**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ТАРАСА ШЕВЧЕНКА**
**Факультет інформаційних технологій**
**Кафедра інтелектуальних технологій**

# ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

## Мобільний рекомендаційний туристичний застосунок

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 42

Бузюрова А.О.
(прізвище та ініціали)

Керівник Федусенко О.В.
(прізвище та ініціали)
Кандидат технічних наук, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

***КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА***

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

**ЗАВДАННЯ**
**НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ**

Бузюровій Анні Олександровні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Мобільний рекомендаційний туристичний застосунок

затверджена протоколом засідання кафедри від « 23 » грудня 2021 р. № 4

0. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року
0. Вихідні дані до проекту (роботи)

Список туристичних місць, закон України про «Захист персональних даних»

0. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналіз процесу надання рекомендацій туристам
2. Розробка архітектури туристичної рекомендаційної системи
3. Програмне забезпечення рекомендаційного туристичного мобільного додатку

0. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Об'єкт, предмет, мета та завдання дослідження (1 слайд), порівняльний аналіз існуючих рішень (1 слайд), проектування інформаційної системи (3 слайди), математичне забезпечення (2 слайди), інформаційне забезпечення (1 слайд), структура програмного забезпечення (1 слайд), огляд процесу тестування (1 слайд), висновок (1 слайд)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Федусенко О.В. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Бузюрова А.О. /
(підпис) (ПІБ)

## КАЛЕНДАРНИЙ ПЛАН

| Пор. № | Назва етапів випускної кваліфікаційної роботи | Термін виконання етапів випускної кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1. | Аналіз літературних джерел, аналіз існуючих методів, аналіз основних процесів предметного середовища, постановка задачі | 15.02.2022 – 06.03.2022 | Виконано |
| 2. | Проектування рекомендаційної туристичної системи | 07.03.2022 – 03.04.2022 | Виконано |
| 3. | Розробка та тестування мобільного рекомендаційного туристичного застосунку | 04.04.2022 – 01.05.2022 | Виконано |
| 4. | Оформлення пояснювальної записки, підготовка презентації | 02.05.2022 – 08.05.2022 | Виконано |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент-дипломник _____ / Бузюрова А.О. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Федусенко О.В. /
(підпис) (ПІБ)

# АНОТАЦІЯ

Бузюрова Анна Олександрівна виконала випускну кваліфікаційну роботу на тему «Мобільний рекомендаційний туристичний застосунок» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних туристичних рекомендаційних систем, розглянуто архітектуру системи, розроблено математичне, інформаційне та програмне забезпечення, що надає можливість отримання туристичних рекомендацій користувачами.

Ключові слова: туризм, рекомендаційна система, мобільний застосунок.

# ANNOTATION

The degree project: «Mobile travel recommendation application» has completed by Buzyurova Anna specialty 122 – «Computer Sciences».

In the final qualification work the analysis of modern tourist recommendation systems is carried out, the architecture of the system is considered, the mathematical, informational and software which gives the chance to receive tourist recommendations by users is developed.

Key words: tourism, recommendation system, mobile application.

# ЗМІСТ

# ВСТУП

Сучасний світ стає все більш різноманітним у сфері інформаційних технологій. Все більше галузей використовують технологічні рішення для покращення та спрощення своєї роботи.

Однією з таких галузей є туристична сфера. Існує безліч застосунків для зручнішої організації процесів туристичних агенств, але не настільки багато – для самостійного туризму. При цьому останній – також дуже популярна діяльність людей, які хочуть не тільки відпочити, але і познайомитись ближче із цим світом, відвідавши визначні місця. Але при цьому, у кожного є своя мета і свої переваги, які програмний застосунок повинен враховувати.

Об’єктом дослідження даної роботи є процеси організації туристичних поїздок.

Предметом дослідження роботи є алгоритми отримання туристичних рекомендацій.

Мета даного дипломного проекту полягає в створенні мобільного програмного застосунку для забезпечення туристів зручним інтерфейсом планування поїздок та отримання ними відповідних рекомендацій на основі інтересів та історії відвідувань. Це дозволить полегшити процес самостійного туризму та заощадити час під час планування поїздок.

Для досягнення даної мети потрібно виконати наступні завдання:

- провести аналіз існуючих технологічних рішень туристичних рекомендаційних систем
- провести аналіз основних методів видачі рекомендацій
- сформувати функціональні та нефункціональні вимоги для розроблюваного застосунку
- розглянути процеси даної предметної області
- розробити архітектуру туристичної рекомендаційної системи
- розробити та протестувати програмний продукт

# РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ НАДАННЯ РЕКОМЕНДАЦІЙ ТУРИСТАМ

## 1.1 Аналітичний огляд літератури за темою досліджень рекомендаційні туристичні системи

Рекомендаційні системи є одним із найбільш популярних додатків інтелектуального аналізу даних і машинного навчання в сфері інтернет-бізнесу. Вони аналізують поведінку користувачів інтернет-сервісу, після чого дають кількісну та якісну оцінку вподобання користувачем того чи іншого об’єкту. Об’єктами рекомендацій можуть бути товари в інтернет-магазині, перелік розділів веб-сайту, медіа-контент або інші споживачі веб-сервісу. Рекомендаційні системи віднайшли широке застосування у таких сферах як електронна комерція, соціальні мережі, веб-додатки тощо, де акцент робиться на користувача даних [1].

Загалом, рекомендаційні системи можуть служити двом різним цілям. З одного боку, їх можна використовувати, щоб стимулювати користувачів робити щось, наприклад, купувати конкретну книгу або дивитися певний фільм. З іншого боку, рекомендаційні системи також можна розглядати як інструменти для боротьби з інформаційним перевантаженням, оскільки ці системи мають на меті вибрати найцікавіші елементи з більшого набору. Таким чином, дослідження рекомендаційних систем також сильно вкорінені в області пошуку інформації та фільтрації інформації [2].

Туризм – один з найбільш швидкозростаючих секторів, і багато країн залежать від нього як від основного джерела прибутку. Його можна розділити на різні категорії, виходячи з основних мотивів, таких як медичний, освітній, художній, спортивний туризм і т.д. [3]. Ця область складається з величезної кількості інформації, яка зберігається в цифровому вигляді.

Туризм сьогодні - глобальний комп'ютеризований бізнес, учасниками якого є не тільки туроператори і турагенти, а й великі авіакомпанії, готелі і ресторани усього світу.

Рекомендаційні системи - це відмінна можливість спростити роботу і заощадити час як туристичним агентствам, так і окремим туристам [4].

Туристичні рекомендаційні інформаційні системи – надають користувачеві рекомендації, щодо маршруту подорожі та відповідних туристичних об'єктів, з врахуванням певних критеріїв. Користувач заповнює певного роду анкету, або дає відповіді на запитання системи, що стосуються його уподобань та бажань. Система при цьому генерує певні рекомендації в яких враховує отримані відповіді. Такі системи є корисними не тільки на етапі планування подорожі, а й під час її реалізації та супроводу. [5]

Проблеми, які виникають під час розробки ефективних інформаційних систем прийняття рішень в індустрії туризму, пов’язані з необхідністю реалізації методів для обробки великих обсягів різнорідної інформації та організація і впровадження діалогу з користувачем системи, який досить часто не може чітко визначити критерії для пошуку бажаного місця відпочинку.

Таким чином, головним завданням при розробці рекомендаційних систем у галузі туризму є розробка ефективних механізмів спрямованого пошуку потрібної інформації [4].

«Мобільний інформаційний асистент туриста» - це система яка забезпечуватиме комплексний повнофункціональний інформаційно-технологічний супровід туриста на етапах планування та здійснення ним подорожі з використанням широкого спектру сучасних інформаційних технологій [6].

Близько двох третин сучасних туристів використовують інформаційні технології для планування та супроводу своєї подорожі, при цьому значна частина з них використовує мобільні комп’ютерні та телекомунікаційні

пристрої. Це, своєю чергою, генерує потребу створення якісних мобільних туристичних інформаційних технологій з метою надання користувачу широкого спектра необхідних інформаційно-технологічних послуг для повноцінного планування, супроводу, підтримки та аналізу результатів туристичної подорожі на базі комплексних повнофункціональних програмно-алгоритмічних застосунків, що реалізовуються на мобільній платформі [6].

Існуючі рекомендаційні системи в електронному туризмі зазвичай імітують послуги, що пропонуються туристичними агентами, коли потенційні туристи звертаються за порадою щодо туристичних місць за певних часових та бюджетних обмежень. Користувач зазвичай вказує свої потреби, інтереси та обмеження на основі вибраних параметрів. Потім система співвідносить вибір користувача з каталогизованими місцями призначення, анотованими з використанням того самого вектора параметрів.

Відносно недавній розвиток електронного туризму полягає у використанні мобільних пристроїв як основної платформи для доступу до інформації, що породило сферу мобільного туризму. Унікальні характеристики мобільного туризму висувають нові виклики та можливості для розвитку інноваційних персоналізованих послуг, яким немає місця в сфері електронного туризму. Наприклад, знання точного місцезнаходження користувача створює відповідну основу для надання послуг на основі розташування. Крім того, мобільність користувачів дозволяє використовувати знання про історію мобільності користувачів і скористатися перевагами соціального середовища користувача, розташованого в географічній близькості [7].

### 1.2 Аналіз існуючих туристичних інформаційних систем

Мобільні туристичні системи стрімко розвиваються у сфері туризму і на даний момент туристи вже мають змогу їх використовувати під час своїх подорожей. Наведемо та проаналізуємо деякі з них.

Tripadvisor - найбільша у світі (станом на 2019 рік) платформа про подорожі, щомісяця допомагає 463 мільйонам мандрівників робити кожну подорож незабутньою. Мандрівники по всьому світу використовують сайт та програму Tripadvisor для того, щоб переглянути більш ніж 859 мільйонів відгуків та коментарів про 8,6 мільйонів варіантів житла, ресторанів, розваг, авіаліній та круїзів. На етапі планування та під час самої поїздки мандрівники звертаються до Tripadvisor для порівняння низьких цін на готелі, авіарейси та круїзи, бронювання популярних екскурсій, у тому числі й відомих пам'яток, а також резервування столиків у хороших ресторанах. Tripadvisor, незамінний помічник для мандрівників, доступний у 49 регіонах світу 28 мовами [8].

izi.TRAVEL – мультимедійний гід з метою інформування туристів про історію різних міст та музеїв. Програма дозволяє обрати тур, побачити маршрут екскурсії на карті та прослухати інформацію про всі цікавинки обраних місць. Також є можливість створити власний аудіогід на будь-яку туристичну тему.

Guides by Lonely Planet – програма-путівник, яка надає ряд послуг для всіх, кому подобається подорожувати. Завдяки ній можна отримати рекомендації щодо місць, які варто відвідати, поспілкуватися з іноземцями завдяки аудіо-розмовникам, бронювати готелі, ресторани та інші місця, подивитися їх на карті. Програма підтримує офлайн режим, а отже можна отримати всі послуги не маючи з’єднання з інтернетом.

Foursquare – колись це був сервіс для так званих «чекінів», що давало можливість отримувати різні знижки та акції у відвіданних місцях, але тепер це геолокаційна рекомендаційна система, яка дозволяє отримати рекомендацію щодо відвідування міцсь на основі відгуків, залишених у іншій системі, пов’язаній з даною.

На основі аналізу різних туристичних інформаційних систем визначимо фактори їх порівняння:

1. підтримка різних мобільних операційних систем, таких як IOS та Android;

2. визначення геолокації, яка буде використовуватись у подальшому для отримання рекомендації та побудови маршрутів до обраних місць або за обраними турами;
3. врахування вподобань користувача при визначенні рекомендацій, а саме відштовхуючись від обраних улюблених категорій та найбільш відвідуванних місць за певний період;
4. можливість використання карти для отримання візуальної інформації місцезнаходження бажаних до відвідування об’єктів, а також відображення маршрутів;
5. наявність різних мов, як самої програми, так і окремого компоненту «аудіо-гіда»;
6. офлайн режим, тобто можливість отримати всю необхідну інформацію без підключення до інтернету;
7. наявність аудіо-гіда, який би розповів історію міста, конкретного музею, архітектурну пам’ятку тощо;
8. можливість поділитися інформацією в соц-мережах, щоб знайомі та друзі могли також побачити або почути про визначне місце, або отримати деталі подорожі;
9. можливість ставити оцінки та залишати відгуки, щоб туристи мали змогу дізнатися думки інших людей, вже відвідавших дане місце та вирішити, чи хочуть вони побачити його власними очима.
10. можливість додавати свої туристичні місця, тобто доповнювати вже існуючі списки мало відомими місцями.

Проведемо порівняльний аналіз існуючих рішень за наведеними вище факторами.

*Таблиця 1.1. Порівняння існуючих рішень за факторами*

| | Tripadvisor | izi.TRAVEL | Guides by Lonely Planet | Foursquare |
|---|---|---|---|---|
| підтримка різних мобільних операційних систем | + | + | + | + |

| | | | | |
|---|---|---|---|---|
| визначення геолокації | + | + | + | + |
| врахування вподобань користувача | - | - | - | - |

*Продовження таблиці 1.1*

| | Tripadvisor | izi.TRAVEL | Guides by Lonely Planet | Foursquare |
|---|---|---|---|---|
| можливість використання карти | + | + | + | + |
| наявність різних мов | + | + | - | + |
| офлайн режим | - | + | + | - |
| наявність аудіо-гіда | - | + | - | - |
| можливість поділитися інформацією в соц-мережах | + | + | + | + |
| можливість ставити оцінки та залишати відгук | + | + | - | + |
| можливість додавати свої туристичні місця | + | + | - | - |

Отже, можна зробити висновок, що існуючі туристичні інформаційні системи об'єднують в собі досить багато функцій, але при цьому жодна з них не враховує вподобання користувача. Рекомендації надаються на основі тільки геолокації або відгуків інших користувачів. Метою даної роботи є поєднання усіх вище перерахованих функцій для комфортного туризму, а також можливість отримання рекомендацій на основі обраних користувачем категорій та вже відвіданних місць.

### 1.3 Аналіз існуючих методів отримання рекомендацій

Оскільки дана система є рекомендаційною, а отже буде створена для надання користувачам рекомендацій щодо туристичних місць, то вона буде використовувати певний алгоритм їх отримання.

Будь-який алгоритм рекомендацій передбачає початковий збір даних. Загалом, це головна і найбільша частина роботи системи. Існує дві основні методики збору даних: явний і неявний збір даних.

Явний збір даних означає, що користувач сам надає необхідні для роботи матеріали. Наприклад, це може бути пошуковий запит, який він вводить, персональні дані, які він заповнює при реєстрації, прохання програми на початку оцінити якісь дані, проставити рейтинги.

Неявний збір даних означає, що програма буде таємно слідкувати за користувачем. Отримувати з інших джерел інформацію про його вподобання, оцінки, покупки тощо. Взагалі, даний варіант є не дуже етичним і правильним з точки зору захисту персональних даних, але таким чином система може отримати більше корисної інформації, яка допоможе розрахувати кращі рекомендації.

Оскільки дана рекомендаційна система передбачає реєстрацію користувача та вибір ним улюблених категорій, а також стеження за подальшою історією відвідувань та рейтингами, які проставляє сам користувач, то буде використовуватись явний збір даних.

Далі розглянемо самі алгоритми (рис. 1.1). Більшість рекомендаційних систем використовують один із двох основних методів: колаборативну фільтрацію або фільтрування на основі вмісту. Варто відзначити, що на практиці зазвичай використовуються гібридні методи, що поєднують в собі переваги приведених нижче підходів.

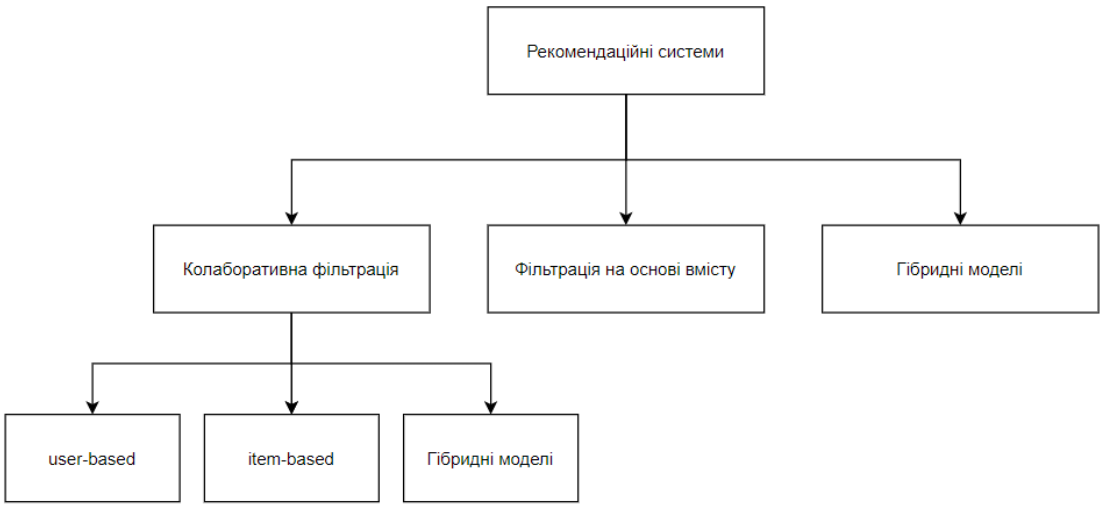



Рисунок 1.1 – Класифікація рекомендаційних систем

Моделі фільтрації на основі вмісту передбачають, що вподобання користувачів визначаються властивостями об’єктів, які вони переглядали раніше. Рекомендаційні системи з фільтрацією на основі змісту (контентною фільтрацією) беруть до уваги схожість об'єктів з інформацією відомою про користувача.

Інформація про користувача може бути отримана з його профілю та/або зібрана з його дій на веб-сайті – написаних відгуків та коментарів, придбаних товарів, переглянутих веб-сторінок тощо. Така інформація може бути зібрана як явними способами (дані з профілю користувачів та сторінок опису об’єктів), так і неявними способами (вилучення ключових слів та фраз, а також їх емоційного забарвлення з коментарів та постів користувачів).

Для створення рекомендацій такі системи аналізують інформацію про користувача, формують ключові слова про його інтереси та вподобання. Також формуються ключові слова для об’єктів системи з їх описів, отриманих тегів тощо. Рекомендуватися будуть об’єкти з бази даних, вибрані на основі визначених ключових слів. Тому таку фільтрацію ще можна назвати фільтрацією по ключовим словам [9].

Методи колаборативної фільтрації, що використовують моделі сусідства, здійснюють прогнозування вподобань користувачів на основі ступеню сусідства між елементами рекомендаційної системи. Ступінь

сусідства визначається за допомогою коефіцієнтів подоби, обчислення яких здійснюється на основі різних параметрів та метрик.

Тож першим кроком методів, заснованих на колаборативній фільтрації, є обчислення коефіцієнтів подоби користувачів та/або об’єктів системи, обчислення яких здійснюються найчастіше на основі оцінок, які користувачі виставляють об’єктам. Крім оцінок можуть використовуватися дані про здійснені перегляди, поставлені теги, написані коментарі тощо. У найбільш простому випадку збираються дані про поставлені користувачами оцінки та записуються у матрицю рейтингів (рис. 1.2).

| | **Об’єкт 1** | **Об’єкт 2** | **…** | **Об’єкт n** |
|---|---|---|---|---|
| **Користувач 1** | 5 | 3 | … | 3 |
| **Користувач 2** | 4 | – | … | 2 |
| **…** | … | … | … | … |
| **Користувач m** | – | 4 | … | 4 |

Рисунок 1.2 - Матриця рейтингів

В матриці рейтингів значення є оцінками конкретного користувача для конкретного об’єкту. Відсутні значення в таблиці рейтингів є невідомими, тобто, для певного об’єкту певний користувач не виставив оцінку.

Розглянемо метрики, що можуть використовуватися для визначення коефіцієнтів подоби у колаборативній фільтрації.

Якщо об’єкт (або користувач), що описується m ознаками, представити точкою у k-мірному просторі, то подібність об'єктів один з одним буде визначатися як відстань в даному метричному просторі. У випадку з матрицею рейтингів таке представлення можливе – ознаками будуть в такому разі оцінки об’єктам. Найбільш поширені метрики подоби, що використовуються в такому випадку: евклідова відстань (1.1), зважена евклідова відстань (1.2), відстань Хемінга (Манхеттенська відстань) (1.3), відстань Мінковського (1.4), відстань Махаланобіса (1.5), кореляція Пірсона (1.6), косинусна подоба (1.7):

$$d(x_1,x_2)=\sqrt{\sum_{i=1}^{m}(x_{1i}-x_{2i})^2}, \quad (1.1)$$

$$d(x_1,x_2)=\sqrt{\sum_{i=1}^{m}w_i(x_{1i}-x_{2i})^2}, \quad (1.2)$$

$$d(x_1,x_2)=\sum_{i=1}^{m}|x_{1i}-x_{2i}|, \quad (1.3)$$

$$d(x_1,x_2)=\left(\sum_{i=1}^{m}|x_{1i}-x_{2i}|^p\right)^{1/p}, \quad (1.4)$$

$$d(x_1,x_2)=\sqrt{(\bar{X}_1-\bar{X}_2)^{\delta}\sum^{-1}(\bar{X}_1-\bar{X}_2)}, \quad (1.5)$$

$$d(x_1,x_2)=\frac{\sum_{i=1}^{m}(x_{1i}-\bar{x}_1)(x_{2i}-\bar{x}_2)}{\sqrt{\sum_{i=1}^{m}(x_{1i}-\bar{x}_1)^2}\sqrt{\sum_{i=1}^{m}(x_{2i}-\bar{x}_2)^2}}, \quad (1.6)$$

$$d(x_1,x_2)=\frac{\bar{X}_1\cdot\bar{X}_2}{\|\bar{X}_1\|\cdot\|\bar{X}_2\|}=\frac{\sum_{i=1}^{m}x_{1i}\cdot x_{2i}}{\sqrt{\sum_{i=1}^{m}(x_{1i})^2}\sqrt{\sum_{i=1}^{m}(x_{2i})^2}}, \quad (1.7)$$

де $d(x_1,x_2)$ – відстань між об'єктами $x_1$ і $x_2$; $x_{1i},x_{2i}$ – значення і-ї ознаки відповідно у 1-го та 2-го об'єкту; $W_i$– вага, що привласнюється і-ій змінній; $^{-1}$ – матриця зворотна коваріаційній матриці, розрахованій по всій вибірці; X1 , X2 – вектори значень ознак у 1-го та 2-го об'єкту; $\underline{x}_1,\underline{x}_2$ – середні значення ознак відповідно у 1-го та 2-го об'єкту; m – кількість ознак.

Методи колаборативної фільтрації на основі моделі сусідства поділяється на два види:

1. Засновані на схожості користувачів (User/User, User-based).

2. Засновані на схожості об’єктів (Item/Item, Item-based) [9].

Перший базується на знаходженні користувачів-сусідів та об’єднання їх, наприклад у кластери. Таким чином, якщо користувачу зі схожими інтересами подобається якийсь об’єкт, скоріше за все він сподобається і даному і попаде у рекомендації.

Другий орієнтований на конкретного користувача та схожість самих об’єктів між собою. У даній рекомендаційній системі буде використовуватись саме другий вид, а отже розглянемо його детальніше.

Основна ідея даних методів полягає у тому, щоб для кожного об’єкту заздалегідь визначити множину схожих на нього об’єктів. Тоді для формування рекомендацій певному користувачу достатньо буде знайти ті об’єкти, яким він поставив найбільші оцінки, та створити зважений список N об’єктів, максимально схожих на них. Результати порівняння об’єктів змінюються не так часто, як результати порівняння користувачів. Тож на першому кроці необхідно дослідити всі наявні дані, а подальші перерахунки можна робити рідко, вибираючи моменти часу, коли навантаження на веб-сайт мінімальне.

Методи колаборативної фільтрації засновані на схожості об’єктів працюють швидше, ніж методи засновані на схожості користувачів, так як багато обчислень можна здійснити заздалегідь. Тож їх можна з успіхом використовувати на великих об’ємах даних.

Для прогнозування оцінки на основі даного підходу треба знайти зважене середнє для оцінених користувачем об’єктів:

$$\widetilde{r}_{i,q}=\overline{r}_i+\frac{\sum_{p=1}^{n}\left(r_{j,p}-\overline{r}_i\right)\cdot k_{q,p}}{\sum_{p=1}^{n}\left|k_{q,p}\right|},$$

де kq, p – коефіцієнт кореляції між q-тим об’єктом та p-тим об’єктом; n – довжина списку схожих на q-тий об’єкт об’єктів.

Використовуючи методи колаборативної фільтрації засновані на схожості об’єктів можна рідше перераховувати коефіцієнти подоби, ніж в методах заснованих на схожості користувачів, адже коефіцієнти подоби для об’єктів змінюються рідше, ніж для користувачів. Також Item-based методи мають більшу стійкість до інформаційних атак, ніж User-based методи [9].

Обидва методи мають спільний недолік – на початку роботи системи невідомі ні оцінки даного користувача, ні загальні рейтинги нових місць. Але оскільки в даній системі на початку обираються улюблені категорії туристичних місць та використовується геолокація, то вже на початку роботи програма знає, які рекомендації можна надати користувачу.

### 1.4 Аналіз основних процесів предметного середовища

Наступним етапом проаналізуємо основні процеси, побудувавши відповідні діаграми. Створимо контекстну діаграму «ЯК Є» та її декомпозицію, які продемонструють процеси предметного середовища без використання мобільного рекомендаційного туристичного застосунку.

На першій діаграмі відображено процес «планування туристичної подорожі» (рис. 1.3). На вході ми отримуємо обране місто, куди хоче поїхати турист або де він вже знаходиться на даний момент, та вподобання туриста. Друге включає в себе категорії місць, які полюбляє відвідувати людина та куди б захотіла піти у першу чергу. Наприклад, це можуть бути визначні пам’ятки, музеї, місця на природі тощо.

На виході, після виконаного процесу, ми отримуємо план подорожі. План містить у собі визначені для поїздки місця до відвідування, а також передбачає нанесення їх на карту для отримання певного маршруту.

Серед елементів керування для даного процесу зазначено документи для подорожі, адже це дуже важлива частина будь-якої подорожі. Турист обов’язково повинен мати при собі паспорт, якщо подорожує своєю країною, або закордонний паспорт, якщо планує поїздку за кордон, авіабілети або залізничні квитки для пересування або необхідні документи на власний транспорт. Якщо турист подорожує за кордон, то також потрібно попіклуватись про наявність медичного страхового полісу, підтвердження наявності грошей для туризму, бронювання готелю або запрошення від громадянина відповідної країни.

Механізмами для процесу «планування туристичної подорожі» є сам турист та ресурси, які він використовує – інтернет та карта.

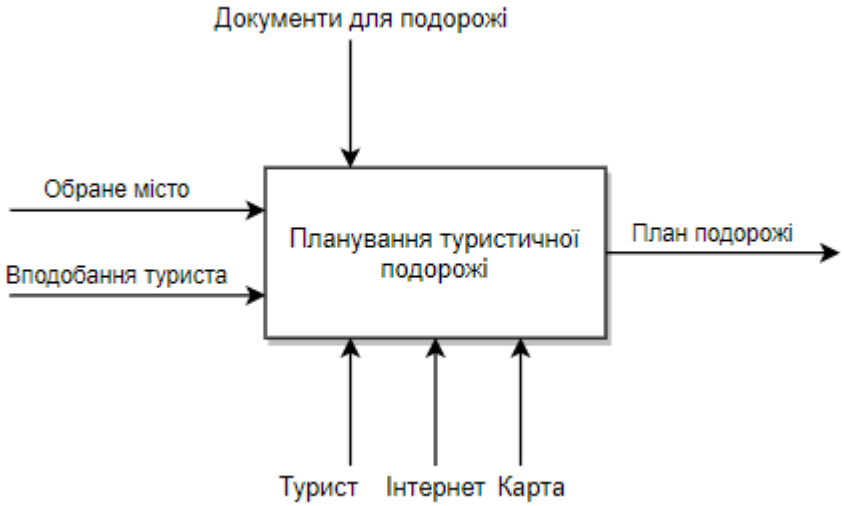



Рисунок 1.3 – Контекстна діаграма ЯК Є

Розглянемо детальніше описаний процес, побудувавши декомпозицію його основних функцій (рис. 1.4). У першу чергу, коли планується подорож, турист здійснює пошук туристичних місць в обраному місті за допомогою мережі інтернет. Прочитавши різні форуми та статті людина вже складає певний список обраних місць. Після цього турист дбає про наявність карти, без якої орієнтуватись у незнайомому місті буде важко. На цьому кроці можна або купити паперовий варіант, або скачати карту на телефон. Наступним кроком потрібно обрати місця вже на карті, щоб було легше побудувати відповідний маршрут та встигнути відвідати все, що запланувалось.

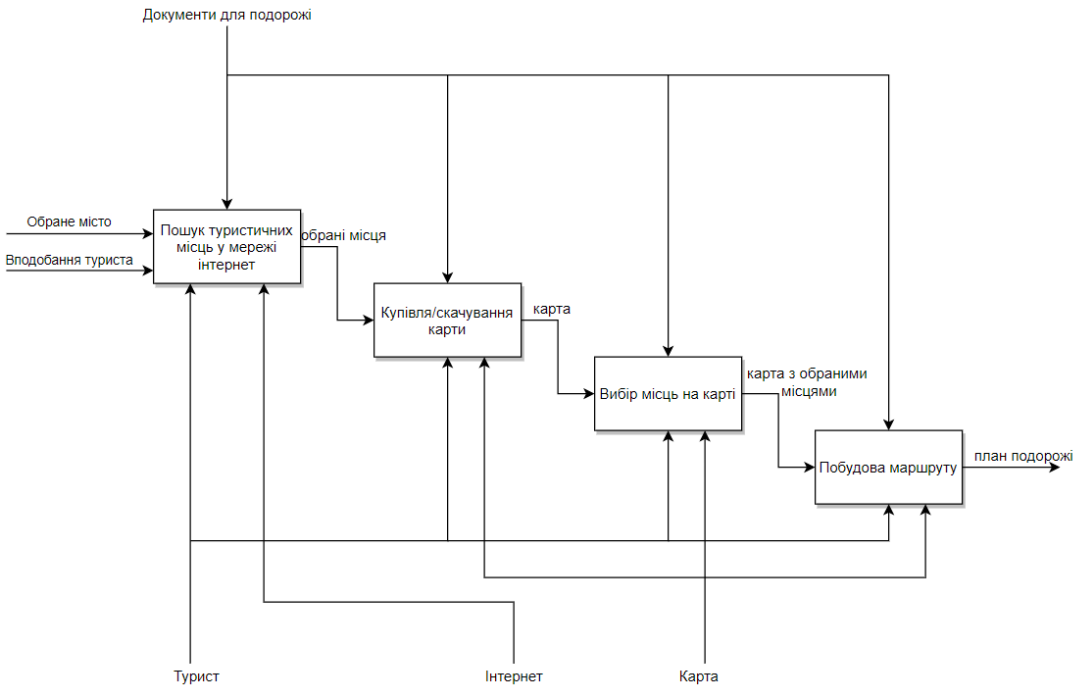



Рисунок 1.4 - Декомпозиція основних функцій діаграми ЯК Є

Отже, процес планування туристичної подорожі потребує значних зусиль, багато часу і ресурсів для отримання результату, який задовольнить туриста, який збирається цікаво та корисно провести час у іншому місті. Щоб точно встигнути подивитися все, що хочеться, та отримати справжнє задоволення від поїздки треба ретельно підготуватися. Пошук місць, які сподобаються саме цьому туристу, читання відгуків інших людей, нанесення місць на карту та отримання задовільного маршруту – все це необхідно зробити, аби гарно провести вихідні. За допомогою системи, що розробляється, цей процес стане простіше та зручніше: треба тільки зайти у програму та подивитися рекомендації. Таким чином турист зможе заощадити багато часу та швидко і надійно побудувати свій маршрут. Відповідно буде більше вільного часу, щоб зібрати необхідні речі для поїздки та перевірити документи і не переживати, куди піти та що подивитися.

## 1.5 Постановка задачі на розробку мобільного рекомендаційного туристичного застосунку

Функціональні вимоги:

- реєстрація у системі

Користувач, який перший раз зайшов у систему повинен мати можливість створити новий акаунт. Під час створення він заповнює декілька форм, де вказує персональні дані, такі як наприклад електронна пошта, пароль, ім’я тощо, а також обирає улюблені категорії туристичних місць, які хотів би відвідати.

- можливість увійти в системі в свій акаунт (логін)

Зареєстрований користувач повинен мати можливість увійти у вже створений акаунт для того, щоб отримати рекомендації, подивитися або редагувати власні списки, персональну інформацію, продивитися свою історію відвіданих місць та отримати доступ до функцій, пов’язаних з цими даними.

- можливість обрати улюблені категорії місць

Користувач повинен мати можливість обирати категорії туристичних місць, яким надає перевагу під час візиту до іншого міста або країни. Вони будуть враховуватись у подальшому для визначення рекомендацій. Також на основі історії відвіданих місць система буде пропонувати видалити категорії, місця з яких турист давно не відвідував, або, навпаки, додати нові, якщо місця з них часто відвідуються.

- визначення геолокації користувача

Система повинна запитувати у користувача дозвіл на отримання інформації про його геолокацію та, у разі дозволу, визначати її для підбору рекомендацій. Таким чином турист матиме змогу одразу бачити місця в місті, де він зараз знаходиться, та ті, які скоріше за все йому сподобаються. Використовуючи дану функцію користувач буде відчутно економити час, який він зазвичай витрачає на пошук місць, куди слід піти або поїхати. Крім цього, гелокація буде використовуватись для відображення на карті місцезнаходження туриста та побудови маршруту до обраних ним місць.

- отримання рекомендацій щодо відвідування туристичних місць

Користувач повинен мати змогу отримувати рекомендації щодо відвідування певних туристичних місць. Під час розрахування рекомендацій

будуть враховуватись і відгуки інших користувачів, і місцезнаходження туриста, і обрані ним категорії. Таким чином будуть обиратися найбільш цікаві та варті візиту місця. Також під час відображення вони будуть відсортовані за рейтингом так, щоб зверху списку одразу було видно місця з найвищою оцінкою. При цьому місця з дуже низьким рейтингом відображатись у списку не будуть.

- пошук всіх туристичних місць в обраному місті

Крім отримання рекомендацій користувач повинен мати можливість подивитися всі туристичні місця. При цьому список не буде обмежений тільки з геолокації, турист буде мати змогу обрати будь-яке місто на свій розсуд. Ця функція буде дуже зручною під час планування майбутніх поїздок, ще до безпосереднього приїзду. Наприклад, якщо людина ще не знає, куди точно хоче поїхати, або вже знає, що на місці в неї не буде інтернет зв’язку та хоче заздалегідь скласти план подорожі та скачати все необхідне. При цьому, як і для рекомендацій, місця будуть розташовані відсортовано за рейтингом.

- можливість переглянути місця на карті

Користувач повинен мати можливість переглядати обрані туристичні місця на карті для отримання інформації про їх розташування. Це дозволить скласти точніше маршрут та оцінити час і витрати на дорогу.

- відображення маршруту на карті

Користувач повинен мати можливість бачити складений маршрут на карті. Так йому буде простіше та швидше знайти необхідні об’єкти. Крім самого маршруту, якщо надано відповідний дозвіл, буде можливість побачити своє місцезнаходження та зрозуміти, у який бік рухатись далі.

- наявність аудіо-гіда

Користувач повинен мати можливість використання аудіо-гіда. Тобто він зможе прослуховувати інформацію про різні архітектурні пам’ятки, визначні місця, експонати у музеях тощо. Таким чином, навіть подорожуючи самостійно, без використання екскурсій, можна буде поринути в історію

відвіданих місць та дізнатися багато нового, а завдяки саме аудіо режиму не потрібно буде при цьому відволікатись та можна буде продовжувати роздивлятись детальніше об’єкти.

- позначення вже відвіданих місць

Користувач повинен мати можливість позначити вже відвідані місця. При цьому турист буде обирати дату відвідання і вони будуть збережені в окремий список профілю «історія відвіданих місць». Після цього вони не будуть відображатись в рекомендаціях, але будуть використовуватись для редагування улюблених категорій.

- перегляд історії відвіданих місць

Користувач повинен мати можливість переглядати історію вже відвіданих місць. Місця у даному списку будуть відображатись відсортовано за датами: найстаріші будуть внизу списку, а нещодавні – зверху. При цьому для кожного місця буде відображатись його назва, місто та країна, де воно знаходиться, а також рейтинг, проставлений користувачем та дата візиту. За бажанням дату можна буде відредагувати.

- можливість ставити оцінки та залишати відгуки

Користувач повинен мати можливість ставити оцінки та залишати відгуки на відвідані туристичні місця. Завдяки цьому інші туристи зможуть дізнатися, чи дійсно цікаве дане місце та чи варто додавати його в свій список. Проставлена оцінка буде впливати на загальний рейтинг місця та, відповідно, його відображення у рекомендаціях.

- створення списків бажаних місць до відвідування

Користувач повинен мати можливість створювати власні списки бажаних місць, які він планує відвідати. Таким чином він зможе зберегти місця, які йому сподобалися, розділити їх по різним списками, наприклад за містом або країною, та за допомогою системи розробити маршрут, який відобразиться на карті. Таким чином у будь-який момент користувач зможе одразу подивитися, куди він хотів піти, та знайти необхідну інформацію.

- можливість поділитися інформацією в соц-мережах

Користувач повинен мати можливість поділитися інформацією в соц-мережах зі своїми рідними, друзями та знайомими. Тоді люди, яких він обере, зможуть побачити відповідні місце, список, маршрут тощо. Користувач зможе самостійно обрати, чим саме ділитися через соц-мережі, та який доступ надавати цим людям. Наприклад, до списків можна буде надати доступ «тільки читання» або «читання та редагування». Тоді люди, які подорожують разом, зможуть створити список місць, який задовольнить всіх.

Нефункціональні вимоги:

- Підтримка різних мобільних операційних систем

Дана програма повинна підтримуватись різними мобільними операційними системами. У сучасному світі мобільні пристрої дуже стрімко розвиваються та вже на даний момент існує багато виробників телефонів. Відповідно розроблено різні операційні системи, найпоширенішими з яких є IOS та Android. Оскільки кожна з них дуже популярна серед користувачів мобільних технологій, то і дана система повинна підтримуватись обома, тобто бути кросплатформною.

- Підтримка офлайн режиму

Користувач повинен мати можливість зберегти необхідну інформацію на пристрій, щоб вона була доступною без інтернету. Цей режим буде дуже корисним для тих, хто подорожує за кордон, та користується там тільки підключенням до мережі «вай-фай». Але вона є далеко не всюди, а отже важливо, щоб не дивлячись на це, списки обраних місць та маршрути всеодно були доступні до перегляду.

- Наявність різних мов

Система повинна підтримувати різні мови, при чому як самого інтерфейсу, так і компоненту «аудіо-гід». Оскільки програма буде використовуватись на території не тільки однієї країни, то повинна бути можливість змінити мову на рідну або хоча б знайому. Першою буде розроблятись українська версія. Найпоширенішою на сьогоднішній день є англійська мова, а отже вона повинна бути обо’язково наступною. Також у

список найпопулярніших мов входять китайська, іспанська, російська, японська, арабська. Але оскільки розповсюджуватись дана програма буде скоріше у Європі, то планується включити в список основні європейські мови, такі як німецька, французька, італійська, польська.

### 1.5 Висновки до першого розділу

Отже, в результаті ретельного огляду літератури за темою «рекомендаційні туристичні системи» було оцінено сучасний стан та поняття рекомендаційних систем у сфері туризму. Також було проаналізовано вже існуючі інформаційні системи за даною темою досліджень. Було обрано фактори та порівняно відповідні системи між собою. Таким чином були сформовані нові, відносно існуючих рішень, вимоги до системи, яка розроблюється.

Також було проаналізовано основні процеси предметного середовища та побудудовано діаграми, які їх відображають. В результаті, ми переконались в актуальності даної рекомендаційної системи. Заключним етапом було сформовано задачу на розробку, а саме – визначення всіх функціональних і нефункціональних вимог та їх опис.

# РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ ТУРИСТИЧНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

## 2.1 Розробка архітектури

### 2.1.1 Функціональний аналіз

Для отримання повної картини бізнес-процесів розроблюваної системи представимо їх у вигляді дерева функцій (рис. 2.1). У вершині вказано головний модуль, а саме – модуль туристичного рекомендаційного додатку. Функції даного застосунку діляться на клієнтські та адміністративні.

До клієнтських функцій системи входять реєстрація, ведення профілю та пошук туристичних місць. Реєстрація розділяє функції для авторизованих користувачів та гостей. Також вона передбачає створення профілю. Його ведення містить функції редагування та роботу з туристичними місцями, адже вони доступні тільки авторизованим користувачам. Друга функція включає в себе різноманітні дії з туристичними місцями, такі як отримання рекомендацій, створення списків до відвідування, додавання власних місць та створення маршруту.

До адміністративних функцій відносяться робота з клієнтами та робота з контентом. Перше містить в собі функції формування звітів по клієнтам та створення «чорних списків», якщо наприклад користувачі постійно залишають тільки негативні відгуки, які можуть включати і різного роду образи. Робота з контентом передбачає додавання нових туристичних місць, редагування вже існуючих та роботу з категоріями місць, таку як їх додавання, видалення та редагування.

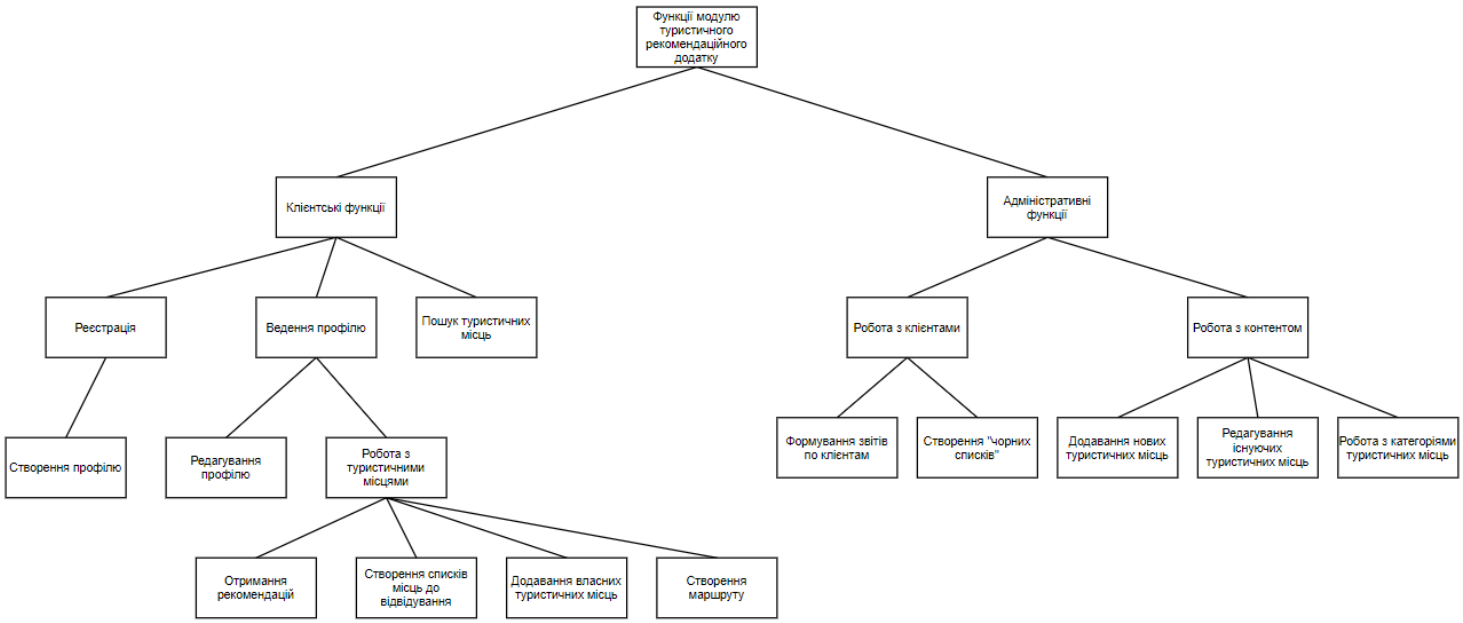



Рисунок 2.1 – Дерево функцій

Опишемо функціонування створюваного туристичного мобільного додатку. Робота починається, коли користувач відкриває завантажений додаток на своєму пристрої. Далі потрібно авторизуватись, тобто створити новий акаунт або увійти у вже створений. Для цього використовується база даних користувачів, в якій зберігаються записи про акаунти. Якщо користувач реєструється, то створюється новий запис, а якщо входить у вже існуючий профіль – то система надсилає запит до бази даних з перевіркою наявності введених даних, а саме – логіну та паролю.

Після успішної авторизації відбувається перехід на першу сторінку додатку, на якій повинна міститись інформація про розраховані пропозиції. Якщо на цей момент геолокація на пристрої не увімкнена, то користувач повинен прийняти рішення, чи вмикати її. Якщо він вирішив включити її, то процес повертається на розгалуження, тобто знову перевіряється, чи включена геолокація. Якщо навпаки, не хоче ділитися з програмою своїм місцезнаходженням, то на сторінці з’явиться відповідне повідомлення та рекомендації не будуть видані. В той же час можна буде подивитися всі можливі варіанти на другій сторінці. Якщо ж геолокація вже включена, то

відбувається завантаження пропозицій та відображення їх на відповідній сторінці. Для цього буде використовуватись база даних туристичних місць, з якої будуть зчитуватись дані для розрахунку рекомендацій та відображення відповідної інформації про них. Дані процеси відображено на діаграмі Event-Driven Process Chain (рис. 2.2).

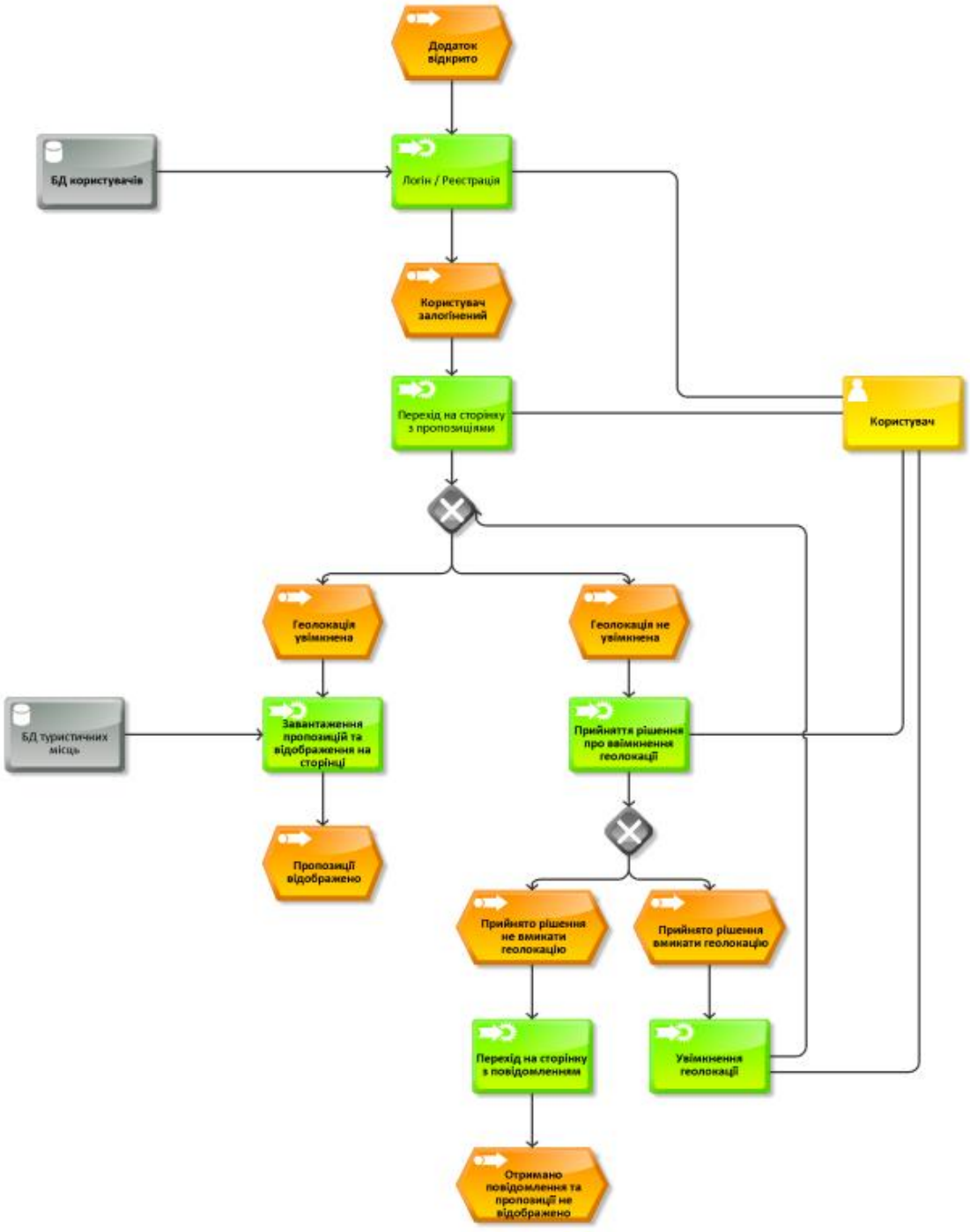


Рисунок 2.2 – Діаграма Event-Driven Process Chain для функціонування мобільного додатку

Наступна діаграма описує ланцюжок процесів, що відбуваються під час функціонування мобільного додатку (рис. 2.3). На найвищому рівні знаходиться процес «Робота з туристичним додатком». Його підпроцесами є «Реєстрація у туристичному додатку» та «Користування туристичним додатком» (відношення частина-ціле). Перший має також підпроцеси – «Заповнення основних даних», «Вибір вподобаних категорій» та «Підтвердження реєстрації». Між цими підпроцесами зв’язком є відношення попередній процес - наступний процес, тобто дії виконуються послідовно. У «Заповнення основних даних» також є свої підпроцеси – «Заповнення логіну», «Заповнення імені» та «Заповнення паролю».

Аналогічним чином побудований другий підпроцес «Користування туристичним додатком». Його підпроцесами є «Отримання пропозицій» та «Зміна особистих даних». Перший містить процеси «Ввімкнення геолокації» та «Перехід на сторінку з пропозиціями». Для другого бачимо також власні підпроцеси: «Перехід на сторінку налаштувань», «Зміна потрібних полів» та «Збереження даних», які повинні виконуватись саме в такому порядку.

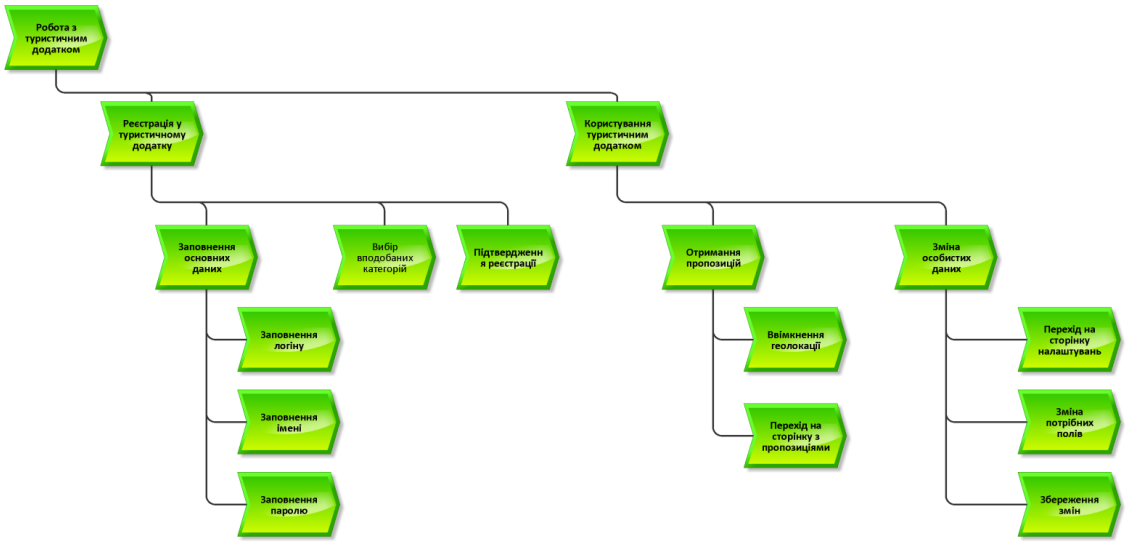



Рисунок 2.3 – Діаграма Value Added Chain Diagram для функціонування мобільного додатку

### 2.1.2 IDEF0 процесу видачі рекомендацій за допомогою інформаційної системи

Представимо виконувані процеси у предметному середовищі за допомогою діаграм «ЯК БУДЕ» у нотації IDEF0, тобто вже з використанням мобільного рекомендаційного туристичного застосунку.

Першою побудуємо контекстну діаграму (рис. 2.4). На вході система отримує гелокацію користувача, яку він увімкнув на своєму пристрої, та дані для авторизації, тобто логін та пароль. На виході повертаються сформовані рекомендації для користувача та оптимальний маршрут для обраних місць подорожі, яка планується.

Елементом керування є закон України про «Захист персональних даних», тому що на кожному етапі зберігаються особисті дані користувача: при авторизації - це логін та пароль, при отриманні рекомендацій – це обрані улюблені категорії та місцезнаходження. Також спостерігається історія вже відвіданих місць, яка використовується в подальшому для обчислення категорій, які потенційно можуть також сподобатись користувачу, або, навпаки, тих, які є улюбленими у користувача, але він рідко відвідає місця з них.

Механізмами у даному випадку є турист, тобто користувач системи, та розроблювана туристична рекомендаційна інформаційна система.

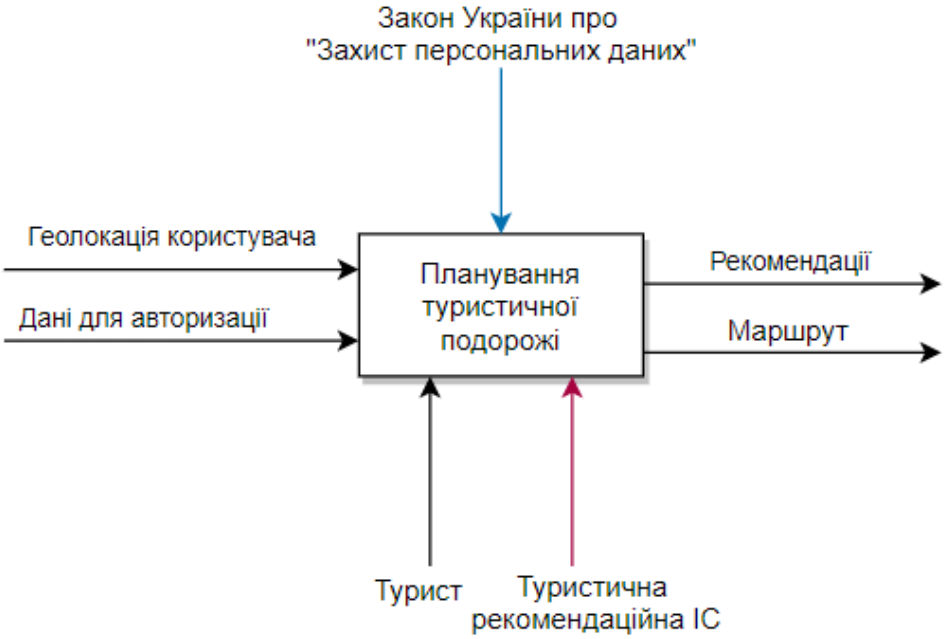



Рисунок 2.4 - Контекстна діаграма ЯК БУДЕ

На наступному етапі проведемо декомпозицію основних процесів діаграми ЯК БУДЕ (рис. 2.5).

Першим процесом є авторизація, бо без неї система не отримає достатньо інформації про користувача для розрахування його персональних рекомендацій.

В результаті отримуємо дані користувача, завдяки яким стає можливим наступний процес, а саме – вибір або редагування улюблених категорій. На цьому етапі формується список категорій туристичних місць, бажаних до відвідування. Їх змінює користувач самостійно, але також система на основі отриманих даних про вже відвідані місця може пропонувати додати нову категорію або видалити вже додану.

Після цього відбувається аналіз отриманих даних та видача рекомендацій. Цей процес виконується самою інформаційною системою, адже відбуваються певні математичні розрахунки, які будуть описані далі.

Наступним етапом є створення списку місць для поїздки. За допомогою наданого інтерфейсу користувач має можливість створити список, в який буде додавати туристичні місця, які захоче відвідати. Таким чином він зможе

швидко знайти те, що йому сподобалось. При чому додавати можна не тільки ті місця, які рекомендує система, а і зі всього списку.

Останнім кроком є побудова оптимального маршруту. Цей процес виконує інформаційна система, використовуючи сформований список місць. Сам маршрут можна буде подивитись, якщо перейти по відповідному посиланню зі списку. При цьому відбується перехід на карту, але можна і відфільтрувати список, тобто місця будуть показані у порядку їх відвідання.

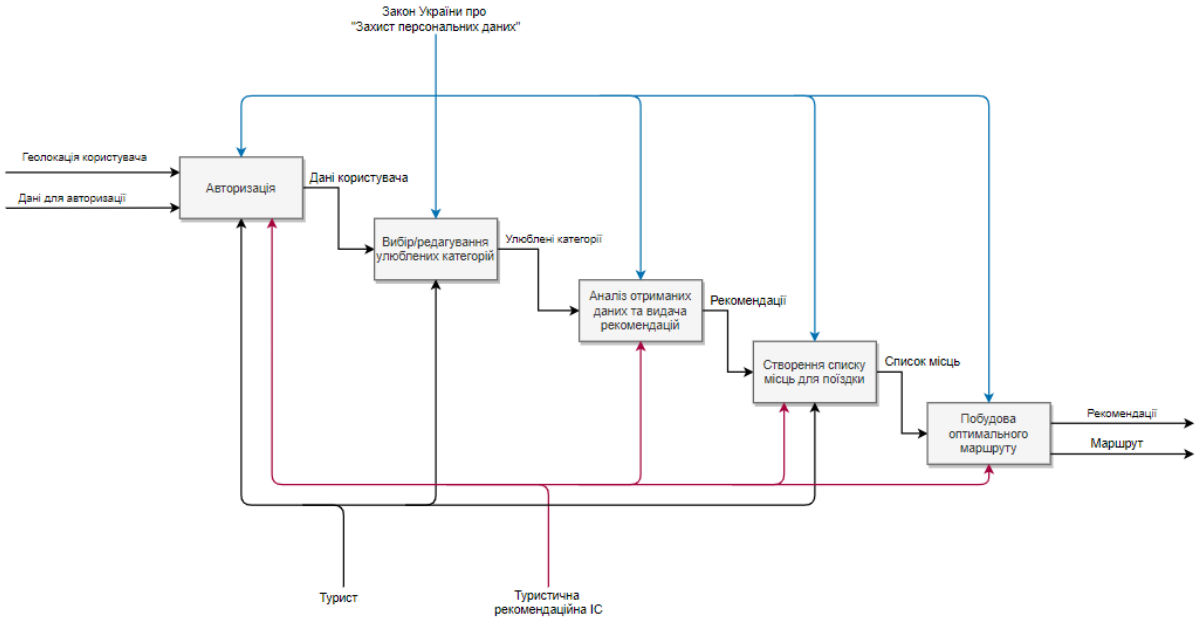



Рисунок 2.5 – Декомпозиція основних процесів діаграми ЯК БУДЕ

### 2.1.3 Архітектура інформаційної системи

Наступним етапом є побудова архітектури розроблюваної інформаційної системи (рис. 2.6). Дана схема відображає модель, структуру, виконувані функції й взаємозв'язок компонентів. Дана система ділиться на модулі роботи з клієнтами та адміністративний, відповідно до дерева функцій.

Модуль роботи з клієнтами включає модулі реєстрації, роботи з профілем, рекомендаційний, пошуку туристичних місць та роботи з картою. Пошук туристичних місць також містить у собі компонент роботи з аудіо-гідом. Модуль роботи з картою також включає модуль створення маршруту.

Адміністративний модуль ділиться на підсистему роботи з базою даних і модуль звітності. Перший включає в себе роботу з підбазами клієнтів та туристичних місць. Оскільки у системі передбачено вибір різних категорій місць, то вони винесені в модуль роботи з підбазою категорій туристичних місць. А також є окремий модуль роботи з підбазою аудіо-гіда.

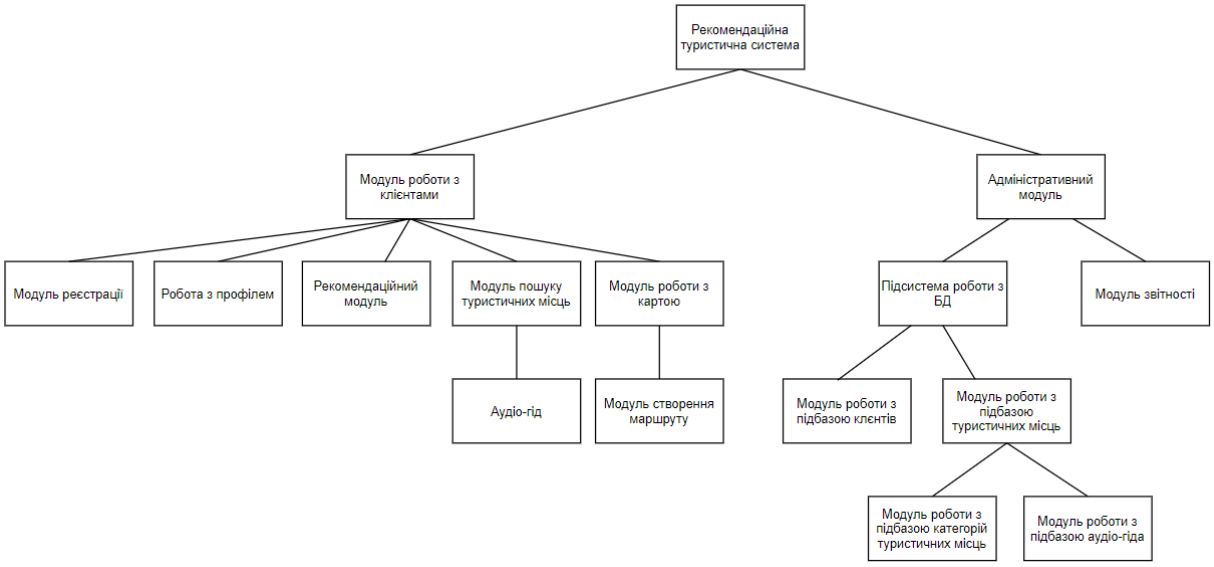



Рисунок 2.6 – Архітектура системи

## 2.2 Математичне забезпечення рекомендаційної туристичної системи

Розглянемо детальніше математичне забезпечення даної рекомендаційної туристичної системи. А саме, опишемо алгоритм, який буде реалізовано для видачі рекомендацій користувачу.

Як вже було сказано, буде використовуватись алгоритм item-to-item колаборативної фільтрації.

На самому початку будується таблиця, яка містить деякі елементи та користувачів, які оцінили ці елементи. Оцінка є чіткою і складається за шкалою від 1 до 5. Кожен запис у таблиці позначає оцінку, надану i-м користувачем j-му елементу. У більшості випадків більшість клітинок порожні, оскільки користувач оцінює лише кілька елементів.

Далі потрібно утворити пари предметів (місць). Після цього ми знаходимо всіх користувачів, які оцінили обидва елементи в парі. Формуємо

вектор для кожного елемента та обчислюємо подібність між двома елементами. Схожість між парами предметів можна знайти різними способами. Одним з найпоширеніших методів є використання косинусної подібності, який і буде тут використовуватись.

Формула косинусної подібності:

$$Similarity(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| * ||\vec{B}||}$$

Також потрібно знайти відсутні оцінки для відповідного користувача. На цьому етапі рекомендаційна система використовує елементи (вже оцінені користувачем), які найбільш схожі на відсутній елемент, щоб створити оцінку. Ми намагаємося генерувати прогнози на основі рейтингів подібних продуктів. Ми обчислюємо це за формулою, яка обчислює рейтинг для конкретного товару (місця), використовуючи зважену суму рейтингів інших подібних продуктів [10].

$$rating(U, I_i) = \frac{\sum_j rating(U, I_j) * s_{ij}}{\sum_j s_{ij}}$$

На завершальному етапі отриманий вектор прогнозованих оцінок сортується за спаданням. Перші 10 елементів і є списоком рекомендацій.

## 2.3 Інформаційне забезпечення рекомендаційної туристичної системи

### 2.3.1 Аналіз інформаційних потоків

Розглянемо перебіг інформації у системі, побудувавши діаграми типу Data Flow Diagram.

Першою є контекстна діаграма (рис. 2.7), яка відображає загальну картину та показує зв’язки між системою та зовнішніми сутностями. Розроблювана туристична рекомендаційна інформаційна система взаємодіє із двума сутностями – «Користувач» та «Адміністратор». Кожен з них надає певну інформацію системі та отримує відповідну від неї. Таким чином, користувач під час роботи із програмою вказує власні дані, тобто дані про користувача як наприклад логін, пароль для входу у систему, ім’я тощо,

обирає окремо улюблені категорії, критерій пошуку та надає дані про свою геолокацію. Від системи він отримує рекомендації, інформацію про туристичні місця та список місць до відвідування, який він формує завдяки програмі.

Друга сутність – адміністратор – підтримує нормальне функціонування системи. Для цього він вносить зміни в дані про туристичні місця, а отримує інформацію про користувачів для подальшого прийняття рішень про ті чи інші дії у відношенні до них, наприклад, про блокування користувача в разі неналежної поведінки, створення звітів тощо.

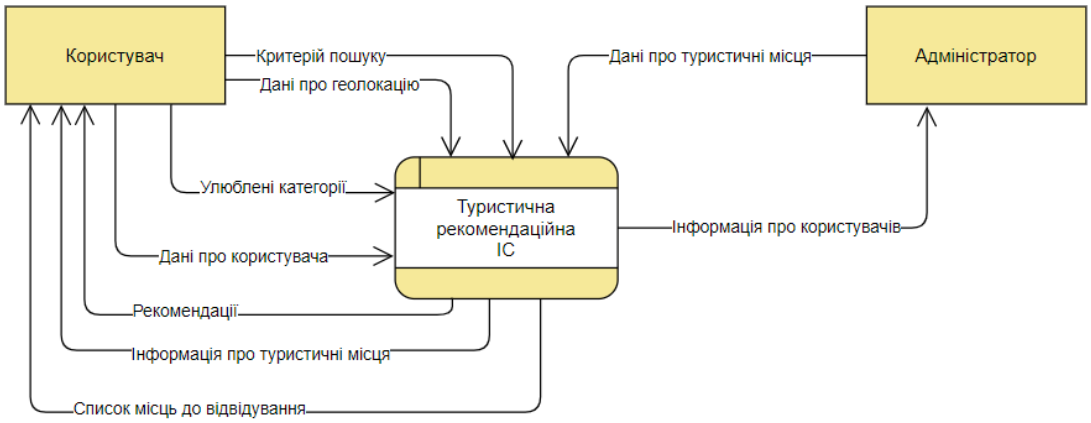



Рисунок 2.7 – Контекстна діаграма DFD

Наступною діаграмою представимо більш детально процеси, які відбуваються у системі з точки зору користувача (рис. 2.8). Дані про нього приходять у систему в першу чергу та записуються до бази даних. Далі відбувається процес формування рекомендацій, для якого потрібні дані про користувача, які вже приходять з бази даних, інформація про туристичні місця, яка також підтягується зі сховища даних, надані користувачем дані про геолокацію та улюблені категорії. Після цього сформовані рекомендації відправляються до бази даних та переходять у наступний процес – роботу з туристичними місцями. Сюди також потрапляє інформація про туристичні місця, а також додається заданий критерій пошуку. Наступним кроком

користувач обирає місця, які переходять в останній процес – формування списку місць до відвідування. Відповідний список повертається користувачу.

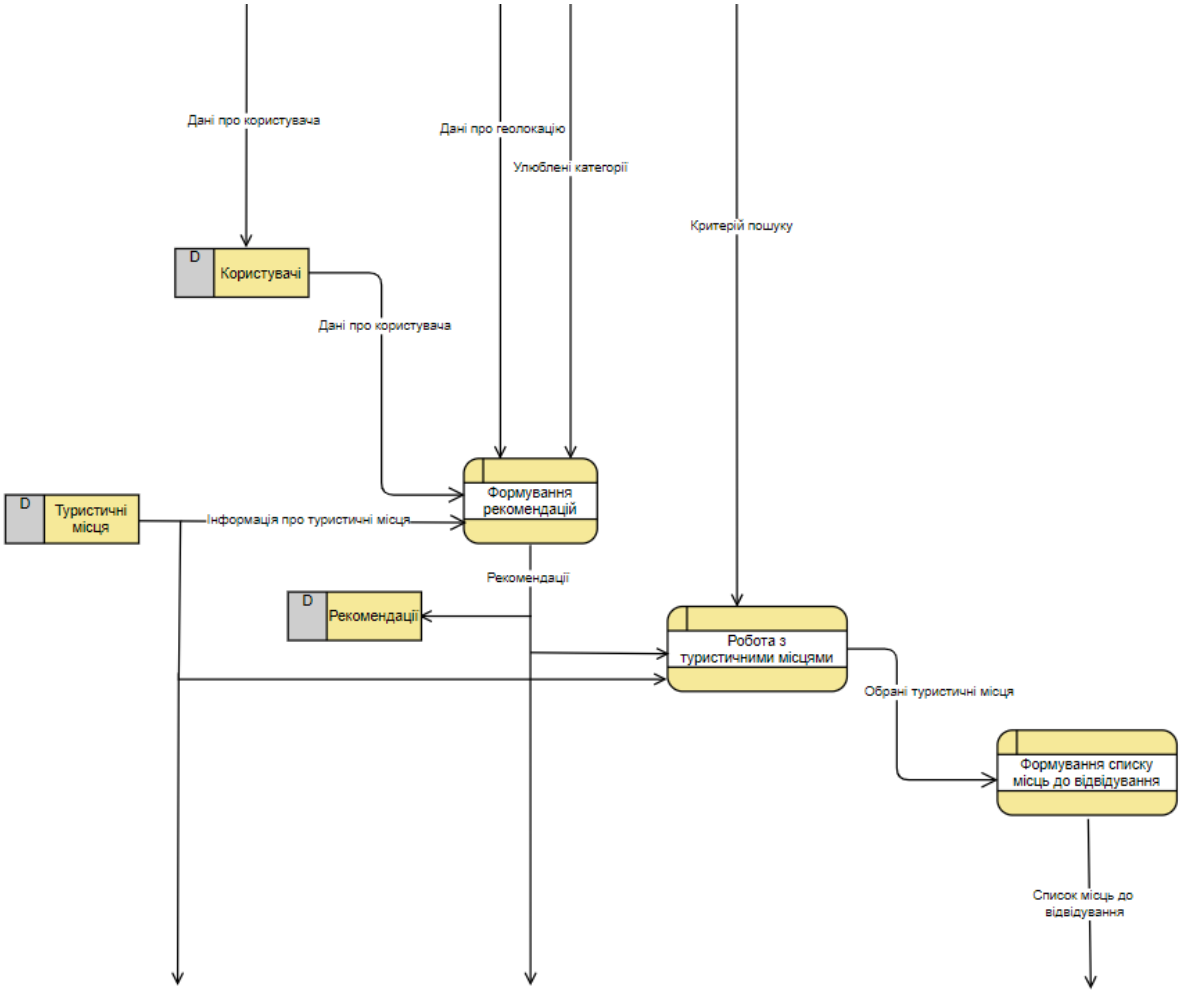



Рисунок 2.8 – Діаграма DFD першого рівня

### 2.3.2 Розробка інформаційного забезпечення

Як вже було зазначено раніше, дана інформаційна система для правильної роботи використовує певні дані. Для збереження цих даних буде розроблена база даних, яка зберігатиме та видаватиме потрібну інформацію.

Спочатку розробимо концептуальну модель бази даних (рис. 2.9). На ній для заданої предметної області відображено наступні сутності:

- Користувач
- Рекомендація
- Туристичне місце
- Категорія
- Рейтинг

- Місто
- Країна

Кожна з них має свої атрибути, тобто майбутні поля у таблицях. Наприклад, «Користувач» має логін, пароль та ім’я, які він вводить при реєстрації. «Рекомендація» містить дату, коли вона була видана. «Туристичне місце» має назву, вартість його відвідування, загальний рейтинг та адресу, завдяки якій місце буде відображатись на карті. «Категорія», «Місто» та «Країна» містять тільки назву. «Рейтинг» містить унікальний номер користувача, унікальний номер туристичного місця та значення рейтингу. Також майже для всіх сутностей є атрибут «id», який задає унікальний номер для кожного запису. Окремим випадком є «Користувач», для якого унікальним є логін.

Усі сутності пов’язані між собою зв’язками, які відображають взаємозалежність даних. Опис зв’язків між сутностями наведено у табл. 2.1.

*Таблиця 2.1 Опис зв’язків між сутностями*

| **№** | **Сутності, що утворюють зв’язок** | **Тип зв’язку** | **Пояснення** |
|---|---|---|---|
| 1 | Користувач - Рекомендація | Один до багатьох | Один користувач отримує багато рекомендацій. Одна рекомендація призначена одному користувачу. |
| 2 | Рекомендація – Туристичне місце | Один до багатьох | Одна рекомендація містить одне туристичне місце. По одному місцю може бути надано багато рекомендацій. |
| 3 | Користувач – Категорія | Багато до багатьох | Один користувач обирає багато категорій. Одна категорія обирається багатьма користувачами. |
| 4 | Туристичне місце – Категорія | Багато до багатьох | Одне туристичне місце відповідає одній категорії. Одна категорія має багато туристичних місць. |
| 5 | Користувач - Рейтинг | Один до багатьох | Один користувач виставляє багато рейтингів. Один рейтинг проставляється одним користувачем. |

*Продовження таблиці 2.1*

| № | Сутності, що утворюють зв'язок | Тип зв'язку | Пояснення |
|---|---|---|---|
| 6 | Рейтинг – Туристичне місце | Один до багатьох | Один рейтинг належить одному туристичному місцю. Одне туристичне місце має багато рейтингів. |
| 7 | Туристичне місце – Місто | Один до багатьох | Одне туристичне місце знаходиться в одному місті. Одне місто може мати багато туристичних місць. |
| 8 | Місто - Країна | Один до багатьох | Одне місто розташоване в одній країні. Одна країна має багато міст. |

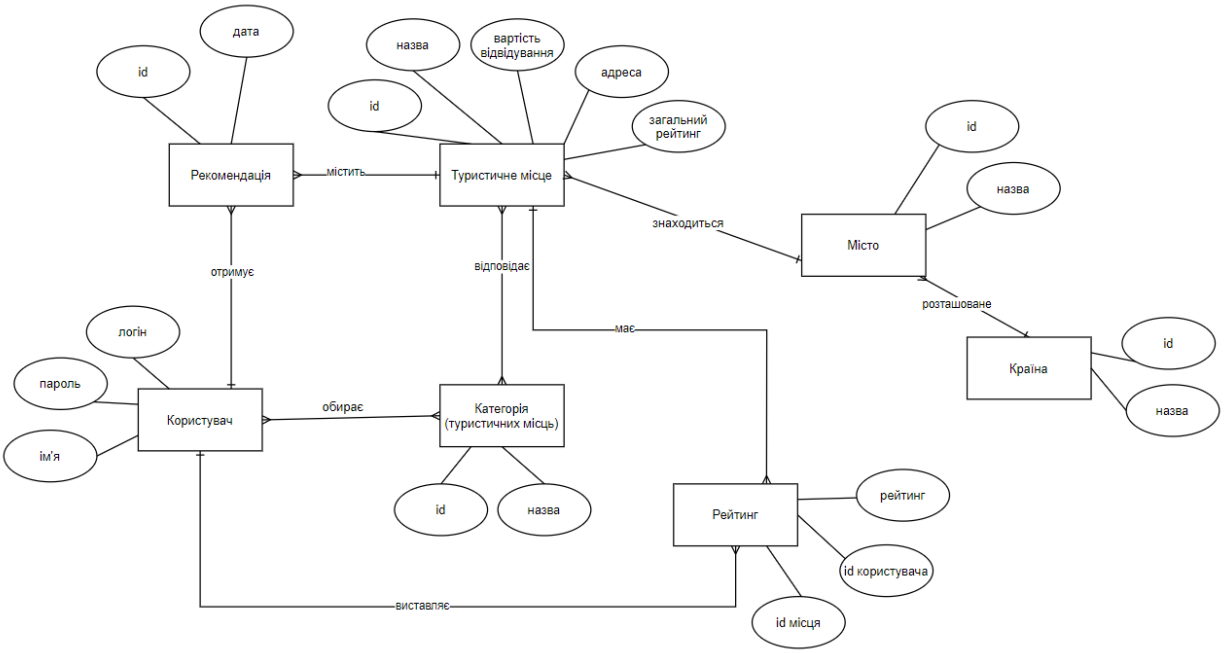



Рисунок 2.9 – Концептуальна модель бази даних

Наступною є логічна модель (рис. 2.10). Вона також, як і концептуальна, описує дані на досить абстрактному рівні, але все ж більше наближеному до реальних баз даних. Також, на відміну від попередньої моделі, ми вже вказуємо первинні ключі, якими в даному випадку є поля «id», і для користувача («User») – login, та зовнішні ключі, які пов'язують таблиці між собою.

Також з'являються додаткові таблиці, які розділяють зв'язки багато до багатьох. Всього таких буде дві – «UserCategory» та «CategoryPlace».

Першою ми розділяємо зв’язок між таблицями «User» («Користувач») та «Category» («Категорія»). Тобто між основними таблицями до проміжної встановлюємо зв’язки один до багатьох. Другою додатковою таблицею ми розділяємо зв’язок таблиць «Category» та «TourPlace» («Туристичне місце»).

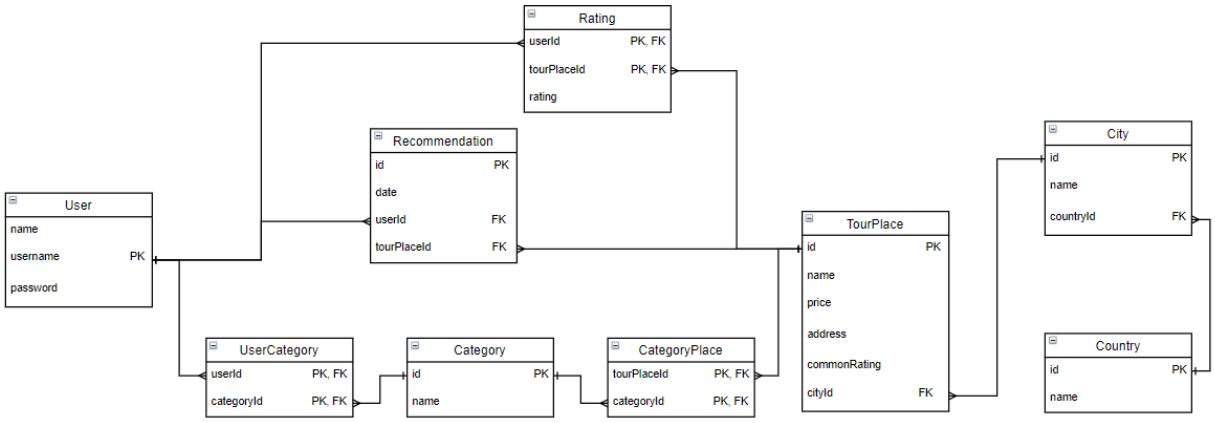



Рисунок 2.10 – Логічна модель бази даних

Останньою є фізична модель бази даних (рис. 2.11), розроблена на основі даталогічної. Вона вже відображає, як дані насправді зберігаються в базі даних. На ній вже можна побачити детальніші дані про атрибути, а саме – тип даних та обмеження для деяких з них. Опишемо їх у таблиці 2.2.

*Таблиця 2.2 Опис атрибутів таблиць*

| **№ п/п** | **Назва елемента даних** | **Тип елемента даних** | **Обов’язкове значення** | **Обмеження** | **Ключ** |
|---|---|---|---|---|---|
| **Таблиця User «Користувач»** | | | | | |
| 1 | username | Символьний | так | 100 | ПК |
| 2 | password | Символьний | так | 20 | |
| 3 | name | Символьний | так | 20 | |
| **Таблиця Category «Категорія»** | | | | | |
| 1 | id | Ціле число | так | | ПК |
| 2 | name | Символьний | так | 50 | |
| **Таблиця Recommendation «Рекомендація»** | | | | | |
| 1 | id | Ціле число | так | | ПК |

*Продовження таблиці 2.2*

| № п/п | Назва елемента даних | Тип елемента даних | Обов'язкове значення | Обмеження | Ключ |
|---|---|---|---|---|---|
| 2 | date | Дата | так | не раніше, ніж дата реєстрації користувача | |
| 3 | userId | Ціле число | так | 100 | ЗК |
| 4 | tourPlaceId | Ціле число | так | | ЗК |
| **Таблиця TourPlace «Туристичне місце»** | | | | | |
| 1 | id | Ціле число | так | | ПК |
| 2 | name | Символьний | так | 100 | |
| 3 | price | Ціле число | | | |
| 4 | address | Символьний | так | 150 | |
| 5 | townId | Ціле число | так | | ЗК |
| 5 | commonRating | Ціле число | так | Від 1 до 5, де 1 – зовсім не сподобалось, 5- дуже сподобалось | |
| **Таблиця City «Місто»** | | | | | |
| 1 | id | Ціле число | так | | ПК |
| 2 | name | Символьний | так | 50 | |
| 3 | countryId | Ціле число | так | | ЗК |
| **Таблиця Country «Країна»** | | | | | |
| 1 | id | Ціле число | так | | ПК |
| 2 | name | Символьний | так | 50 | |
| **Таблиця UserCategory** | | | | | |
| 1 | userId | Ціле число | так | | ПК, ЗК |
| 2 | categoryId | Ціле число | так | | ПК, ЗК |
| **Таблиця CategoryPlace** | | | | | |
| 1 | tourPlaceId | Ціле число | так | | ПК, ЗК |
| 2 | categoryId | Ціле число | так | | ПК, ЗК |
| **Таблиця Rating «Рейтинг»** | | | | | |
| 1 | userId | Символьний | так | 50 | ПК, ЗК |

*Продовження таблиці 2.2*

| № п/п | Назва елемента даних | Тип елемента даних | Обов'язкове значення | Обмеження | Ключ |
|---|---|---|---|---|---|
| 2 | tourPlaceId | Ціле число | так | | ПК, ЗК |
| 3 | rating | Ціле число | так | Від 1 до 5, де 1 – зовсім не сподобалось, 5- дуже сподобалось | |

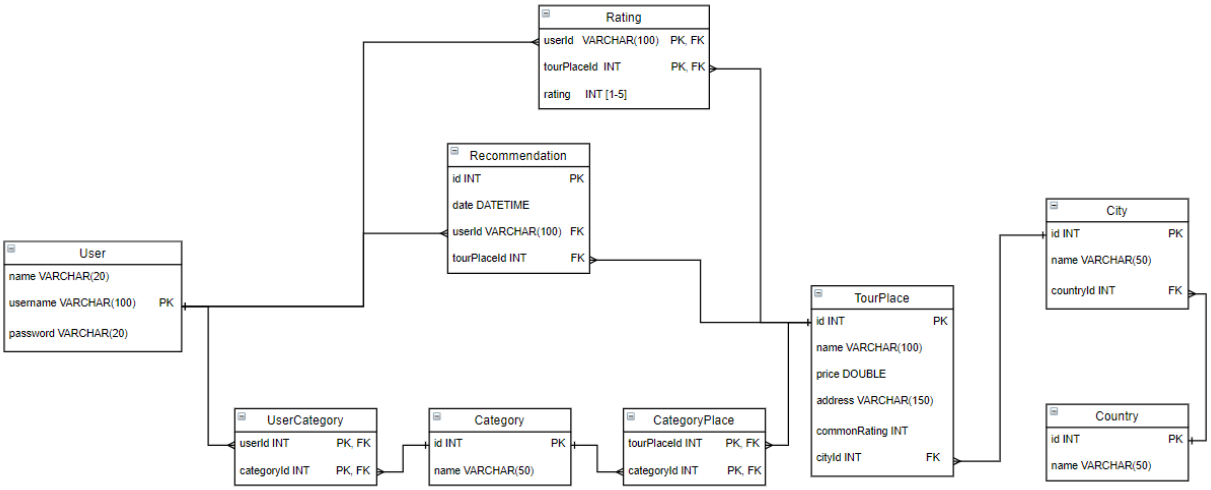



Рисунок 2.11 – Фізична модель бази даних

## 2.4 Висновки до другого розділу

В результаті проведення етапу розробки туристичної рекомендаційної системи було спроектовано систему для подальшої її реалізації.

Було проведено функціональний аналіз та побудовано дерево функцій, яке відображає бізнес-процеси інформаційної системи. Також було розроблено діаграми Event-Driven Process Chain та Value Added Chain Diagram для функціонування мобільного додатку. На даних схемах було зображено та описано логіку роботи програми.

Також було представлено діаграми IDEF0 "ЯК БУДЕ", які показують процеси видачі рекомендацій, виконувані з використанням даної туристичної системи.

Було сформовано архітектуру системи для відображення її внутрішніх компонентів та їх взаємозв'язок.

Було проведено аналіз методів математичного забезпечення системи та обрано алгоритм для розрахування рекомендацій туристичних місць для користувачів.

Крім цього, було розроблено інформаційне забезпечення додатку, а саме – діаграми потоків даних DFD, відповідні схеми бази даних та їх детальний опис.

# РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РЕКОМЕНДАЦІЙНОГО ТУРИСТИЧНОГО МОБІЛЬНОГО ДОДАТКУ

## 3.1 Обґрунтування вибору програмних засобів

Для розробки застосунку було використано ряд інструментів та методів, за допомогою яких створено інформаційну систему з відповідною базою даних та користувацьким інтерфейсом.

Ionic – це SDK для створення гібридних мобільних застосунків: набір CSS та JS компонентів, створений на основі AngularJS, SASS і Apache Cordova.

Гібридний додаток – це браузер на весь екран мобільного пристрою. Доступ до ресурсів пристрою здійснюється за допомогою cordova-плагінів, а власне Ionic – це інтеграція Cordova та AngularJS з bootstrap-схожою розміткою і набір стандартних ionic-компонентів (HTML, CSS(SASS), JS). Стандартні ionic-компоненти дозволяє створювати додатки з елементами: заголовками, табами, кнопками. [11].

Angular — це платформа дизайну додатків і платформа для створення ефективних і складних односторінкових програм. Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі [12].

TypeScript - це типізована надмножина JavaScript, призначена для виявлення помилок на етапі компіляції. Програма на TypeScript компілюється у простий код JavaScript, який виконується у будь-якому браузері. Розробляється мова програмування як проект з відкритими вихідними текстами, тому використовувати його може будь-хто охочий безкоштовно і без особливих обмежень [13].

MongoDB – це документоорієнтована система управління базами даних із відкритим вихідним кодом, яка не потребує опису схеми таблиць.

MongoDB класифікується як NoSQL і використовує JSON-подібні документи і схему бази даних. [14].

PHP (Hypertext Preprocessor (Препроцесор гіпертексту)) - це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Перевагою PHP є надання web-розробникам можливості швидкого створення динамічних web-сторінок. Значною відзнакою PHP від якого-небудь коду, що виконується на стороні клієнта, наприклад, JavaScript, є те, що PHP-скрипти виконуються на стороні сервера [15].

### 3.2 Структура програмного забезпечення

Надалі розглянемо структуру розробленого програмного забезпечення туристичної рекомендаційної системи.

Для візуального розділення сторінок застосунку та сервісів для відповідних обчислень їх було зображено різними способами:

- сторінки відображені трьома фрагментами розширень .ts, .html та .css;
- сервіси відображено тільки розширенням .ts.

Побудована структура зображена на рис. 3.1.

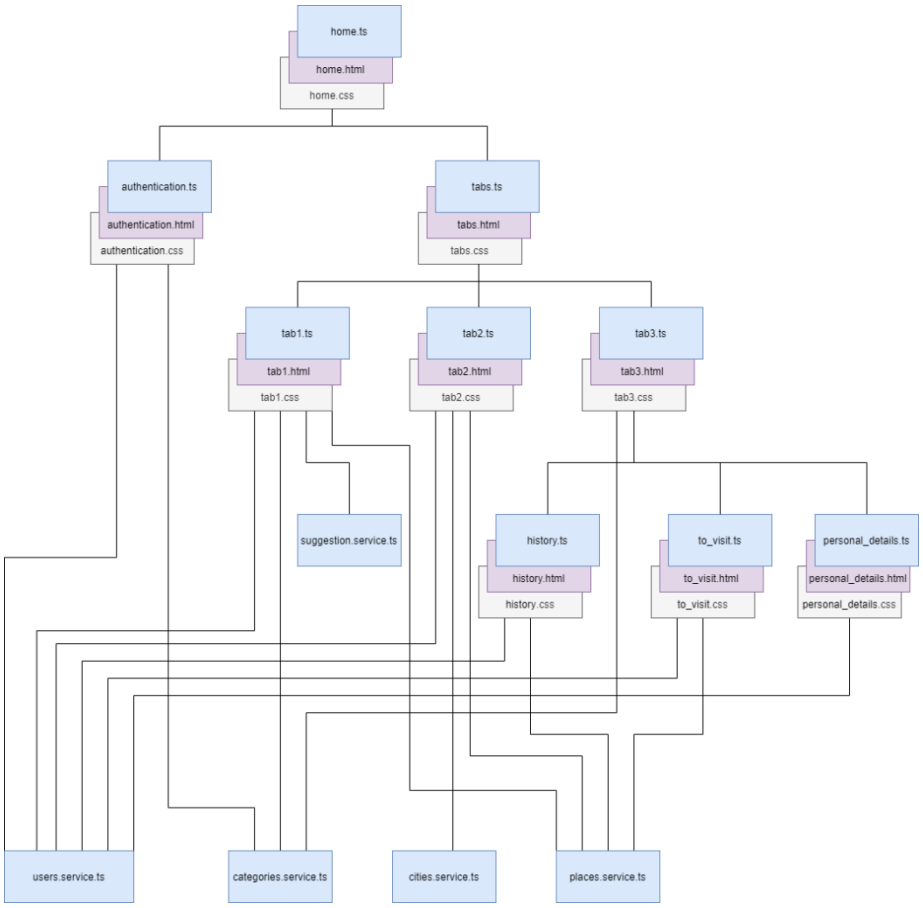



Рисунок 3.1 – Структура програмного забезпечення

Опишемо детальніше специфікацію програмних модулів (табл. 3.1). Оскільки для виконання роботи було обрано фреймворк Ionic, то відповідно розглянемо опис файлів з розширенням .ts.

*Таблиця 3.1 Специфікація програмних модулів*

| Модуль | Опис |
|---|---|
| home.ts | Початковий модуль, який відображає привітання користувача та дозволяє обрати вид доступу.<br>Вхідна інформація:<br>-<br>Вихідна інформація:<br>Вид доступу, а саме гостьовий вхід, вхід вже існуючого користувача, або реєєстрація. |
| authentication.ts | Модуль аутентифікації користувача, тобто вхід вже існуючого користувача або реєстрація нового.<br>Вхідна інформація:<br>Дані користувача, а саме:<br>- при вході: логін та пароль |

| | |
|---|---|
| | - при реєстрації: ім’я, логін, пароль, улюблені категорії<br>Вихідна інформація:<br>Зареєстрований та залогінений користувач, його дані |
| tabs.ts | Модуль переходу між сторінками основної частини програми.<br>Вхідна інформація:<br>Вид доступу, бажана сторінка<br>Вихідна інформація:<br>Відповідний інтерфейс сторінок |
| tab1.ts | Модуль першої сторінки (таби) основного меню (частини програми) для відображення рекомендацій користувача.<br>Вхідна інформація:<br>Вид доступу, дані про користувача, якщо він увійшов в свій акаунт<br>Вихідна інформація:<br>- якщо користувач не залогінений, то відображення повідомлення про неможливість розрахувати рекомендації.<br>- якщо користувач залогінений, то відображення рекомендацій за обраними категоріями та розрахованими на основі історії відвіданих місць користувача |

*Продовження таблиці 3.1*

| Модуль | Опис |
|---|---|
| tab2.ts | Модуль другої сторінки (таби) основного меню для відображення усіх можливих місць та їх пошуку за містом або ключовим словом.<br>Вхідна інформація:<br>Вид доступу<br>Вихідна інформація:<br>Відображення усіх місць з бази даних, або за критерієм пошуку<br>Якщо користучач залогінений – можливість зміни списків історії відвіданих місць та бажаних до відвідування |
| tab3.ts | Модуль третьої сторінки (таби) основного меню для відображення інформації про користувача та навігації до його списків.<br>Вхідна інформація:<br>Вид доступу<br>Вихідна інформація:<br>- Якщо користучач залогінений - дані про користувача<br>- Якщо користувач не залогінений – повідомлення про неможливість відобразити дані та пропозиція увійти або зареєструватись |
| history.ts | Модуль сторінки з історією відвіданих місць користувачем.<br>Вхідна інформація:<br>Дані про користувача<br>Вихідна інформація:<br>Список місць, які користувач відмітив відвіданими та коротка інформація про них. |
| to_visit.ts | Модуль сторінки з бажаними до відвідування місцями користувача.<br>Вхідна інформація:<br>Дані про користувача<br>Вихідна інформація:<br>Список місць, які користувач позначив бажаними до відвідування. |

*Продовження таблиці 3.1*

| Модуль | Опис |
|---|---|
| personal_details.ts | Вхідна інформація:<br>Дані про користувача<br>Вихідна інформація:<br>Змінені дані про користувача |
| users.service.ts | Модуль сервісу для відправлення запитів до бази даних користувачів.<br>Вхідна інформація:<br>Дані про користувача, тип запиту<br>Вихідна інформація:<br>Результати запитів |
| categories.service.ts | Модуль сервісу для відправлення запитів до бази даних категорій.<br>Вхідна інформація:<br>Дані про користувача (в залежності від запиту), тип запиту<br>Вихідна інформація:<br>Результати запитів |
| cities.service.ts | Модуль сервісу для відправлення запитів до бази даних міст.<br>Вхідна інформація:<br>тип запиту<br>Вихідна інформація:<br>Результати запитів |
| places.service.ts | Модуль сервісу для відправлення запитів до бази даних туристичних місць.<br>Вхідна інформація:<br>Дані про користувача (в залежності від запиту), критерій пошуку (в залежності від запиту), тип запиту<br>Вихідна інформація:<br>Результати запитів |
| suggestion.service.ts | Модуль розрахунку рекомендацій туристичних місць для користувача.<br>Вхідна інформація:<br>Туристичні місця, їх рейтинги, айді користувача<br>Вихідна інформація:<br>Результати розрахунків, а саме рекомендації туристичних місць для обраного користувача |

Представимо переходи між модулями програми у вигляді графу переходів (рис. 3.2). Вершини графу зображують модулі системи, а стрілки вказують, куди користувач може переходити із цих модулів.

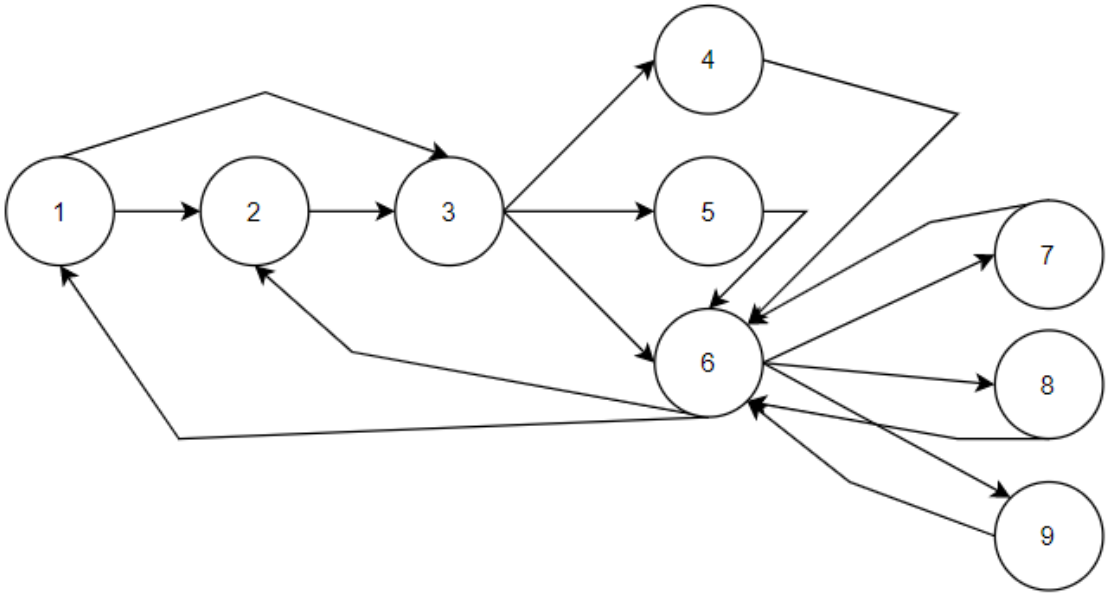



Рисунок 3.2 – Граф переходів

Опишемо кожну вершину графу у вигляді таблиці (табл. 3.2).

*Таблиця 3.2 Опис графу переходів*

| Номер | Опис |
|---|---|
| 1 | Сторінка home.html, яка пов’язана з файлом home.ts та відображає екран привітання та вибору доступу. |
| 2 | Сторінка authentication.html, яка пов’язана з файлом authentication.ts та відображає процес логіну або реєстрації користувача. |
| 3 | Сторінка tabs.html, яка пов’язана з файлом tabs.ts та відображає меню навігації між сторінками (табами). |
| 4 | Сторінка tab1.html, яка пов’язана з файлом tab1.ts та відображає рекомендації користувача. |
| 5 | Сторінка tab2.html, яка пов’язана з файлом tab2.ts та відображає список всіх туристичних місць. |

*Продовження таблиці 3.2*

| Номер | Опис |
|---|---|
| 6 | Сторінка tab3.html, яка пов’язана з файлом tab3.ts та відображає інформації про користувача. |
| 7 | Сторінка personal_details.html, яка пов’язана з файлом personal_details.ts та відображає дані про користувача, які можна змінити. |
| 8 | Сторінка history.html, яка пов’язана з файлом history.ts та відображає список відвіданих користувачем туристичних місць. |
| 9 | Сторінка to_visit.html, яка пов’язана з файлом to_visit.ts та відображає список місць, бажаних користувачем до відвідування. |

### 3.3 Керівництво користувача

Наступним етапом опишемо керівництво користувача з прикладами з готової програми.

Коли користувач відкриває туристичний рекомендаційний додаток, першою він бачить головну сторінку (рис. 3.3). На ній він може використати форму для логіну, щоб увійти у вже створений акаунт. Форма має два поля для обов’язкового заповнення – логін та пароль, а також кнопку для безпосереднього входу.

Також користувач може обрати функцію гостьового входу або перейти на процес реєстрації.

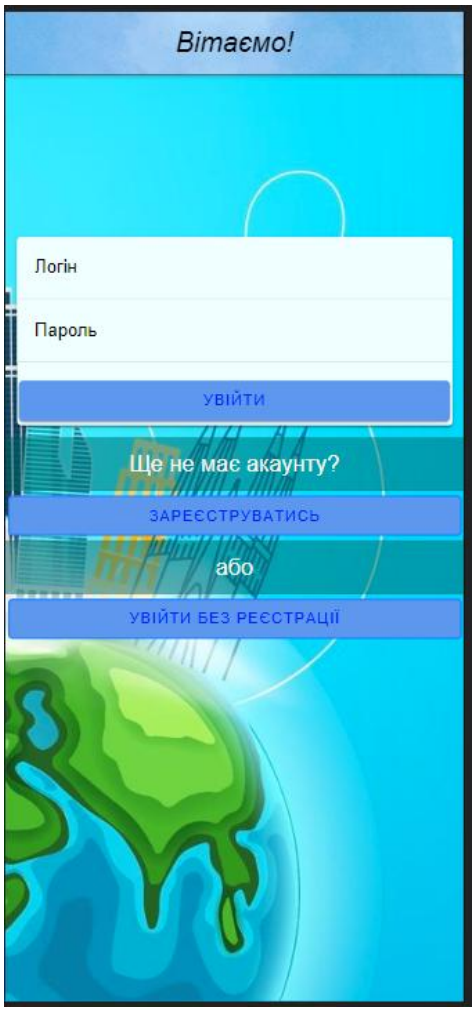



Рисунок 3.3 – Головна сторінка

Якщо користувач обирає перехід на процес реєстрації, то спочатку він потрапляє на перший етап для заповнення основних даних (рис. 3.4). Такими даними є його ім’я, логін, який є електронною поштою, та пароль.

Якщо користувач заповнює дані невірно або взагалі залишає поля пустими та намагається перейти на наступний крок, то відповідні поля підсвічуються червоним кольором та він бачить повідомлення про помилку заповнення.

Якщо ж усі дані заповненні вірно, то користувач переходить на другу сторінку реєстрації (рис. 3.5). На другому кроці йому потрібно обрати хоча б дві улюблені категорії туристичних місць. Тільки після обрання він має змогу завершити реєстрацію та увійти в основну частину додатку.

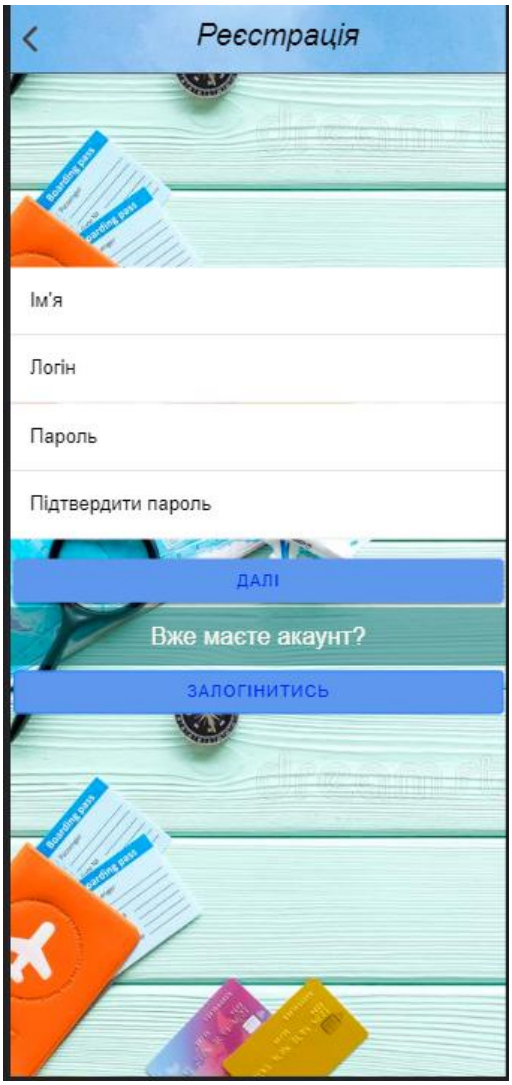



Рисунок 3.4 – Перша сторінка реєстрації

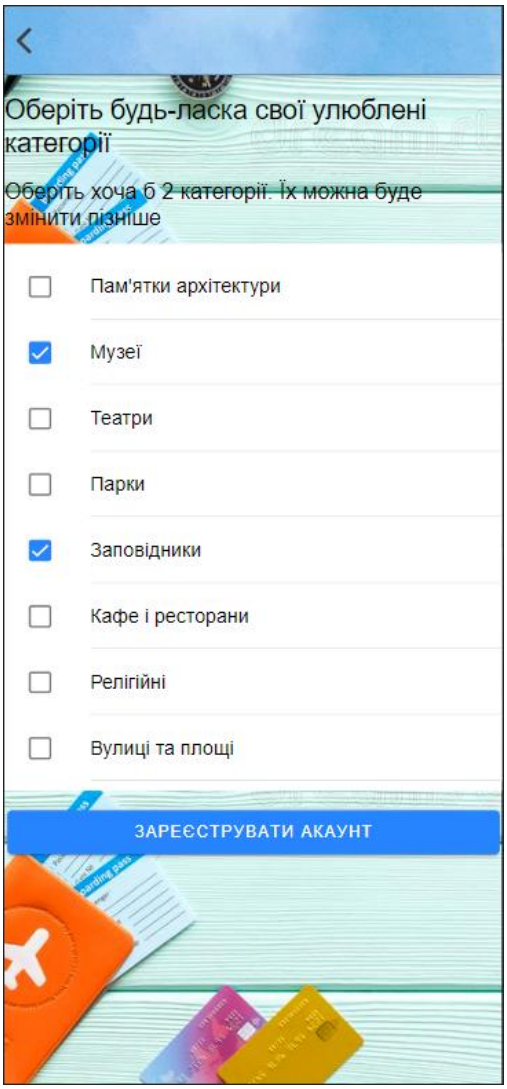



Рисунок 3.5 – Друга сторінка реєстрації

Якщо користувач обирає функцію гостьового входу, то він потрапляє в головну частину програми, але з обмеженими можливостями.

Першою він бачить сторінку з усіма наявними в базі даних туристичними місцями (рис. 3.7). На даній сторінці він може побачити інформацію про місця, а саме – їх назву, зображення, адресу та ціну відвідування. Також він може шукати місця за ключовими словами за допомогою поля пошуку. Наявна і функція сортування місць за містом, яке можна обрати у дропдауні зверху сторінки.

У нижньому меню користувач може побачити та перейти на сторінки отримання рекомендацій та профілю. На першій (рис. 3.6) він отримує

повідомлення про недоступність даної функції. Це пов’язано із відсутністю можливості додавати місця в історію або у бажані до відвідування місця для гостя.

На сторінці профілю (рис. 3.8) користувач також отримує повідомлення та буде запропоновано увійти в свій акаунт, якщо він вже існує або зареєструватись. Відповідні сторінки для логіну та реєстрації були описані вище.

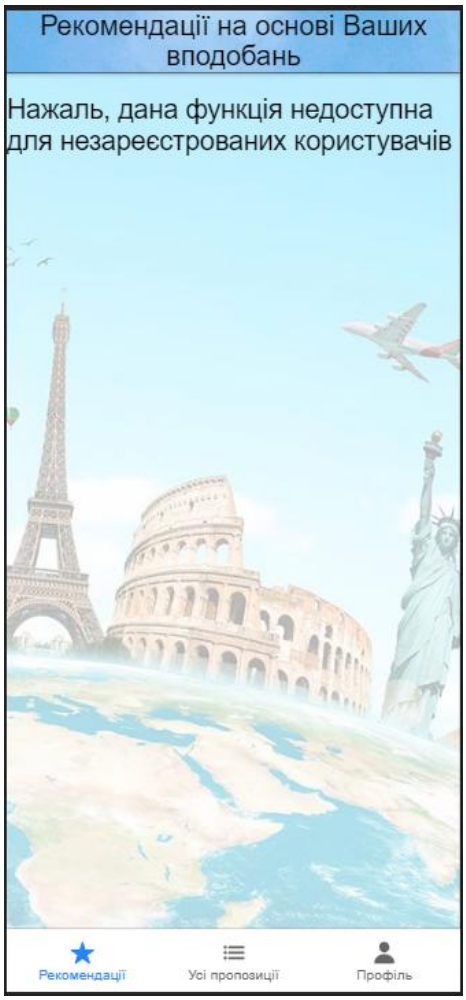



Рисунок 3.6 - Рекомендації

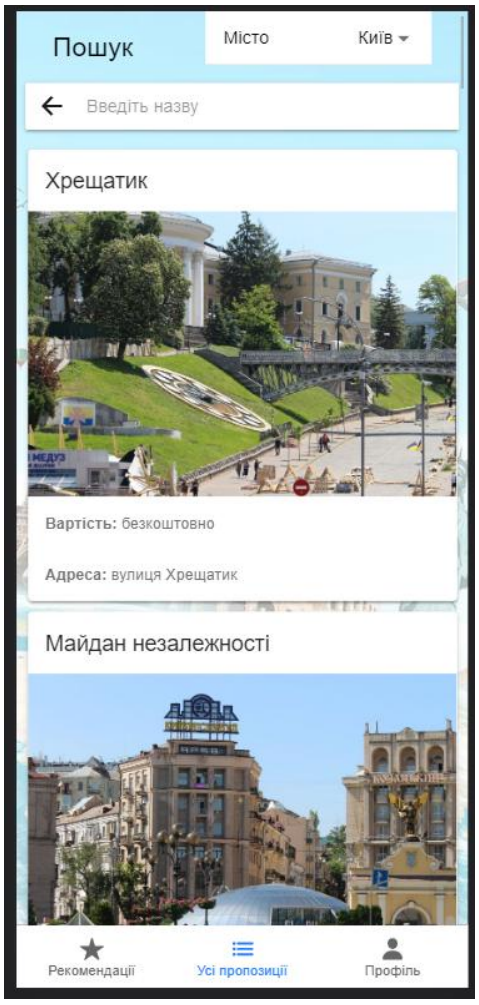



Рисунок 3.7– Усі місця

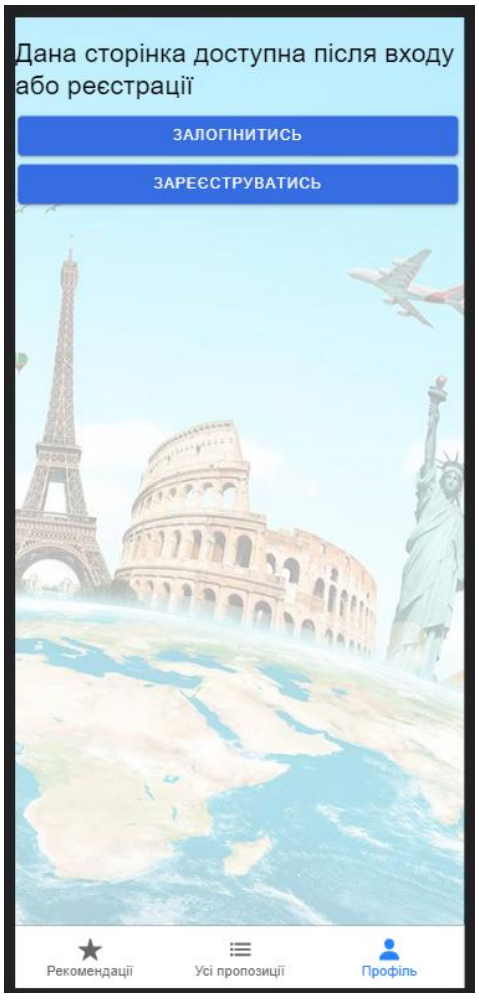



Рисунок 3.8 - Профіль

У випадку, якщо користувач увійшов у свій акаунт, то він переходить на сторінку з рекомендаціями (рис. 3.9). Зверху сторінки він бачить категорії, які він обирав раніше. Якщо нажати на відповідну назву, то відкриється список місць з цієї категорії. Під категоріями користувач може побачити рекомендації (тобто туристичні місця), які були обраховані за допомогою алгоритму. Вони відсортовані від самого кращого до найгіршого. Дані місця

відображаються тільки у випадку, якщо користувач вже додавав місця до списку відвіданих та оцінював їх. Також алгоритм може порівняти місця тільки якщо їх оцінило як мінімум двоє користувачів. Інакше вони враховуватись не будуть.

На другій сторінці «Усі пропозиції» (рис. 3.12) відображено все те ж саме, що було описано для гостя, але також користувач бачить іконки на картках з туристичними місцями для додавання їх до відповідних списків. Коли місце додається до списку бажаних до відвідування, то відповідна іконка підсвічується. Коли до списку історії, тобто вже відвіданих – то з'являється вікно (рис. 3.13), на якому програма просить користувача оцінити відповідне місце за шкалою від 1 до 5, яка описана словами: дуже сподобалось, сподобалось, досить цікаво, не дуже або зовсім не сподобалось. Тільки після оцінки місце додається до списку та також підсвічується потрібна іконка.

На третій сторінці, а саме сторінці профілю (рис. 3.10) користувач вже бачить власну інформацію, тобто привітання з вказаним ім'ям та улюблені категорії, які він обирав при реєстрації. Також на даній сторінці користувач може обрати одну з трьох функцій – перейти на сторінку редагування даних, перейти до історії вже відвіданих місць, перейти до списку бажаних до відвідування місць або вийти з акаунту.

Якщо користувач обирає останню функцію, то він переходить на головну сторінку (рис. 3.3).

Якщо він хоче відредагувати свої дані, то він натискає відповідну кнопку та переходить на сторінку редагування (рис. 3.11). На ній він бачить поля для зміни імені та паролю, а також чекбокси для обрання улюблених категорій. Коли користувач заповнить вірно всю бажану інформацію, то він натискає кнопку «Підтвердити» та повертається на попередню сторінку.

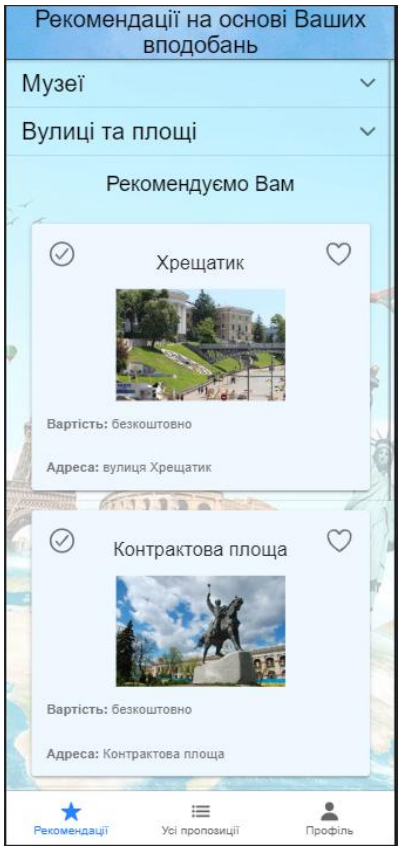



Рисунок 3.9- Рекомендації

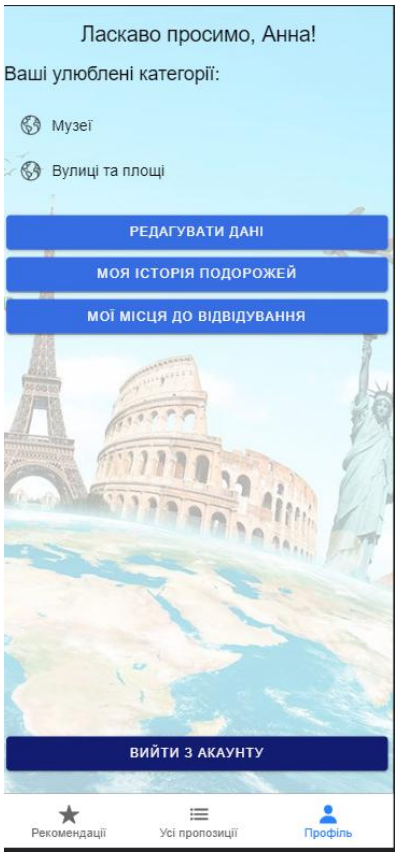



Рисунок 3.10 - Профіль

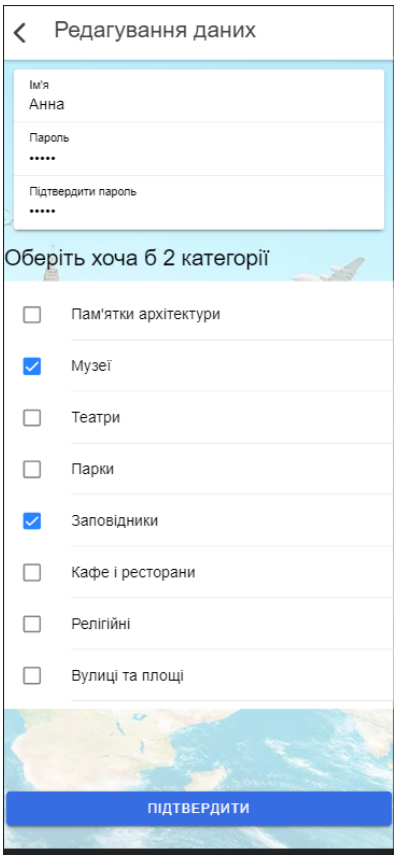
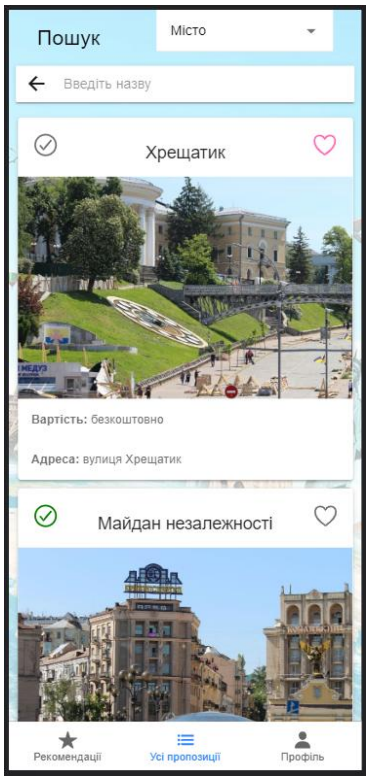



Рисунок 3.11 – Редагування профілю



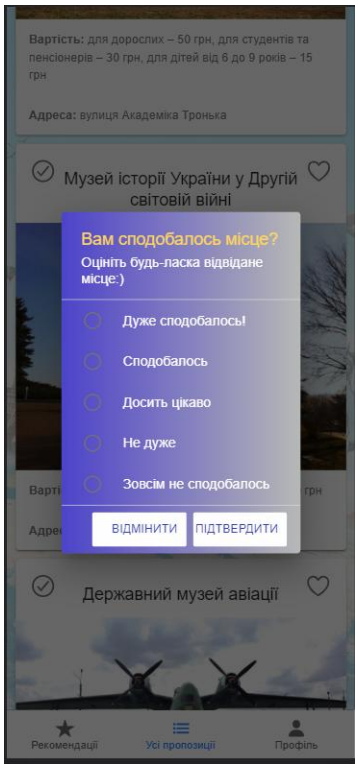



Рисунок 3.12 – Усі місця Рисунок 3.13 – Додавання до історії

Якщо користувач бажає переглянути туристичні місця, які він додав до списку бажаних до відвідування, то він переходить на відповідну сторінку за допомогою кнопки «Мої місця до відвідування» (рис. 3.15). На сторінці він бачить картки з описом місць, а також дві функції для кожного з них – видалити або позначити відвіданним. У другому випадку, як і на попередніх сторінках, з’являється вікно для проставлення рейтингу. Після завершення дії місце прибирається із даного списку та додається до історії.

За допомогою кнопки «Моя історія подорожей» користувач переходить на сторінку зі списком місць, які він вже відвідав (рис. 3.14). Для кожного місця прибирається поле ціни та додається інформація про рейтинг, або оцінку, яку проставив користувач. Також є можливість видалити місце з даного списку.

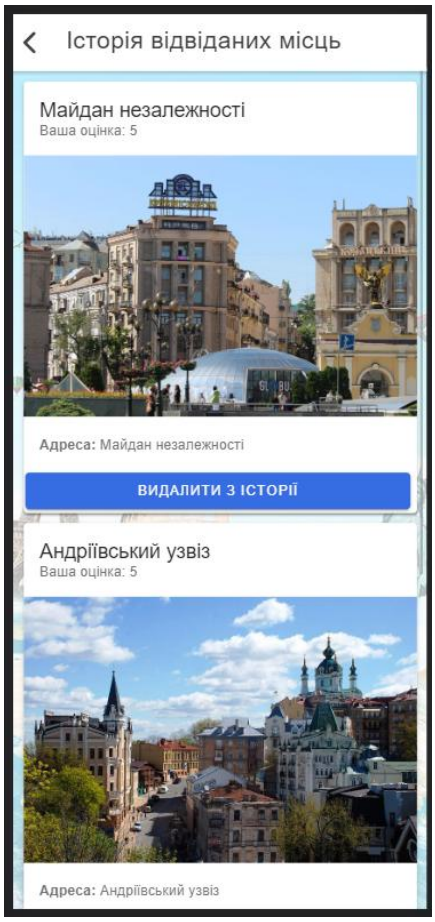



Рисунок 3.14 – Історія відвіданих місць

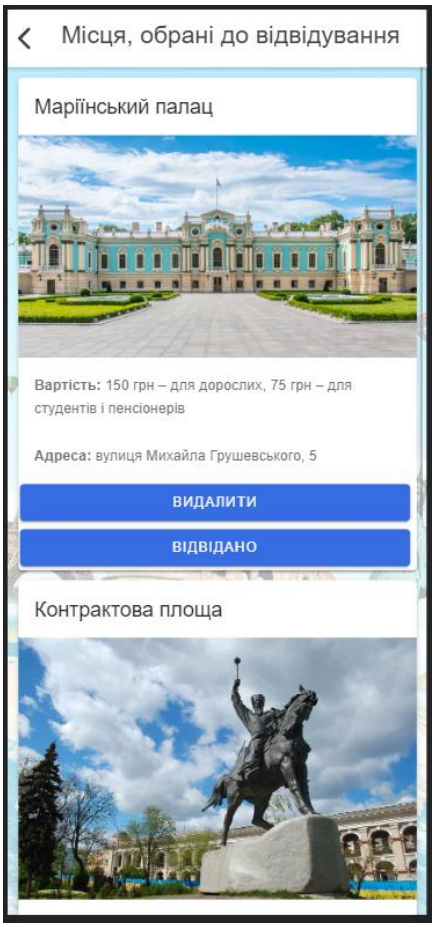



Рисунок 3.15 – Бажані місця до відвідування

### 3.4 Огляд процесу тестування

Останнім кроком оглянемо процес тестування рекомендаційного туристичного додатку. Для цього напишемо тести у вигляді так званих тест кейсів (табл. 3.3).

*Таблиця 3.3 Опис тестів*

| id | Пріоритет | Модуль | Кроки | Очікувані результати |
|---|---|---|---|---|
| 1 | найвищий | home.ts | **Можливість увійти в свій акаунт**<br>Кроки:<br>1. Ввести правильний логін<br>2. Ввести правильний пароль<br>3. Натиснути кнопку «Увійти» | 1. Логін відображено у полі логіну<br>2. Пароль відображено як крапки у полі паролю<br>3. Користувач переходить на сторінку з рекомендаціями |
| 2 | найвищий | home.ts | **Можливість зареєструватись**<br>Кроки:<br>1. Натиснути кнопку «Зареєструватись»<br>2. Заповнити всі обов’язкові поля<br>3. Натиснути кнопку «Далі»<br>4. Обрати хоча б дві улюблені категорії<br>5. Натиснути кнопку «Зареєструватись» | 1. Користувач переходить на першу сторінку для реєстрації<br>2. Всі обов’язкові поля заповнені<br>3. Користувач переходить на другу сторінку для реєстрації<br>4. Категорії обрано<br>5. Користувач переходить на сторінку з рекомендаціями |
| 3 | високий | home.ts | **Перевірка невірно заповненого паролю для входу**<br>Кроки:<br>1. Ввести правильний логін<br>2. Ввести неправильний пароль<br>3. Натиснути кнопку «Увійти» | 1. Логін відображено у полі логіну<br>2. Пароль відображено як крапки у полі паролю<br>3. Користувач залишається на гловній сторінці та бачить повідомлення про помилку |

*Продовження таблиці 3.3*

| id | Пріоритет | Модуль | Кроки | Очікувані результати |
|---|---|---|---|---|
| 4 | найвищий | tab1.ts | **Перегляд місць за обраними категоріями**<br>Кроки:<br>1. Увійти як зареєстрований користувач<br>2. Розгорнути категорію | 1. Користувач переходить на сторінку з рекомендаціями та бачить туристичні місця за обраними категоріями у вигляді згорнутих списків<br>2. Користувач бачить список місць обраної категорії |
| 5 | найвищий | tab1.ts | **Отримання рекомендацій туристичних місць**<br>Кроки:<br>1. Увійти як зареєстрований користувач | 1. Користувач переходить на сторінку з рекомендаціями та бачить рекомендації туристичних місць під категоріями, серед яких нема вже відвіданих |
| 6 | високий | tab2.ts | **Додавання місць до історії відвіданих**<br>Кроки:<br>1. Увійти як зареєстровний користувач<br>2. Перейти на сторінку з усіма місцями<br>3. Натиснути ліву іконку для відповідного місця<br>4. Оцінити місце | 1. Користувач переходить на сторінку з рекомендаціями<br>2. Користувач переходить на сторінку з усіма місцями<br>3. З’являється вікно для виставлення оцінки (рейтингу) місця<br>4. Місце додане до списку відвіданих, та іконка підсвічується зеленим кольором |

### 3.5 Висновки до третього розділу

Для безпосередньої реалізації рекомендаційного туристичного мобільного застосунку було проведено ряд робіт.

На початку було обрано та обгрунтовано програмні інструменти, які використовувались під час розробки додатку.

Було визначено структуру програмного застосунку, побудовано її у вигляді схеми та описано специфікацію програмних модулів у вигляді таблиці. Також було представлено граф переходів між цими модулями та описано кожну вершину графу.

Крім цього, було описано керівництво користувача, використовуючи вже створений додаток. Для цього було зроблено «скріншоти» відповідних сторінок та описано кожну з них.

Заключним етапом був огляд процесу тестування створеного рекомендаційного туристичного застосунку, а саме – було створено ряд тест кейсів для перевірки функціоналу. Кожен тест було виконано вручну та отримано позитивні результати.

## Висновок

Туризм завжди був, є і залишається цікавою сферою діяльності. Люди цікавляться світом, у якому живуть, хочуть побачити нові місця, а отже дана тематика буде актуальною і надалі.

В ході виконання даної дипломної роботи було проведено аналіз існуючих сучасних рішень для самостійного туризму та сформовано відповідні вимоги до мобільного туристичного рекомендаційного застосунку. Було проаналізовано математичні методи розрахування та отримання рекомендацій користувачами, один з яких було вирішено використовувати під час реалізації.

Для розробки даного застосунку було спроектовано базу даних, дерево функцій, архітектуру додатку, його структуру.

В результаті реалізації було створено рекомендаційний туристичний застосунок з використанням таких інструментів, як MongoDB, Ionic, Angular, TypeScript та PHP. Також додаток було протестовано відповідно до описаних тест кейсів.

Розроблений додаток можна покращувати та доповнювати новими функціями у майбутньому, що дозволить зробити його ще більш зручним та корисним для майбутніх туристів.

## Список використаних джерел


1. R. Francesco, R. Lior, S. Bracha, K. B. Paul. Recommender Systems Handbook. Dordrecht:Springer, 2015. –1009 p
2. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems: An Introduction. Cambridge University Press, 2010 – 353 p.
3. Dareddy, M. Challenges in Recommender Systems for Tourism / M. Dareddy // Workshop on Recommenders in Tourism held in conjunction with the 10th ACM Conference on Recommender Systems (RecSys) (Boston, September 15, 2016 y.). Boston, 2016: P. [1–3]
4. Puhretmair, F. Extended Decision Making in Tourism Information Systems / F. Puhretmair, H. Rumetshofer, E. Schaumlechner // EC-Web 2002: E-Commerce and Web Technologies : Third International Conference, EC-Web 2002: proceedings (Aix-en-Provence, September 2–6, 2002 y.). Aix-en-Provence, 2002, Vol. 2455: P. 57–66
5. Damianos G. Mobile recommender systems in tourism / Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou // Journal of Network and Computer Applications., March 2014, Volume 39: P. 319-333
6. В. В. Пасічник, В. В. Савчук ІНТЕЛЕКТУАЛЬНА СИСТЕМА «МОБІЛЬНИЙ ІНФОРМАЦІЙНИЙ АСИСТЕНТ ТУРИСТА»: ФУНКЦІОНАЛЬНІ ТА ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ. SISN. 2015; Випуск 832, Номер 3: сс. 225 – 241
7. Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas and Grammati Pantziou Mobile Recommender Systems in Tourism. Journal of Network and Computer Applications, March 2014, 39(1): p. 319–333
8. Про Tripadvisor [Електронний ресурс] – Режим доступу: https://tripadvisor.mediaroom.com/us-about-us
9. Є.В. Мелешко, «МЕТОДОЛОГІЯ ЗАБЕЗПЕЧЕННЯ СТІЙКОСТІ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДО ДЕСТАБІЛІЗУЮЧИХ

ФАКТОРІВ У КОМП'ЮТЕРНИХ МЕРЕЖАХ», Національний технічний університет «Харківський політехнічний інститут» Міністерство освіти і науки України Черкаський державний технологічний університет, Черкаси, УДК 004.89+004.942, 2021

10. Item-to-Item Based Collaborative Filtering – Режим доступу: https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/
11. Особливості створення гібридних мобільних застосунків на Ionic – Режим доступу: https://codeguida.com/post/830
12. Introduction to the Angular Docs – Режим доступу: https://angular.io/docs
13. IT технології: TypeScript – Режим доступу: https://apeirondb.com/ua/blog/it-technologies-typescript
14. Найпопулярніші системи управління базами даних – Режим доступу: https://www.management.com.ua/partners/2021/02/21/5-najpopulyarnishih-sistem-upravlinnya-bazami-danih-2020-roku/
15. PHP – Режим доступу: http://programming.in.ua/web-design/allphp/30-about-php.html

## ДОДАТОК А

### Лістинг Front-end частини

**tab1.page.ts**

```
import {Component, OnInit} from '@angular/core';
import {CategoriesService} from "../services/categories.service";
import {PlacesService} from "../services/places.service";
import {UserService} from "../services/users.service";
import {AlertController} from "@ionic/angular";
import {SuggestionService} from "../services/suggestion.service";
import {VisitedPlace} from "../interfaces/visitedPlace";

@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss']
})

export class Tab1Page implements OnInit {

  isLoggedIn: boolean = false;
  categories: [{ _id: { $oid: string }, name: '' }];
  places: [[]] = [[]];
  visited = [];
  buttonsState = [];
  historyState = [];
  suggestionPlaces = [];
  users = [];

  constructor(private categoriesService: CategoriesService, private placesService:
PlacesService,
              private userService: UserService, public alertController: AlertController,
              private suggestionService: SuggestionService) {
  }

  ngOnInit() {
    this.isLoggedIn = !!(localStorage.getItem("email"));
    if (this.isLoggedIn) {
      let userId = localStorage.getItem("userId");

      this.categoriesService.getUserCategories(userId).subscribe((data: any) => {
        if (data == 'Failure') {
        } else {
          this.categories = data;
          this.categories.forEach(category =>
this.placesService.getPlacesByCategoryId(category._id.$oid)
            .subscribe((data: any) => {
              if (data == 'Failure') {
              } else {
                this.places.push(data);
```

```
            this.placesService.getPlacesHistoryByUserId(userId).subscribe((data: any) =>
{
              if (data == 'Failure') {
              } else {
                this.visited = data;
                this.places.forEach((category: [], index1) => {
                  if (index1 != 0) {
                    category.forEach((place: any, index2) => {
                      if (index2 === 0) {
                        this.historyState[index1] = [];
                      }
                      this.historyState[index1][index2] = this.visited.some((element: any) =>
element._id?.$oid === place._id?.$oid);
                    });
                  }
                });
              }

              this.getSuggestions(userId);
            });
            this.placesService.getPlacesToVisitByUserId({
              id: userId,
              tovisit: 'tovisit'
            }).subscribe((data: any) => {
              if (data == 'Failure') {
              } else {
                let tovisit: [] = data;
                this.places.forEach((category: [], index1) => {
                  if (index1 != 0) {
                    category.forEach((place: any, index2) => {
                      if (index2 === 0) {
                        this.buttonsState[index1] = [];
                      }
                      this.buttonsState[index1][index2] = tovisit.some((element: any) =>
element._id?.$oid === place._id?.$oid);
                    });
                  }
                });
              }
            });
          }
        }))
      }
    });

  }
}

activeCheckHistory(item, i, j) {
  this.presentAlert(item, i, j);
}
```

```
activeCheck(item, i, j) {
  let tovisitstr = localStorage.getItem("tovisit");
  let tovisit = tovisitstr.split(',');
  let visitedstr = localStorage.getItem("visited");
  let visited: [VisitedPlace] = JSON.parse(visitedstr);

  const index1 = tovisit.indexOf(item._id.$oid, 0);
  if (index1 < 0) {
    if (tovisit[0] === 'undefined' || tovisit[0] === '') {
      tovisit[0] = item._id.$oid;
    } else {
      tovisit.push(item._id.$oid)
    }
    const index = visited.map(function (e) {
      return e.id;
    }).indexOf(item._id.$oid);
    if (index > -1) {
      visited.splice(index, 1);
    }
  } else {
    tovisit.splice(index1, 1);
  }
  localStorage.setItem("tovisit", tovisit.toString());
  localStorage.setItem("visited", JSON.stringify(visited));
  this.userService.update({
    username: localStorage.getItem("email"),
    name: localStorage.getItem("name"),
    visited: JSON.stringify(visited),
    tovisit: tovisit.toString()
  }).subscribe((data: any) => {
    if (data == 'Failure') {
    } else {
      this.buttonsState[i][j] = !this.buttonsState[i][j];
    }
  });
}

async presentAlert(item, i, j) {
  const alert = await this.alertController.create({
    cssClass: 'basic-alert',
    header: 'Вам сподобалось місце?',
    subHeader: 'Оцініть будь-ласка відвідане місце:)',
    inputs: [
      {
        name: '5',
        type: 'radio',
        label: 'Дуже сподобалось!',
        value: '5'
      },
      {
        name: '4',
```

```
        type: 'radio',
        label: 'Сподобалось',
        value: '4'
      },
      {
        name: '3',
        type: 'radio',
        label: 'Досить цікаво',
        value: '3'
      },
      {
        name: '2',
        type: 'radio',
        label: 'Не дуже',
        value: '2'
      },
      {
        name: '1',
        type: 'radio',
        label: 'Зовсім не сподобалось',
        value: '1'
      }
    ],
    buttons: [
      {
        text: 'Відмінити',
        role: 'cancel',
        cssClass: 'secondary',
        handler: () => {
          let rate = undefined;
          this.checkHistory(rate, item, i, j);
        }
      }, {
        text: 'Підтвердити',
        handler: (alertData) => {
          this.checkHistory(alertData, item, i, j);
        }
      }
    ]
  });
  await alert.present();
}

checkHistory(value, item, i, j) {
  if (value != undefined) {
    let tovisitstr = localStorage.getItem("tovisit");
    let tovisit = tovisitstr.split(',');
    let visitedstr = localStorage.getItem("visited");
    let visited: [VisitedPlace] = JSON.parse(visitedstr);

    const index1 = visited.map(function (e) {
      return e.id;
```

```
    }).indexOf(item._id.$oid);
    if (index1 < 0) {
      if (visited[0] === undefined) {
        visited[0] = {id: item._id.$oid, rate: value};
      } else {
        visited.push({id: item._id.$oid, rate: value});
      }
      const index = tovisit.indexOf(item._id.$oid, 0);
      if (index > -1) {
        tovisit.splice(index, 1);
      }
    }
    localStorage.setItem("visited", JSON.stringify(visited));
    localStorage.setItem("tovisit", tovisit.toString());
    this.userService.update({
      username: localStorage.getItem("email"),
      name: localStorage.getItem("name"),
      visited: JSON.stringify(visited),
      tovisit: tovisit.toString()
    }).subscribe((data: any) => {
      if (data == 'Failure') {
      } else {
        this.historyState[i][j] = !this.historyState[i][j];
      }
    });
  }
}

private getSuggestions(userId) {
  this.userService.getUsers().subscribe((data: any) => {
    if (data == 'Failure') {
    } else {
      this.users = data;

      let tourPlaces = [];
      this.places.forEach((category: []) => {
        category.forEach((place: any) => {
          let arr = [];
          this.users.forEach(user => {
            let idOfUser = user._id.$oid;
            let visited: [VisitedPlace] = JSON.parse(user.visited);
            const index = visited.map(function (e: any) {
              return e.id;
            }).indexOf(place._id.$oid);
            if (index > -1) {
              arr.push({userId: idOfUser, rating: visited[index].rate});
            }
            let item = {id: place._id.$oid, usersRating: arr, similarityWithOther: []};
            tourPlaces.push(item);
          })
        })
      })
```

```
        let suggestions = [];
        // @ts-ignore
        let notVisited = this.places[1].filter(item => !this.visited.map(v =>
      v._id.$oid).includes(item._id.$oid));
        notVisited.forEach((visit: any) => {
          suggestions.push({
            id: visit._id.$oid,
            rate: this.suggestionService.makeSuggestion(tourPlaces, visit._id.$oid, userId)
          });
        })
        suggestions.sort((suggest1, suggest2) => suggest2.rate - suggest1.rate);
        for (let i = 0; i < suggestions.length; i++) {
            this.placesService.getPlaceById(suggestions[i].id).subscribe((data: any) => {
              if (data == 'Failure') {
              } else {
                this.suggestionPlaces[i] = data[0];
              }
            });
          }
      }
    });
  }
}
```

**suggestion.service.ts**

```
import {Injectable} from '@angular/core';

interface Film {
  id: string;
  name: string;
  usersRating: Array<{ userId: string, rating: number }>;
  similarityWithOther: Array<{ filmId: string, similarity: number }>;
  suggestRating?: number;
}

@Injectable({
  providedIn: 'root'
})
export class SuggestionService {

  constructor() {
  }

  public makeSuggestion(films: Film[], filmId: string, userId: string) {
    let currentFilm = this.findFilmById(filmId, films);
    if (currentFilm.suggestRating) {
      return currentFilm.suggestRating;
    }
    const evaluatedFilms = films.filter(film => film.usersRating.some(rate => rate.userId
  === userId));
```

```
    if (!evaluatedFilms.length) {
      return -1
    }

    films.forEach(film => {
      if (film.id !== filmId) {
        const similarity = this.makeSimilarity(this.findFilmById(filmId, films),
this.findFilmById(film.id, films));
        similarity !== -1 && currentFilm.similarityWithOther.push({filmId: film.id,
similarity: similarity});
      }
    })
    const ratings: number[] = [], similarities: number[] = [];
    currentFilm.similarityWithOther.forEach(similarity => {
      evaluatedFilms.forEach(evaluatedFilm => {
        if (similarity.filmId === evaluatedFilm.id) {
          ratings.push(evaluatedFilm.usersRating.find(rate => rate.userId ===
userId)!.rating);
          similarities.push(similarity.similarity);
        }
      })
    })
    let nominator = 0, denominator = 0
    for (let i = 0; i < ratings.length; i++) {
      nominator += ratings[i] * similarities[i];
      denominator += similarities[i];
    }
    currentFilm.suggestRating = nominator / denominator;
    return currentFilm.suggestRating;
  }

  private makeSimilarity(film1: Film, film2: Film) {
    const v1: number[] = [], v2: number[] = [], users = [];
    film1.usersRating.forEach(rate1 => {
      film2.usersRating.forEach(rate2 => {
        if (rate1.userId === rate2.userId) {
          users.push(rate1.userId);
          v1.push(rate1.rating);
          v2.push(rate2.rating);
        }
      })
    })

    if (users.length < 2) {
      return -1;
    }

    let nominator = 0, denominator: number[] = [0, 0]

    for (let i = 0; i < v1.length; i++) {
      nominator += v1[i] * v2[i];
      denominator[0] += v1[i] * v1[i];
```

```
      denominator[1] += v2[i] * v2[i];
    }
    denominator[0] = Math.sqrt(denominator[0]);
    denominator[1] = Math.sqrt(denominator[1]);
    const resultDenominator = denominator[0] * denominator[1];

    return nominator / resultDenominator;

  }

  private findFilmById(id: string, films: Film[]): Film {
    return films.find(film => film.id === id)!;
  }

}
```

## ДОДАТОК Б

### Лістинг Back-end частини

**users.php**

```
<?php
header('Access-Control-Allow-Origin: *');

header('Access-Control-Allow-Methods: GET, POST, PUT');

header("Access-Control-Allow-Headers: X-Requested-With, Content-Type");


if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["username"])) {
if (extension_loaded("mongodb")) {
try {
	$email = $_GET["username"];
	$password = $_GET["password"];

	$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");

	$filter_arr = ['username' => $email, 'password' => $password];
	$query = new MongoDB\Driver\Query($filter_arr,[]);

	$cursor = $manager->executeQuery('tourismdb.User', $query);
	$result = [];

	foreach($cursor as $document) {
		array_push($result, $document);
	}

	if (empty($result)){
		echo json_encode("Failure");
	}
	else {
		echo json_encode($result);
	}

} catch (MongoConnectionException $e){
	var_dump($e);
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "POST") {

	if (extension_loaded("mongodb")) {
	try {
		$body = json_decode(file_get_contents('php://input'));
		$manager = new
MongoDB\Driver\Manager("mongodb://localhost:27017");
		$bulk = new MongoDB\Driver\BulkWrite;
```

```
                $username = $body -> username;
                $name = $body -> name;
                $pass = $body -> password;

                $document1 = [          'username' => $username,
                                        'name' => $name,
                                        'password' => $pass];

                $bulk->insert($document1);
                $result1 = $manager->executeBulkWrite('tourismdb.User', $bulk);

                $filter_arr = ['username' => $username, 'password' => $pass];
                $query = new MongoDB\Driver\Query($filter_arr,[]);

                $cursor = $manager->executeQuery('tourismdb.User', $query);
                $result = [];

                foreach($cursor as $document) {
                        array_push($result, $document);
                }

                if (empty($result)){
                        echo 'Failure';
                }
                else {
                        echo json_encode($result);
                }


        } catch (MongoConnectionException $e){
                var_dump($e);
        }
}
}

else if ($_SERVER["REQUEST_METHOD"] == "PUT" && isset($_GET["username"])
&& isset($_GET["password"])) {

        if (extension_loaded("mongodb")) {
        try {
                $body = json_decode(file_get_contents('php://input'));
                $manager                                        =                                       new
MongoDB\Driver\Manager("mongodb://localhost:27017");
                $bulk = new MongoDB\Driver\BulkWrite;


                $username = $_GET["username"];
                $password = $_GET["password"];

                $bulk->update(
```

```
                ['username' => $username],
                ['$set' => ['password' => $password]],
                ['multi' => false, 'upsert' => false]
            );
            $result1 = $manager->executeBulkWrite('tourismdb.User', $bulk);

            $filter_arr = ['username' => $username];
            $query = new MongoDB\Driver\Query($filter_arr,[]);

            $cursor = $manager->executeQuery('tourismdb.User', $query);
            $result = [];

            foreach($cursor as $document) {
                array_push($result, $document);
            }

            if (empty($result)){
                echo 'Failure';
            }
            else {
                echo json_encode($result);
            }


        } catch (MongoConnectionException $e){
            var_dump($e);
        }
}
}

else if ($_SERVER["REQUEST_METHOD"] == "PUT") {

        if (extension_loaded("mongodb")) {
        try {
            $body = json_decode(file_get_contents('php://input'));
            $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
            $bulk = new MongoDB\Driver\BulkWrite;


            $username = $body -> username;
            $name = $body -> name;
            $visited = $body -> visited;
            $tovisit = $body -> tovisit;

            $bulk->update(
                ['username' => $username],
                ['$set' => ['name' => $name, 'visited' => $visited, 'tovisit' => $tovisit]],
                ['multi' => false, 'upsert' => false]
            );
            $result1 = $manager->executeBulkWrite('tourismdb.User', $bulk);
```

```
                $filter_arr = ['username' => $username];
                $query = new MongoDB\Driver\Query($filter_arr,[]);

                $cursor = $manager->executeQuery('tourismdb.User', $query);
                $result = [];

                foreach($cursor as $document) {
                        array_push($result, $document);
                }

                if (empty($result)){
                        echo 'Failure';
                }
                else {
                        echo json_encode($result);
                }


        } catch (MongoConnectionException $e){
                var_dump($e);
        }
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET") {
if (extension_loaded("mongodb")) {
try {

        $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");

        $filter_arr = [];
        $query = new MongoDB\Driver\Query($filter_arr,[]);

        $cursor = $manager->executeQuery('tourismdb.User', $query);
        $result = [];

        foreach($cursor as $document) {
                array_push($result, $document);
        }

        if (empty($result)){
                echo json_encode("Failure");
        }
        else {
                echo json_encode($result);
        }

} catch (MongoConnectionException $e){
        var_dump($e);
}
}
```

```
}
```

## places.php

```
<?php
header('Access-Control-Allow-Origin: *');

header('Access-Control-Allow-Methods: GET, POST');

header("Access-Control-Allow-Headers: X-Requested-With, Content-Type");

if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["cityId"])) {
if (extension_loaded("mongodb")) {
try {
        $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
        $cityId = $_GET["cityId"];
        $filter = ['cityId' => $cityId];
        $query = new MongoDB\Driver\Query($filter,[]);

        $cursor = $manager->executeQuery('tourismdb.TourPlace', $query);
        $result = [];

        foreach($cursor as $document) {
                array_push($result, $document);
        }

        echo json_encode($result);

} catch (MongoConnectionException $e){
        var_dump($e);
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["placeId"])) {
if (extension_loaded("mongodb")) {
try {
        $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
        $placeId = $_GET["placeId"];
        $filter1 = ['_id' => new MongoDB\BSON\ObjectID($placeId)];
        $query1 = new MongoDB\Driver\Query($filter1,[]);
        $cursor1 = $manager->executeQuery('tourismdb.TourPlace', $query1);

        $result = [];
        foreach($cursor1 as $document1) {
                array_push($result, $document1);
        }

        echo json_encode($result);

} catch (MongoConnectionException $e){
        var_dump($e);
```

```
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["categoryId"])) {
if (extension_loaded("mongodb")) {
try {
	$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
	$categoryId = $_GET["categoryId"];
	$filter = ['categoryId' => $categoryId];
	$query = new MongoDB\Driver\Query($filter,[]);

	$cursor = $manager->executeQuery('tourismdb.TourPlace', $query);
	$result = [];

	foreach($cursor as $document) {
		array_push($result, $document);
	}

	echo json_encode($result);

} catch (MongoConnectionException $e){
	var_dump($e);
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["userId"]) && isset($_GET["tovisit"])) {
if (extension_loaded("mongodb")) {
try {

	$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
	$userId = $_GET["userId"];
	$filter = ['_id' => new MongoDB\BSON\ObjectID($userId)];
	$query = new MongoDB\Driver\Query($filter,[]);

	$cursor = $manager->executeQuery('tourismdb.User', $query);
	$result = [];

	foreach($cursor as $document) {
		$document = json_decode(json_encode($document),true);
	}

	$tovisit = explode(",", $document['tovisit']);

	foreach($tovisit as $item) {
		$filter1 = ['_id' => new MongoDB\BSON\ObjectID($item)];
		$query1 = new MongoDB\Driver\Query($filter1,[]);
		$cursor1 = $manager->executeQuery('tourismdb.TourPlace', $query1);
```

```
                foreach($cursor1 as $document1) {
                        array_push($result, $document1);
                }
        }

        echo json_encode($result);

} catch (MongoConnectionException $e){
        var_dump($e);
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["userId"])) {
if (extension_loaded("mongodb")) {
try {

        $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
        $userId = $_GET["userId"];
        $filter = ['_id' => new MongoDB\BSON\ObjectID($userId)];
        $query = new MongoDB\Driver\Query($filter,[]);

        $cursor = $manager->executeQuery('tourismdb.User', $query);
        $result = [];
        $ids = [];
        $doc;

        foreach($cursor as $document) {
                $doc = json_decode(json_encode($document),true);
        }

        $visited = json_decode($doc['visited']);

        foreach($visited as $item) {
                $filter1 = ['_id' => new MongoDB\BSON\ObjectID($item->id)];
                $query1 = new MongoDB\Driver\Query($filter1,[]);
                $cursor1 = $manager->executeQuery('tourismdb.TourPlace', $query1);

                foreach($cursor1 as $document1) {
                        array_push($result, $document1);
                }
        }

        echo json_encode($result);

} catch (MongoConnectionException $e){
        var_dump($e);
}
}
}

else if ($_SERVER["REQUEST_METHOD"] == "GET") {
```

```
if (extension_loaded("mongodb")) {
try {
        $manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");

        $query = new MongoDB\Driver\Query([],[]);

        $cursor = $manager->executeQuery('tourismdb.TourPlace', $query);
        $result = [];

        foreach($cursor as $document) {
                array_push($result, $document);
        }

        echo json_encode($result);

} catch (MongoConnectionException $e){
        var_dump($e);
}
}
}
```