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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
"___"_____2024 р.

Магістерська дисертація

на здобуття ступеня другого (магістерського) рівня вищої освіти
за освітньою програмою "Цифрові технології в енергетиці"
за спеціальності 122 "Комп'ютерні науки"

на тему Побудова веломаршрутів на основі моделей штучного інтелекту

Виконав: студент _2_курсу, групи ТР-32мп
ТОПОЛЮК Кирило Михайлович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент каф. цифрових технологій в енергетиці,
к.т.н., доц., Юлія СИДОРЕНКО _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Рецензент доцент каф. теплової та альтернативної енергетики,
к.т.н., доц., Артур РАЧИНСЬКИЙ _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Н.контроль доцент каф. цифрових технологій в енергетиці,
д.ф., Артем ГУРІН _____
(посада, ім'я ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

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

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – другий (магістерський)
спеціальність 122 "Комп'ютерні науки"
Освітньо-професійна програма "Цифрові технології в енергетиці"

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА_

«___»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

ТОПОЛЮКУ Кирилу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Побудова веломаршрутів на основі моделей штучного інтелекту

керівник роботи Сидоренко Юлія Всеволодівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «11» листопада 2024 р. № 5040-с

2. Термін подання роботи студентом 06.12.2024 року

3. Вихідні дані до роботи _мова програмування JavaScript, інструмент веб-розробки React, середовище розробки Visual Studio Code, геопросторові дані.

4. Перелік питань, які потрібно розробити: аналіз існуючих програмних систем, аналіз методів, що виконують завдання рекомендації маршрутів, реалізація користувацького інтерфейсу та рекомендаційної системи, тестування та налагодження програмної системи.

5. Орієнтований перелік ілюстративного матеріалу постановка завдання, діаграма прецедентів, логічна структура системи, ілюстрації роботи системи

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	02.09 – 13.09.2024	виконано
2.	Вивчення та аналіз задачі	16.09 – 20.09.2024	виконано
3.	Проєктування архітектури розроблюваної системи	23.09 – 27.09.2024	виконано
4.	Розробка та реалізація модулю побудови маршрутів	30.09 – 4.11.2024	виконано
5.	Розробка та реалізація рекомендаційної системи	11.11 – 15.11.2024	виконано
6.	Тестування та налагодження системи	18.11 – 22.11.2024	виконано
7.	Оформлення пояснювальної записки	23.09 – 06.12.2024	виконано
8.	Захист програмного продукту	24.10.2024	виконано
9.	Передзахист	28.11.2024	виконано
10.	Захист	16.12.2024	

Студент _____ Кирило ТОПОЛЮК
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Юлія СИДОРЕНКО
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Актуальність теми. На сьогодні велосипед стає все більше популярним транспортним засобом завдяки ефективності та екологічності. У світі швидкої урбанізації велотранспорт є не лише способом розваги, а й важливим елементом повсякденного транспорту. Побудова веломаршрутів з врахування індивідуальних вподобань, не лише спрошує а й пришвидшує планування подорожей. Тому дослідження та побудова відповідної системи є актуальною та має практичне значення;

Метою роботи є створення системи, що зможе будувати персоналізовані веломаршрути та надавати рекомендації користувачам;

Завдання дослідження:

- провести аналіз сучасних систем побудови веломаршрутів, методи проектування та архітектуру;
- проаналізувати методи побудови користувацьких рекомендацій;
- оцінити фактори, що впливають на побудову маршрутів, такі як відстань, тип доріг та погодні умови;
- розробити інструмент побудови веломаршрутів з використанням проаналізованих факторів;
- розробити рекомендаційну модель для користувачів на основі збережених маршрутів;
- провести тестування розробленої програмної системи;
- створити стартап-проєкт для розробленої програмної системи;
- провести апробацію розробленої програмної системи;

Об’єкт дослідження: комп’ютерна система побудови веломаршрутів;

Предмет дослідження: комп’ютерна система побудови персоналізованих веломаршрутів з використанням методів штучного інтелекту;

Практична цінність здобутих в роботі результатів полягає в наданні можливості створювати веломаршрути на основі індивідуальних вподобаннях користувачів враховуючи такі фактори як відстань, тип дороги, погоду та

наявність цікавих місць. Принцип до побудови маршрутів динамічний, що завжди надаватиме користувачу унікальні маршрути;

Апробація результатів дисертації. Результати роботи були оприлюднені на VII Міжнародній студентській конференції «Діджиталізація науки як виклик сьогодення», м. Полтава, 25 жовтня 2024 року;

Дисертація складається з вступу, п’яти розділів та висновків. Повний обсяг складає 102 сторінок, в тому числі 90 сторінок основного тексту, 22 таблиці, 34 рисунки, 51 джерело в переліку;

Ключові слова: *штучний інтелект, рекомендаційна система, персоналізовані маршрути*;

ABSTRACT

Relevance of the research topic. Today, bicycles are becoming increasingly popular as a means of transportation due to their efficiency and environmental friendliness. In a world of rapid urbanization, cycling is not only a way of entertainment, but also an important element of everyday transportation. Building bicycle routes based on individual preferences not only simplifies but also speeds up travel planning. Therefore, researching and building an appropriate system is relevant and of practical importance;

Purpose of the work is to create a system that can build personalized cycling routes and provide recommendations to users;

Research Tasks:

- analyze modern bicycle route construction systems, design methods, and architecture;
- analyze methods of building user recommendations;
- assess the factors that influence route construction, such as distance, road type, and weather conditions;
- develop a tool for building cycling routes using the analyzed factors;
- develop a recommendation model for users based on the saved routes;
- to test the developed software system;
- create a startup project for the developed software system;
- perform a comparative evaluation of the models required to perform this work;

Object of research: a computer system for building cycling routes;

Subject of research: a computer system for building personalized cycling routes using artificial intelligence methods;

The practical value of the results obtained in this work is to provide an opportunity to create cycling routes based on individual user preferences, taking into account such factors as distance, road type, weather and the availability of interesting places. The principle of route construction is dynamic, which will always provide the user with unique routes;

Approval of Dissertation Results. The results of the work were presented at VII International Student Conference “Digitalization of Science as a Challenge of Today”, Poltava, October 25, 2024;

The dissertation consists of an introduction, five chapters and conclusions. The full volume is 102 pages, including 90 pages of the main text, 22 tables, 34 figures, 51 sources in the list;

Keywords: *artificial intelligence, recommendation system, personalized routes, coordinates*;

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

OSM – платформа для отримання картографічних даних

GIS – геоінформаційна система

GPS – глобальна навігаційна система

POI – точка інтересу

API – програмний інтерфейс для зв’язку з іншими системами

GPX – формат обміну географічними координатами для веломаршрутів

Firebase – база даних, що використовується в системі

JavaScript – мова програмування розробленої системи

React – бібліотека для побудови веб-систем

ВСТУП

На сьогодні велосипед стає все більше популярним транспортним засобом завдяки ефективності та екологічності. У світі швидкої урбанізації велотранспорт є не лише способом розваги, а й важливим елементом повсякденного транспорту. Його все частіше асоціюють з здоровим способом життя, свободою пересування та зручністю. В багатьох країнах світу велосипед є одним з найпоширеніших видів транспорту. Зокрема, у Нідерландах станом на 2023 рік понад 28% поїздок у 2023 році здійснено на велосипедах [1], що є стає можливим завдяки постійному розвитку велоінфраструктури в країні.

Для того, щоб подорожі були комфортними, а головне безпечними, необхідні сучасні системи побудови веломаршрутів, що враховують не лише відстань, а й тип доріг, погодні умови і наявності точок інтересу. У цьому контексті надзвичайно важливим стає завдання створення системи побудови маршрутів на основі моделей штучного інтелекту, здатної аналізувати різноманітні фактори. Однак, розробка таких маршрутів є непростим викликом через необхідність враховувати великої кількості параметрів для забезпечення оптимального маршруту.

Актуальність дипломної роботи полягає в зростаючій потребі розробки більш ефективних та зручних методів побудови веломаршрутів, що враховують індивідуальні потреби користувача. Це особливо важливо в умовах активного розвитку міської інфраструктури та зростання популярності екологічного транспорту. Використання штучного інтелекту дозволяє автоматизувати процес побудови маршрутів, зважаючи на такі фактори, як дистанція, рельєф, час, погода, що надає користувачам безпечні та кофмортні веломаршрути.

1 ПОБУДОВИ ВЕЛОМАРШРУТІВ НА ОСНОВІ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

В цьому розділі описана постановка задачі розробленої системи. Проведено аналіз сучасних підходів та рішень у галузі геоінформаційних систем, зокрема застосування штучного інтелекту для побудови веломаршрутів. Мета розділу детально оглянути сучасні тенденцій та виклики у галузі геоінформаційних систем.

1.1 Задачі системи побудови веломаршрутів

Головною задачею є розробити веб-сервіс, який надаватиме користувачам можливість будувати маршрути на основі власних індивідуальних уподобань, таких як дистанція, тип дороги, погода та інші фактори. Потрібно розробити зручний інструмент для планування подорожей, що включатиме інтерактивну карту та можливість вибору бажаних налаштувань для маршрутів. Також веб-сервіс надаватиме можливість зберігати маршрути в потрібному форматі для подальшого використання в інших сервісах. Рекомендаційна система пропонувати користувачу нові маршрути на основі збережених.

Для досягнення поставленого завдання описано наступні задачі:

- провести аналіз та огляд існуючих аналогічних програмних систем, навести їх переваги, недоліки та підходи до побудови;
- визначити критерії, які є найважливішими для користувача під час побудови маршрутів, зокрема такі як відстань, час, тип доріг, наявність points of interest;

- оглянути та проаналізувати алгоритми, що використовуються для побудови маршрутів;
- проаналізувати моделі штучного інтелекту, що застосовуються для побудови рекомендацій;
- забезпечити можливість зберігати та вивантажувати маршрути у форматі «.gpx» для використання в інших навігаційних додатках та пристроях GPS;
- побудувати систему «BikeNet», що матиме інтерактивний інтерфейс з можливістю побудови веломаршрутів за індивідуальними вподобаннями;

Таким чином, розроблена система допомагатиме велосипедистам, туристам та спортивним любителям у плануванні та виборі маршрутів. Завдяки інтеграції з різними сервісами система забезпечить універсальний підхід до планування поїздок. Розробка цієї системи має на меті популяризувати та заохотити велоспорт для більшої кількості людей навіть у величезних містах з погано розвиненою інфраструктурою.

1.2 Огляд і аналіз існуючих аналогів

Важливим етапом розробки даної системи є детальний аналіз систем-конкурентів. Саме такий аналіз виявить сильні та слабкі сторони існуючих рішень, а також допоможе визначити потреби користувачів, які ще не охоплені на ринку та реалізувати їх в своїй системі. Популярними аналогами є Strava, Komoot, Google Maps. Ці системи пропонують різноманітні функції для планування веломаршрутів, аналізу активності та навігації і деякі певною мірою навіть є соціальними мережами, де користувачі діляться своїми досягненнями, об’єднуються в спільноти, організовують групові поїздки та можуть навіть залишати коментарі та спілкуватись з іншими користувачами.

Strava є однією з найпопулярніших платформ для велосипедистів та бігунів, яка дозволяє користувачам відстежувати свої тренування, планувати

маршрути та аналізувати результати. Сервіс дозволяє створювати та зберігати власні маршрути, а також досліджувати маршрути інших користувачів [2].

Окрім того, веб-сервіс інтегрується з популярними пристроями для фітнесу, як от Garmin, Wahoo та інші, що додає ще більше зручності при використанні. Унікальною особливістю є можливість брати участь в сегментних змаганнях, тобто на визначених ділянках маршрутів, де користувачі змагаються за найкращий час. Таким чином, даний сервіс комбінує покращення індивідуальних показників та інтерактивну взаємодію з іншими спортсменами.

Strava має такі переваги:

- персоналізація маршрутів, враховуючи індивідуальні вподобання;
- актуальні та точні картографічні дані, що підвищує точність маршрутів;
- інтуїтивно зручний та зрозумілий інтерфейс;

Серед недоліків можна виділити:

- безкоштовна версія не дозволяє створювати маршрути;
- відсутність можливості генерувати випадкові маршрути;
- проблеми з конфіденційністю;

Отже, Strava – це потужна платформа для відстеження фізичної активності, яка спеціалізується на велосипедному та біговому спорті. Однак серед недоліків можна зазначити обмежений функціонал в безкоштовній версії та відсутність генерації випадкових маршрутів.

Komoot – це популярна платформа для планування маршрутів, яка орієнтована на таку категорію користувачів, як велосипедисти, туристи та просто любителі активного спорту [3]. Вона дозволяє користувачам створювати детальні маршрути, враховуючи специфіку рельєфу, фізичну підготовку користувачів та покриття доріг.

Komoot має наступні переваги:

— детальна інформація про рельєф;

— величезна спільнота користувачів;

— наявність офлайн карт;

— широкий вибір активностей;

Недоліками Komoot є:

— обмеження безкоштовної версії;

— відсутність можливості генерувати випадкові маршрути;

— проблеми з точністю;

Загалом Komoot відомий своєю варіацією типів активності, а саме велоспорт, пішохідні та туристичні прогулянки. Це дозволяє охопити більшу аудиторію та об’єднати їх в єдиному застосунку. Проте, недоліками є також обмеження безкоштовної версії, відсутність можливості генерувати випадкові маршрути та нижча в порівнянні з конкурентами точність.

Google Maps є однією з найпопулярніших картографічних платформ, що надає користувачам можливість знаходити маршрути для різних видів транспорту включаючи автомобіль, піший хід, велосипед, а також громадський транспорт і навіть літак [4]. Потужним інструментом є саме пошук громадського транспорту з наявністю актуального розкладу та бази маршрутів. Google Maps підтримує широкий спектр функцій, що допомагають користувачам орієнтуватися в міських умовах та знаходити оптимальні маршрути.

Перевагами Google Maps є:

- Сервіс підтримує багато видів транспорту;
- платформа регулярно оновлює свої дані;
- Google Maps надає велику кількість інформації про навколишні об'єкти, такі як ресторани, магазини, готелі та інші корисні місця;
- легко інтегрується з іншими сервісами Google, такими як Google Earth та Google Street View;

Але платформа має й свої недоліки:

- у випадку веломаршрутів, не завжди можна отримати оптимальні варіанти, особливо в країнах з менш розвиненою велоінфраструктурою;

— використання сервісу пов’язане з обробкою особистих даних, що викликає питання щодо конфіденційності;

Тому, Google Maps є універсальним інструментом для навігації, що охоплює всіх види транспорту. Завдяки своїй великій базі даних, детальним картам та актуальній інформації про дорожній рух, Google Maps дозволяє користувачам швидко знаходити оптимальні маршрути, враховуючи різні фактори, такі як затори, будівництва та інші перешкоди.

Висновки до розділу 1

В цьому розділі визначено задачі та завдання розробленої системи. Також наведено приклади аналогічних програмних систем. Проаналізовано їх переваги, недоліки. Загальною рисою для систем є те, що усі вони пропонують можливість планування маршрутів з інтерактивною взаємодією з мапою та враховуючи індивідуальні вподобання. Такі системи використовують штучний інтелект для оптимізації навігації, персоналізації маршрутів та для створення прогнозів.

Кожна система має свої унікальні особливості. Одні орієнтовані на максимально швидку побудову веломаршрутів, інші – на детальне врахування абсолютно всіх деталей для більш точної побудови маршруту, але з більшою витратою часу при плануванні.

Визначено критерій та потреба для користувачів, що не враховується в вище перелічених системах, а саме – відсутня можливість генерувати випадкові маршрути з персоналізованими налаштуваннями. Це змушує користувача витрачати більше часу на планування, особливо для досвідчених спортсменів, які прагнуть знаходити нові маршрути або менш популярні шляхи або ж туристам на нових локаціях, які не знають місцевості та не мають інформації про цікаві місця та можливі небезпечні ділянки.

2 АЛГОРИТМИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПОБУДОВИ ВЕЛОМАРШРУТІВ

Найважливішим компонентом розробленої системи є картографічні дані, які є основою при плануванні та навігації маршрутів. Дані відіграють ключову роль при побудові не лише інтерактивної системи, а й при створенні рекомендації.

У розділі описані вхідні дані до розробленої системи, опис сервісів, які слугують джерелами даних. Також розглянуто алгоритми та інструменти, що використовуються для створення персоналізованих рекомендацій для користувачів шляхом обробки та аналізу вхідних даних користувача.

2.1 Вхідні дані до системи

Під час розробки системи у ролі головного джерела картографічних даних використовувався сервіс OSM або ж OpenStreetMaps та Overpass у ролі інтерфейсу для вибірки даних.

OSM – це сервіс, що надає відкриті картографічні дані, що були створені і редагуються спільнотою користувачів з усього світу. Дані OSM включають інформацію про дороги, будівлі, природні об’єкти, а саме – ліси, поля, водойми будь-яких типів, точки інтересу (POI) та інші географічні елементи. Кожен елемент представлений у вигляді [5]:

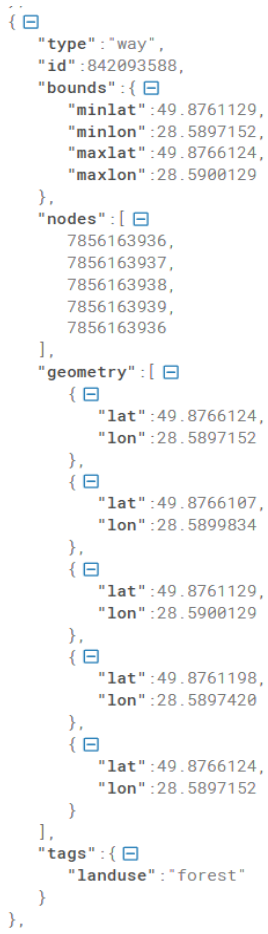
- точка (node), які є окремими координатами і позначають місця на карті;
- лінія (ways), що є послідовністю точок, які формують лінійні об’єкти, такі як дороги та річки;
- зв’язки (relation), що є групою точок та ліній, які описують складні структури, наприклад межі територій;

Кожен з елементів має атрибути, які зазвичай є опціональними, у вигляді пари «ключ-значення», що надають додаткову інформацію про елемент.

Для вибірки та фільтрації даних OSM застосовувався інтерфейс Overpass, що дозволяє отримувати підмножини даних OSM за певними критеріями, які задає користувач у вигляді спеціальної мови Overpass QL. Ця мова запитів дозволяє будувати складні запити, включаючи локацію та типи об’єктів для більш точної вибірки [6]. Під час розробки системи використовувались різні типи даних, зокрема ті, що позначаються тегом «natural», що вказує на природні об’єкти. Цей тег дозволяє обирати наступні елементи:

– ліси (woods), а саме природні та штучні насадження лісів;
– річки (water), включає озера, ставки, струмки і відповідно річки;
– болота (wetland), території з надмірною вологістю;
– гори (peak), а саме вершини та горби;
– піски (sand), включаючи пляжі та пустелі;

Для коректної побудови персоналізованих маршрутів враховувались два основні теги – «woods» та «water». Формуючи запит Overpass QL у вигляді аргументів передавались бажані теги, локація шуканого місця у форматі (широта, довгота) та радіус пошуку в метрах [7]. Як відповідь, система отримує дані у форматі JSON з масивом знайдених точок, що зображено на рисунку 2.1. Блок «bounds» містить опис географічної області, в межах якої виконується запит. Він визначає просторові межі та представлений у вигляді прямокутника з мінімальними та максимальними координатами області, але даний блок не використовуються надалі в програмі через те, що координати є лише приблизними. Саме для побудови персоналізованого маршруту використовувався блок geometry, що містить вже точні координати об’єкта, представлені полями «lat» та «lon». Згодом ці точки об’єднувались з точками маршруту у певній послідовності для формування логічного маршруту.



```
{
   "type":"way",
   "id":842093588,
   "bounds":{
      "minlat":49.8761129,
      "minlon":28.5897152,
      "maxlat":49.8766124,
      "maxlon":28.5900129
   },
   "nodes":[
      7856163936,
      7856163937,
      7856163938,
      7856163939,
      7856163936
   ],
   "geometry":[
      {
         "lat":49.8766124,
         "lon":28.5897152
      },
      {
         "lat":49.8766107,
         "lon":28.5899834
      },
      {
         "lat":49.8761129,
         "lon":28.5900129
      },
      {
         "lat":49.8761198,
         "lon":28.5897420
      },
      {
         "lat":49.8766124,
         "lon":28.5897152
      }
   ],
   "tags":{
      "landuse":"forest"
   }
},
```


Рисунок 2.1 – Overpass response

Кожен елемент «geometry» відповідає елементу «nodes» у форматі «ключ-пара». У цьому контексті «ключ» є унікальним ідентифікатором вузла (node) у системі OpenStreetMap (OSM) та точно визначає об’єкти на мапі.

Блок «tags» містить інформацію набір тегів, які асоціюються з об’єктом. Теги складаються з пар «ключ-значення» і описують категорію або характеристику об’єкта. Наприклад вони можуть містити назву, тип використання, функцію та інші особливості. Теги охоплюють широкий спектр об’єктів, як от географічні, соціальні інфраструктурні та навіть екологічні. В розроблені системі дана інформація використовувалась лише для ідентифікації об’єкта для користувача. Будь-які теги також можна задавати при вибірці в запиті Overpass QL.

2.2 Метрика схожості

Метрика схожості – це важливий інструмент в аналітичних системах, які обробляють дані, зокрема у побудові веломаршрутів. Основою метою є оцінка рівня подібності між об'єктами або наборами даних, такими як от маршрути. Метрика схожості дозволяє адаптувати систему до потреб користувачів додаючи персоналізовані рекомендації, які враховують їхню попередню поведінка [8], а у випадку розробленої системи – збережені маршрути.

У контексті веломаршрутів метрика схожості порівнює такі фактори маршрутів як географічна схожість, а саме координати точки старту, дистанцію, тип дороги та погодні умови маршрутів. Порівнюючи збережені маршрути користувача з наявними маршрутами в сховищі даних – система рекомендує користувачу новий унікальний маршрут.

Косинусна схожість є широко застосовується в інформаційному пошуку та пов'язаних з ним дослідженнях як важлива метрика. Вона моделює набір даних як вектор ознак. За цією моделлю схожість між двома наборами даних може бути отримана шляхом обчислення значення косинуса між векторами ознак двох об’єктів [9]. Дана метрика в розробленій системі є особливо корисною через свою стійкість до масштабування. Не зважаючи на великий обсяг порівняльних даних – дана метрика ефективно, а головне швидко працює з даними та знаходить схожі маршрути для рекомендації користувачам.

Набір даних представлений у вигляді вектора ознак, розмірність якого відповідає кількості ознак, що містяться у цьому наборі [10]. Кожне вимірювання характеризується частотою появи відповідної ознаки в наборі. Загальна формула косинусної схожості між двома векторами A та B, заданими в n-вимірному просторі має вигляд (2.1):

$$\text{cosine\_similarity}(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 * \sum_{i=1}^{n} B_i^2}} \quad (2.1)$$

де A, B – вектори, що порівнюються,

$\sum_{i=1}^{n} A_i B_i$ – скалярний добуток двох векторів,

$\sqrt{\sum_{i=1}^{n} A_i^2}$ – довжина вектора A,

$\sqrt{\sum_{i=1}^{n} B_i^2}$ – довжина вектора B.

Діапазон значень косинусної схожості знаходиться в межах від -1 до 1, де:

- 1, якщо вектори мають один напрямок, тобто максимальна схожість;
- 0, якщо вектори ортогональні, тобто схожість відсутня;
- -1, якщо вектори мають протилежний напрямок, тобто мінімальна схожість;

Таким чином, в розробленій системі косинусна схожість може застосовуватись для порівняння маршрутів та визначення їх подібності за спільними факторами. Даний метод забезпечує як точність так і швидкодію системи, адже математичні операції доволі швидко обробляються в рамках системи.

2.3 Алгоритми машинного навчання

Машинне навчання є потужним інструментом для аналізу даних та у випадку розробленої системи для побудови рекомендацій велосипедних маршрутів для користувачів. В будь-який системі вибір алгоритму машинного навчання повинен бути обґрунтований, враховуючи переваги, недоліки та специфіку поставленої задачі, адже кожен алгоритм має своє призначення і звісно ж обмеження. Набір моделей, що можуть вирішити поставлену задачу рекомендації маршруту включає логістичну регресію, метод K–найближчих сусідів, дерева рішень та метод опорних векторів.

Логістична регресія добре справляється з задачами бінарної класифікації , де важливо оцінити ймовірність належності до певного класу. Головною перевагою є інтерпретованість, тобто модель дозволяє зрозуміти як кожна змінна системи впливає на результат. В контексті розробленого проєкту, алгоритм логістичної регресії може визначати чи належить маршрут до класу рекомендацій, аналізуючи його характеристики та критерії. Таким чином, нерелевантні варіанти будуть відсіюватись, а коректні рекомендуватись користувачу.

Алгоритм k-найближчих сусідів є ефективним у задачах, де важлива локальна структура даних для порівняння схожості між об’єктами, однак цей алгоритм залежить від обраної метрики відстані k, а також є обчислювально-дорогим для великих вибірок.

На противагу, алгоритми дерев рішень та опорних векторів пропонують інший, але більш гнучкий та точний підхід до визначення рекомендації для користувача. Дерева рішень добре підходять для задач, де потрібно інтуїтивно зрозуміле та легко інтерпретоване рішення, як от класифікація маршрутів за певними критеріями. По схожості з деревами рішень, метод опорних векторів також можна використовувати для визначення відповідності маршруту певним критеріям. Але перевагою цього алгоритму є можливість працювати з даними високої розмірності.

Логістична регресія – це статистичний метод машинного навчання, що застосовується для моделювання ймовірності належності об’єкта до одного з двох класів, тобто бінарна класифікація [11]. По суті, логістична регресія шукає математичну залежність між вхідними змінними та факторами і на основі їх аналізу модель навчається і визначає, наприклад, чи є відноситься певний маршрут до категорії «рекомендація». Для вирішення задач алгоритм застосовує методи максимальної ймовірності або градієнтного спуску. Метод максимальної ймовірності оцінює параметри моделі та знаходить їх найімовірніше значення на основі вже наявних даних. У випадку градієнтного спуску, відбувається обчислення

градієнта, що вказує на напрямок найшвидшого зменшення помилки. Тому це сприяє поступовому покращенню параметрів, так як метод використовується багаторазово, оскільки працює ітеративно. Обидва методи є важливими компонентами алгоритму логістичної регресії та інших методів машинного навчання. Логістична функція представлена на рисунку 2.2.

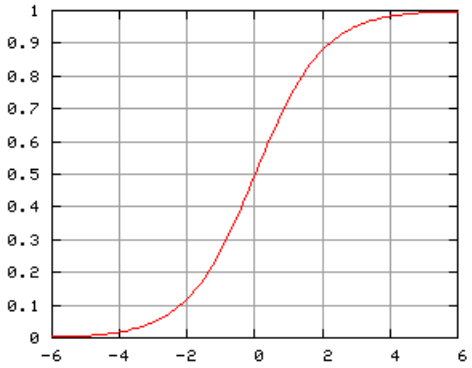


Рисунок 2.2 – Функція Сигмоїди

Функція має S-подібну форму, а діапазон значень знаходиться в межах [0,1], де при високих значеннях, тобто максимальна схожість – функція наближається до 1, а при мінімальних до 0. За допомогою формули, що має вигляд (2.2), функція перетворює значення лінійної комбінації вхідних змінних z на ймовірність p, яка вже як згадувалось, знаходиться в діапазоні від 0 до 1 [12]. Тобто, значення для z можуть варіюватись від $-\infty$ до $+\infty$ і все одно перетворюватись в ймовірність p в певному діапазоні.

$$p = \frac{1}{(1 + e^{-z})} \qquad (2.2)$$

де e – основа натурального логарифма, число Ейлера,

z – лінійна комбінація незалежних змінних та їх коефіцієнтів

Логістична регресія є універсальним інструментом для моделювання бінарних залежностей. Однією з основних переваг є можливість чіткої інтерпретації результатів, оскільки коефіцієнти модель показують вплив кожної змінної на ймовірність настання подій [13]. Окрім того, логістична регресія дозволяє використовувати як числові так і категоріальні змінні, що дозволяє аналізувати різні набори даних. Проте, модель також має свої обмеження, зокрема модель логістичної регресії передбачає саме лінійну залежність між логарифмами ймовірностей, що може бути некоректним у випадку складних та нелінійних зв'язків.

Метод k-найближчих сусідів (k-NN) є методом машинного навчання, що застосовується для задач класифікації та регресії. Основна ідея методу в тому, що для прогнозування класу враховуються класи або значення його *k* найближчих сусідів в навчальній вибірці [14], з цього і назва методу. Вибір базується на основі обчислення відстаней за допомогою формули (2.3) між об'єктами у багатовимірному масиві даних, що дає змогу визначити їх схожість.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2.3)$$

Після етапу підготовки даних та обчислення відстаней між кожним об'єктом навчальної вибірки відбувається сам процес визначення найближчих сусідів, а саме обираються k-найближчі сусіди, тобто об'єкти, що мають найменші значення метрики відстані до нового об'єкта [15]. Під час класифікації метод підраховує, які класи є найбільш поширеними серед k сусідів і відповідно присвоює новому об'єкту цей клас за принципом голосування більшості. Щодо регресії, метод обчислює середнє або вагоме середнє значення серед сусідів з використанням вагів (2.4), де ваги можуть бути обернено пропорційні до відстані.

$$w_i = \frac{1}{d(x, y_i)} \tag{2.4}$$

де w_i – вага сусіда y_i.

В формулі обчислення ваг відбувається обчислення відстані, де $\mathrm{d(x, y)}$ – відстань між новим об'єктом x для якого здійснюється прогноз та сусідом y_i з навчальної вибірки. Згодом новий об'єкт отримує прогнозоване значення, яке є або числовим або категорійним на основі інформації, наданої найближчими сусідами. Графічно метод k-найближчих сусідів показано на рисунку (2.3)

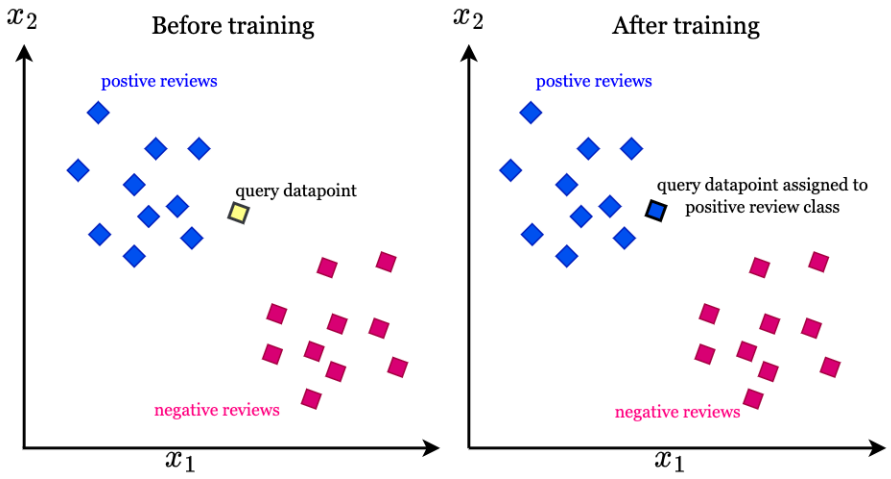


Рисунок 2.3 – Процес навчання методу

Головною перевагою методу є те, що він не потребує навчання моделі, так як обчислення відбувають під час прогнозування. Це в свою чергу спрощує процес інтеграції в систему, проте впливає на оптимізацію та швидкодію. Тому тут і формується недолік, так як метод потребує високої обчислювальної потужності залежно від набору даних.

Випадковий ліс – це метод машинного навчання, що використовує поняття дерева рішень. Дерева рішень є алгоритмами, що моделюють процес прийняття

рішень у вигляді дерева, де структура представлена у вигляді вузлів та гілок. Вузол відповідає за перевірку певної ознаки, а гілки – за можливі значення цієї ознаки.

Випадковий ліс є ансамблевих типом методів, тобто даний підхід об’єднує прогнози саме кількох базових моделей, щоб отримати найбільш точний з них. Ідея полягає в комбінуванні результатів в результаті чого з слабких моделей або з моделей з середньої точністю, можна створити сильну модель, яке перевершує попередні [16]. Таким чином, метод поєднує кілька дерев рішень для формування «лісу» моделей, що зображено на рисунку 2.4.

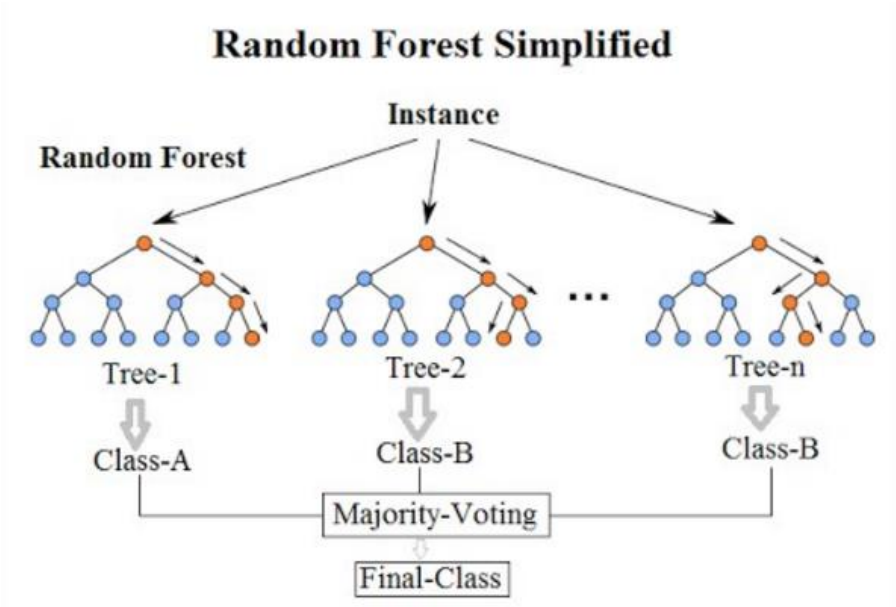


Рисунок 2.4 – Демонстрація роботи випадкового лісу

Суть випадкового лісу полягає в поєднанні багатьох дерев рішень, де кожне дерево навчається на випадковій підмножині даних, щоб зменшити ймовірність перенавчання та максимізувати точність моделі. Підхід дозволяє створити певний ансамбль моделей, які, по перше, працюють незалежно одна від одної, а, по друге, їх результати комбінуються для отримання більш узагальненого прогнозу. На

початковому етапі для кожного дерева в лісі створюється випадкова вибірка з навчальних даних з використанням методу бутстрепу.

Метод бутстрепу є статистичним методом, що створює багаторазові випадкові вибірки, де елементи можуть повторюватись більше одного разу з початкового набору даних [17]. Припустимо, що існує початкова вибірка з n елементів. Даний метод створює нові вибірки шляхом випадкового відбору об’єктів. Наприклад вибірка [x1,x2,x3,x4] після бутстрепу матиме вигляд [x1,x2,x2,x4] і подібні. Такий підхід зменшує кореляцію між деревами завдяки випадковим наборам і тому прогнози стають менш залежними між собою, що в загальному покращує точність ансамблю.

На наступному етапі після бутстрепу, алгоритм виконує випадковий відбір ознак. Основний завдання – це відібрати з кожного вузла дерева випадкова підмножина ознак з усіх доступних ознак. Згодом розгалуження вузла ґрунтується лише на одній з ознак цієї підмножини, які максимізує якість розбиття. Це важливо тому, що якщо всі дерева аналізуватимуть однаковий набір ознак, вони можуть продукувати схожі моделі і таким чином збільшується кореляція між ними, що призводить до низької точності алгоритму. Також випадковий відбір ознак прибирає домінування сильних ознак і змушує модель враховувати менш очевидні ознаки, які можуть суттєво впливати на прогноз.

Агрегація є фінальним етапом випадкового лісу. Під час нього об’єднуються результати усіх дерев ансамблю для отримання фінального прогнозу [18]. В цьому процесі враховуються усі індивідуальні прогнози вузлів і формується єдине рішення, яке є найбільш точним, ніж прогнози інших моделей.

В контексті рекомендації веломаршрутів алгоритм може відбирати маршрути за різними категоріями, такими як тип дорожнього покриття, погодні умови або ж наявність точки інтересу (POI). Наприклад, для міських маршрутів алгоритм може надавати перевагу асфальтованим покриття, тоді як для заміських – маршрутам з мінімальним впливом трафіку, як от лісові місцевості або поля.

Однією з важливих переваг алгоритму є те, що він вміє працювати з різнорідними даними, такими як числові показники, як от довжина маршруту, так і з категорійними змінними такими як покриття доріг або погодні умови. Random Forest може оцінювати, які з вище наведених ознак є найважливішими при формуванні рекомендацій. І вже узагальнене рішення про рекомендований для користувача маршрут будуватиметься шляхом голосування прогнозів дерев.

Опорні вектори (SVM) – це метод машинного навчання, що базується на концепції (рисунок 2.5) пошуку оптимальної гіперплощини для подальшого розділення даних на класи в багатовимірному просторі ознак. У сфері класифікації, алгоритм відзначається вмінням ефективно визначати оптимальні межі розділення між класами в наборі даних. Тому він особливо ефективним під час робити з багато розмірними наборами даних, де класи маю чіткі відмінності [19].

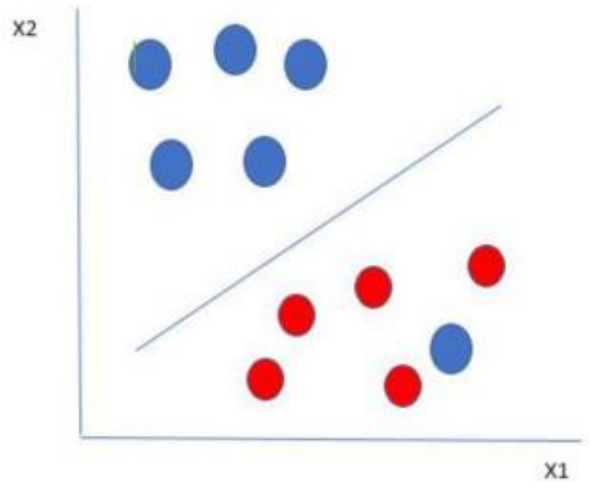


Рисунок 2.5 – Концепція роботи SVM

Алгоритм спрямований на розв’язання задачі оптимізації, метою якої є не лише правильне розділення даних, але й забезпечення узагальнення моделі на нових, ще не досліджених даних. Завдяки цьому SVM демонструє високу

продуктивність у задачах, де дані мають чіткі межі розділення. Метою методу є пошук межі, що розділяє точки (дані) на класи [20].

Якщо вхідні дані не є лінійно-роздільними, а саме між класами в наборі неможливо провести лінію або ж гіперплощину, то метод застосовує техніку ядрових функцій (kernel trick), що трансформує дані у простір більшої розмірності, де вже розділення даних є можливим. Тому SVM може працювати з різними типами даних та зв'язків. Метод підтримує кілька типів ядрових функцій [21]:

- лінійне ядро, що використовується для лінійно-роздільних даних, тобто тоді коли ми можемо відокремити класи однією прямою лінією або гіперплощиною. Дана ядрова функція є найпростішою, так як не потребує трансформації даних;
- поліноміальне ядро, що враховує нелінійні залежності між ознаками. Ядро моделює складні межі між класами. Поліноміальне ядро є обчислювально витратним для великих наборів даних, тому його часто застосовується, коли дані мають низьку розмірність, але в той же час містять складні зв'язки;
- радіально-базисне ядро є найбільш універсальним і популярним методом для даних з складною та нелінійною природою. Функція ефективно працює з даними, що складно розділити в початковому просторі, тому вона переносить їх у простір вищої розмірності, де вони вже стають лінійно-роздільними;
- сигмоїдне ядро, є функцією, поведінка якої нагадує функцію сигмоїди в нейронних мережах. Дане ядро використовується у випадках, коли дані мають складні, але вже зрозумілі залежності, наприклад в задачах прогнозування або класифікації текстів. Однак функція схильна до перенавчання, тому критично важливим є правильне її налаштування;

У системах рекомендації веломаршрутів SVM може бути використаний для класифікації маршрутів за різними критеріями, такими як рівень складності, тип

покриття, наявність підйомів тощо. Наприклад, на основі історичних даних про використання маршрутів та їх характеристик, SVM може навчитися розрізняти маршрути, які підходять для початківців або досвідчених велосипедистів. Алгоритм добре працюватиме з комбінованими типами даних, як от погодні умови або ж тип дороги. Саме використання ядрових функцій алгоритм моделюватиме складні нелінійні зв'язки між ознаками маршрутів, що є головною перевагою з поміж інших алгоритмів.

2.4 Алгоритми глибокого навчання

Глибоке навчання, як підгалузь машинного навчання, є потужними інструментами для аналізу великих і складних наборів даних. Вони застосовуються для моделювання нелінійних залежностей та автоматичного визначення ознак та створення високорівневих абстракцій, що робить можливим обробляти дані у різних формах від аудіо до відео та числових рядів. Алгоритми цього типу базуються на використанні багатошарових нейронних мереж, де кожен шар виконує послідовні нелінійні трансформації вхідних даних. Як результат, модель поступово виділяє ключові особливості та певні абстракції.

Серед різноманітних архітектур нейронних мереж особливо вирізняються глибокі нейронні мережі (DNN) та рекурентні нейронні мережі (RRN), кожна архітектура має свої унікальні характеристики і саме вони були розглянуті в цьому розділі.

Глибокі нейронні мережі (DNN) – це технологія, що змінила представлення та підхід до обробки даних і прийняття рішень в штучному інтелекті. DNN є структурою, схожою на людський мозок, де кожен шар нейронів відповідає за свій рівень обробки інформації. На початкових рівнях мережа розпізнає базові елементи, як от контури чи кольори, а на більш глибоких вже починає розуміти складніші деталі, як от обличчя чи тип об'єкта на певному зображені [22].

Глибокі нейронні мережі працюють як багатошарова система фільтрів. Кожен шар нейронів бере інформацію від попереднього (рисунок 2.6), перетворює та передає її далі. Таким чином, мережа поступово витягує все більш складні та складні особливості з даних і передає далі.

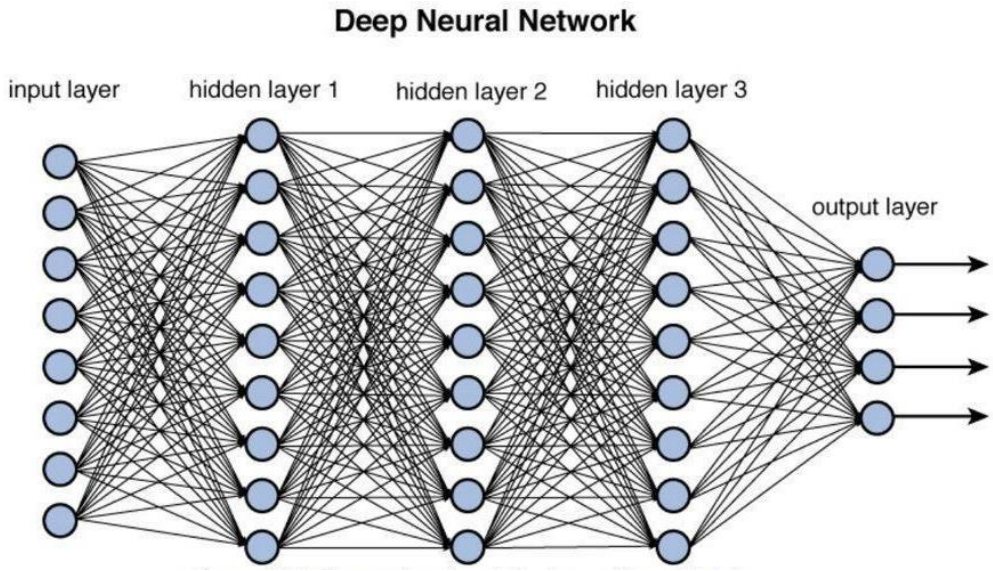


Рисунок 2.6 – Загальна архітектура DNN

Основні компоненти глибоких нейронних мереж включають:

– базисним компонентом DNN є вхідні дані, які мережа отримує у формі числових векторів або ж матриць. Залежно від типу вхідних даних до вхідному наборі застосовуються різні способи перетворення. Для зображень виконується перетворення в матриці пікселів, де кожне значення відповідає інтенсивності кольору. Текст в свою чергу перетворюється у векторі представлення за допомогою методів Word2Vec або embeddings. І також звукові дані кодуються у формі спектрограм або інших сигналів. Фінальні числові представлення стають вхідними сигналами для нейронів у першому шарі;

- наступний важливий компонент це шари нейронів. Мережа складається з вхідного, прихованого та вихідного шару. Вхідний шар приймає дані у сирій або попередньо обробленій формі [23]. Прихований шар є ядром моделі, де відбувається більшість обчислень і відповідних трансформацій даних. І вихідний шар, що вже генерує прогноз або рішення, як от класифікація або регресія. Кожен нейрон у шарі з'єднаний з нейронами відповідного попереднього шару, передаючи дані через ваги;
- ваги є частиною з'єднань між нейронами, адже кожен зв'язок має свою власну вагу, що представлено числовим значенням. Вага визначає важливість сигналу. Під час навчання ваги змінюються з метою мінімізації помилки в прогнозах;
- після отримання вхідного сигналу кожен нейрон обчислює свою активацію за допомогою нелінійних функцій. Серед таких функцій існують ReLU (Rectified Linear Unit), сигмоїдна функція та гіперболічний тангенс [23]. Перша функція пропускає лише позитивні значення, відсікаючи негативні. Сигмоїдна функція перетворює значення в діапазон від 0 до 1 та використовується для ймовірностей. І остання функція, подібна до сигмоїди, але значення вже змінюються від -1 до 1;
- Forward propagation або ж передача вперед є процесом передачі даних від вхідного шару до вихідного. Дані проходять через шари усіх нейронів, де на кожному етапі відбувається обробка вагами та функціями активації, тобто дані проходять лише вперед і не повертаються назад;
- Back propagation або ж зворотне поширення є методом оптимізації, що дозволяє оновлювати свої ж ваги задля мінімізації похибки між прогнозами та реальними значеннями. Саме це є ключовим етапом нейронної мережі;
- функція втрат оцінює те, наскільки добре модель працює, порівнюючи її прогноз з реальними значеннями. Метою є мінімізація значення функції

втрат, що відповідає зменшенню похибки між прогнозами та істинною. Основними видами функції втрат є перехресна ентропія, що використовується для задач класифікації, середньоквадратична помилка, та абсолютна похибки, що застосовуються для задач регресії;

– градієнтний спуск є оптимізаційним алгоритмом, що використовується для мінімізації втрат шляхом зміни ваг моделі. Суть полягає в русі у напрямку градієнта функції втрат, що вказує на те, як зменшити помилку. Важливим пунктом при використанні градієнтного спуску є правильний вибір швидкості навчання. Якщо швидкість завелика, модель може «перестрибувати» через функцію втрат, що призведе до нестабільного навчання та розбіжностей [24]. З іншого боку, якщо функція занадто мала, процес навчання буде надзвичайно повільним і модель потребуватиме багато епох, щоб досягти хоча б якогось прийнятного мінімум;
– епохою називають один повний прохід через весь набір навчальних даних. Під час кожної епохи дані передаються через мережу, виконуючи як forward так і backward propagation. А по завершенню епохи модель оновлює свої ваги на основі отриманих даних. Аналогічно потрібно бути уважним з кількістю епох, адже завелика кількість призведе до перенавчання;
– ітерація існує всередині епохи. Це є одним кроком навчання під час якого обробляється міні-батч даних. Кількість ітерацій в епосі залежить від розміру даних і відповідно міні-батча та обраховується як відношення розміру даних до розміру батча;

Тому глибокі нейронні мережі (DNN) є важливими в сучасних система штучного інтелекту, адже дозволяють вирішувати складні завдання, зокрема в сфері побудови велосипедних маршрутів та пошуку маршрутів для рекомендацій. За допомогою багатошарової структури, мережі здатні автоматично визначати

ключові ознаки з даних про маршрути і визначати маршрути, які підійдуть для кожного.

Рекурентні нейронні мережі (RNN) – це клас штучних нейронних мереж, що спеціалізуються на обробці саме послідовних даних, таких як от текст, аудіо чи часові ряди. На противагу DNN, дана мережа має також зворотні зв’язки, що надає змогу зберігати інформацію про попередні стани і враховувати контекст при обробці поточних вхідних даних [25]. Саме робота з послідовними даними є перевагою даної мережі.

Рекурентні мережі відповідно мають унікальну архітектуру, що відповідає їх перевагам. Вихідні дані з одного кроку (рисунок 2.7) повертаються як вхідні дані до наступного, що дозволяє зберігати інформацію з попередніх станів. Завдяки цьому RNN чудово працюють з завданнями, де важливий контекст з попередніх кроків.

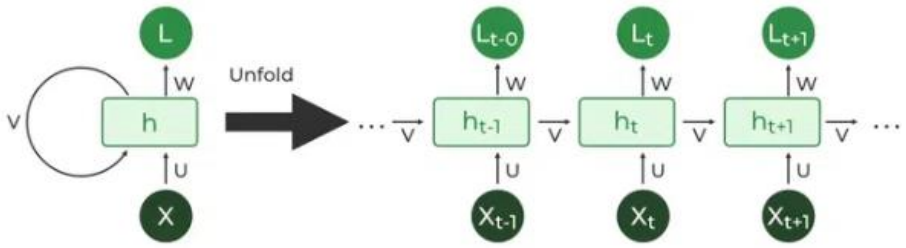


Рисунок 2.7 – Структура RNN [26]

Особливістю RNN є її так званий прихований стан або ж стан пам’яті, що зберігає інформацію з попередніх вхідних даних. Таким чином, використовуючи однакові параметри на усіх етапах, мережа працює узгодження на усіх входах, тим самим зменшуючи складність параметрів в порівнянні з традиційними нейронними мережами.

Архітектура RNN включає наступні ключові компоненти:

- вхідний шар, що є першим шаром в мережі та приймає дані в зрозумілому для моделі форматі, а саме числові значення, вектори, матриці. Кожен елемент послідовності обробляється окремо;
- прихований шар, що є центральним компонентом мережі. Тут відбувається обробка інформації з врахуванням попередніх станів. Особливість прихованого шару в здатності «запам’ятовувати» інформацію про попередні кроки та використовувати її для обробки поточного вводу [27];
- зворотній зв’язок забезпечує передачу результатів попередніх обчислень вперед на наступні кроки. Саме завдяки зворотному зв’язку враховується контекст попередніх даних. Цей компонент перетворює дану мережу з звичайної на мережу з пам’яттю;
- вихідний шар формує кінцевий результат обробки на основі прихованого стану. У випадку класифікації, на виході мережа надаватиме ймовірність приналежності до певного класу. Якщо регресія, то на виході мережа повертатиме числове значення. І у випадку генерації послідовностей, мережа генерує слово за словом;

Але варто виділити, що під час навчання цих мереж виникають проблеми, пов’язані з процесом зворотного поширення помилки через час (Backpropagation Through Time). Двома основними проблемами є зникання та вибух градієнтів. Дані проблеми ускладнюють навчання RNN, особливо при роботі з довгими послідовностями даних.

Зникання градієнтів виникає, коли сигнали під час передачі через мережу для оновлення ваг, з часом стають занадто малими [28]. Причина цьому багаторазове множення похідних функцій активації, значення яких лежать від 0 до 1. В особливо довгих послідовностях такі малі значення експоненційно зменшуються, внаслідок чого градієнт наближається до нуля. Як результат, мережа ніби забуває далекі залежності в даних, так як градієнти для ваг стають надто малими і модель стає нездатною ефективно використовувати інформацію і формувати результат.

Вибух градієнтів є протилежною проблемою, яка вже виникає, коли значення градієнтів стає занадто великими. Це трапляється за умови, коли ваги мережі та сигнали на зворотному поширені експоненційно саме збільшуються через повторне множення значень більших на 1. Така поведінка призводить до нестабільності моделі, а у найгірших випадках обчислювання стає некоректним.

Таким чином, RNN є фундаментальним інструментом для роботи з послідовними даними різних типів. Завдяки своїй здатності враховувати часові залежності та контекст – RNN стали основою для багатьох сучасних технологій як от машинний переклад та системи голосового управління. В контексті розробленої системи дані мережа здатна враховувати взаємозв'язки між різними ділянками маршруту та прогнозувати маршрут залежно від індивідуальних вподобань.

2.5 Засоби розробки

Правильний вибір інструментів та технологій для розробки є надзвичайно важливим етапом під час створення будь-якої системи. Від цього залежить не лише швидкодія програмної системи, а й можливість обслуговування, покращення та подальше масштабування. Саме тому вибір засобів розробки повинен враховувати специфіку розроблюваної системи, масштабування та довгострокові перспективи.

Мова програмування JavaScript – це основа мова програмування навколо якої побудована розроблена веб-система. Серед інших мов програмування JavaScript є однією з найпопулярніших завдяки широкому спектру застосувань, як на стороні сервера так і на стороні клієнта. Дана мова програмування є універсальною і використовується для задач розробки веб-систем, систем штучного інтелекту, для мобільних додатків і навіть ігор. JavaScript дозволяє працювати як з фронтенд частиною системи, тобто те, що бачить користувач, так і з бекенд частиною, що забезпечує зв'язок з базою даних.

Об’єктно-орієнтованість є важливою особливістю JavaScript і робить дану мову програмування універсальною для малих та складних і масштабованих систем. Класи, прототипи та метод структурують код у чіткі та зрозумілі модулі, які поєднують логіку однією сутності. Модульність компонентів розподіляє код на незалежні частини, що надає можливість працювати з кожним компонентом окремо та використовувати їх неодноразово в системі [29].

Важливо визначити, що кожен засіб розробки має як свої переваги так і недоліки. Серед переваг JavaScript можна виділити наступне:

– простота використання, що є найголовнішою перевагою, адже мова має низький поріг входження, що і робить її доступною для новачків;
– величезна екосистема бібліотек та фреймворків, такі як React, Vue, Angular та багато інших. Мова постійно підтримується та покращується завдяки активній спільноті розробників;
– JavaScript може працювати на будь-якій платформі з веб-браузером. Також дана мова програмування підтримується усіма сучасними операційними системами завдяки NodeJs;
– JavaScript вважається стандартом веб-розробки, адже є основною мовою для створення динамічних та інтерактивних веб-сторінок;

Однак JavaScript також має недоліки, а саме:

– нижча швидкість в порівнянні з компільованими мовами програмування, такими як C++ чи Golang;
– асинхронна природа є як перевагою такі недоліком, адже структури callback та promise можуть привести до проблем [30];
– так як JavaScript працює безпосередньо в браузері, це робить її вразливою до XSS атака, якщо не дотримуватись практик безпеки;

Мова розмітки гіпертексту (HTML) – це мова розмітки для створення структури веб-сторінок. Вона є основою всього, що створено в мережі інтернет, дозволяючи веб-браузерам інтерпретувати та показувати текст, зображення,

мультимедіа та інші елементи системи. Синтаксис HTML передбачає використання тегів у вигляді кутових дужок «<» та «>» всередині яких зазначається назва елементу, які мають свою власну функцію та роль в структура веб-документа. Такі теги зазвичай мають парну структуру, тобто є відкриваючих тег, що починається з назви елемента та закриваючий тег [31]. Теги працюють у тісній взаємодії як з мовою стилізації (CSS) так і з JavaScript.

Особливість мови розмітки гіпертексту є її семантичність, а саме те, що теги не лише описують те, як контент виглядає, а й також надають інформацію про його значення. Завдяки цьому, HTML є доступним для пошукових систем, що підвищує SEO-оптимізацію.

HTML є універсальним інструментом, що також використовується для розробки мобільних додатків у поєднанні з різними фреймворками для різних пристроїв на Android та IOS та інших операційний системах. Щодо адаптивності, то це є можливим в комбінації разом з CSS, що надає можливість створювати веб-сайти, які будуть коректно відображатись на будь-яких пристроях та екранах.

Найголовнішим аспектом є те, що HTML постійно розвивається і додається новий функціонал. Саме тому мова постійно є актуальною і сумісною з сучасними технологіями.

Каскадні таблиці стилів (CSS) – це мова стилів, що використовується для розробки візуальної складової веб-сторінок. Вона дозволяє створювати інтерфейси, визначати кольори, шрифти, розміри тексту і навіть поведінку та розташування HTML елементів. Мова CSS підтримує різноманітні анімації, трансформації, що додають динаміку та інтерактивність веб-інтерфейсам.

Важливою перевагою CSS є те, що вона забезпечує модульність. Маючи величезну HTML структуру з важкими зв’язками та компонентами, CSS дозволяє організувати стилів в окремих файлах і змінивши їх в одному місці, зміниться весь проєкт. Аналогічно модульність дозволяє повторне перевикористання стилів для різних елементів [32].

В розробленій системі використовувався як чистий CSS так і в поєднанні з інструментами Bootstrap та Tailwind, що пропонують вже готові стилі та компоненти і, як результат, прискорюють процес розробки.

Бібліотека React – є однією з найпопулярніших бібліотек, створених на основі JavaScript та використовується для розробки швидких, а головне динамічних користувацьких інтерфейсів. Головна ідея React в можливості будувати веб-сервіси у вигляді модульних компонентів, які легко об'єднувати та повторно використовувати.

Ключовим принципом React є використання віртуальної DOM, що оновлюється лише там, де це потрібно. Віртуальна DOM є копією реальної DOM, тобто структури сайту. Коли дані змінюються, React порівнює нову версію віртуальної DOM з попередньою, знаходить різницю і оновлює лише той інтерфейс, що потребує змін [33]. Таким чином, зайві оновлення відсутні і зберігається швидкодія системи. Саме такий «реактивний» підхід забезпечує миттєве відображення інтерфейсу без необхідності оновлювати сторінку.

Стани або ж state є однією з найважливіших концепцій бібліотеки React. Вони використовуються для зберігання даних, що можуть змінюватись протягом роботи компонента. Наприклад, це може бути певна активна вкладка чи сторінка списку. При зміні стану автоматично оновлюється інтерфейс забезпечуючи синхронізацію між даними та відображенням. Окрім стану також застосовуються пропси (props) та ефекти (effects). Пропси дають можливість передавати будь-які дані між компонентами. Ефекти в свою чергу дозволяють виконувати певні дії після рендерингу компонента, наприклад завантаження даних з API [34]. Усі ці компоненти працюють разом та дозволяють створювати динамічні та функціональні веб-додатки.

Платформа Node.js є серверною платформою, що дозволяє виконувати JavaScript код поза браузером. Це потрібно для того, щоб одна мова програмування використовувалась як на стороні клієнта, тобто у браузері, так і на

сервері, що значно спрощує розробку. Сфера використання Node.js охоплює створення масштабованих серверів, API сервісів та веб-додатків.

Подієво-орієнтовна архітектура є одним з ключових принципів Node.js. Вона будується на принципі обробки подій через певний цикл подій, що дозволяє не очікувати відповіді від бази даних чи іншого сервісу, а обробляти інші запити, поки попередні ще виконуються. Іншими слова це асинхронна архітектура і вона є надзвичайно ефективною для завдань з великою кількістю одночасних підключень.

Node.js підтримує модульність завдяки npm (Node Package Manager) [35]. Це надає доступ до мільйонів бібліотек та модулів. Також платформа дозволяє створювати навіть власні модулі, що збільшує гнучкість розробки.

Популярність Node.js з часом зростає і все зростає завдяки активній спільноті розробників та широкому функціоналу Її використовують у великих компаніях, таких як Netflix, LinkedIn, Uber, що показує надійність та універсальність платформи. Враховуючи усі ці фактори, Node.js був обраний для розробленої програми.

Firebase є хмарною платформою для розробки веб- та мобільних додатків, розроблена Google, що надає широкий спектр інструментів для управління даними, автентифікації, аналітики та хостингу. Платформа підтримує як прості так і складні проєкти та забезпечує синхронізацію даних між клієнтами через базу даних в режимі реального часу [36]. Firebase пропонує такі методи автентифікації для користувачів такі як електронна пошта, соціальні мережі або ж номер телефону.

Переваги Firebase:

– Firebase дозволяє синхронізувати дані в реальному часі, що є особливо корисним для додатків, які потребують миттєвого оновлення інформації;

- автоматичне масштабування ресурсів в залежності від навантаження на додаток дозволяє ефективно керувати навантаженням без необхідності вручну налаштовувати сервери;
- Firebase надає потужні механізми автентифікації та безпеки даних. Це включає підтримку автентифікації через електронну пошту, соціальні мережі, а саме Google та Facebook і анонімну автентифікацію [37];
- можливість безкоштовного хостингу веб-додатків з використанням SSL сертифікатів. Завдяки цьому можна розгортати додатки швидше і з високим рівнем безпеки;

Середовище розробки Visual Studio Code є зручним та універсальним інструментом для розробки програм будь-якою мовою програмування. Середовище підтримує JavaScript, Python, C++, Java, PHP, Go та багато інших. Це можливо завдяки модульності середовища, що реалізовано через офіційний «магазин», де можна завантажити потрібне розширення. Розширення додають в середовище функції підтримки різних мов програмування, інструменти тестування, автоматизація та стилізація коду. Тому розробники мають можливість налаштувати Visual Studio Code під свої потреби.

Інтелектуальний автодоповнювач (IntelliSense) є найпопулярнішою особливістю середовища. Даний інструмент аналізує код в режимі реального часу та підказує можливі виправлення, функції за допомогою яких можна вирішити завдання та змінні, що використовуються в системі [38].

Також потрібно відзначити вбудований термінал. Він дозволяє виконувати команди безпосередньо в середовищі. Розробникам не потрібно перемикатись інші інструменти, адже Visual Studio Code об’єднує це в собі і робить можливим виконувати інші скрипти, запускати сервери, взаємодіяти з git не залишаючи редактор коду.

Visual Studio Code постійно оновлюється та покращується. Спільнота користувачів активно створює нові розширення та інструменти. На основі всього вище наведеного, дане середовище було обрано для розробки.

Фреймворк Cypress – це сучасний фреймворк створений для автоматизованого тестування веб-додатків. Він створений для полегшення процесу написання та виконання тестів. Cypress працює безпосередньо в веб-браузері і тестує додаток ніби як звичний користувач. Cypress підтримує такі типи тестування:

– енд-то-енд тестування перевіряє весь робочий процес додатка з точки зору кінцевого користувача. Відбувається тестування повного сценарію, наприклад від входу в систему до генерації маршруту. Таке тестування гарантує, що усі компоненти працюють разом та без помилок;
– інтеграційні тести в свою чергу перевіряють взаємодію між окремими модулями або ж компонентами [39]. Інтеграційні тести дозволяють переконатись, що інтегровані між собою компоненти коректно працюють разом;
– юніт тести є стандартним типом тестування і фокусуються на перевірці роботи окремих функцій, але в ізоляції. Таким чином можна переконатись, що кожен модуль правильно працює сам по собі, без ніякої залежності від інших компонентів;

Перевагами Cypress є його простота використання, інтуїтивний синтаксис, широка документація та гнучкість у роботі з різними видами тестів. Саме тому цей фреймворк було обрано для тестування системи.

GitHub – це хмарний сервіс для зберігання, спільного використання та управління коду. Сервіс базується на системі контролю версій Git. Сервіс є надважливим додатком під час розробки будь-якого продукту, особливо якщо це відбувається в команді розробників.

Особливістю сервісу є підтримка спільної роботи над проєктами. Розробники мають можливість створювати сховища зберігання коду та надавати доступ іншим учасникам команди. В будь-якому проєкті можна побудувати ієрархію за якою відповідальна особа, зазвичай це найдосвідченіший розробник, перевірятиме зміни в коді перед їх інтеграцією в основну гілку. Основа гілка є центральним місцем, де зберігається основний прогрес проєкту і служить базою для розробки або виправлення помилок [40].

Github інтегрується з іншими інструментами розробки, а у випадку розробленої системи – з Visual Studio Code. Така інтеграція дозволяє керувати проєктом та гілками одразу ж в програмному середовищі. Будь-які зміни в коді чи в файловій структури, як от видалення чи додавання файлу, будуть автоматично записані і інші розробники зможуть це легко відстежити.

Висновки до розділу 2

В цьому розділі було проаналізовано алгоритми машинного та глибокого навчання, описано їх характеристики та переваги/недоліки. Серед алгоритмів машинного навчання розглянуто метод логістичної регресії, випадковий ліс, метод k-найближчих сусідів та метод опорних векторів (SVM). Також було розглянуто алгоритми глибокого навчання, а саме глибокі нейронні мережі (DNN) та рекурентні нейроні мережі (RNN) та їх роль в вирішенні задачі побудови рекомендації маршрутів на основі збережених.

Щодо набору програних засобів, які використовувались, було розглянуто технічний стек, що включає React, Firebase та Node.js. Середовищем розробки слугувало Visual Studio Code, тестування відбувалось за допомогою фреймворку Cypress, а контроль версій відбувався з використанням GitHub.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ПОБУДОВИ ВЕЛОМАРШРУТІВ

У цьому розділі представлено архітектуру програмного забезпечення та описано його функціональні можливості. Основна увага приділяється структурі системи, способу взаємодії між її компонентами, а також ключовим аспектам реалізації функціоналу. Описані методи дозволяють зрозуміти, як саме побудована система та яким чином реалізуються її основні завдання.

3.1 Архітектура системи

Під час проєктування програмного продукту, було розроблено діаграму прецедентів, що містить два основних елементи:

- актор(учасник) – певна роль в системі, що виконується в взаємодії з прецедентами чи сутностями. В розробленій системи є актори користувач та адміністратор;
- прецедент в свою чергу є описом конкретного аспекту поведінки системи і показує не результат, а тільки що саме виконується [41]. Наприклад, генерувати, зберігати маршрути;

Діаграма прецедентів для розробленої системи зображена на рисунку 3.1. Основний актор системи це користувач, який взаємодіє з сутністю маршрути і може виконувати базові дії над ними, а саме:

- перегляд інформації про маршрут;
- генерувати випадковий маршрут;
- генерувати маршрут з точки А в тому В;
- зберігати маршрут;

– видаляти маршрут;

– вивантажувати маршрут;

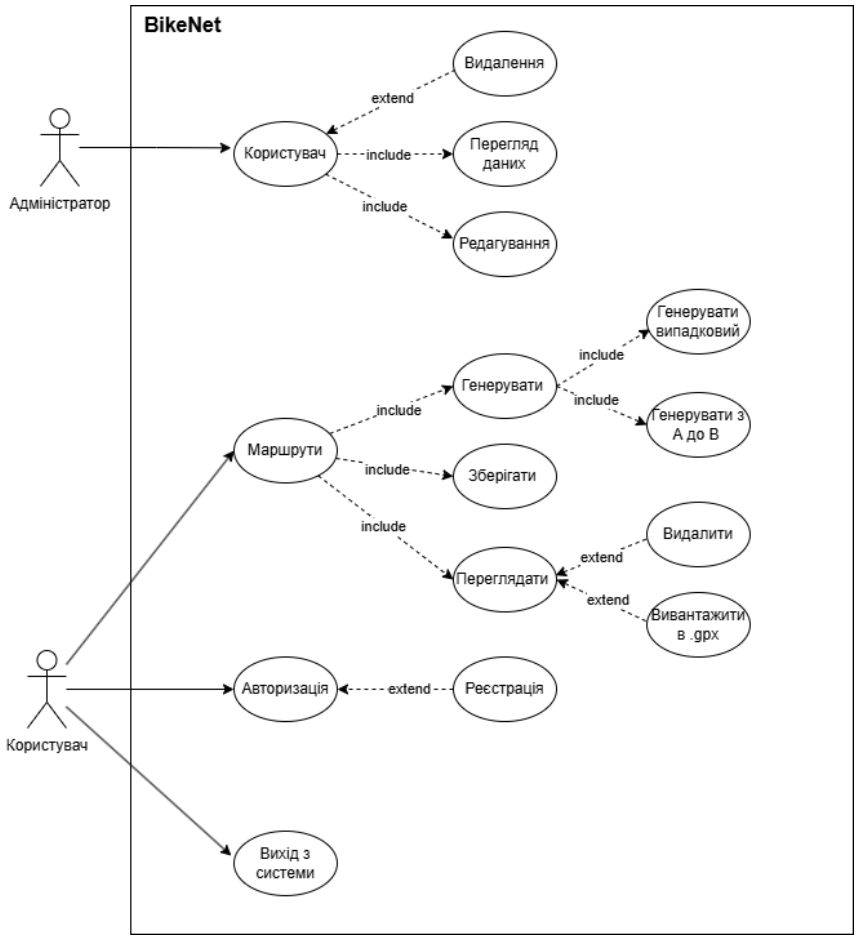


Рисунок 3.1 – Діаграма прецедентів

Адміністратор є підсутністю актора «Користувач» і його функції є додатковими. Вони передбачають керування сутностями користувач і його маршрутами, а саме перегляд, видалення та оновлення.

Окрім діаграми прецедентів, також була розроблені діаграма класів системи. Діаграма класів детально описує основі об’єкти системи, атрибути та зв’язки між ними. За допомогою діаграми формується чітке уявлення про загальну структуру розроблюваної системи [42]. Основними компонентами в

розробленій системі побудови веломаршрутів на основі моделей штучного інтелекту є користувач, маршрути та похідні об’єкти, що використовуються для формування фільтрів маршруту. На рисунку 3.2 зображено загальну структуру діаграми класів в розробленій системи.

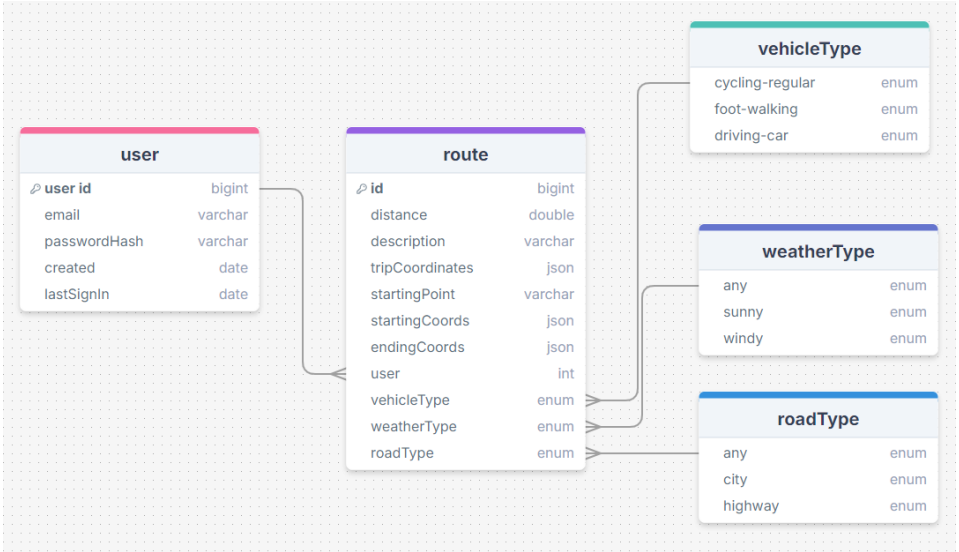


Рисунок 3.2 – Діаграма класів

Центральним об’єктом є маршрут (route). Він як є сутністю з якою взаємодіє користувач, так і є результатом, що продукує система. Зв’язком, що підтримується між ними є користувачем та маршрутом є один до багатьох, тобто один користувач може мати багато маршрутів в системі. Кожен маршрут є унікальним в системі, адже будується на строгих координатах, тому неможливо ситуації, коли кілька користувачів матимуть однаковий маршрут.

Таблиці weatherType, roadType, vehicleType є додатковими і використовуються для формування фільтрів під час побудови маршрутів. Таблиця weatherType містить інформацію про типи погоди, що користувач може обрати під час побудови маршруту. Таблиця roadType містить про можливі типи

дороги в системі, а таблиця vehicleType містить в собі інформацію про типи транспорту в системі.

Кожен маршрут пов’язаний з одним типом погоди, адже кожен маршрут унікальний та має власні налаштування. Однак один і той самий тип погоди може застосовуватись для багатьох маршрутів. Таким чином вони мають зв’язок «багато до одного», де кожен кілька записів «routes» можуть посилатись на один і той самий тип погоди. Поле weatherType в «routes» є зовнішнім ключем, що пов’язує маршрути з «weatherType», де зберігаються можливі типи погоди. Аналогічним чином відбувається і з таблицями «roadType», «vehicleType».

3.2 Сервіси API

Важливим ресурсом даної програми – є дані, які використовуються для створення персоналізованих і оптимізованих маршрутів. Для досягнення цього в системі використовуються різні API, зокрема OpenStreatMaps, Overpass, OpenRouteService та LocationIQ. Кожен з цих сервісів відіграє свою роль у зборі, обробці та візуалізації даних.

Правильне налаштування API є надзвичайно важливим для коректної роботи системи, адже він якості, швидкості доставки даних залежить і швидкодія системи та загальний досвід користувача. Саме тому при інтеграції API були використані сучасні підходи проєктування.

Картографічні дані від **OpenStreatMaps** система отримує за допомогою Tile-серверу, який вже надає готові зображення карти у вигляді тайлів. Тайл – це невеликий квадратний фрагмент мапи, який покриває певну ділянку. Система динамічно завантажує тайли під час перегляду мапи або ж руху по ній. Запит до серверу в системі зображено на рисунку 3.3. Запит до сервера потребує наступних параметрів:

– {s}, субдомен для розподілення навантаження;

– {z}, рівень масштабування;
– {x}{y}, координати тайла на мапі;

```
<TileLayer
  url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
  attribution='&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>contributors'/>
```

Рисунок 3.3 – Запит даних до OpenStreatMaps

Наступним важливим API сервісом є **Overpass**, що використовується в системі для отримання структурованих даних про місцевість з сервісу OpenStreetMap у форматі JSON. Як і в попередньому сервісі, Overpass не потребує приватного API ключа, що спрощує процес інтеграції та використання даних.

Запит до сервісу (рисунок 3.4) Overpass складається з набору інструкцій, де визначається область пошуку та параметри фільтрації. Основні частини запиту містять:

– форматі відповіді у форматі json [out:json];
– блок фільтрації даних, де визначаються об’єкти з тегом ["natural"="water"], а саме природні водойми та виключаються об’єкти з тегом ["water"!="reservoir"], тобто водосховища;
– блок місцезнаходження містить радіус пошуку та координати точки навколо якої проводиться пошук;
– формат вихідних даних вказує на те, що відповідь від API повинна містити геометрію об’єктів.

```
const query = `
   [out:json];
   (
     nwr["natural"="water"]["water"!="reservoir"](around:2000, ${pointRaw["lat"]},
${pointRaw["lng"]});
   );
   out geom;
 `;
 const response = await fetch(
   `https://overpass-api.de/api/interpreter?data=${encodeURIComponent(query)}`
 );
```

Рисунок 3.4 – Виклик сервісу Overpass

OpenRouteService є потужним геоінформаційним сервісом, що використовується в системі для побудови маршрутів [43]. Для отримання даних система надсилає запит, що зображено на рисунку 3.5. Запит складається з наступних параметрів:

– this.state.vehicle, тип транспорту, наприклад велосипед, автомобіль чи піший хід;
– coordinates, координати точку старту;
– extra_info, інформація про маршрут, яку ми бажаємо отримати. Додаткова інформація може містити інформація про нахили, висоти та іншу корисна інформацію щодо маршруту;
– length, бажана довжина маршруту в метрах;
– avoid_polygons, полігони у форматі geojson, які потрібно уникати під час побудови маршруту;
– API_CODE, унікальний ключ доступу, який дозволяє сервісу ідентифікувати користувача;
– Accept для налаштування бажаного формату для отримання даних, а саме json, geojson, gpx та інші;

```
const body = JSON.stringify({
        coordinates: [[startingLon, startingLat]],
        extra_info: ["surface", "steepness"],
        options: {
          round_trip: {
            length: roundTripLength * 1000,
            points: 3,
            seed: seed
          },
          avoid_polygons: {
            type: "MultiPolygon",
            coordinates: avoidBorders
          }
        }
      });

await fetch(
      `https://api.openrouteservice.org/v2/directions/${this.state.vehicle}/geojson`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json;",
          Accept:
            "application/json, application/geo+json, application/gpx+xml, img/png;
charset=utf-8",
          Authorization:
            "API_CODE",
        },
        body: body
      }
    )
```

Рисунок 3.5 – Виклик сервісу OpenRouteService

І останнім API сервісом**,** що використовувався в розробленій системі є **LocationIQ.** Сервіс надає можливості геокодування, автозаповнення, пошук адрес. LocationIQ підтримує інтеграцію з додатками, де потрібна робота з геолокаційною інформацією [44]. В розробленій системі, коли користувач вводить свою адресу в текстовому форматі з номером вулиці та будинком, сервіс геокодує даний текст та надає його у форматі точних координат, які згодом використовуються в системі.

Запит до сервісу потребує лише одного важливого параметру, а саме startingPoint або точку старту. Після цього система чекає відповідь від сервісу і отримує список з можливих місць. Приклад запиту зображено на рисунку 3.6.

```
var startingURL =
  "https://eu1.locationiq.com/v1/search.php?key=" +
  geocodingKey +
  "&q=" +
  this.state.startingPoint +
  "&format=json";
```

Рисунок 3.6 – Виклик сервісу LocationIQ

Надзвичайно важливим є етап поєднання розробленої системи з платформами постачальниками даних. Правильна інтеграція гарантує безперебійний доступ до даних та стабільність системи навіть під час високої активності з боку користувачів. Також коректна інтеграція важлива з боку безпеки. Стандартизовані канали обміну є захищеними до чуттєвих даних, як от географічні дані користувача. Саме тому була розроблена лінійна архітектура з асинхронним підходом. Архітектура інтеграції зображена на рисунку 3.7.

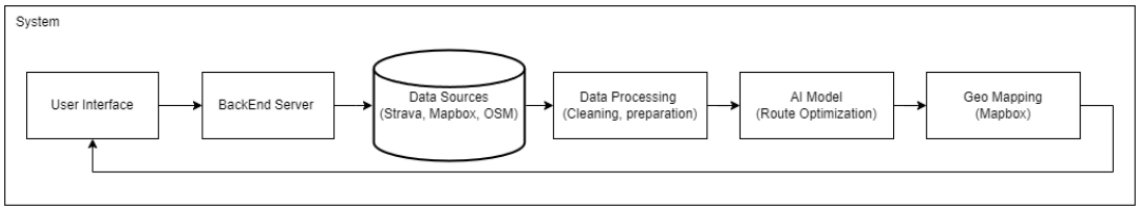


Рисунок 3.7 – Інтеграція з сервісами API [45]

Лінійна архітектура була обрана через наступні причини: прозорість роботи, модульність, швидкість роботи та гнучкість. Прозорість досягається завдяки чіткій структурі, адже завдяки ній можна легко відслідковувати кожен етап обробки даних. Модульність забезпечує те, що кожен компонент виконує свою чітку функцію. Це значно спрощує процес розробки та підтримки різних компонентів. Лінійна обробка запитів мінімізує затримки та дозволяє швидко отримувати результати. Останньою перевагою є гнучкість, адже система може легко масштабуватись.

Архітектура інтеграції с сервісами API складається з таких компонентів:

– користувач (user). Він є центральним компонентом в архітектурі, що напряму взаємодіє лише з сервером додатку. Взаємодія відбувається через зручний інтерфейс системи. Користувач заповнює початкові дані

маршруту, а саме точки старту, відстань та інші параметри після чого дані передаються до серверу;

- сервер (backend server). Компонент приймає запит від користувача, обробляє його та надсилає відповідні API запити до постачальників даних; Функціями компоненту є авторизація, верифікація даних та обмін інформацією між іншими компонентами;
- постачальники даних (data sources). Це компонент представлений сервісами, які використовуються для отримання даних. OpenStreatMaps передає картографічні дані, OpenRouteService обраховує маршрути, LocationIQ проводить геокодування адрес в координати та навпаки, Overpass дозволяє отримувати дані від OSM сервісу. Після того як сервер отримав запит від користувача та інформацію про маршрут, він передає запити постачальникам сервісу та отримує від них дані;
- моделі штучного інтелекту (AI models) аналізують отримані дані та персоналізують їх відповідно до індивідуальних вподобань користувача. Модель також бере до уваги історію поїздок та збережені маршрути користувача і будує на основі них наступну рекомендацію.
- геокодування (geo mapping). Компонент відображає підготовлені дані на інтерактивні карті за допомогою інструментів Leaflet. Також компонент відповідальний за відображення маршруту та ключових елементів, а саме точки інтересу чи географічні особливості.

3.2 База даних

Використання Firebase у даному проєкті зумовлено тим, що ця база даних ідеально підходить для роботи з динамічними даними, які оновлюються в реальному часі. Веломаршрути залежать від багатьох змінних, таких як обрані користувачем налаштування, поточні умови на маршруті, погода та інші

параметри. Firebase дозволяє швидко синхронізувати всі ці дані між користувачами та базою даних.

Двома головними колекціями в реалізованій базі даних є користувач (user) та маршрут (route). Між ними підпримується зв'язок один до багатьох тобто один користувач може мати багато різних маршрутів. Також існують об'єкти типу enum, такі як vehicleType, weatherType, roadType, що містять значення можливі типу транспорту, типу погоди та типу дороги відповідно. Можливі значення для даних фільтрів зберігаються окремо в інших таблиця.

Основною таблицею бази даних та сутністю для «user» є таблиця «routes». Вона забезпечує побудову маршрутів та містить наступну інформацію:

1. id – ідентифікатор маршруту;
2. distance – довжина маршруту;
3. description – опис маршруту;
4. tripCoordinates – координати маршруту;
5. startingCoords – координати початкової точки;
6. endingCoords – координати кінцевої точки;
7. user – зовнішній ключ до таблиці user;
8. vehicleType – тип транспорту;
9. weatherType – тип погоди;
10. roadType – тип дороги;

Деякі параметри є опціональними та не завжди присутні в об'єкті класу «маршрут». Наприклад, endingCoords не заповнюється при випадковій поїздці, адже startingCoords і є кінцевою в цьому випадку. Поле description також є опціональним і тому в базі маршрути можуть бути з пустими назвами. Усі інші параметри є обов'язковими для маршруту і заповнюються коли користувач зберігає маршрут. Таблиця «routes» зображена на рисунку 3.8.

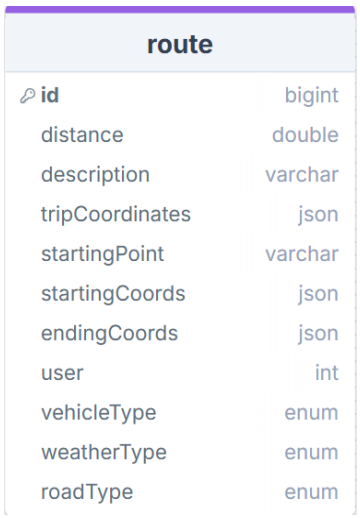


Рисунок 3.8 – Таблиця «route»

Наступною важливою таблицею є «user», що зберігає інформацію про користувача. Зв’язок між таблицями «user» та «route» побудований як один до багатьох, тобто один користувач може мати кілька маршрутів. Таблиця «user» зображена на рисунку 3.9 та складається з таких полів як:

1. user id – ідентифікатор користувача;
2. email – пошта користувача;
3. passwordHash – захешований пароль;
4. created – дата створення облікового запису;
5. lastSignIn – дата останнього входу в систему;

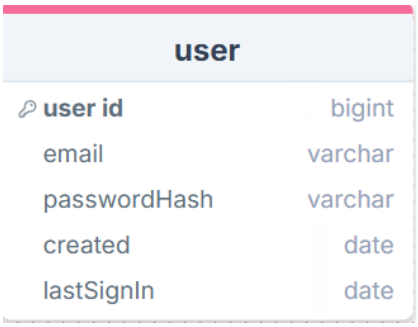


Рисунок 3.9 – Таблиця «users»

Додатковими таблицями є vehicleType, weatherType та roadType. Перша таблиця містить можливі значення типу транспорту, а саме велосипед, автомобіль та піший хід. Таблиця weatherType зберігає інформацію про можливі значення типу погоди, що може обрати користувач, а саме вітряно, сонячно або будь-яка погода. І таблиця roadType, що містить інформацію про типи дороги, з поміж який користувач може обрати. Можливі значення включають міську дорогу, шосе або ж будь-яка дорога.

3.3 Специфікація функцій

В межах розробленої системи побудови веломаршрутів специфікація функцій описує ключові функції системи, їх призначення та принцип роботи.

Метод calculateDistance є важливим, коли користувач намагається побудувати маршрут з врахуванням погодних умов або точок інтересу. Загальна логіка погодного фільтру наступна: якщо надворі сонячно, маршрут будуватиметься навколо водойм або ж як мінімум одна точка маршруту буде проходити біля водойми, а якщо ж погода вітряна, то маршрути генеруватимуть навколо захисних природних об’єктів, а саме лісів.

Після активації погодного фільтру система отримує дані від сервісу Overpass про лісові та водні об’єкти навколо шуканої області. Будується випадковий маршрут і кожна точка побудованого маршруту порівнюється з точками об’єктів отриманих даних та знаходить відстань між ними за допомогою формули зображеної на рисунку 3.10. Формула Гаверсинуса застосовується для обчислення відстаней між точками на землі [46]. Вся інформація зберігається в список та точка, що знаходиться найближче до створеного маршруту і в межах одного кілометру – додається до маршруту. Якщо ж поруч відсутні потрібні

об’єкти, користувач побачить повідомлення, про те, що маршрут не було знайдено.

```
function calculateDistance(lat1, lon1, lat2, lon2) {
  const R = 6371;
  const dLat = ((lat2 - lat1) * Math.PI) / 180;
  const dLon = ((lon2 - lon1) * Math.PI) / 180;
  const a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos((lat1 * Math.PI) / 180) *
      Math.cos((lat2 * Math.PI) / 180) *
      Math.sin(dLon / 2) *
      Math.sin(dLon / 2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  return R * c;
}
```

Рисунок 3.10 – Функція обчислення відстані між точками

Метод login відповідає за вхід користувача в систему. Для успішного виконання вона потребує пошту та пароль. Після натискання кнопки входу, дані введені користувачем передаються в метод signInWithEmailAndPassword, що працює всередині Firebase. Введені користувачем дані відправляються до Firebase Authentication, і якщо обліковий запис з таким іменем існує і пароль вірний – користувач успішно входить в систему. Якщо дані не вірні, то повертається помилка, що містить код помилки та повідомлення.

Метод signup виконує реєстрацію користувача. Так як само як і login, метод потребує пошту, пароль, але і також нове поле – ім’я користувача. Дані передають в метод createUserWithEmailAndPassword, що створює новий обліковий запис за умови якщо такого користувача не існує та якщо пароль та логін відповідають правильній складності та формату. Після успішного створення користувач бачить повідомлення з вітанням та описом, що акаунт створено.

Компонент SavedRouteCard відповідає за відображення збережених маршрутів користувача. Компонент відображає інформацію про опис, початкову/кінцеву точки, тип транспорту, дистанцію та час створення маршруту. Даний компонент створений з дотриманням принципів React, тому

використовується багаторазово в системі як для відображення збережених, так і для відображення рекомендованих маршрутів, адже це залишається тою ж сутністю.

Компонент routing є доволі важливим в системі та відповідає за обробку даних створеного маршруту та побудову порядку дій проходження маршруту. На вхід routing отримує інформацію про точки маршруту, а саме їх географічні координати та будує інформацію про рух по маршруту, наприклад «Поверніть праворуч через 500 метрів по вулиці…». Однак, API сервіси не підтримують українську мову, тому було створено масив можливих значень інструкцій, що зображено на рисунку 3.11.

```
const instructionMapping = {
  'Head east': 'Прямуйте на схід',
  'Head northeast': 'Прямуйте на північний схід',
  'Head south': 'Прямуйте на південь',
  'Head southwest': 'Прямуйте на південний захід',
  'Head southeast': 'Прямуйте на південний схід',
  'Head west': 'Прямуйте на захід',
  'Head northwest': 'Прямуйте на північний захід',
  'Turn sharp left': 'Різко поверніть ліворуч',
  'Turn left': 'Поверніть ліворуч',
  'Turn slight left': 'Трохи поверніть ліворуч',
  'Turn right': 'Поверніть праворуч',
  'Turn sharp right': 'Різко поверніть праворуч',
  'Keep left': 'Тримайтеся ліворуч',
  'Keep right': 'Тримайтеся праворуч',
  'Continue straight': 'Продовжте рухатися прямо',
  'Enter the roundabout and take the 1st exit': 'Заїдьте на кільце і вийдіть на 1-й виїзд',
  'Arrive at': 'Приїдьте на',
  'Arrive at your destination': 'Приїдьте до вашого місця призначення',
  'Arrive at your destination, on the right': 'Приїдьте до вашого місця призначення, справа',
  'Arrive at your destination, on the left': 'Приїдьте до вашого місця призначення, зліва',
  'Turn right onto': 'Поверніть праворуч на',
  'Turn left onto': 'Поверніть ліворуч на',
  'Continue straight onto': 'Продовжте рухатися прямо на',
  'Keep right onto': 'Тримайтеся праворуч на',
  'Keep left onto': 'Тримайтеся ліворуч на',
  'Arrive at, on the right': 'Приїдьте на, справа',
  'Arrive at, on the left': 'Приїдьте на, зліва',
  'onto вулиця': 'по вулиці',
  'on вулиця': 'по вулиці',
  'onto': 'по',
  'Turn right': 'Поверніть праворуч',
  'Turn left': 'Поверніть ліворуч',
};
```

Рисунок 3.10 – Функція обчислення відстані між точками

Кожен елемент отриманого масиву інструкцій відбирається та посилаючись на instructionMapping, функція ніби «перекладає», але по суті замінює частинку кожної інструкції англійською на українську мову.

Зберігання API ключів до сервісів даних та інша конфіденційна інформація в файлі типу «.env» є стандартною практикою під час розробки програмних систем. Перевагами практики є безпека, зручність налаштування, легкість масштабування та гнучкість. На рисунку 3.11 зображено заповнення .env файлу.

```
REACT_APP_API_KEY=your_api_key_here
DATABASE_PASSWORD=your_database_password
```

Рисунок 3.11 – Заповнення .env файлу

Не менш важливим блоком є обробка натискання мапи користувачем. Під час інтерактивного режиму, кожен клік на мапі фіксується та надсилається в обробник. Далі за допомогою параметру event, що містить координати обраної точки система зберігає дані в спеціальні зміні. Обробка кліку на мапі зображена на рисунку 3.12.

```
handleClickOnMap = (event) => {
  const { isMapClickableStart } = this.state;
  const { isMapClickableEnd } = this.state;

  const lat = event.latlng.lat.toFixed(6);
  const lng = event.latlng.lng.toFixed(6);

  if (isMapClickableStart) {
    this.setState({
      startingPoint: `${lat}, ${lng}`,
    });
  }
  else if (isMapClickableEnd) {
    this.setState({
      endPoint: `${lat}, ${lng}`,
    });
  }
};
```

Рисунок 3.12 – Обробка натискання на мапі

3.4 Вимоги до технічного забезпечення

Система призначена для генерації маршрутів в режимі реального часу. Так як система передбачає зв’язок з API сервісами поза розробленої системи, пристрої повинні маті відповідно технічні характеристики для швидкої та стабільної роботи. Мінімальні та рекомендовані вимоги до обладнання наступні:

- процесор (CPU). Мінімум, що необхідний для стабільної роботи це двоядений процесор з тактовою частотою 2.0 ГГц. Серед моделей це може бути Intel Core i3 або ж еквівалент AMD;
- оперативна пам’ять (RAM). Мінімум для роботи з системою це 4 гігабайти. Цього достатньо для базової роботи з системою без великого навантаження. Рекомендований розмір пам’яті 8 гігабайтів або ж більше, що дозволяє плавну обробку мапи та швидке завантаження маршрутів;
- відеокарта (GPU). Мінімально достатньою є інтегрована відеокарта, як от Intel HD або еквівалент. Для якісною роботою з OpenGL рекомендована вже дискретна відеокарта Nvidia 1050-тої серії та вище або ж еквівалент AMD Radeon RX 550;
- операційна система. Обмеження лише стосується старіших версій систем. Загалом Windows 10,11, macOS та Ubuntu покривають більшість потреб користувачів;
- екран. Мінімально рекомендована роздільна здатність 1366x768, рекомендованою ж є 1920x1080 або вище для зручного перегляду та взаємодії з мапами;
- швидкість інтернету є надзвичайно важливою вимогою при роботі з системою. Мінімальна 2 Мбіт/с, що достатньо для базового завантаження мапи. Рекомендованою є 10 Мбіт/с і головне підключення повинно бути стабільним. Висока швидкість інтернету робить можливим швидку взаємодію з API, картами та інструментами системи;

- браузер. Важливою вимогою є оновлена версія браузера з підтримкою JavaScript, оскільки система використовує динамічний контент та бібліотеки. Серед браузер можливі Google Chrome, Mozilla Firefox та Microsoft Edge;
- визначення місця розташування (gps). Для повного набору функцій та максимально точного налаштування маршрутів пристрій повинен підтримувати геолокацію;

Вищезазначені вимоги стосуються лише кінцевого користувача. Для стабільного розгортання системи та підтримки стабільності роботи необхідно врахувати наступні рекомендовані параметри:

- процесор (CPU). Мінімум 4 ядра з тактовою частотою 2.5 ГГц. Рекомендовано 8-16 ядер для обробки величезної кількості одночасних запитів. Для цього підійде Xeon серія процесорів Intel або ж AMD EPYC;
- оперативна пам’ять (RAM). Мінімум потрібно 16 гігабайт для обробки невеликого трафіку. Щоб мати змогу якісно працювати з великою кількістю запитів користувачів потрібно 32-64 гігабайт;
- сховище повинно бути лише на SDD накопичувачах, адже вони швидші. Мінімум 250 гігабайт для операційної системи, бази даних та роботою з кешем. Рекомендований розмір 1 терабайт з високою швидкістю запису та читання та для роботи з картографічними даними;
- операційною системою для серверу є Ubuntu Server з найновішою версією;
- у ролі веб серверу може слугувати Nginx або Apache. Для розробленої системи найкраще підійде саме Nginx, адже його перевагою є швидка робота з статичними файлами та балансування навантаження [47];
- як хмірні сервіси рекомендується застосовувати таких постачальників як AWS, Microsoft Azure, Google Cloud Platform;
- для моніторингу краще застосовувати Prometeus, а для інтерактивної візуалізації як використання даних так і швидкодії системи Grafana;

– для зменшення затримки при завантажені карт рекомендується застосовувати глобальну розподілену мережу серверів Content Delivery Network (CDN). Вона забезпечуватиме швидку доставку контенту , а саме карт, зображень, відео прямо до кінцевого користувача; Найпопулярнішим CDN провайдером є Cloudflare. Однією з переваг є інтеграція інструментів кібербезпеки та аналітики. Сервіс надає захист від DDoS-атак аналізуючи підозрілий трафік та блокує масові запити на сервер [48];

Вище визначені рекомендовані параметри для користувача та для системи в цілому. Для користувача представлено мінімальний та рекомендований набір, що дозволяє безпроблемна використовувати систему та отримувати готові маршрути. Для системи, а саме для серверної частини представлено вимоги до апаратного забезпечення, а саме процесор, оперативна пам’ять, розмір сховища та інші. Також представлені не лише вимоги до апаратного, а й до програмного забезпечення включаючи веб-сервер, хмарні сервіси та сервіси доставки контенту в систему.

Висновки до розділу 3

В даному розділі описано архітектуру програмної системи, побудовано діаграмою прецедентів та описано її головні компоненти. Розглянуто використання API сервісів OpenStreatMaps, Overpass, OpenRouteService та LocationIQ саме з технічного боку та подальше використання в системі. Описано структуру бази даних та її головні елементи, а саме сутності користувач, маршрут та їх зв’язки. Наприкінці описані основі методи системи, їх призначення та принцип роботи.

4 ВЗАЄМОДІЯ КОРИСТУВАЧА З ВЕБ-СИСТЕМОЮ

В даному розділі показано те, як користувач може взаємодіяти з системою та її методами. Описано його основні дії та результат роботи програми. Також представлено логічну структуру веб-системи з деталізацією взаємозв'язків між основними компонентами та їх функціоналом.

4.1 Інтерфейс веб-системи

Після запуску системи користувач бачить головний елемент системи, а саме карту, що центрована навколо університету. Головне меню складається з декількох блоків, а саме блок реєстрації/входу, блок побудови маршруту та блок мапа, що зображені на рисунку 4.1.

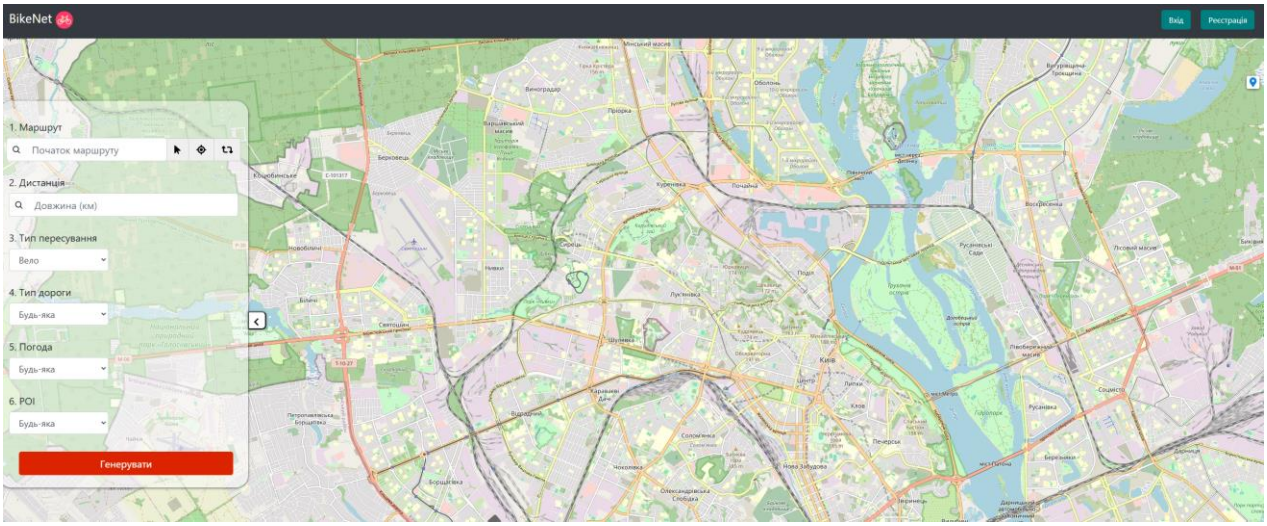


Рисунок 4.1 – Головне меню системи

Користувач, що вперше, може згенерувати маршрут за допомогою форми, що наведена зліва. Є два способи, задати початкову точу або ж задати старт та кінець подорожі. Додатковими функціями є задання точки на мапі та використання поточного місцезнаходження користувача. На рисунку 4.2 наведена форма для заповнення для створення маршруту.

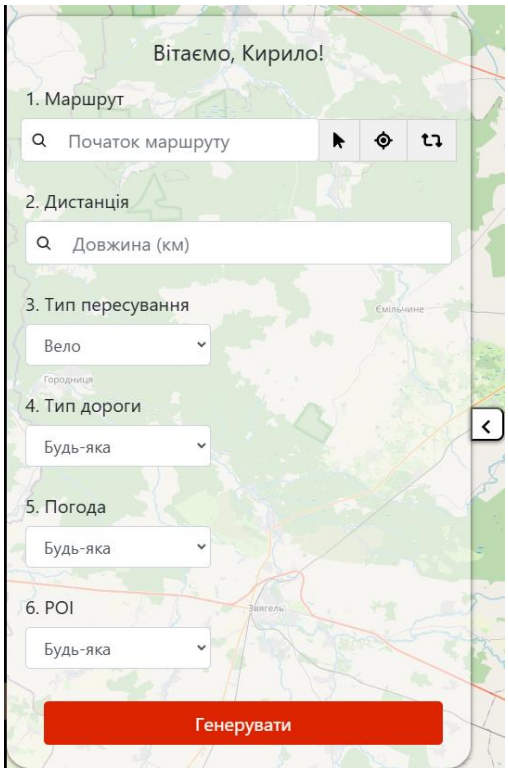


Рисунок 4.2 – Заповнення форми випадкової поїздки

Обов’язковими параметрами при створенні випадкового маршруту є точка старту та довжина в кілометрах. Тип пересування за замовчуваннях є велосипед, але також користувач може змінити його на автомобіль або ж піший хід, як показано на рисунку 4.3. Для велосипеда в такому випадку маршрут буде оптимізовано для велодоріжок, а автомагістралі будуть уникатись. Тип транспорту автомобіль буде уникати доріг, що заборонені для автомобіля. Враховуються обмеження швидкості та наявність тунелей або ж мостів. Біг або

піший хід передбачає побудову безпечних маршрутів саме по тротуарах, парках чи пішохідних зонах.

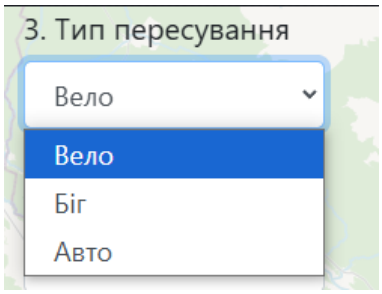


Рисунок 4.3 – Заповнення типу пересування

Типом дороги для веломаршруту за замовчуванням є будь-який, тобто система не звертає увагу на цей параметр. Серед інших варіантів користувач може обрати з поміж міською та шосе, що зображено на рисунку 4.4. Міська дорога передбачає побудову маршрутів всередині певних населених пунктів і лише всередині них, незалежності від відстань. Тобто усі можливі маршрути генеруватимуться в межах населеного пункту, який обрав користувач. Головна ідея шосе в побудові маршрутів саме по основних транспортних дорогах в шуканому місці для максимальної швидкості та якості дороги.

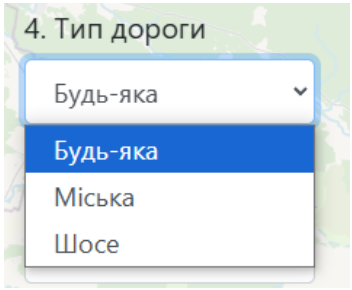


Рисунок 4.4 – Заповнення типу дороги

За потреби, користувач може налаштувати враховування погодних умов при побудови маршруту. Серед можливих значень доступні будь-яка погода, що встановлено за замовчуванням, сонячна та вітряна погода, що проілюстровано на рисунку 4.5. Сонячна погода передбачає побудову маршрутів навколо водойм, а у випадку вітряної погоди, побудова навколо захисних природних об’єктів.

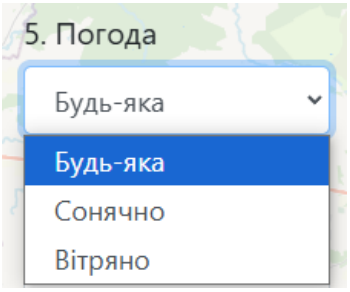


Рисунок 4.5 – Заповнення інформації про погодні умови

Другий спосіб це додати початкову та кінцеву точку самостійно. Це дозволить згенерувати маршрут з точки А в точку В. Форма статичної побудови маршруту на рисунку 4.6.

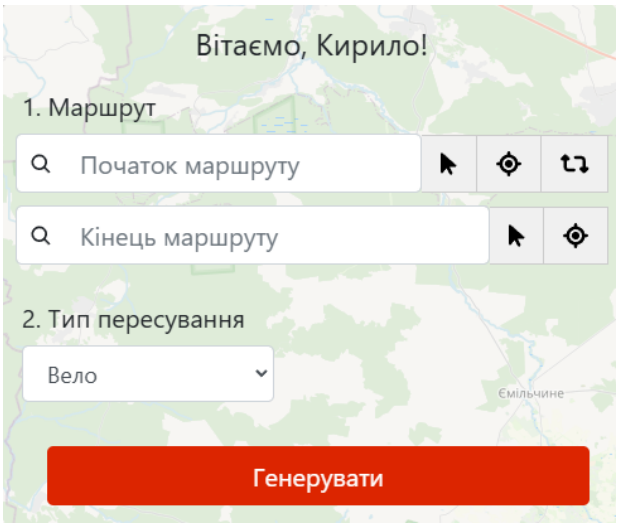


Рисунок 4.6 – Заповнення маршруту з точки А в точку В

Результатом заповнення форми є створення випадкового маршруту, що відповідає введеним фільтрам. При повторній генерації маршрут змінюється, тому користувач завжди може обрати щось до вподоби. Після генерації користувач побачить точково створений маршрут, кожну точку точку якого можна переміщати та відповідно редагувати його власноруч. Згенерований маршрут зображено на рисунку 4.7.

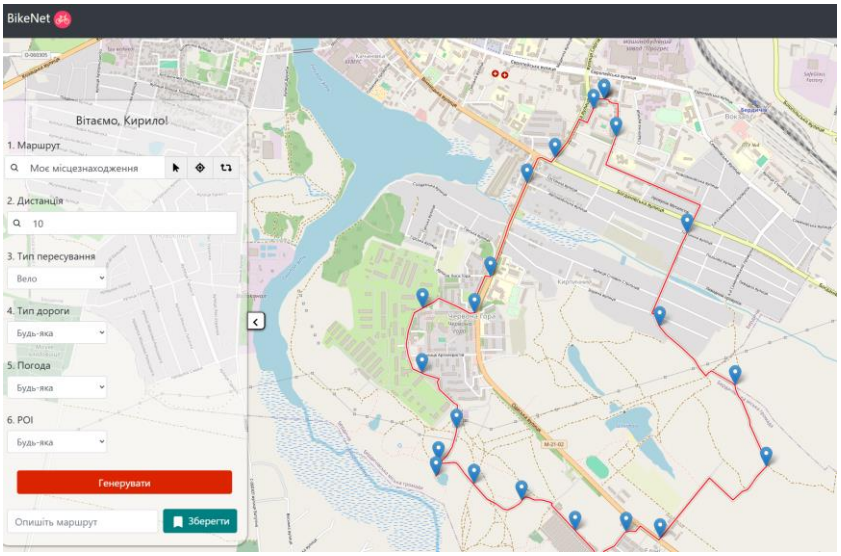


Рисунок 4.7 – Згенерований маршрут

Як тільки маршрут згенеровано та зображено на мапі, користувач може зберегти його до власної вибірки за допомогою форми, що зображена на рисунку 4.8. Для збереження користувач повинен описати маршрут або ж лише його назвати. Після натискання з’являється повідомлення, що маршрут збережено і користувач надалі зможе його бачити в списку своїх маршрутів.

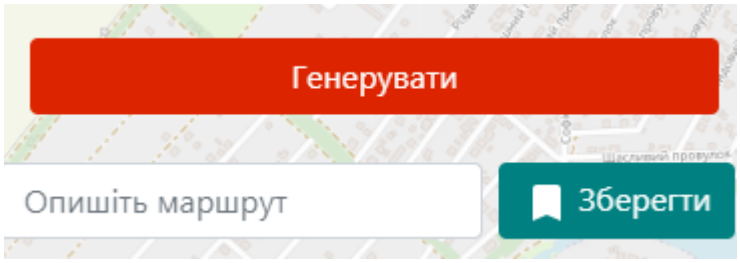


Рисунок 4.8 – Форма збереження маршруту

Разом з згенерованим маршрут користувач може побачити його деталі в правій частині мапи, а саме маршрутизацію, що включає покрокові інструкції проходження маршруту, що показані на рисунку 4.9. При наведені відповідна точка зображується на мапі і користувач в режимі реального часу може переглядати усі деталі.

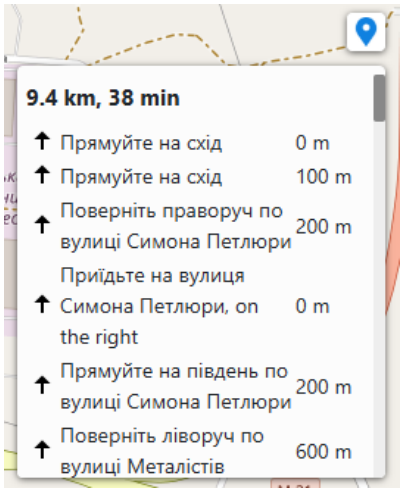


Рисунок 4.9 – Згенерований маршрут

Користувач без облікового запису не має можливості зберігати маршрути до власного списку. Для цього користувач повинен зареєструватись, заповнивши форму, де повинен ввести електронну пошту та пароль та назву акаунту як зображено на рисунку 4.10. Під час реєстрації, існують певні обмеження, а саме

коректний формат пошти та мінімальна кількість для паролю, що становить 5 символів.

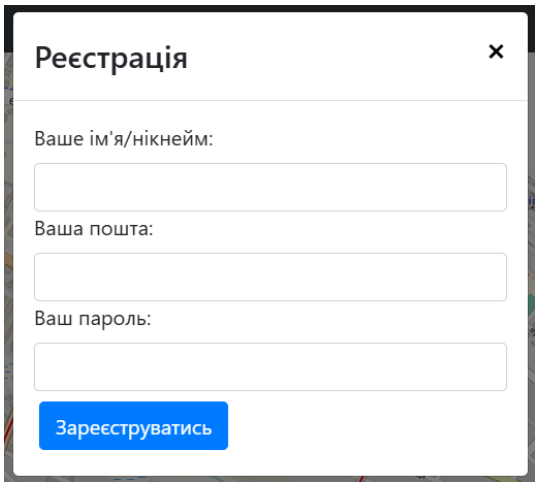


Рисунок 4.10 – Форма реєстрації

Форма авторизації є схожою до попередньої форми окрім того, що вона не потребує імені користувача, а лише пошту та пароль. Якщо користувач вводить правильні дані, система надає доступ до системи. Заповнення форми авторизації зображено на рисунку 4.11.

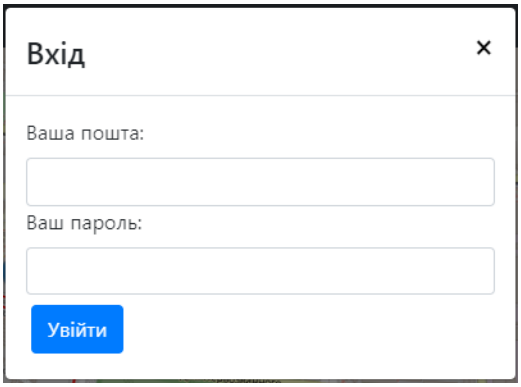


Рисунок 4.11 – Форма авторизації

Після входу в систему з’явиться можливість переглядати збережені маршрути. Якщо користувач не має маршрутів, то список буде пустий. Якщо ж має, то він може переглянути сам маршрут на карті і короткі відомості про нього, такі як Початкова точка, довжина, час створення, що зображено на рисунку 4.12.

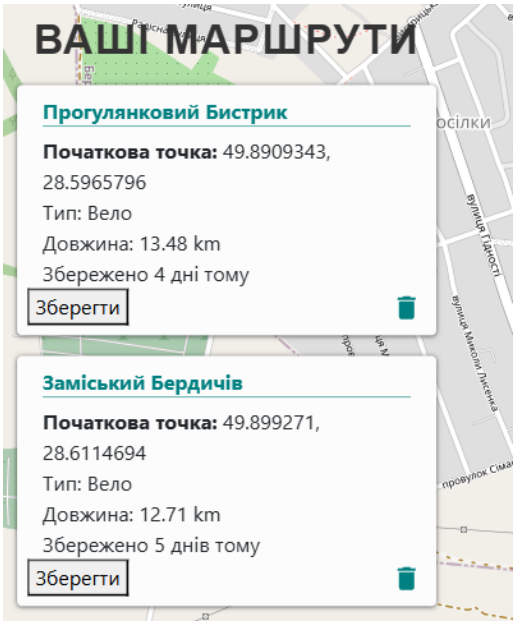


Рисунок 4.12 – Збережені маршрути користувача

Блок збереженого маршруту передбачає можливість переглядати його, видаляти та вивантажувати в .gpx форматі і далі застосовувати в інших сервісах та навігаційних додатках. При натисканні кнопки видалення, маршрут буде безповоротно видалено з бази даних і за кілька секунд зникне з списку збережених маршрутів. Якщо користувач не має маршрутів, то даний блок буде пустим, з повідомленням «Тут поки нічого». Кількість збережених маршрутів для певного користувача є необмеженою і чим більша їх кількість, тим краще рекомендація.

Блок рекомендації знаходиться над блоком збереженого маршруту, але використовує його сутності як відображення рекомендацій, що зображено на

рисунку 4.13. Головною різницею є те, що користувач не може їх видалити, адже вони оновлюються кожного разу при переході в блок збережених маршрутів.

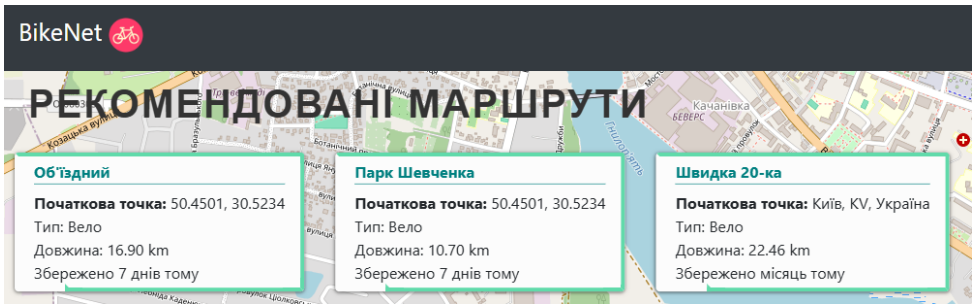


Рисунок 4.13 – Рекомендовані маршрути користувача

Блок задання адреси містить блок інтерактивних кнопок (рисунок 4.14), що містять три основні кнопки, а саме визначення адреси на мапі, адреса за місце знаходженням та зміну типу побудови маршруту.

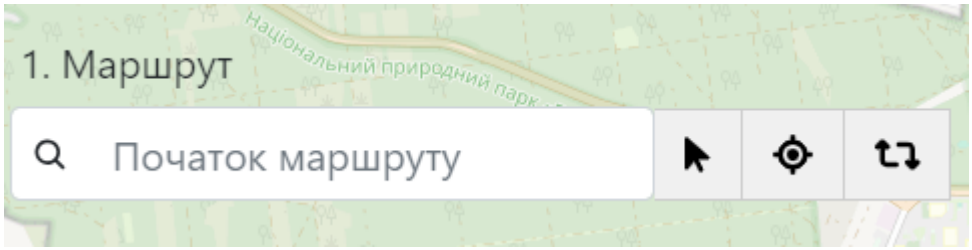


Рисунок 4.14 – Блок інтерактивних кнопок під час пошуку

Блок визначення адреси на мапі дозволяє користувачу бажану адресу початку маршруту одразу ж на мапі. При натисканні активується режим роботи з мапою і кожне наступне натискання на мапі буде визначати точку старту маршруту. У вигляді тексту в полі будуть вказуватись координати обраного місця. Таким чином, користувачу не потрібно задавати точну адресу текстово, а лише за допомогою кліку на мапі.

Блок вибору адреси за місцезнаходженням доволі дозволяє користувачу визначити точку старту маршруту як власне місцезнаходження пристрою, що використовується користувачем. Місцезнаходження не є точний, а обмежується певною областю в межах якої знаходиться користувач.

Блок зміни типу побудови маршруту виконує зміну типу маршруту відповідно. Якщо користувач хоче згенерувати статичний маршрут, а саме вказати як точку старту так і точку кінця, то при натисканні останньої кнопки змінюється форма створення маршруту на форму, що зображена на рисунку 4.15.

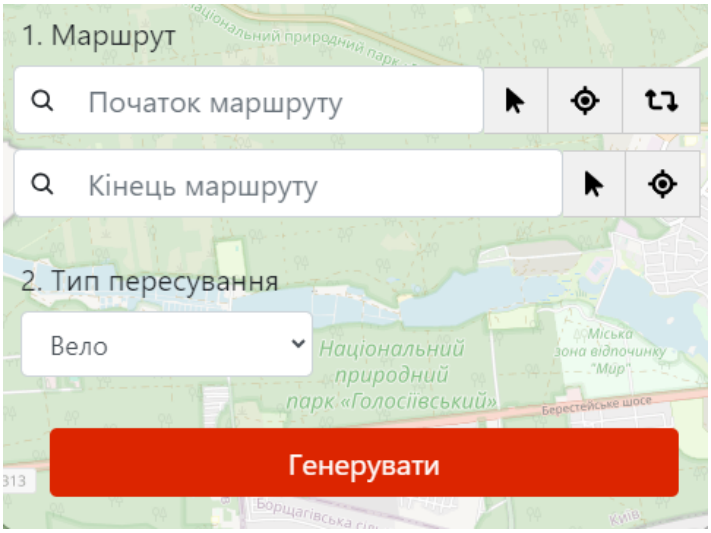


Рисунок 4.15 – Блок побудови статичного маршруту

Для створення статичного маршруту все що потрібно зазначити користувачу це початок маршруту, кінець та тип пересування. Визначити точки початку та кінця користувач може в текстовому форматі, задавши точку на мапі або ж визначивши точку на основі власного місцезнаходження. Як результат, користувач отримає фіксований статичний маршрут з точки А до точки В. Такий маршрут буде завжди однаковий без зміни параметрів.

4.2 Логічна структура веб-системи

В межах розробленої системи побудована логічна структура веб-системи, що показує як організовані функціональні блоки системи і визначає, які дії користувач повинен подолати, щоб досягти поставленої мети. На рисунку 4.16 зображена логічна структура веб-системи. Діаграма є навігаційною системою, що показує як певні блоки пов’язані і іншими системними блоками.

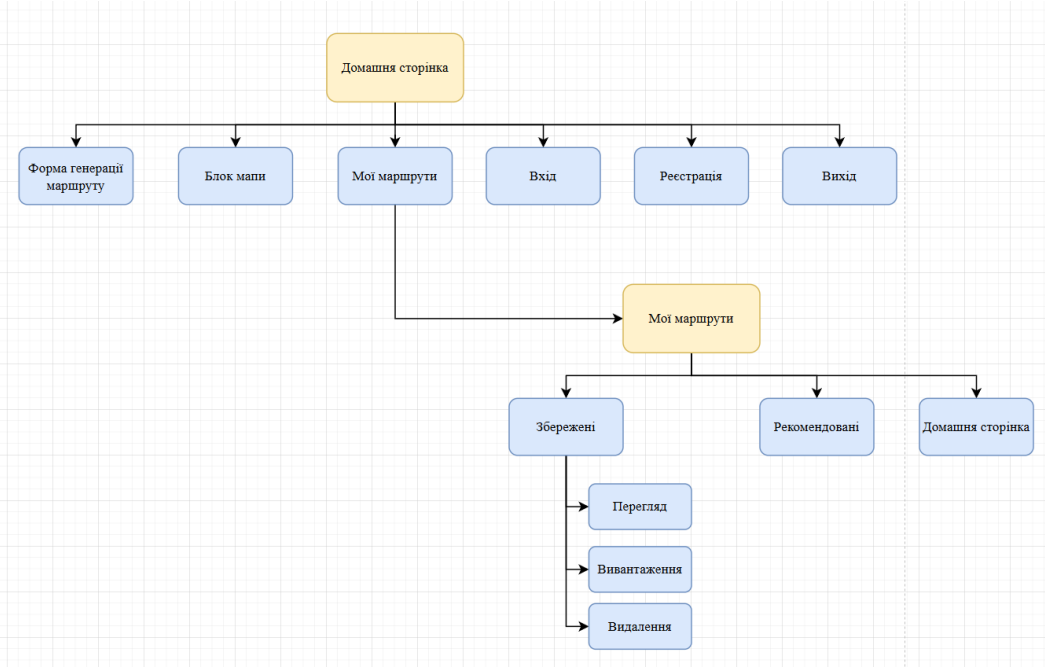


Рисунок 4.16 – Логічна структура веб-системи

Логічна структура веб-системи має два основі блоки, а саме домашня сторінка, що містить найважливіший елемент системи – інтерактивна мапа та блок маршрутів користувача. Кожен блок має посилання на інший блок системи, що дозволяє вільно переміщатись між ними. Логічна структура показує взаємозв’язок між модулями системи.

Домашня сторінка міститься посилання на усі функціональні блок системи, а саме форма генерації маршруту, блок мапи та збережених маршрутів, блок авторизації та реєстрації користувача. Це є початковим блоком та перше, що бачить користувач при взаємодії з системою.

Блок маршрутів користувача містить інформацію про збережені, рекомендовані маршрути та має посилання на домашню сторінку. Збережені маршрути можуть переглядатись, видалятись або ж вивантажуватись.

Висновки до розділу 4

В розділі описано взаємодію користувача з системою, проілюстровано роботу з кожним функціональним блоком, наведено їх опис. Розглянуто основні можливі процеси використання системи, а саме, форма створення маршруту та кожен блок, що відповідає за створення. Також показано процес входу в систему, що включає автентифікацію, перевірку прав доступу та обмеження, які встановлюються на введені дані для забезпечення їхньої коректності.

Окрім того, у розділі представлено логічну структуру веб-системи, де описано компоненти, які взаємодіють між собою. Показано, як користувач може переміщатись між блоками виконуючи певні функції. Логічна структура також надає користувачеві навігаційний посібник, допомагаючи зрозуміти, які функціональні можливості доступні й як вони інтегровані у загальну роботу системи.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Правильно сформований стратегічний підхід прямим чином впливає на успіх будь-якого стартапу, адже він залежить не лише від технічної реалізації, а й від стратегії виходу на ринок.

У розділі описано стартап-проєкт, а саме ідею, переваги, технологічний аудит, потенційні клієнти та іншу важливу інформацію, що визначає конкурентні переваги та перспективи реалізації проєкту. Особлива увага приділяється аналізу ринкових можливостей, оцінці потреб цільової аудиторії.

5.1. Опис ідеї стартап–проєкту

В межах підпункту потрібно описати саму ідею стартап–проєкту, напрямки застосування та вигоди для користувача. Опис ідеї стартап- проєкту наведено на таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап–проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Побудова веломаршрутів на основі моделей штучного інтелекту	Розумне планування маршрутів	Дозволяє економити час завдяки швидшому плануванню маршрутів.
	Персоналізація маршрутів	Індивідуальний підхід до побудови маршруту відповідно до вимог користувача

Визначення сильних, слабких та нейтральних характеристик власної системи та системи конкурентів є доволі важливим етапом планування проєкту. Саме

реалістичне планування дозволяє адекватно оцінити можливості стартапу. Також це допоможе ідентифікувати потенційні ризики, що можуть перешкодити розвитку стартапу, тому завчасне виявлення знижує ризики провалу проєкту [49]. І на остаток, визначення сильних характеристик допомагає зрозуміти переваги проєкту, які роблять його конкурентоспроможним В таблиці 5.2 наведено сильні та слабкі сторони розробленої системи та систем конкурентів.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик

No п/п		(потенційні) товари/концепції конкурентів			
		Мій проект	Strava	Komoot	Google Maps
1	W слабка сторона	Мала база даних активних користувачів	Сервіс має велику базу користувачів з усього світу	Сервіс має значну базу користувачів, переважно туристів та велосипедистів	Величезна база користувачів, які щодня взаємодіють з сервісом
2	S сильна сторона	Висока швидкодія	Середня швидкодія системи	Швидкість роботи середня, можливі підвисання	Висока швидкодія з можливими підвисаннями
3		Прийнятна вартість	Вартість вище ніж в конкурентів	Ціна вища ніж середня	Сервіс є безкоштовним
4	N нейтральна сторона	Кросплатформність	Кросплатформність	Кросплатформність	Кросплатформність

Сильними сторонами розробленої системи є її швидкодія, адже розмір системи значно менший від конкурентів, прийнятна вартість для користувачів, однак недоліком є менша база активних користувачів ніж в конкурентів, які на ринку вже давно і мають певну довіру.

5.2. Технологічний аудит ідеї стартап–проєкту

Аудит технологій ідеї стартап-проєкту є важливим з декількох причин. По першу, він оцінює чи зможе проєкт бути життєздатним в довгій перспективі та виявляє можливі ризики та залежності з питань технологій в проєкті. По друге, аудит також дозволяє зрозуміти фінансовий бік підтримки проєкту і враховуючи це формувати можливу ціну підписки. Аудит технологій наведено у таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту

No п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Інтерфейс користувача	React, HTML та CSS	Наявна	Доступні безкоштовно
2	Обробка даних	JavaScript	Наявна	Доступна безкоштовно
3	База даних	Firebase	Наявна	Доступна платно
4	Сервіси даних API	OpenRouteService, Overpass, LocationIQ	Наявна	Доступні платно
5	Кросплатформне рішення	React, HTML та CSS	Наявна	Доступні безкоштовно
Висновок: проект можна розробити з місячними чи річними витратами та сервіси та базу даних.				

Основні технології розробки та підтримки системи є доступними та безкоштовними, однак система потребує даних з сервісів API, які надаються платно, якщо відбувається перевищення квоти безкоштовної версії, що передбачувано. Також очікуються витрати на підтримку бази даних.

5.3. Аналіз ринкових можливостей запуску стартап–проекту

За допомогою аналізу ринкових можливостей оцінюються перспективи запуску стартап-проєкту побудови веломаршрутів на основі моделей штучного інтелекту. Аналіз включає оцінку кількості конкурентів, загальний обсяг продажів, динаміка ринку та інші важливі метрики. Аналіз ринку стартап-проєкту показано на таблиці 5.4.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап–проекту

No п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	4800000 грн. річних продажів
3	Динаміка ринку (якісна оцінка)	Стрімко зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	20-40%

Ринок, де планується запуск стартап-проєкту стрімко зростає та сприятливі умови для нових гравців. Навіть незважаючи на трьох великих конкурентів, відсутність будь-яких обмежень з боку держави чи подібних питань з сертифікації, дозволяє легко зайти в цю галузь.

Загальний обсяг річних продажів 4.8 мільйонів гривень в рік, що обраховувалось на основі очікуваної кількості користувачів. Середня рентабельність на рівні 20-40% робить можливим стабільний прибуток.

Наступним важливим етапом є визначення потенційних груп клієнтів та їх характеристики, що зображена на таблиці 5.5

Таблиця 5.5 – Характеристика потенційних клієнтів стартап–проекту

No п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Побудова випадкових веломаршрутів з врахуванням індивідуальних потреб користувачів	Велотуристи, спротсмени	Значні відмінності між поведінкою різних груп цільової аудиторії відсутні. Купівля послуг сервісу відбувається за допомогою підписки терміном від місяця до року.	Швидкодія системи Прийнятна ціна Тестовий період для користувачів Персоналізація маршрутів Актуальність даних

Необхідним етапом є визначення факторів загроз та можлива реакція на ці загрози. Даний етап дозволяє зрозуміти і передбачити можливі ризики в системі. Фактори загроз розробленої системи зображено на таблиці 5.6.

Таблиця 5.6 – Фактори загроз

No п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Можливі технічні збої	Технічні збої, затримки в обробці запитів та низька швидкодія системи під час масштабування системи	Оптимізація коду, постійний моніторинг витрати ресурсів та використання хмарних сервісів
2	Конкуренція з великими платформами	Системи мають багаторічний досвід, базу користувачів, а головне довіру. Цього немає в достатній мірі на початку розробки стартап-проєкту	Фокус на унікальні функції, цінова доступність

Додатково, фактори можливостей допомагають оцінити переваги системи та майбутні можливості. Ці фактори детально описують можливості, які можуть бути реалізовані в стартап-проєктів. Фактори можливостей зображено на таблиці 5.7.

Таблиця 5.7 – Фактори можливостей

No п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Необхідність персоналізованих маршрутів для різних видів спорту	Такі види спорту як біг та звичайна хода та інші теж потребують персоналізованих випадкових маршрутів	Охоплення більшої кількості типів транспорту в системі
2	Збільшення кількості змінних побудови маршрутів	У випадку ретельного планування користувачу потрібна більша кількість налаштувань для побудови маршрутів	Розширення функціональності системи
3	Підтримка міського планування	Не лише сервіси, а й міста зацікавлені в створені безпечної інфраструктури. Сервіс може аналізувати популярні маршрути та виявляти проблемні	Впровадження API сервісу, що надаватиме міським службам доступ до аналітики та для інтеграції з їх платформами

Наступним етапом є ступеневий аналіз конкуренції на ринку, що враховує такі аспекти як тип конкуренції та підтипи, а саме конкуренція за рівнем боротьби, за галузевою ознакою, за видами товарів, характером переваг та інші. Такий аналіз визначає позицію проєкту відносно його прямих та непрямих конкурентів, допомагає оцінити сильні та слабкі сторони. Аналіз конкурентного середовища зображено на таблиці 5.8.

Таблиця 5.8. – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції – монополія/олігополія/ монополістична/чиста	Монополія Ринок має кількох сильних конкурентів, що пропонують схожі послуги	Створення унікальної особливості, що не пропонують конкуренти
2. За рівнем конкурентної боротьби – локальний/національний/міжнародний	Міжнародна конкуренція	Усі сервіси працюють на глобальному рівнів. Перспективною є впровадження інтеграції з різними місцевими сервісами та використання публічних даних щодо особливостей місцевості
3. За галузевою ознакою – міжгалузева/ внутрішньогалузева	внутрішньогалузева	Орієнтація на задоволення конкретних потреб користувача, що не охоплено конкурентами
4. Конкуренція за видами товарів: – товарно–родова – товарно–видова – між бажаннями	товарно–видова	Змагання з іншими платформами, пропонуючи кращі та унікальні рішення
5. За характером конкурентних переваг – цінова / нецінова	цінова	Надання більш доступних цін та преміум функцій
6. За інтенсивністю – марочна/не марочна	не марочна	Зосередження на функціональних характеристиках продукту, а не на бренді.

В таблиця 5.9 проаналізовано конкуренцію в газулі з М. Портером.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари–замінники
	Strava, Komoot, Google maps	BikeMap	-	Споживамають альтернативи	Замінники існують на ринку
Висновки:	Сильна конкурентна боротьба	Потенційні клієнти не мають тих функцій, які присутні в розробленому у продукті	Системи підлаштовуєтьс я під постачальників даних	Клієнти диктують вимоги згідно з умовами експлуатації	Середній рівень загрози, адже замінники надають базові функції

Таблиця 5.10 показує фактори конкурентоспроможності системи та їх обґрунтування.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

No п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Динамічний підхід до побудови маршрут	Аналоги обмежені та мають статичний підхід до побудови маршрутів, де користувач повинен ввести точку старту та кінця
2	Персоналізація маршрутів	Врахування індивідуальних потреб користувачів, таких як тип дороги, погодні умови та наявність цікавих місць поруч

Цифровий аналіз сильних та слабких сторін системи надає можливість об’єктивно визначити успіх і місця для вдосконалення. Аналіз наведено в таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

No п/п	Фактор конкурентоспроможності	Бали 1–20	Рейтинг товарів–конкурентів у порівнянні з розробленою системою (даним продуктом)						
			–3	–2	–1	0	1	2	3
1	Користувацький інтерфейс	15						+	
2	База користувачів	5							+
3	Швидкість систем	15			+				
4	Залежність від даних	20				+			
5	Ціна	10		+					

Фінальним етапом ринкового аналізу можливостей є складання SWOT–аналізу. Він виділяє сильні, слабкі сторони та загрози, можливості. SWOT аналіз зображено на таблиці 5.12.

Таблиця 5.12 – SWOT–аналіз стартап–проекту

Сильні сторони: Персоналізація маршрутів Швидкодія Створення випадкових маршрутів Доступні ціни	**Слабкі сторони:** Мала база користувачів Висока конкуренція Залежність від сторонніх API
Можливості: Додаткові функції Інтеграція з місцевими сервісами Охоплення нових видів транспорту	**Загрози:** Зміна в політиці API постачальників Технічні проблеми при масштабуванні Регулювання та правові обмеження

Визначення альтернатив ринкового впровадження дозволяє будувати так звані додаткові плани на випадок адаптації до ринкових умов. Альтернативи зменшують ризики так як є вибір одного або кількох можливих сценарії розробки, що наведені в таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап–проекту

No п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Впровадження більшої кількості видів транспорту та додавання функцій до них	70 %	1 місяць
2	Зміна архітектури подачі даних, додавання об'ємних об'єктів за допомогою street.gl	40 %	5 місяців
3	Орієнтація системи на gps додаток	30 %	1 рік

5.4. Розроблення ринкової стратегії стартап–проєкту

Наступним критичним етапом формування ринкової стратегії стартап-проєкту. В аналіз входить визначення цільових потенційних користувачів, стратегії розвитку, конкурентної поведінки, позиціонування та багато інших важливих стратегій.

Визначення сегментації ринку дозволяє сфокусувати власні ресурси саме на тих користувачах, які найбільш зацікавлені у використанні системи та як результат є найбільш цікавими системі [50]. Вибір цільової аудиторії показано на таблиці 5.14

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

No п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Рекреаційні велосипедисти	Готові	Високий	Середня	Просто
2	Велотуристи	Готові	Високий	Середня	Просто
3	Велоспортсмени	Потрібен час	Низький	Середня	Складно
Які цільові групи обрано: Рекреаційні велосипедисти та велотуристи					

Найважливішою цільовою аудиторією для системи є рекреаційні велосипедисти та велотуристи. Рекреаційні велосипедисти – це люди, які використовують велосипед для активного відпочинку, прогулянок або подорожей у вільний час. Їх потреби дуже прості, а саме легкі та мальовничі маршрути. Велотуристи використовують велосипед для подорожей на довгу дистанцію. І перший і другий тип доволі легко задовільнити. Щодо велоспортсменів, простота входу в сегмент є складною, адже вони потребують не лише маршрутів, а й аналітичну інформацію про саму поїздку. Вони використовуються велосипед для тренувань та підготовки до змагань. Їм важливий аналіз швидкості, інформація про проходження висоти та багато інших метрик, які відсутні в розробленій системі. Саме тому було обрано перші два типи велосипедистів як цільова аудиторія системи.

Наступним критичним етапом є формування ринкової стратегії стартап-проєкту. Для роботи в обраних вище сегментах потрібно обрати сформувати стратегію, яку

буде дотримуватись стартап-проєкт. Базова стратегія розвитку системи показана на таблиці 5.15.

Таблиця 5.15 – Визначення базової стратегії розвитку

No п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Рекреаційні велосипедисти	Концентрація на вузькому сегменті користувачів	Рекреаційні велосипедисти потребуються легких та гарних маршрутів	Постійно проводити аудит системи, прислухатись до користувачів
2	Велотуристи	Концентрація на вузькому сегменті користувачів	Велотуристи потребують можливість створювати багатоденні маршрути та враховувати точки відпочинку.	Постійно проводити аудит системи, прислухатись до користувачів

Стратегія заохочення ринку спрямована на концентрацію на вузький сегмент користувачів. Так як цільовою аудиторією стратегії є рекреаційні велосипедисти та велотуристи, система забезпечує для них спеціалізовані маршрути за допомогою штучного інтелекту. В якості маркетингової стратегії активно використовуються соціальні мережі, а саме Facebook та Instagram та SEO оптимізація.

Наступним кроком є вибір стратегії конкурентної поведінки, що зображено на таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

No п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні, наявні аналоги	Компанія буде забирати існуючих споживачів	Компанія аналізуватиме та копіюватиме основні функції конкурентів та буде намагатись їх покращити.	Стратегія, коли користувачу будуть пропонуватись такі ж та навіть краще за вигіднішими умовами

В таблиці 5.17 зображено стратегію позиціонування стартап-проєкту.

Таблиця 5.17 – Визначення стратегії позиціонування

No п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто–спроможні позиції власного стартап–проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Стабільність роботи Коректність роботи Прийнятна ціна Широкий функціонал системи	Залучення більше нових клієнтів	Користувачам потрібно швидко та якісно отримувати готові маршрути з врахуванням їх вподобань	Зручність, швидкість, якість, різноманіття

5.5. Розроблення маркетингової програми стартап–проєкту

Для розробки маркетингової стратегії потрібно зробити висновок та визначити результати попередніх досліджень. В таблиці 5.18 наведено ключові переваги концепції потенційного товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

No п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Потреба в швидких персоналізованих маршрутах	Лише кілька ключових параметрів і користувач отримає випадковий маршрут, що відповідає його вподобанням.	Програмна система BikeNet має динамічну архітектуру побудови веломаршуртів, що вирізняє його серед конкурентів, які мають лише статичних підхід, що потребує більше часу та прибирає ефект зацікавленості в певних моментах.
2	Невисока ціна	BikeNet пропонує доступні ціни та наявність безкоштовної версії	Програмна система BikeNet в рази доступніша ніж сервіси-конкуренти, завдяки гнучкішій ціновій політиці.
3	Врахування специфічних потреб	BikeNet враховує погодні умови, тип дорог та наявність точок інтересу при побудові	Програмна система має унікальні функції при побудові маршрутів, яких не мають системи аналоги.

Наступним етапом є розробка трирівневої маркетингової моделі товару, що зображена на таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Побудова веломаршрутів на основі моделей штучного інтелекту</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Швидкодія</td></tr>
<tr><td>Блок генерації маршрутів</td><td>1000 мс.</td></tr>
<tr><td>Блок збережених маршрутів</td><td>500 мс.</td></tr>
<tr><td>Блок рекомендації маршрутів</td><td>1500 мс.</td></tr>
<tr><td colspan="2">Пакування: продаж та дистрибуція товару відбувається через основні цифрові платформи у зручному форматі</td></tr>
<tr><td></td><td colspan="2">Марка: маркування основного логотипу бренду в системі та назва</td></tr>
<tr><td>III. Товар із підкріпленням</td><td colspan="2">Потенційний користувач може ознайомитись з системою абсолютно безкоштовно, адже додаток розміщується на цифрових платформах безкоштовно. Для доступу до додаткових функцій користувач оформлює пробний період і по закінченню може оформити підписку.</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання:

товар захищений сервісами Google Play та App Store лише всередині яких підтримується підписка. Прив’язка відбувається до акаунту і оплату контролюють/обмежують цифрові платформи.</td></tr>
</table>

На кожному рівні моделі відображаються характеристики проєкту. Важливою для розробленої системи є її системи. Описано очікуваний час, що витрачатиметься

на роботу кожного блоку. Також зазначено особливості пакування, маркування та опис захисту ліцензії потенційного товару.

Наступним кроком є визначення цінових меж, що зображено на таблиці 5.20. Цими межами керується стартап-проєкт при встановленні ціни на підписку.

Встановлення регіональних цін не відповідає стратегії стартап проєкту, адже проєкт орієнтований на глобальний ринок і головною задачею є створення універсального продукту, що доступний всім. Серед причин встановлення єдиної ціни є те, що проєкт однаково добре працює для будь-якої частини світу та для усіх клієнтів, незалежно від локації. Також процес єдиної ціни зручніше адмініструвати, адже регіональна ціна потребує постійних як фінансових, так і технічних ресурсів, що може ускладнити управління системою, тим паче на ранніх стадіях розвитку стартап-проєкту.

Таблиця 5.20 – Визначення меж встановлення ціни

No п/п	Рівень цін на товари–замінники	Рівень цін на товари–аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	0 грн	800…1400 грн/рік	> 120000 грн/рік	600…800 грн/рік

Далі в таблиці 5.21 показано формування системи збуту для стартап-проєкту. Стратегія формування збуту складається з визначення функцій збуту, глибини каналу та оптимальної системи збуту. Враховуючи специфіку проєкту, акцент зроблено на цифрових каналах збуту, що відповідають потребам користувачів. Значна увагу приділена оптимальним система збуту, покладаючись на автоматизацію платежів завдяки цифровим платформам. Це вирішує питання швидкого виходу на ринок та питання зростання кількості клієнтів під час масштабування.

Таблиця 5.21 – Формування системи збуту

No п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Система надаватиметься за підпискою, але перед цим клієнт матиме змогу використати систему під час тестового періоду	Просування товару, забезпечення легкого продажу для користувача, забезпечення стабільного доступу	Прямий канал збуту. Розповсюдження через офіційний веб-сайт та мобільний дотаток	Прямий збут одразу ж кінцевим користувача через офіційні цифрові платформи

Останньою важливою частиною маркетингової програми є створення концепції маркетингових комунікацій, що залежить від попередньо визначеної стратегії позиціонування та специфіки поведінки клієнтів. Розроблена концепція зображена на таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

No п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Купівля підписки всередині додатку або ж одразу на веб-сайті.	Веб–сайт, додаток	Різноманіття Швидкодія Персоналізація	Показ основних та унікальних функцій системи, демонстрація швидкодії	Інноваційно, зручно, доступно

Створення якісної та продуманої маркетингової програми стартап-проєкту є одним з найважливіших кроків на етапі створення продукту. В межах підрозділу для розробленого стартап-проєкту окреслено функціональні переваги концепції проєкту, описано три рівні моделі товару, визначено межі встановлення цін, формування системи збуту та концепція маркентингових комунацій. Щодо фінансової складової, проаналізовано та сформовано різні рівні цін, а саме ціни в конкурентів та ціни, що будуть встановлені в розробленій системи.

Також визначено плановані канали збуту стартап-проєкту, а саме концентрація як на веб-версії системи так і через додаток та цифрові платформи розповсюдження як от Google Play та App Store. Дані платформи контролюють процес розповсюдження та оплати товару з боку користувача, хоч і мають свої комісії.

Висновки до розділу 5

В даному розділі було всебічно проаналізовано основну ідею стартап-проєкту, що спрямований на створення системи побудови веломашрутів на основі моделей штучного інтелекту. Опрацьовано ідею стартап-проєкту, здійснено технологічний аудит ідеї, проаналізовано ринкові можливості запуску проєкту, розроблено ринкову та маркетингову стратегії.

Проаналізовано конкурентне середовище та визначено прямих і непрямих конкурентів. Серед них Strava, Google Maps та Komoot. Повністю визначено характер конкуренції на ринку для стартап-проєкту за рівнем боротьби, галузевою приналежністю та особливістю конкурентних переваг.

Таким чином, виконано повний цикл дій, які необхідні для реалізації стартапу.

ВИСНОВКИ

1. Проаналізовано існуючі програмні комплекси, що виконують подібну задачу, а саме Strava, Komoot та Google Maps. Описано їх архітектуру, переваги та недоліки. Визначено потреби, що не охоплені програмними комплексами конкурентів.
2. Проаналізовано методи машинного та глибинного навчання такі як: логістична регресія, метод k-найближчих сусідів, випадковий ліс, метод опорних векторів і глибокі та рекурентні нейронні мережі. Обґрунтовано використання методу логістичної регресії для побудови рекомендації маршруту для користувача.
3. Визначено та описано набір програмних засобів, що використовуються для створення системи побудови веломаршрутів на основі моделей штучного інтелекту.
4. Описано дані, які використовуються в розробленій системі та API сервіси, які застосовуються для побудови маршрутів.
5. Розроблено систему побудови персоналізованих веломаршрутів та рекомендаційну модель для користувачів на основі збережених маршрутів.
6. Розроблено стартап-проєкт, що описує основну ідею проєкту, аналіз технічної частини системи, конкурентний аналіз, формування стратегій поведінки та позиціонування проєкту. Таким чином, проведено комплексне дослідження життєспроможності розробленої системи.

Результати роботи оприлюднені на VII Міжнародній студентській конференції «Діджиталізація науки як виклик сьогодення», м. Полтава, 25 жовтня 2024 року.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. KiM Netherlands Institute for Transporation Policy Analysis. URL: https://english.kimnet.nl/publications/publications/2024/01/10/cycling-facts-2023 (дата звернення: 25.09.2024).
2. How to Use Strava: A Guide to Strava’s Popular Features. URL: https://bikexchange.com/strava-popular-features-guide/ (дата звернення: 05.10.2024);
3. What exactly is Komoot? URL: https://bikerumor.com/what-exactly-is-komoot-and-why-should-you-be-using-it-to-plan-your-rides/ (дата звернення: 05.10.2024).
4. The Algorithms Behind The Working Of Google Maps. URL: https://blog.codechef.com/2021/08/30/the-algorithms-behind-the-working-of-google-maps-dijkstras-and-a-star-algorithm/ (дата звернення: 05.10.2024).
5. Open Street Maps documentation. URL: https://www.openstreetmap.org/help/ (дата звернення: 13.10.2024).
6. Overpass API documentation. URL: https://osmlab.github.io/learnoverpass/en/docs/ (дата звернення: 13.10.2024).
7. Overpass Tutorial. URL: https://osm-queries.ldodds.com/tutorial/index.html (дата звернення: 14.10.2024).
8. Salton G., Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management. 1988. Т. 24, № 5. С. 513–523. URL: https://doi.org/10.1016/0306-4573(88)90021-0 (дата звернення: 18.10.2024).
9. Mukherjee S., Sonal R. A reconciliation between cosine similarity and Euclidean distance in individual decision-making problems. Indian Economic Review. 2024. URL: https://doi.org/10.1007/s41775-023-00206-8 (дата звернення: 18.10.2024).
10. Cosine Similarity. URL: https://www.geeksforgeeks.org/cosine-similarity/ (дата звернення: 18.10.2024).

11. What is Logistic Regression? URL: https://aws.amazon.com/what-is/logistic-regression/ (дата звернення: 19.10.2024).
12. Hilbe J. M. Practical Guide to Logistic Regression. Chapman and Hall/CRC, 2016. URL: https://doi.org/10.1201/b18678 (дата звернення: 20.10.2024).
13. Hosmer D. W., Lemeshow S., Jr H. D. W. Applied Logistic Regression. Wiley & Sons, Incorporated, John, 2013. 392 с.
14. Duda R. O. Pattern Classification, 2Nd Ed. wiley, 2007. 678 с.
15. K-Nearest Neighbor(KNN) Algorithm URL: https://www.geeksforgeeks.org/k-nearest-neighbours/ (дата звернення: 21.10.2024).
16. Mukherjee S., Sonal R. A reconciliation between cosine similarity and Euclidean distance in individual decision-making problems. Indian Economic Review. 2024. URL: https://doi.org/10.1007/s41775-023-00206-8 (дата звернення: 21.10.2024).
17. Resampling methods: A practical guide to data analysis. Boston : Birkhäuser, 1999. 269 с.
18. Random Forest: A Complete Guide for Machine Learning URL: https://builtin.com/data-science/random-forest-algorithm (дата звернення: 28.10.2024).
19. Cristianini N., Shawe-Taylor J. Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2013.
20. Wang L. Support Vector Machines: Theory and Applications. Springer Berlin / Heidelberg, 2010.
21. Burges C.J.C. A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, С. 121–167, 1998
22. Deep Learning and Parallel Computing Environment for Bioengineering Systems. Elsevier, 2019. URL: https://doi.org/10.1016/c2018-0-00906-8 (дата звернення: 01.11.2024).

23. Aggarwal C. C. Neural Networks and Deep Learning. Cham : Springer International Publishing, 2018. URL: https://doi.org/10.1007/978-3-319-94463-0 (дата звернення: 01.11.2024).
24. Hinton G. E. A fast learning algorithm for deep belief nets / G. E. Hinton, S. Osindero, Y. Teh //Neural Computation. – 2006. – Vo1. 18. – C. 1527-1554.
25. Yu W., Rubio J., Li X. Recurrent Neural Network Training with Stable Risk-Sensitive Kalman Filter Algorithm // Proc. of IJCNN'05. – Monreal, Canada, 2005. – July 31 – Aug. 4. – C. 700 – 705
26. Recurrent Neural Networks URL: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (дата звернення: 08.11.2024).
27. Salem F. M. Recurrent Neural Networks: From Simple to Gated Architectures. Springer International Publishing AG, 2021.
28. The Exploding and Vanishing Gradients Problem in Time Series URL: https://medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22 (дата звернення: 08.11.2024).
29. JavaScript Guide URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 13.11.2024).
30. Eloquent JavaScript: A Modern Introduction to Programming. 3-тє вид. San Francisco, California, United States of America : No Starch Press, 2018. 472 с.
31. HTML: HyperText Markup Language URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 16.11.2024).
32. CSS: Cascading Style Sheets URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 16.11.2024).
33. React Documentation URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 18.11.2024).
34. Learning React: Modern Patterns for Developing React Apps. O'Reilly Media, 2020. 310 с.

35. Node.js documentation URL: https://nodejs.org/docs/latest/api/ (дата звернення: 21.11.2024).
36. Firebase documentation URL: https://firebase.google.com/docs (дата звернення: 21.11.2024).
37. Yahiaoui H. Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. Packt Publishing, 2017. 288 с.
38. Visual Studio Code documentation URL: https://code.visualstudio.com/docs (дата звернення: 24.11.2024).
39. Cypress documentation URL: https://docs.cypress.io/app/get-started/why-cypress (дата звернення: 24.11.2024).
40. GitHub documentation URL: https://docs.github.com/en (дата звернення: 25.11.2024).
41. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 25.11.2024).
42. What is Class Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/ (дата звернення: 25.11.2024).
43. Openrouteservice URL: https://giscience.github.io/openrouteservice/getting-started (дата звернення: 26.11.2024).
44. LocationIQ URL: https://docs.locationiq.com/docs/introduction (дата звернення: 26.11.2024).
45. Тополюк К. М., Сидоренко Ю. В. Побудова веломаршрутів на основі моделей штучного інтелекту. Діджиталізація науки як виклик сьогодення : Матеріали VII Міжнар. студ. конф., м. Полтава, 25 жовт. 2024 р. С. 601–602.
46. Haversine formula to find distance between two points on a sphere URL: https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/ (дата звернення: 26.11.2024).

47. Що таке NGINX і навіщо він потрібний URL: https://vps.ua/wiki/ukr/nginx/ (дата звернення: 26.11.2024).
48. Pollard B. HTTP/2 in Action. Manning Publications Co. LLC, 2019.
49. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.
50. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, 2011. 336 c.

Додаток А

Побудова веломаршрутів на основі моделей штучного інтелекту

Публікації за темою роботи

УКР.НТУУ“КПІ ім. Ігоря Сікорського” _ІАТЕ_ЦТЕ_ТР-32мп

Аркушів 4

МАТЕРІАЛИ VII МІЖНАРОДНОЇ СТУДЕНТСЬКОЇ НАУКОВОЇ

КОНФЕРЕНЦІЇ

ДІДЖИТАЛІЗАЦІЯ НАУКИ ЯК ВИКЛИК СЬОГОДЕННЯ

М. ПОЛТАВА, УКРАЇНА

25 ЖОВТНЯ 2024 РІК

Тополюк Кирило Михайлович, здобувач вищої освіти
кафедри цифрових технологій в енергетиці
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна

Науковий керівник: Сидоренко Юлія Всеволодівна, канд. техн. наук,
доцент, доцент кафедри цифрових технологій в енергетиці
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна

ПОБУДОВА ВЕЛОМАРШРУТІВ НА ОСНОВІ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

Станом на сьогодні велосипедний транспорт набуває дедалі більшої популярності завдяки своїй екологічності та ефективності. В умовах швидкої індустріалізації велотранспорт є не лише засобом розваги, а й повноцінним транспортним засобом в повсякденному житті. Велосипед все більше асоціюється не просто з екологічно чистим транспортом, але й з здоровим і активним способом життя, свободою пересування та зручністю. Велотранспорт є одним із найпопулярніших засобів пересування в багатьох країнах світу, особливо в Нідерландах, де станом на 2023 рік понад 28% усіх поїздок [1] здійснюються на велосипедах. Країна постійно займається розвитком спеціалізованої інфраструктури.

Однак для того, щоб забезпечити зручність і безпеку велоподорожей, потрібні ефективні системи побудови маршрутів, які враховуються не лише відстань, але й такі фактори, як наявність велодоріжок, тип покриття дороги, погоду і також небезпечні ділянки, де можуть виникати аварійні ситуації. Тому актуальною задачею є розробка системи побудови велосипедних маршрутів на основі моделей штучного інтелекту, яка враховує різноманітні фактори. Проте, розробка оптимальних маршрутів для велосипедистів є складною задачею через потребу врахування вищезазначених численних факторів: безпека, зручність, трафік, рельєф, погода тощо.

Сучасні технології дають змогу розробляти подібні інструменти для планування веломаршрутів, застосовуючи такі методи як алгоритми машинного навчання [2], нейронні мережі, алгоритми на основі графів та інші.

Розглядаючи перший інструмент, а саме алгоритми машинного навчання – це методи, що дозволяють системам автоматично навчатись з даних і постійно покращувати свої рекомендації. Такі методи використовують статистичні підходи для аналізу даних і виявлення закономірностей, патернів. У контексті побудови веломаршрутів ці алгоритми аналізуватимуть дані щодо дорожньої інфраструктури.

Таким чином, у даній роботі пропонується розроблення системи, що будуватиме веломаршрути на базі вхідних даних від користувача. Враховуючи вхідні фактори, але першочергово, це відстань та час, буде проводитись аналіз можливих маршрутів для подорожей. В першу чергу буде вирішуватись проблема планування маршруту, що зазвичай забирає багато часу. Користувач визначатиме вхідні фактори і обиратиме на основі згенерованих маршрутів. Також важливо

передбачити актуальність і своєчасне оновлення геоданих, що є надзвичайно важливим при плануванні.

Для реалізації системи пропонується база веб-технологій HTML, CSS, JavaScript і також мови програмування Python. React у формі JavaScript застосовується для побудови інтерфейсу користувача. Мова Python забезпечить розробку та запуск моделі штучного інтелекту, особливо для автоматизації маршрутів. API сервіси Mapbox та Strava надають необхідні геодані та дані популярних веломаршрутів. Використання API забезпечує модульність, розділяючи систему на окремі компоненти, що спрощує розробку [3]. Перевагами обраного технічного стеку є гнучкість та швидкість розробки і найголовніше висока продуктивність. Останнє дозволяє ефективніше використовувати ресурсів та легше масштабуватись. Виходячи з вищезазначеного попередньо було розроблено архітектуру системи, що зображено на рис. 1.

Рис. 1. **Архітектура системи**

Архітектура складається з таких ключових елементів: Інтерфейс користувача (User Interface) - найперший елемент через який користувач взаємодіє з системою; Серверна частина (BackEnd Server) - елемент, що обробляє запити від користувачів і слугує зв'язуючим між інтерфейсом і AI моделями та джерелами даних; Джерела даних (Data Sources) - елемент, що охоплює такі зовнішні ресурси як Strava API, Mapbox API, OSM; Модуль обробки даних (Data Processing) - відповідає за очищення та підготовку даних для аналізу AI моделі, яка в свою чергу виконує аналіз вхідних даних та генерує маршрути; Модуль геомапування (Geo Mapping) - інтегрує результати з AI моделі та візуалізує маршрути на карті. Як результат, користувач отримує набір згенерованих маршрутів, оптимізованих з урахованням його вподобань.

Підсумовуючи, розроблювана система надасть унікальні інструменти для побудови веломаршрутів, що дозволить значною мірою зекономити час при плануванні подорожі та забезпечити зручність та безпеку. Система має потенціал до масштабування, серед яких врахування місцевих умов і вимог, специфіка інфраструктури і популярні маршрути поруч. В перспективі систему можна адаптувати під різні види транспорту.

Список використаних джерел:

1. KiM Netherlands Institute for Transport Policy Analysis. URL: https://english.kimnet.nl/publications/publications/2024/01/10/cycling-facts-2023 (дата звернення: 15.10.2024).
2. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition. Packt Publishing, 2019. 770 p.
3. Geewax J. J. API Design Patterns. Manning Publications Co. LLC, 2021. 480 p.