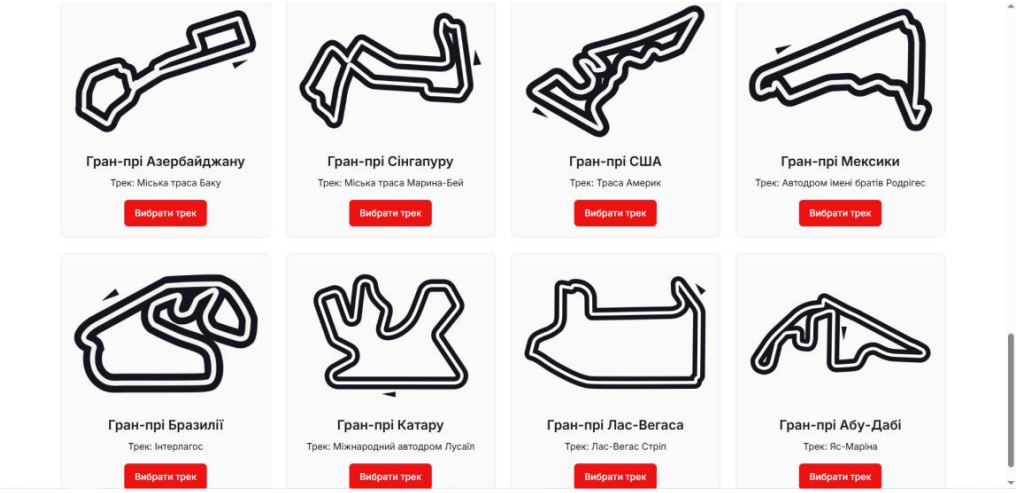


Рисунок Г.2, аркуш 2

Після вибору треку, користувача перекидає на сторінку вибору команди. Принцип дії аналогічний, як і на попередній сторінці. Щоб побачити більше команд, потрібно рухати повзунок донизу. На рисунку Г.3 представлено сторінку вибору команд.

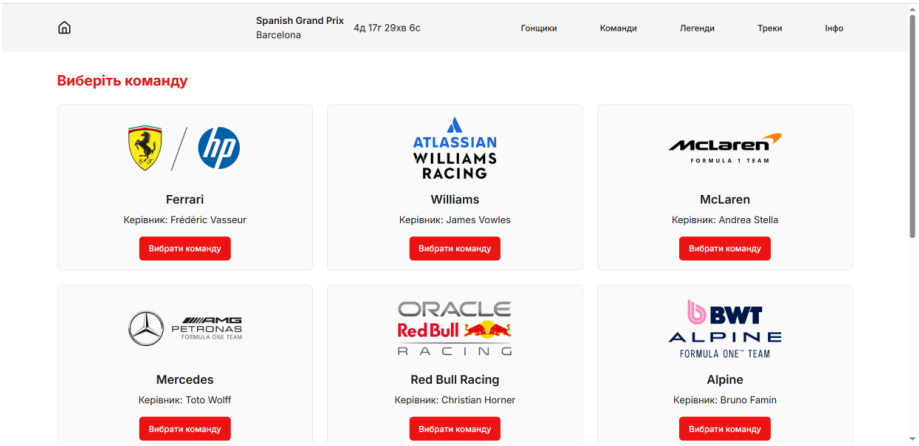


Рисунок Г.3 – Загальний вигляд інтерфейсу сторінки вибору команди

Потім користувачу буде потрібно вибрати гонщика серед двох, які виступають за обрану команду. На рисунку Г.4 представлено сторінку вибору гонщика.

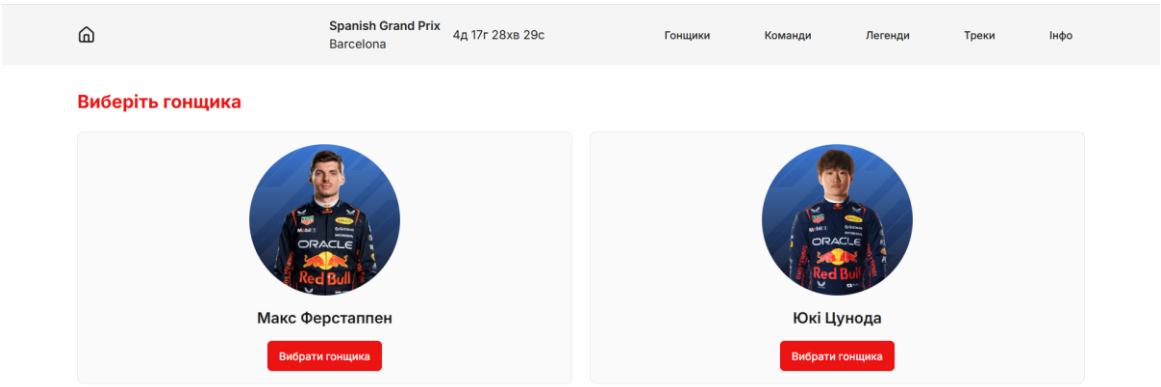


Рисунок Г.4 – Загальний вигляд інтерфейсу сторінки вибору гонщика

Після того як користувач вибере бажаного гонщика його переадресує на сторінку підбору стратегії. На ній користувач може переобрати трасу, команду та гонщика, відповідно як це було на етапі вибору. Ще однією важливою функцією є вибір погоди на конкретний тип сесії: чи практика, чи кваліфікація, чи для вибору стратегії на гонку. На рисунку Г.5 показана сторінка підбору стратегії на гонку з врахуванням різної погоди.

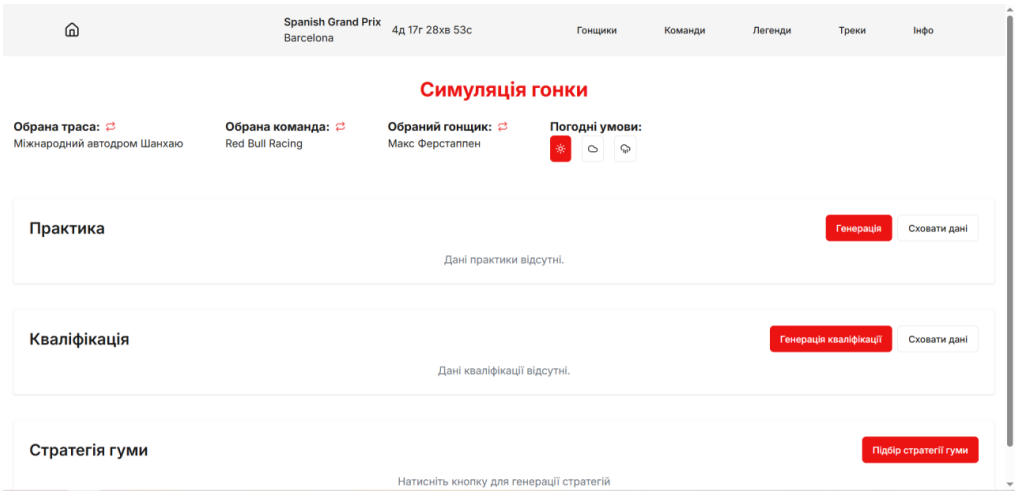


Рисунок Г.5 – Загальний вигляд інтерфейсу сторінки підбору стратегії з врахуванням різної погоди

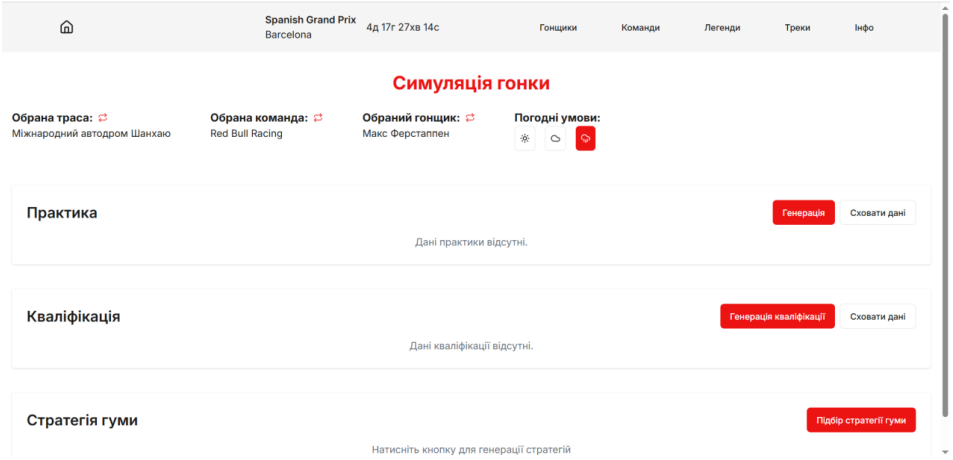


Рисунок Г.5, аркуш 2

У разі якщо вибрано трек на якому сильні дощі не відбуваються, то платформа повідомить, що тут можливі варіанти лише сухі або хмарні погодні умови. На рисунку Г.6 подано варіант сторінки, коли не можна використовувати дощову погоду.

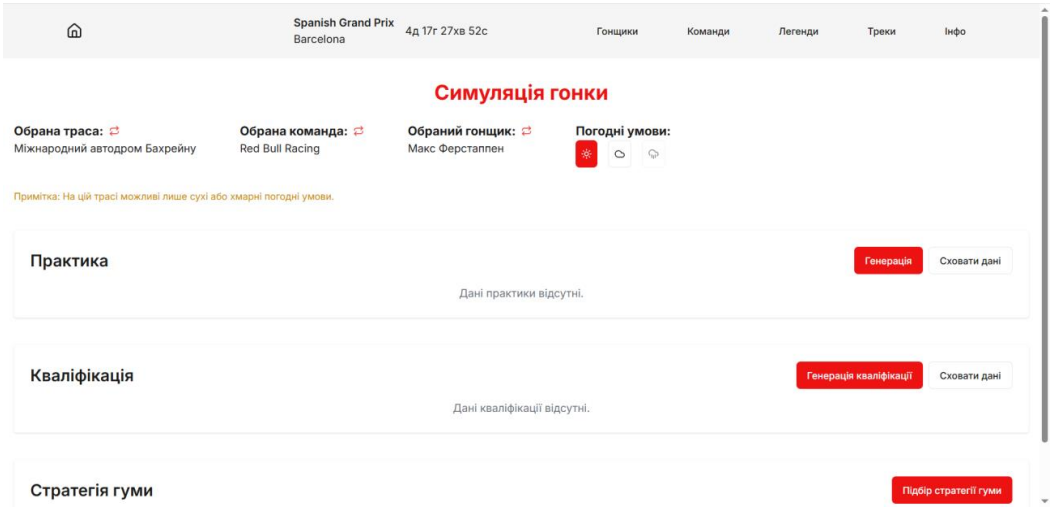
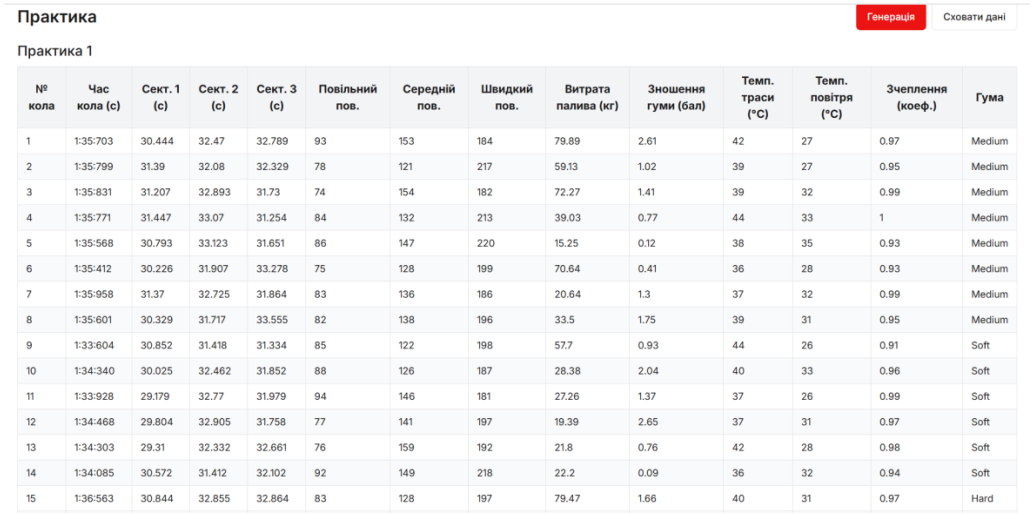


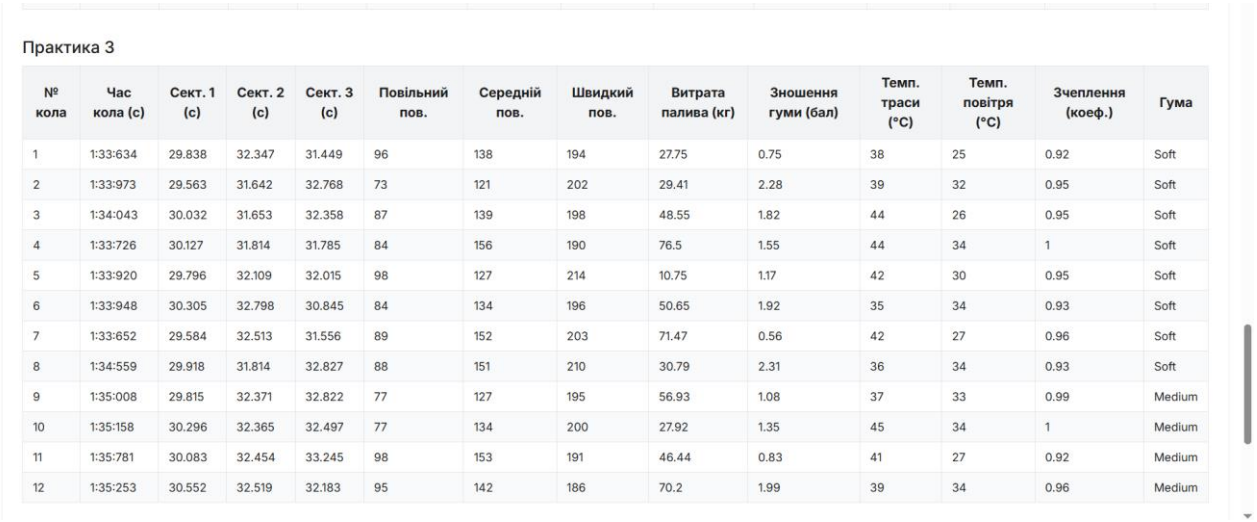
Рисунок Г.6 – Варіант загального вигляд інтерфейсу сторінки, коли не можна використовувати дощову погоду

Користувач може згенерувати дані про 1 – 3 практики, які потім будуть використовуватися для підбору стратегії. Після чого їх можна буде сховати натиснувши кнопку «Сховати дані». Важливим моментом є те, що результати практик можна перегенерувати, натиснувши на «Генерація». Результати генерації можна побачити на рисунку Г.7.



№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:35:703	30.444	32.47	32.789	93	153	184	79.89	2.61	42	27	0.97	Medium
2	1:35:799	31.39	32.08	32.329	78	121	217	59.13	1.02	39	27	0.95	Medium
3	1:35:831	31.207	32.893	31.73	74	154	182	72.27	1.41	39	32	0.99	Medium
4	1:35:771	31.447	33.07	31.254	84	132	213	39.03	0.77	44	33	1	Medium
5	1:35:568	30.793	33.123	31.651	86	147	220	15.25	0.12	38	35	0.93	Medium
6	1:35:412	30.226	31.907	33.278	75	128	199	70.64	0.41	36	28	0.93	Medium
7	1:35:958	31.37	32.725	31.864	83	136	186	20.64	1.3	37	32	0.99	Medium
8	1:35:601	30.329	31.717	33.555	82	138	196	33.5	1.75	39	31	0.95	Medium
9	1:33:604	30.852	31.418	31.334	85	122	198	57.7	0.93	44	26	0.91	Soft
10	1:34:340	30.025	32.462	31.852	88	126	187	28.38	2.04	40	33	0.96	Soft
11	1:33:928	29.179	32.77	31.979	94	146	181	27.26	1.37	37	26	0.99	Soft
12	1:34:468	29.804	32.905	31.758	77	141	197	19.39	2.65	37	31	0.97	Soft
13	1:34:303	29.31	32.332	32.661	76	159	192	21.8	0.76	42	28	0.98	Soft
14	1:34:085	30.572	31.412	32.102	92	149	218	22.2	0.09	36	32	0.94	Soft
15	1:36:563	30.844	32.855	32.864	83	128	197	79.47	1.66	40	31	0.97	Hard

Рисунок Г.7 – Загальний вигляд інтерфейсу результату генерації практик 1 – 3

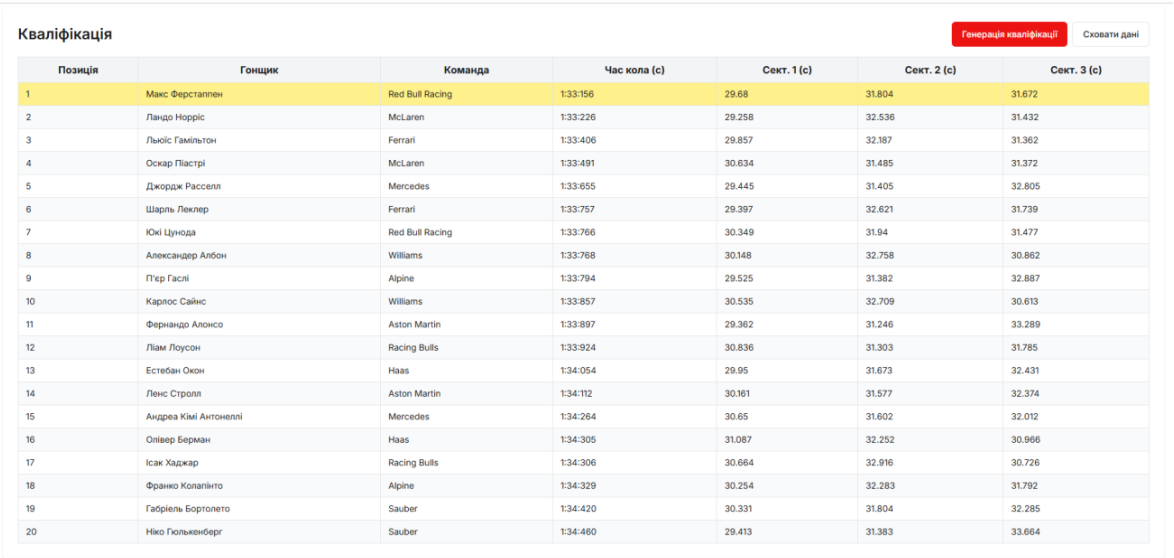


Практика 3

№ кола	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)	Повільний пов.	Середній пов.	Швидкий пов.	Витрата палива (кг)	Зношення гуми (бал)	Темп. траси (°C)	Темп. повітря (°C)	Зчеплення (коеф.)	Гума
1	1:33:634	29.838	32.347	31.449	96	138	194	27.75	0.75	38	25	0.92	Soft
2	1:33:973	29.563	31.642	32.768	73	121	202	29.41	2.28	39	32	0.95	Soft
3	1:34:043	30.032	31.653	32.358	87	139	198	48.55	1.82	44	26	0.95	Soft
4	1:33:726	30.127	31.814	31.785	84	156	190	76.5	1.55	44	34	1	Soft
5	1:33:920	29.796	32.109	32.015	98	127	214	10.75	1.17	42	30	0.95	Soft
6	1:33:948	30.305	32.798	30.845	84	134	196	50.65	1.92	35	34	0.93	Soft
7	1:33:652	29.584	32.513	31.556	89	152	203	71.47	0.56	42	27	0.96	Soft
8	1:34:559	29.918	31.814	32.827	88	151	210	30.79	2.31	36	34	0.93	Soft
9	1:35:008	29.815	32.371	32.822	77	127	195	56.93	1.08	37	33	0.99	Medium
10	1:35:158	30.296	32.365	32.497	77	134	200	27.92	1.35	45	34	1	Medium
11	1:35:781	30.083	32.454	33.245	98	153	191	46.44	0.83	41	27	0.92	Medium
12	1:35:253	30.552	32.519	32.183	95	142	186	70.2	1.99	39	34	0.96	Medium

Рисунок Г.7, аркуш 2

Згодом користувачу потрібно згенерувати дані кваліфікації. Їх як і практики можна сховати або перегенерувати. Гонщика, якого обрав користувач буде підсвічено жовтим, щоб легше зрозуміти позицію гонщика. На рисунку Г.8 представлені результати генерації кваліфікації.



Кваліфікація

Генерація кваліфікації | Сховати дані

Позиція	Гонщик	Команда	Час кола (с)	Сект. 1 (с)	Сект. 2 (с)	Сект. 3 (с)
1	Макс Ферстаппен	Red Bull Racing	1:33:156	29.68	31.804	31.672
2	Ландо Норріс	McLaren	1:33:226	29.258	32.536	31.432
3	Льюїс Гамільтон	Ferrari	1:33:406	29.857	32.187	31.362
4	Оскар Піастрі	McLaren	1:33:491	30.634	31.485	31.372
5	Джордж Расселл	Mercedes	1:33:655	29.445	31.405	32.805
6	Шарль Леклер	Ferrari	1:33:757	29.397	32.621	31.739
7	Юкі Цунода	Red Bull Racing	1:33:766	30.349	31.94	31.477
8	Александер Албон	Williams	1:33:768	30.148	32.758	30.862
9	П'єр Гаслі	Alpine	1:33:794	29.525	31.382	32.887
10	Карлос Сайнс	Williams	1:33:857	30.535	32.709	30.613
11	Фернандо Алонсо	Aston Martin	1:33:897	29.362	31.246	33.289
12	Ліам Лоусон	Racing Bulls	1:33:924	30.836	31.303	31.785
13	Естебан Окон	Haas	1:34:054	29.95	31.673	32.431
14	Ленс Стролл	Aston Martin	1:34:112	30.161	31.577	32.374
15	Андреа Кімі Антонеллі	Mercedes	1:34:264	30.65	31.602	32.012
16	Олівер Берман	Haas	1:34:305	31.087	32.252	30.966
17	Ісак Хаджар	Racing Bulls	1:34:306	30.664	32.916	30.726
18	Франко Колапінто	Alpine	1:34:329	30.254	32.283	31.792
19	Габріель Бортолето	Sauber	1:34:420	30.331	31.804	32.285
20	Ніко Гюлькенберг	Sauber	1:34:460	29.413	31.383	33.664

Рисунок Г.8 – Загальний вигляд інтерфейсу результату генерації кваліфікації

Потім користувачу потрібно натиснути кнопку «Підбір стратегії гуми», система виведе на сторінку найкращий варіант, відповідно згенерованих раніше даних. Також виведе рекомендований оптимальний час їзди на кожному комплекту гуми для різних ситуацій на треку. Результат підбору стратегії на гонку, можна побачити на рисунку Г.9.

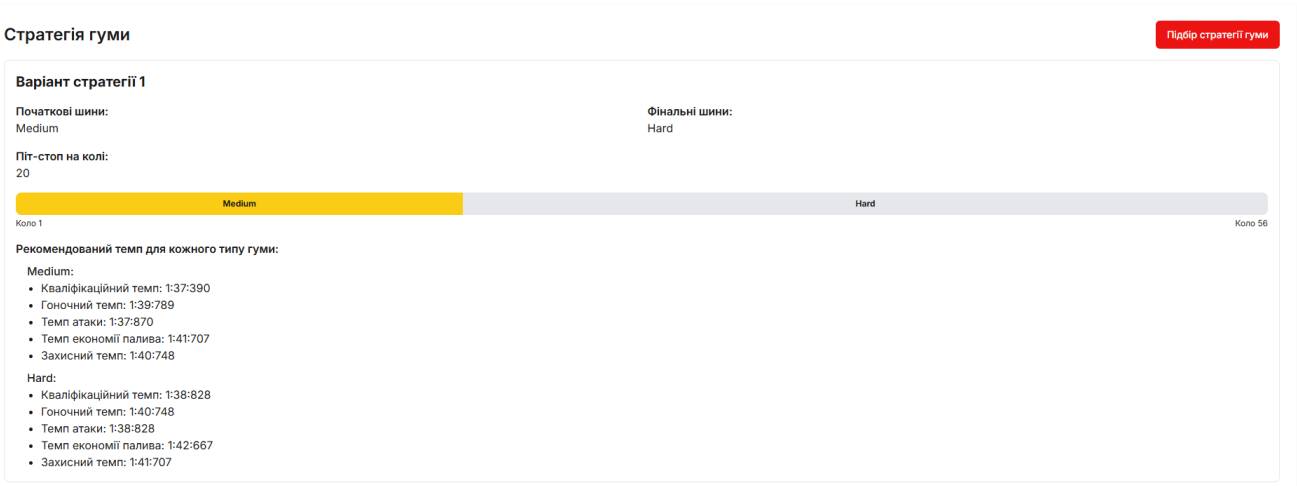


Рисунок Г.9 – Загальний вигляд інтерфейсу результату підбору стратегії на гонку