Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для подорожей визначними місцями. Front-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-7

Макаревич М. П.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Онищенко К. Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Макаревичу Михайлу Павловичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для подорожей визначними місцями. Front-end

Затверджена наказом по університету від 20.05.2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи Розробити сайт для організації подорожей визначними місцями за допомогою React.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Формування вимог до ПЗ	20.04.2024	*виконано*
3	Проектування ПЗ	22.04.2024	*виконано*
4	Розробка ПЗ	20.05.2024	*виконано*
5	Тестування ПЗ	22.05.2024	*виконано*
6	Оформлення пояснювальної записки	03.06.2024	*виконано*
7	Перевірка роботи на антиплагіат та проходження нормоконтролю	04.06.2024	*виконано*
8	Оцінка роботи рецензентом, отримання відгуку від керівника кваліфікаційної роботи	14.06.2024	*виконано*
9	Попередій захист роботи	14.06.2024	*виконано*
10	Здача роботи до електронного архіву, допуск роботи до захисту завідувачем кафедри	15.06.2024	*виконано*
11	Захист кваліфікаційної роботи	18.06.2024	

Дата видачі завдання 08 квітня 2024р.

Студент _____ Макаревич М. П.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Онищенко К. Г.
(підпис) (посада)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 79 стор., 31 рис., 15 джерел.

МАНДРІВКИ, САЙТ ДЛЯ МАНДРІВНИКІВ, ФРОНТЕНД, MATERIAL UI, NEXT.JS, REACT, TYPESCRIPT

Об’єкт розробки – клієнтська частина програмної системи для подорожей визначними місцями.

Мета розробки – спростити планування подорожей для мандрівників, реалізувати інтерфейс для отримання інформації про визначні місця а також запровадити можливість перегляду згадок про минулі мандри.

Метод рішення – фреймворк Next.js, мова програмування TypeScript, UI бібліотека Material UI.

У результаті розробки створено сайт для мандрівників, який допомагає в плануванні подорожей.

TRIPS, TRAVEL SITE, FRONTEND, MATERIAL UI, NEXT.JS, REACT, TYPESCRIPT

The object of development is the client part of a software system for traveling to places of interest.

The purpose of the development is to simplify travel planning for travelers, implement an interface for obtaining information about places of interest, and introduce the ability to view mentions of past trips.

The solution method is the Next.js framework, TypeScript programming language, and the Material UI library.

As a result of the development, a website was created for travelers to help them plan their trips.

Я, Макаревич Михайло Павлович, студент гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для подорожей визначними місцями. Front-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному світі подорожі стали невід'ємною частиною життя багатьох людей. Відкриття нових місць, знайомство з культурами та традиціями інших народів, отримання незабутніх вражень – все це робить їх захопливими та бажаними. Однак, планування мандрівки може бути складним і трудомістким процесом, який вимагає ретельної підготовки та врахування безлічі факторів. Вибір визначних місць, складання маршрутів, пошук інформації про пам'ятки, планування бюджету – ось лише деякі з завдань, які постають перед мандрівниками.

Інтернет та цифрові технології відкрили нові можливості для спрощення процесу планування подорожей. Проте, незважаючи на наявність великої кількості веб-ресурсів та мобільних додатків, багато з них мають обмежену функціональність або не забезпечують комплексного підходу до планування подорожей. Деякі додатки зосереджені лише на пошуку авіаквитків чи бронюванні готелів, інші надають рекомендації щодо визначних місць, але не забезпечують можливості формування маршрутів та планування бюджету.

Саме тому розробка програмної системи для подорожей визначними місцями є актуальною та затребуваною. Така система дозволить спростити процес планування мандри, зробити його більш ефективним та зручним для користувачів.

Метою розробки є створення клієнтської частини програмної системи для подорожей визначними місцями «Tripster», яка допоможе мандрівникам ефективно планувати свої подорожі, знаходити найбільш відповідні для них місця відвідування, отримувати детальну інформацію про пам'ятки, формувати маршрути та бюджет, а також ділитися спогадами від минулих мандрівок. Така система стане корисним інструментом для мандрівників різного рівня досвіду – від початківців до досвідчених туристів, які прагнуть організувати свої подорожі самостійно та максимально ефективно.

Основними завданнями роботи є:

- аналіз існуючих рішень та визначення функціональних вимог до системи з метою виявлення недоліків та переваг наявних програмних продуктів та врахування їх при розробці нової системи;
- вибір та обґрунтування технологій для розробки клієнтської частини, які забезпечать високу продуктивність, масштабованість та зручність використання додатку;
- розробка інтерфейсу та функціоналу для тестування переваг користувача та визначення підходящих місць відвідування з урахуванням його інтересів та уподобань;
- реалізація можливості отримання детальної інформації про визначні пам'ятки та місця, включаючи опис, фотографії, рекомендовані заходи та маршрути;
- впровадження функціоналу для формування плану подорожі з можливістю вибору дат, місць та створення нотаток;
- реалізація зручних інструментів для планування бюджету подорожі з урахуванням витрат на проживання, транспорт, харчування та розваги;
- розробка компонентів для перегляду спогадів від минулих подорожей;
- тестування та налагодження створеної системи.

Результати роботи можуть бути застосовані в туристичній галузі, а також використовуватися індивідуальними мандрівниками для ефективного планування своїх подорожей. Розроблена система допоможе зробити процес планування більш зручним, організованим та персоналізованим відповідно до індивідуальних потреб та уподобань користувачів. Завдяки комплексному підходу та інтеграції різноманітних функцій, система стане корисним помічником для мандрівників на всіх етапах підготовки до подорожі - від пошуку визначних місць до формування маршрутів, складання бюджету та збереження спогадів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Туристична галузь є однією з найбільших і найшвидше зростаючих галузей економіки у світі. Згідно з даними Всесвітньої туристичної організації ООН (UNWTO), у 2019 році туристичний сектор генерував близько 1,5 трильйона доларів США, що становило близько 7% світового експорту товарів та послуг [1]. Крім того, туризм забезпечував приблизно 330 мільйонів робочих місць, що становило кожне десяте робоче місце у світі [2].

Попит на подорожі та відкриття нових місць залишається високим серед людей різного віку та походження. Мандрівники прагнуть відвідувати визначні пам'ятки, занурюватися у культури інших народів, досліджувати природні красоти та збагачувати свій досвід новими враженнями. Ця тенденція особливо помітна серед молодого покоління, для якого подорожі стали способом життя та можливістю для саморозвитку.

Однак, процес планування подорожі може бути складним і трудомістким завданням. Мандрівникам доводиться витрачати багато часу на пошук інформації про визначні місця, складання маршрутів, вибір способів пересування, бронювання житла та визначення бюджету. Крім того, збереження спогадів від минулих подорожей також може бути проблематичним, оскільки фотографії, нотатки та враження можуть бути розпорошені по різних ресурсах.

У відповідь на ці виклики з'явилася низка веб-додатків та мобільних застосунків, призначених для допомоги в плануванні подорожей. Однак, більшість з них зосереджені лише на окремих аспектах подорожі, наприклад, пошуку авіаквитків чи бронюванні готелів. Деякі додатки надають рекомендації щодо визначних місць та маршрутів, але не забезпечують можливості формування бюджету або збереження спогадів.

Розглянемо більш детально деякі з існуючих рішень для планування подорожей.

Tripadvisor є одним з найпопулярніших туристичних веб-сайтів, який надає користувачам інформацію про готелі, ресторани, визначні пам'ятки та авіаквитки (див. рис. 1.1). Він був заснований у 1999 році та на сьогоднішній день налічує

понад 8,7 мільйонів залишених відгуків про різноманітні туристичні локації та об'єкти у понад 190 країнах світу [3].

Рисунок 1.1 – Логотип додатку Tripadvisor (за даними [3])

Ключові функції Tripadvisor:

- пошук та бронювання готелів: користувачі можуть знайти та забронювати готелі у будь-якій точці світу, переглянути ціни, фотографії, відгуки інших мандрівників та порівняти варіанти;
- огляд ресторанів: платформа надає інформацію про ресторани різних типів та цінових категорій, включаючи відгуки, фотографії, меню та ціни;
- інформація про визначні пам'ятки: користувачі можуть переглянути опис визначних пам'яток, години роботи, ціни на квитки, фотографії та відгуки інших відвідувачів;
- пошук авіаквитків: Tripadvisor надає можливість знайти та забронювати авіаквитки на різні напрямки;
- форуми та відгуки: платформа має активну спільноту користувачів, які діляться своїми враженнями, відгуками та порадами щодо подорожей.

Переваги Tripadvisor:

- величезна база даних готелів, ресторанів та визначних пам'яток з детальною інформацією та відгуками;
- зручний інтерфейс для пошуку та фільтрації результатів;
- можливість бронювання готелів та авіаквитків безпосередньо на сайті;
- активна спільнота користувачів, які діляться своїм досвідом.

Недоліки Tripadvisor:

– обмежені можливості для планування власних маршрутів та створення персоналізованих планів подорожей;
– відсутність інструментів для планування бюджету подорожі;
– немає функцій для зберігання та перегляду спогадів від минулих подорожей;
– фокус переважно на готелях, ресторанах та визначних пам'ятках, але менше уваги приділяється іншим аспектам подорожі.

Хоча Tripadvisor є потужним ресурсом для пошуку інформації про визначні місця, готелі та ресторани, він не забезпечує комплексного рішення для планування подорожей. Відсутність можливості створювати власні маршрути, планувати бюджет та зберігати спогади від поїздок є суттєвими недоліками для мандрівників, які прагнуть самостійно організовувати свої подорожі.

Google Trips є мобільним додатком від компанії Google, який допомагає мандрівникам організувати інформацію про поїздку в одному місці (див. рис. 1.2). Додаток був запущений у 2016 році та є безкоштовним для користувачів Android та iOS пристроїв [4].

Рисунок 1.2 – Логотип додатку Google Trips (за даними [4])

Основні функції Google Trips:

– об'єднання інформації про подорож: додаток автоматично збирає дані про резервування готелів, авіаквитків та іншої інформації про поїздку з облікового запису Google користувача;
– рекомендації щодо визначних пам'яток та заходів: Google Trips надає рекомендації щодо популярних визначних пам'яток, ресторанів та заходів у місці призначення, засновані на інтересах користувача та відгуках інших мандрівників;

- офлайн доступ до інформації: додаток дозволяє завантажувати інформацію про поїздку для офлайн доступу, що є корисним для мандрівників, які не мають постійного доступу до Інтернету;
- планування маршрутів: Google Trips надає можливість планувати маршрути між різними визначними місцями та об'єктами;
- збереження нотаток та фотографій: користувачі можуть зберігати свої нотатки, фотографії та враження в додатку для перегляду в майбутньому.

Переваги Google Trips:

- зручний інтерфейс для доступу до всієї інформації про поїздку в одному місці;
- автоматичне збирання даних з облікового запису Google;
- офлайн доступ до інформації;
- можливість планувати маршрути та зберігати нотатки та фотографії.

Недоліки Google Trips:

- обмежені можливості для створення власних персоналізованих маршрутів та планів подорожей;
- відсутність інструментів для планування бюджету подорожі;
- залежність від даних Google, що може бути неповними або неточними в деяких регіонах;
- відсутність деяких додаткових функцій, таких як бронювання готелів чи авіаквитків безпосередньо в додатку.

Google Trips є корисним інструментом для організації та доступу до основної інформації про поїздку в одному місці. Однак, він не забезпечує комплексного рішення для планування подорожей, оскільки не має можливості створювати власні персоналізовані маршрути, планувати бюджет та повноцінно керувати всіма аспектами подорожі.

Roadtrippers є одним з провідних веб-сервісів та мобільних додатків для планування подорожей на автомобілі (див. рис. 1.3). Його основна мета – полегшити процес створення маршрутів, пошуку цікавих місць та пам'яток вздовж шляху, а також забезпечити платформу для обміну досвідом між мандрівниками.

Сервіс пропонує зручний інструмент для планування маршрутів, де користувачі можуть вказати початкову та кінцеву точки подорожі, а система автоматично прокладе найоптимальніший шлях [5]. Під час планування маршруту є можливість додавати проміжні зупинки та визначні місця, які становлять інтерес для мандрівника. Roadtrippers має у своєму розпорядженні базу даних з понад 1 мільйоном різноманітних визначних пам'яток, природних чудес, музеїв, ресторанів та інших цікавих локацій по всьому світі. Користувачі можуть легко знаходити потрібні місця за категоріями, рейтингами, відгуками та відстанню від запланованого маршруту.

Рисунок 1.3 – Логотип додатку Roadtrippers (за даними [5])

Одною з ключових переваг Roadtrippers є активна спільнота мандрівників, які діляться своїми маршрутами, рекомендаціями та враженнями від подорожей. Користувачі можуть переглядати, коментувати та оцінювати маршрути інших мандрівників, а також ділитися своїми досягненнями в соціальних мережах.

Roadtrippers доступний як веб-сервіс та у вигляді мобільних додатків для iOS та Android. Мобільні додатки дозволяють отримувати сповіщення про цікаві місця поблизу під час самої подорожі, а також переглядати свої маршрути та додавати нові місця прямо з мобільного пристрою.

Незважаючи на свою популярність та зручність для планування автомобільних мандрівок, Roadtrippers зосереджений переважно на цьому виді подорожей і не пропонує комплексних рішень для планування поїздок іншими видами транспорту або з урахуванням бюджету та інших факторів, що є важливими для загального планування подорожі.

Sygic Travel – мобільний додаток, який надає інформацію про визначні пам'ятки, маршрути та розклади транспорту (див. рис. 1.4). Однак, він не дозволяє створювати власні маршрути та плани подорожі [6].

Рисунок 1.4 – Логотип додатку Sygic Travel (за даними [6])

Як бачимо, більшість існуючих рішень зосереджені лише на окремих аспектах планування подорожей, що ускладнює для користувачів процес організації комплексної поїздки до визначних місць. Таким чином, виникає потреба у створенні комплексної програмної системи, яка об'єднає всі необхідні функції для ефективного планування подорожей визначними місцями.

1.1 Виявлення та вирішення проблем

Аналіз предметної галузі та існуючих рішень для планування подорожей виявив низку проблем і недоліків, які ускладнюють для мандрівників процес організації комплексних поїздок до визначних місць. Розглянемо основні з них та запропоновані шляхи їх вирішення.

Перша проблема полягає у відсутності єдиної платформи, яка б охоплювала всі етапи планування подорожі. Більшість наявних веб-сайтів та додатків зосереджуються лише на окремих аспектах, таких як пошук авіаквитків, бронювання готелів або отримання інформації про визначні пам'ятки. Мандрівникам доводиться використовувати кілька різних ресурсів, що є незручним та неефективним. Тому розробка комплексної програмної системи, яка об'єднає всі необхідні функції для планування подорожей, включаючи пошук визначних місць, створення маршрутів, планування бюджету та збереження спогадів, стане вагомим вирішенням цієї проблеми.

Наступна проблема стосується обмежених можливостей для створення власних маршрутів та планів подорожей. Більшість існуючих рішень пропонують лише стандартні маршрути або рекомендації, не даючи користувачам змоги створювати власні персоналізовані плани відповідно до їхніх уподобань та інтересів. Тому впровадження функціоналу для створення власних маршрутів та персоналізованих планів подорожей, з можливістю вибору дат, визначних місць, заходів та додавання нотаток, стане вагомим доповненням до розробленої системи.

Ще однією проблемою є відсутність інструментів для планування бюджету подорожі. Більшість наявних рішень не надають користувачам можливості планувати бюджет своєї поїздки, враховуючи витрати на проживання, транспорт, харчування та розваги. Розробка зручних інструментів для планування бюджету подорожі, які дозволять користувачам встановлювати ліміти витрат і контролювати свій бюджет, стане важливим доповненням до функціоналу системи.

Після повернення з подорожі, мандрівникам часто доводиться шукати свої фотографії, нотатки та враження в різних місцях, що ускладнює процес збереження та перегляду спогадів. Впровадження функціоналу для зберігання та перегляду спогадів від минулих подорожей в межах однієї програмної системи, включаючи фотографії, нотатки та рефлексії, вирішить цю проблему.

Нарешті, більшість існуючих рішень надають стандартизовані рекомендації, не враховуючи унікальні інтереси та переваги кожного мандрівника. Впровадження функціоналу для тестування уподобань користувача та надання персоналізованих рекомендацій щодо визначних місць на основі його інтересів забезпечить необхідну персоналізацію та більш точні пропозиції для подорожей.

Вирішуючи ці проблеми, розроблена програмна система для подорожей визначними місцями стане комплексним і зручним рішенням для планування поїздок. Вона дозволить мандрівникам створювати власні персоналізовані маршрути, планувати бюджет, зберігати спогади та отримувати рекомендації на основі їхніх уподобань, об'єднуючи всі необхідні функції в одному інтерфейсі.

1.2 Постановка задачі

На основі проведеного аналізу предметної галузі та виявлених проблем, метою даної кваліфікаційної роботи є розробка клієнтської частини програмної системи для подорожей визначними місцями. Дана система має вирішити основні недоліки існуючих рішень та забезпечити мандрівників зручним інструментом для ефективного планування своїх поїздок.

Головні завдання, які необхідно вирішити в рамках розробки програмної системи:

- створити єдину платформу, яка об'єднає всі необхідні функції для планування подорожей, включаючи пошук визначних місць, формування маршрутів, планування бюджету та збереження спогадів від минулих поїздок;
- реалізувати функціонал для створення власних персоналізованих маршрутів та планів подорожей, з можливістю вибору дат, визначних місць, заходів та додавання нотаток;
- розробити зручні інструменти для планування бюджету подорожі, які дозволять користувачам встановлювати ліміти витрат і контролювати свій бюджет з урахуванням проживання, транспорту, харчування та розваг;
- впровадити функціонал для зберігання та перегляду спогадів від минулих подорожей, включаючи фотографії, нотатки та рефлексії, в межах однієї програмної системи;
- реалізувати можливість тестування уподобань користувача та надання персоналізованих рекомендацій щодо визначних місць на основі його інтересів;
- забезпечити зручний та інтуїтивний користувацький інтерфейс для ефективної взаємодії з системою;
- використати сучасні технології та підходи до розробки веб-додатків, такі як Next.js, TypeScript та Material UI, для забезпечення високої продуктивності, масштабованості та зручності використання системи;

– розробити архітектуру системи, яка дозволить легко розширювати та вдосконалювати її функціонал у майбутньому.

Успішна реалізація даної програмної системи дозволить мандрівникам ефективно планувати свої подорожі до визначних місць, враховуючи їхні індивідуальні уподобання та потреби. Система стане зручним інструментом для пошуку інформації про пам'ятки, створення персоналізованих маршрутів, планування бюджету та зберігання спогадів від минулих поїздок, об'єднуючи всі необхідні функції в одному веб-додатку.

Крім того, розроблене рішення може бути корисним для туристичної галузі, надаючи можливість підвищити якість обслуговування клієнтів та запропонувати їм зручний інструмент для самостійного планування подорожей.

1.3 Цільова аудиторія

Розроблена програмна система для подорожей визначними місцями орієнтована на широку цільову аудиторію, яка охоплює різні категорії мандрівників. Детальний аналіз цільової аудиторії допоможе забезпечити ефективне задоволення потреб та очікувань користувачів під час розробки системи. Тож наведено наступний перелік зацікавлених груп користувачів користувачів:

– індивідуальні мандрівники. Ця категорія включає людей, які прагнуть самостійно планувати свої подорожі, вибираючи визначні місця, складаючи маршрути та контролюючи бюджет. Це можуть бути як досвідчені туристи, так і ті, хто лише починає захоплюватися подорожами. Для них важливо мати зручний інструмент, який дозволить персоналізувати всі аспекти подорожі відповідно до їхніх індивідуальних потреб та уподобань;
– молоді мандрівники. Для молодих людей, для яких подорожі стали способом життя та можливістю для саморозвитку, важливо мати систему, яка надасть персоналізовані рекомендації щодо визначних місць та заходів на основі їхніх інтересів. Вони шукають нові враження та цінують

інноваційні рішення, що роблять процес планування подорожей більш захопливим та креативним;

- сімейні пари або групи друзів. Ця категорія охоплює групи людей, які хочуть спільно спланувати захоплюючу подорож, враховуючи інтереси та уподобання кожного учасника. Для них важливо мати можливість створювати спільні плани подорожей, додавати нотатки та ділитися спогадами від минулих мандрівок;
- любителі історії, культури та визначних пам'яток. Ця категорія включає людей будь-якого віку, які цікавляться історією, культурою та визначними пам'ятками різних країн і міст та прагнуть відвідати їх. Для них важливо мати доступ до детальної інформації про визначні місця, можливість створювати маршрути з урахуванням їхніх інтересів та отримувати рекомендації щодо цікавих заходів;
- туристичні агентства та організатори подорожей. Хоча розроблена система орієнтована переважно на індивідуальних мандрівників, вона також може бути корисною для туристичних агентств та організаторів подорожей. Вони можуть використовувати систему як інструмент для створення персоналізованих турів та маршрутів, а також для надання клієнтам додаткової інформації про визначні місця та рекомендацій щодо заходів.

Врахування потреб і очікувань різних категорій цільової аудиторії під час розробки програмної системи для подорожей визначними місцями забезпечить її зручність та корисність для широкого кола користувачів. Це, в свою чергу, сприятиме популярності та успішному впровадженню системи на ринку туристичних послуг.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Фронтенд частина програмної системи для подорожей визначними місцями відіграє ключову роль у забезпеченні зручного та інтуїтивного користувацького інтерфейсу. Саме через фронтенд користувачі взаємодіятимуть з системою, плануватимуть свої подорожі, шукатимуть інформацію про визначні місця та ділитимуться своїми спогадами. Тому важливо, щоб фронтенд був розроблений з урахуванням сучасних тенденцій веб-розробки, забезпечував високу продуктивність та зручність використання.

2.1 Технології та інструменти

Для розробки фронтенд частини будуть використані такі технології та інструменти:

- Next.js – фреймворк для React, що забезпечує сучасний підхід до розробки веб-додатків, включаючи серверний рендеринг, статичний генератор та інші функції для підвищення продуктивності та поліпшення користувацького досвіду;
- TypeScript – мова програмування, яка додає статичну типізацію до JavaScript. Використання TypeScript допоможе зробити код більш стійким до помилок, покращить підтримку коду та полегшить роботу;
- Material UI – бібліотека компонентів користувацького інтерфейсу для React, яка забезпечує зручний та сучасний дизайн, а також дотримується принципів матеріального дизайну від Google [7];
- React – популярна бібліотека для створення користувацьких інтерфейсів, яка дозволяє будувати швидкі та масштабовані веб-додатки;
- Axios – бібліотека для здійснення HTTP-запитів до серверної частини додатку та отримання необхідних даних;
- React Query – утиліта для кешування і контролю інформації, яка отримана через API.

– стилізація компонентів. Для стилізації компонентів буде використано підхід CSS-in-JS з використанням бібліотеки styled-components.

2.2 Структура та архітектура

Фронтенд частина буде побудована з використанням модульної архітектури та принципів компонентного підходу [8]. Кожен функціональний блок додатку буде представлений окремим компонентом React, що полегшить повторне використання коду та підтримку [9].

Основними модулями фронтенд частини будуть:

– модуль авторизації та реєстрації – відповідатиме за процеси входу та реєстрації користувачів у системі;
– модуль профілю користувача – дозволить користувачам переглядати та редагувати свій профіль, налаштування та переваги;
– модуль пошуку та перегляду визначних місць – забезпечить функціональність пошуку визначних пам'яток, музеїв, природних заповідників та інших цікавих місць, а також надасть детальну інформацію про них;
– модуль планування подорожі – дозволить користувачам створювати, редагувати та переглядати плани своїх подорожей, додавати місця для відвідування, заходи та нотатки;
– модуль управління бюджетом – надасть інструменти для розрахунку та контролю бюджету подорожі;
– модуль спогадів – забезпечить можливість переглядати фотографії, нотатки та враження від минулих подорожей.

Кожен модуль буде складатися з відповідних компонентів React, які будуть відповідати за відображення даних, обробку подій користувача та взаємодію з серверною частиною додатку.

2.3 Дизайн та користувацький інтерфейс

Для створення унікального та захопливого дизайну буде застосовано інноваційний підхід, що поєднує традиційні принципи дизайну з сучасними тенденціями та технологіями, зберігаючи при цьому простоту та зручність реалізації.

Візуальна ідентичність додатку буде заснована на концепції «Exploration Chic» – стилі, натхненному прагненням до відкриттів, пригод та елегантності. Палітра кольорів буде поєднувати насичені, але стримані відтінки, які нагадують про екзотичні пейзажі та культури світу. Акцентні кольори будуть використовуватися для виділення ключових елементів інтерфейсу та створення візуальної ієрархії.

Типографіка додатку буде ретельно підібрана, щоб передавати відчуття пригоди та водночас зберігати зручність читання. Для заголовків та акцентних елементів буде використано стильний шрифт із суміші геометричних та каліграфічних елементів, натхнений старовинними мандрівними картами. Для основного тексту буде використано чистий та легкий для читання шрифт з відкритим начертанням.

Візуальні елементи інтерфейсу будуть натхненні мотивами мандрівок, культури та природи. Ілюстрації та іконки будуть виконані у стилі ліній та силуетів, що нагадують старовинні гравюри та естампи. Використання анімації та паралаксних ефектів створить відчуття глибини та динаміки, занурюючи користувачів у світ пригод та відкриттів.

2.4 Взаємодія з серверною частиною

Фронтенд частина буде взаємодіяти з серверною частиною через API. Воно буде забезпечувати доступ до даних про визначні місця, плани подорожей, спогади користувачів та іншої необхідної інформації.

Для здійснення HTTP-запитів до API буде використано бібліотеку Axios. Ці інструменти дозволять відправляти запити на сервер та отримувати відповіді у зручному форматі JSON [10].

Під час розробки фронтенд частини буде приділено увагу безпеці та захисту даних, включаючи обробку помилок, перевірку вхідних даних та захист від вразливостей, таких як Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF).

2.5 Продуктивність та оптимізація

Забезпечення високої продуктивності та оптимізація додатку є одним з ключових пріоритетів. Для досягнення максимальної швидкості та плавності роботи додатку будуть застосовані такі підходи та техніки:

- серверний рендеринг. Next.js забезпечує можливість серверного рендерингу React-додатків, що дозволяє покращити час завантаження сторінок та показники продуктивності для користувачів;
- статичний генератор. Next.js також надає інструменти для генерації статичних сторінок на етапі збірки, що значно підвищує швидкість завантаження контенту для користувачів;
- код-спліттінг: Розбиття коду на окремі модулі та завантаження їх лише за потребою, що дозволить зменшити розмір початкового завантаження та покращити час реакції додатку;
- ліниве підвантаження – використання техніки лінивого підвантаження для великих або ресурсномістких компонентів, щоб запобігти завантаженню непотрібного коду на початковому етапі [11];
- мемоїзація та кешування – застосування технік мемоїзації та кешування для запобігання повторних обчислень або запитів до серверу для даних, які не змінюються часто;
- віртуалізація списків – для відображення великих списків даних (наприклад, списку визначних місць) буде використана віртуалізація, яка дозволить відображати лише ті елементи, які видимі на екрані, збільшуючи продуктивність;

- оптимізація зображень та мультимедіа. Застосування технік оптимізації зображень та мультимедіа, таких як стиснення, ліниве підвантаження та використання сучасних форматів (наприклад, WebP для зображень);
- моніторинг продуктивності. Буде впроваджено інструменти для моніторингу продуктивності фронтенд частини в реальних умовах, щоб виявляти та вирішувати потенційні проблеми з продуктивністю.

Ці підходи та техніки забезпечать високу продуктивність додатку, що сприятиме кращому користувацькому досвіду та задоволенню від використання додатку.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

У цьому підрозділі будуть представлені діаграми UML, які візуалізують структуру та поведінку фронтенд частини «Tripster». UML є стандартизованою мовою моделювання, яка дозволяє створювати різні види діаграм для опису системи з різних точок зору.

3.1.1 Діаграма прецедентів

Для візуалізації функціональних вимог та взаємодії між акторами (користувачами) та прецедентами (функціями) у фронтенд частині веб-додатку «Tripster» було створено діаграму прецедентів за допомогою UML (див. рис. 3.1). Діаграма прецедентів відображає високорівневу функціональність системи з точки зору користувача та ілюструє, як різні актори взаємодіють із системою.

На діаграмі прецедентів представлено два основні актори: неавторизований користувач та авторизований користувач. Неавторизований користувач має можливість зареєструватися в системі, щоб стати авторизованим користувачем, або авторизуватися, якщо він вже має обліковий запис.

Авторизований користувач має доступ до широкого спектру функціональних можливостей системи. Однією з ключових функцій є планування подорожі, яке включає створення, оновлення та видалення плану подорожі. Користувач може додавати місця до свого плану, вказувати дати, бюджет та інші деталі подорожі.

Інша важлива функція для авторизованого користувача – це перегляд та робота з місцями. Користувач може переглядати список місць, сортувати та фільтрувати їх за різними критеріями, наприклад, за категорією, популярністю або розташуванням. Крім того, користувач може додавати місця до списку вподобаних, щоб швидко отримати доступ до них пізніше, а також додавати місця безпосередньо до свого плану подорожі.

Система також надає персональні рекомендації щодо місць та активностей на основі вподобань та інтересів користувача. Для отримання персональних

рекомендацій користувач може пройти опитування, яке допоможе системі краще зрозуміти його преференції.

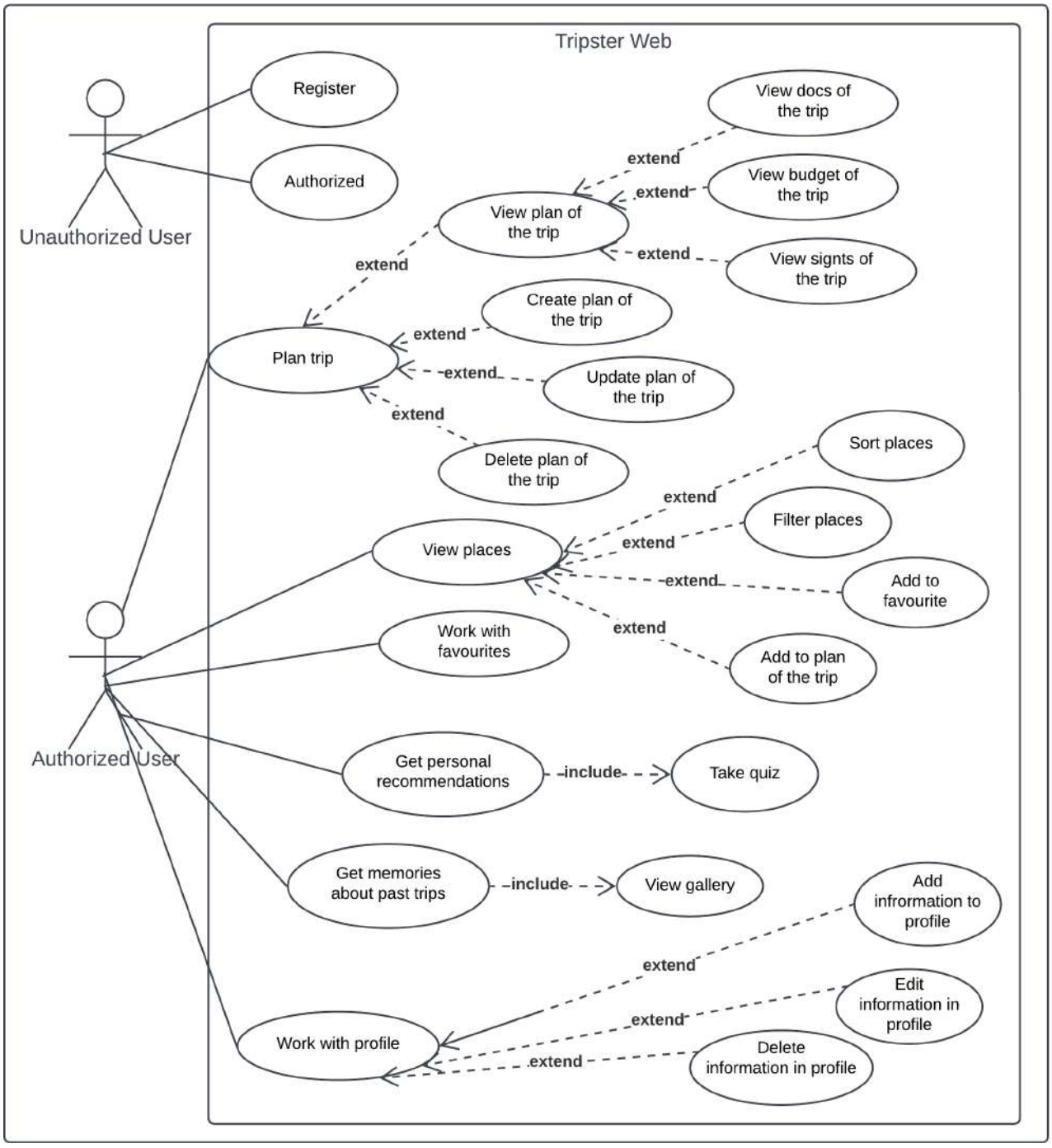


Рисунок 3.1 – Діаграма прецедентів (виконано самостійно)

Авторизований користувач має доступ до функції перегляду спогадів про минулі подорожі. Ця функція дозволяє користувачеві переглядати галерею

фотографій, заміток та вражень від попередніх подорожей, а також додавати нову інформацію до свого профілю.

Робота з профілем – це ще одна важлива функція для авторизованого користувача. Користувач може редагувати особисту інформацію, таку як ім'я, електронна пошта та фотографія профілю, а також видаляти інформацію з профілю за потреби.

Крім того, авторизований користувач має можливість переглядати детальну інформацію про свої подорожі, включаючи план подорожі, бюджет, місця та документи, пов'язані з подорожжю.

Діаграма прецедентів наочно демонструє основні функції фронтенд частини веб-додатку «Tripster» та взаємодію між акторами та прецедентами. Вона допомагає визначити межі системи, зрозуміти вимоги користувачів та забезпечити повноту функціональності системи, а також є важливим інструментом для комунікації між зацікавленими сторонами проекту, включаючи замовників, менеджерів проектів та розробників. Така діаграма слугує основою для подальшого детального проектування та розробки системи, а також для планування та оцінки робіт.

3.1.2 Діаграма класів

Для візуалізації структури та взаємозв'язків між основними класами фронтенд частини «Tripster» було створено діаграму класів (див. рис 3.2). Діаграма класів відображає статичну структуру системи, її класи, атрибути, методи та взаємозв'язки між ними.

Основними класами, представленими на діаграмі, є User (Користувач), Place (Місце), Favorite (Вподобане місце), Quiz (Опитування), Question (Запитання), Answer (Відповідь), QuizResult (Результат опитування), Vacation (Відпустка), VacationDay (День відпустки) та Gallery (Галерея).

Клас User містить інформацію про користувача, таку як ідентифікатор, ім'я, електронну пошту та хеш пароля. Він також має зв'язки з іншими класами, такими

як Favorite, QuizResult та Vacation, що вказує на те, що користувач може мати вподобані місця, результати опитувань та інформацію про відпустки.

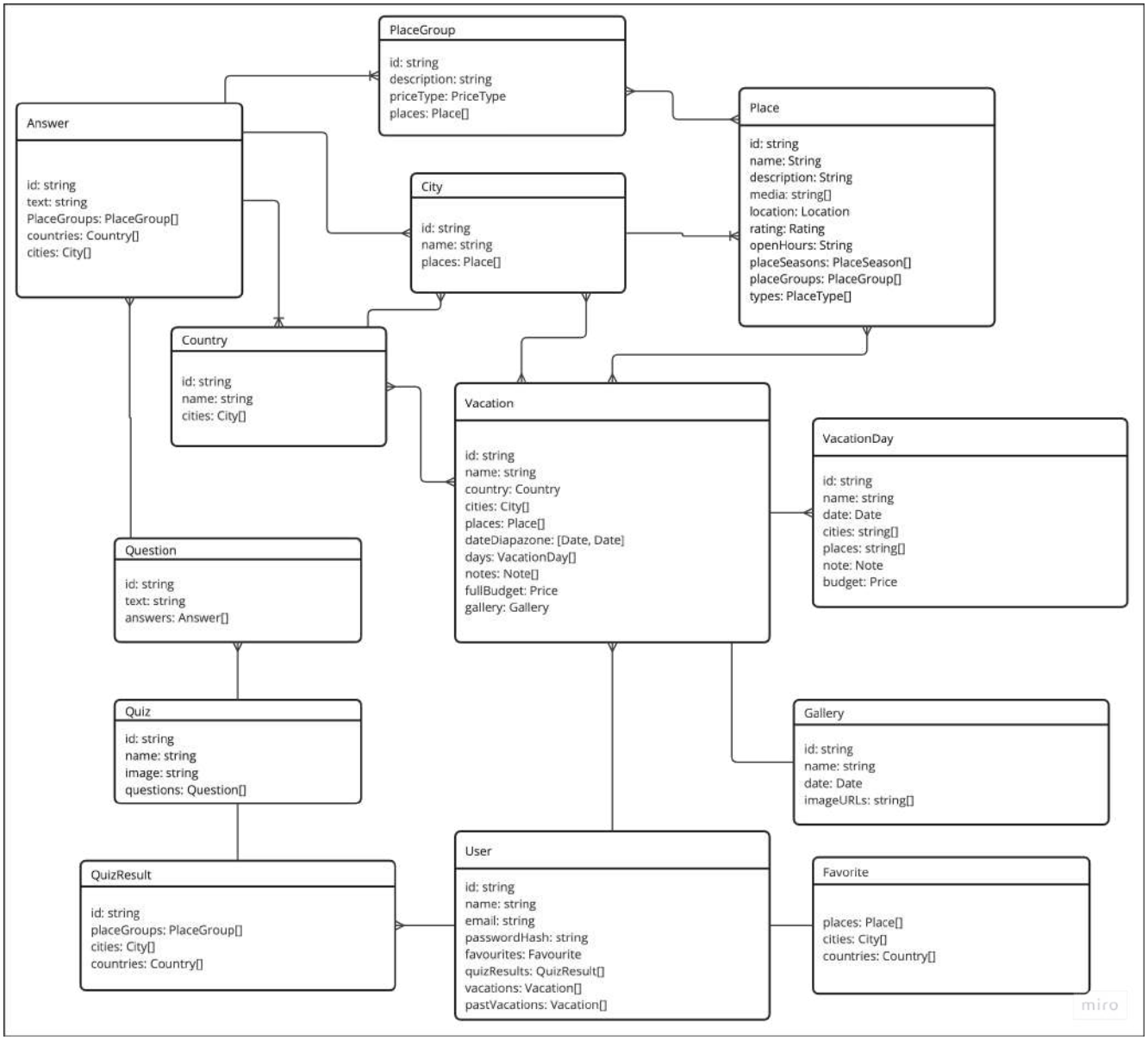


Рисунок 3.2 – Діаграма класів (виконано самостійно)

Клас Place представляє місце, яке користувач може відвідати. Він містить атрибути, такі як ідентифікатор, назва, опис, медіа-файли, розташування, рейтинг, години роботи та сезонність. Клас Place має зв'язки з класами PlaceGroup, Location та PlaceSeason, що дозволяє групувати місця, визначати їх розташування та вказувати сезонність.

Клас Favorite відображає вподобані користувачем місця, міста та країни. Він має зв'язки з класами User, Place, City та Country, що дозволяє користувачеві зберігати свої улюблені місця та отримувати до них швидкий доступ.

Класи Quiz, Question та Answer представляють функціональність опитування користувача для визначення його переваг та інтересів. Клас Quiz містить запитання, кожне з яких представлено класом Question. Клас Answer містить варіанти відповідей на запитання та має зв'язки з класами PlaceGroup, Country та City, що дозволяє рекомендувати відповідні місця на основі відповідей користувача.

Клас QuizResult зберігає результати опитування користувача, включаючи рекомендовані групи місць, міста та країни. Він має зв'язки з класами User, PlaceGroup, City та Country.

Класи Vacation та VacationDay представляють інформацію про відпустку користувача. Клас Vacation містить загальні дані про відпустку, такі як назва, країна, міста, місця, дати, дні відпустки, нотатки, бюджет та галерею. Клас VacationDay представляє окремий день відпустки та містить інформацію про відвідані міста, місця, нотатки та бюджет на цей день.

Клас Gallery представляє галерею зображень, пов'язаних з відпусткою користувача. Він містить атрибути, такі як ідентифікатор, назва, дата та колекцію зображень.

Діаграма класів також включає додаткові класи, такі як PlaceGroup, City, Country та Location, які використовуються для організації та структурування даних про місця, міста, країни та розташування.

Загалом, діаграма класів є важливим інструментом для розуміння архітектури системи, полегшує комунікацію між розробниками та дозволяє ефективно планувати та реалізовувати функціональність додатку. Атрибути класів відображають основні характеристики та властивості кожного об'єкту, а зв'язки між класами показують, як об'єкти різних класів взаємодіють та співвідносяться один з одним.

3.1.3 Діаграма діяльності

Для візуалізації послідовності дій та потоку керування в процесі взаємодії користувача з фронтенд частиною «Tripster» було розроблено діаграму діяльності (див. рис. 3.3). Вона допомагає краще зрозуміти логіку роботи системи та послідовність виконання операцій.

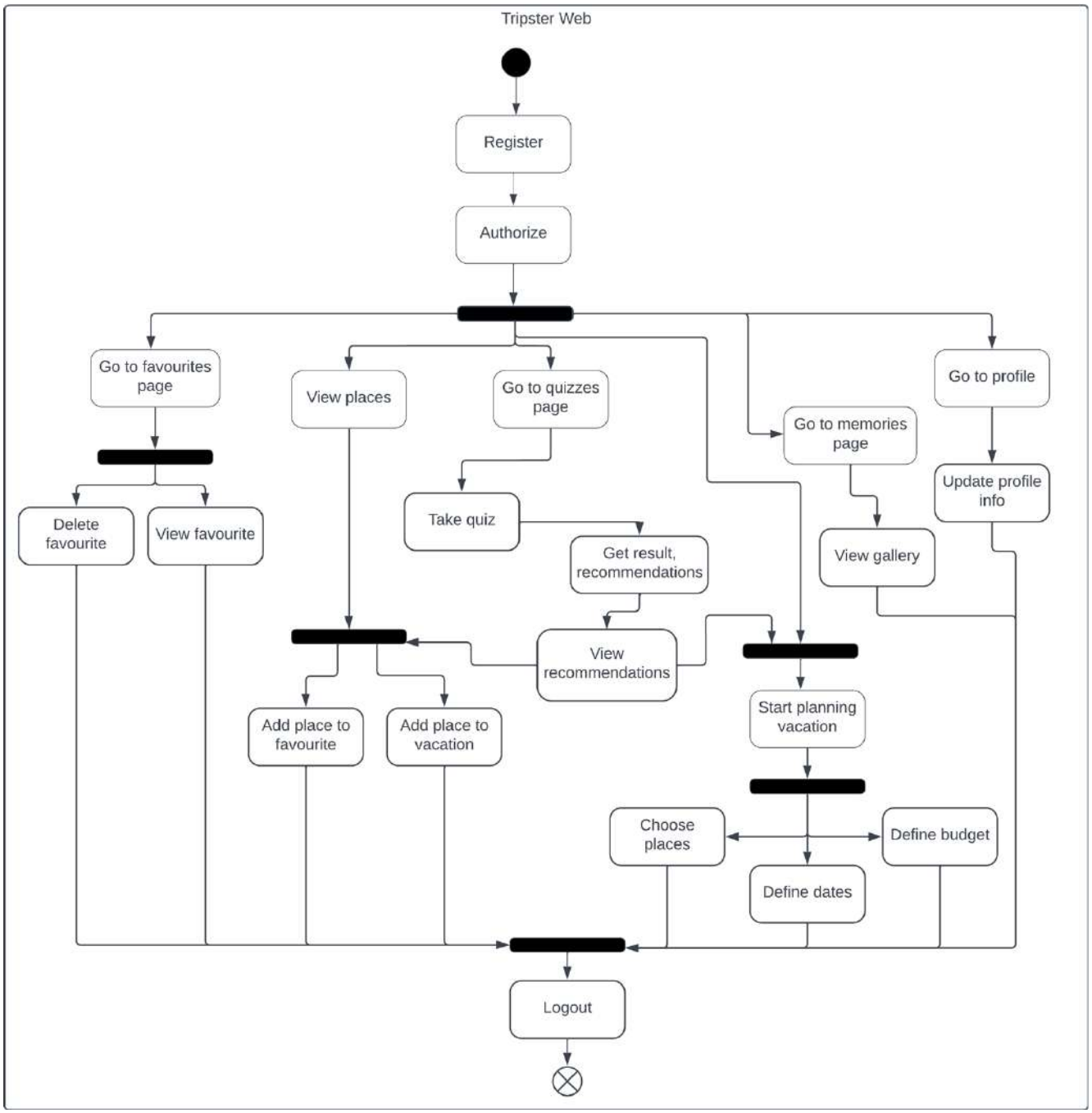


Рисунок 3.3 – Діаграма діяльності (виконано самостійно)

Діаграма починається з точки входу, що представляє початок взаємодії користувача з веб-додатком. Далі користувач має два основні шляхи: реєстрація (Register) або авторизація (Authorize). Якщо користувач ще не має облікового запису, він може пройти процес реєстрації, після чого буде перенаправлений на сторінку авторизації.

Після успішної авторизації користувач потрапляє на головну сторінку веб-додатку, де має доступ до різних функцій та можливостей. Основні дії, які може виконувати користувач, включають перегляд списку місць, перехід до сторінки обраних місць, перехід до сторінки опитувань, перегляд профілю та перегляд сторінки спогадів.

При перегляді списку місць користувач може додавати місця до обраного, додавати місця до плану подорожі або просто переглядати інформацію про місця. Користувач також має можливість видалити місце зі списку обраних.

На сторінці опитувань користувач може пройти опитування, щоб отримати персоналізовані рекомендації щодо місць для відвідування. Після проходження опитування користувач може переглянути результати та рекомендації, а також ознайомитися з запропонованими місцями.

Сторінка профілю користувача дозволяє переглядати та редагувати особисту інформацію. Користувач може оновлювати дані профілю або просто переглядати поточну інформацію.

Сторінка спогадів надає користувачу можливість переглядати збережені спогади про минулі подорожі. Користувач може переглядати галерею фотографій та інформацію про відвідані місця.

Однією з ключових функцій веб-додатку «Tripster» є планування подорожі. Користувач може розпочати процес планування подорожі, вибравши місця для відвідування, визначивши дати подорожі та сформувавши бюджет. Після завершення планування користувач може переглядати детальну інформацію про спланована подорож.

У будь-який момент користувач має можливість вийти з системи, що завершує сеанс роботи з сайтом.

Діаграма діяльності наочно демонструє основні сценарії використання веб-додатку «Tripster», відображаючи послідовність дій та можливі переходи між ними. Вона допомагає розробникам та зацікавленим сторонам краще зрозуміти логіку роботи системи та забезпечити її ефективну реалізацію.

Використання діаграми діяльності дозволяє оптимізувати користувацький досвід, виявити потенційні проблеми та вдосконалити процеси взаємодії з веб-додатком. Крім того, вона слугує основою для подальшої деталізації та реалізації окремих компонентів та функціональних можливостей фронтенд частини системи.

3.2 Проєктування архітектури ПЗ

У цьому підрозділі буде описано архітектуру фронтенд частини «Tripster», включаючи основні компоненти, шаблони проєктування та принципи, які будуть застосовані для забезпечення масштабованості, гнучкості та зручності підтримки.

3.2.1 Архітектурний патерн

Для розробки фронтенд частини додатку «Tripster» було обрано архітектурний патерн «React + Context API». React – це популярна бібліотека для створення користувацьких інтерфейсів, яка дозволяє будувати масштабовані та ефективні веб-додатки. Context API – це вбудований механізм у React для управління станом додатку та передачі даних між компонентами без необхідності передавати пропси вручну через кожен рівень дерева компонентів.

Використання React з Context API забезпечує чіткий поділ відповідальності між компонентами та ефективне управління станом додатку. Цей підхід дозволяє створювати модульну та легко підтримувану архітектуру фронтенд частини веб-додатку.

Основні принципи та переваги використання архітектурного патерну «React + Context API»:

– компонентний підхід. React заохочує розбиття інтерфейсу користувача на окремі, повторно використовувані компоненти. Кожен компонент інкапсулює свою власну логіку та відображення, що робить код більш

модульним та зрозумілим. Це дозволяє розробникам зосередитись на розробці окремих компонентів, які можна легко інтегрувати в загальну структуру додатку;

- декларативний синтаксис. React використовує декларативний підхід до опису користувацького інтерфейсу. Розробники описують, як компоненти повинні виглядати в різних станах, а React ефективно оновлює та відображає зміни в DOM. Це спрощує розробку та зменшує кількість помилок, пов'язаних з ручним маніпулюванням DOM;
- віртуальний DOM. React використовує концепцію віртуального DOM для ефективного оновлення користувацького інтерфейсу. Замість безпосередньої маніпуляції реальним DOM, React створює легку копію DOM у пам'яті, яка називається віртуальним DOM. Коли стан компонента змінюється, React обчислює різницю між новим та попереднім станом віртуального DOM і застосовує тільки необхідні зміни до реального DOM. Це значно покращує продуктивність та швидкість оновлення інтерфейсу користувача;
- односпрямований потік даних. React підтримує односпрямований потік даних, де дані передаються від батьківських компонентів до дочірніх через пропси. Це робить потік даних більш передбачуваним та полегшує відстеження змін у стані додатку. Коли стан компонента змінюється, React автоматично оновлює відповідні дочірні компоненти, забезпечуючи узгодженість даних у всьому додатку;
- управління станом за допомогою Context API. Context API дозволяє ефективно передавати дані через дерево компонентів без необхідності передавати пропси на кожному рівні. Це особливо корисно для глобальних даних, таких як інформація про аутентифікацію користувача, налаштування теми або мовні параметри. За допомогою Context API можна створити постачальників контексту (providers), які зберігають спільний стан та надають його споживачам (consumers) у будь-якому місці дерева компонентів;

- можливість інтеграції зі сторонніми бібліотеками. React може легко інтегруватися з іншими бібліотеками та інструментами, що дозволяє використовувати існуючі рішення для конкретних завдань;
- підтримка та популярність. React має велику та активну спільноту розробників, яка постійно розвиває та підтримує бібліотеку. Це означає, що розробники можуть знайти багато ресурсів, навчальних матеріалів та готових рішень для поширених проблем. Крім того, популярність React гарантує, що бібліотека буде активно підтримуватися та оновлюватися в майбутньому.

Загалом, архітектурний патерн «React + Context API» є потужним та ефективним вибором для розробки сайту «Tripster», забезпечуючи високу продуктивність, модульність та можливість створення привабливих і функціональних інтерфейсів користувача.

3.2.2 Шаблони проєктування

Під час розробки фронтенд частини веб-додатку будуть застосовані такі шаблони проєктування:

- контейнер/презентаційний компонент. Цей шаблон передбачає розділення компонентів на контейнери, які обробляють логіку та управляють даними, та презентаційні компоненти, які відповідають лише за відображення даних;
- вищий компонент порядку (Higher-Order Component, HOC). Шаблон, який дозволяє повторно використовувати функціональність компонентів, обгортаючи їх іншим компонентом;
- провайдер (Provider). Шаблон, що забезпечує централізоване постачання даних для дочірніх компонентів;
- обгортка (Wrapper). Шаблон, який дозволяє додавати функціональність до існуючих компонентів, не модифікуючи їх напряму.

Застосування цих шаблонів проєктування сприятиме модульності, повторному використанню коду, розширюваності та зручності підтримки фронтенд частини додатку.

3.3 Проєктування структури зберігання, управління даними

3.3.1 Клієнт-серверна архітектура

Веб-додаток буде розроблений з використанням клієнт-серверної архітектури, де фронтенд частина (клієнт) буде взаємодіяти з серверною частиною (бекенд) через API (Application Programming Interface).

Серверна частина буде відповідати за зберігання та управління даними про користувачів, визначні місця, плани подорожей та інші необхідні дані. Фронтенд частина буде отримувати необхідні дані від серверної частини через API та відображати їх у зручному для користувача форматі.

3.3.2 Управління станом додатку

Для управління станом сайту буде використано Context API – вбудований механізм у React для передачі даних через дерево компонентів без необхідності передавати пропси на кожному рівні.

Context API дозволяє створювати постачальників контексту (providers), які зберігають спільний стан та надають його споживачам (consumers) у будь-якому місці дерева компонентів. Це спрощує передачу даних між компонентами та робить код більш читабельним та зрозумілим.

Будуть створені кілька контекстів для управління різними частинами стану додатку:

– AuthContext. Цей контекст буде відповідати за зберігання та управління інформацією про аутентифікацію користувача. Він міститиме такі дані, як статус аутентифікації (авторизований чи ні), інформацію про поточного користувача (наприклад, ім'я користувача, електронну пошту) та методи для входу та виходу з системи;

– TripContext. Цей контекст буде відповідати за зберігання та управління даними, пов'язаними з подорожами користувача. Він міститиме інформацію про поточну подорож, список місць для відвідування, дати подорожі, бюджет та інші деталі. Компоненти, які відображають або взаємодіють з даними про подорожі, будуть споживати цей контекст;
– QuizContext. Цей контекст буде відповідати за зберігання та управління даними, пов'язаними з опитуваннями користувача. Він міститиме список доступних опитувань, поточні відповіді користувача та результати опитування. Компоненти, які відображають опитування або результати, будуть споживати цей контекст.

Кожен з цих контекстів буде мати свій власний файл, який експортує постачальника контексту. Ці постачальники будуть обгортати компоненти, які потребують доступу до відповідного контексту.

3.3.3 Кешування даних

Для підвищення продуктивності та зменшення кількості запитів до серверної частини буде впроваджено механізм кешування даних на клієнтській стороні. Це особливо важливо для даних, які не змінюються часто, наприклад, інформація про визначні місця або загальні налаштування користувача.

Кешування даних буде реалізовано з використанням різних підходів, таких як:

– кешування в пам'яті (in-memory caching). Дані будуть зберігатися в оперативній пам'яті браузера, що забезпечить швидкий доступ;
– кешування у сховищі браузера (browser storage caching). Дані будуть зберігатися в localStorage браузера, що дозволить зберігати їх між сесіями;
– кешування на рівні мережі (network caching). Використання механізмів кешування HTTP-запитів на рівні браузера.

Таким чином додаток покращить UX (User Experience), і користувачам буде зручно їм користуватися.

3.4 Приклади найцікавіших алгоритмів та методів

3.4.1 Алгоритм рекомендацій

У фронтенд частині «Tripster» буде реалізовано алгоритм рекомендацій для надання користувачам персоналізованих рекомендацій щодо місць для відвідування. Алгоритм базуватиметься на двох основних підходах: рекомендації на основі результатів проходження квізів та рекомендації на основі популярності місць серед усіх користувачів.

Рекомендації на основі результатів квізів. Користувачі матимуть можливість проходити різні квізи, які допоможуть визначити їхні вподобання та інтереси щодо подорожей. Кожен квіз міститиме набір запитань з варіантами відповідей, і кожна відповідь буде пов'язана з певними категоріями місць. Після проходження квізу система аналізуватиме відповіді користувача і на основі цього формуватиме список рекомендованих місць. Алгоритм працюватиме наступним чином:

- для кожної відповіді користувача визначатимуться відповідні категорії місць;
- система підраховуватиме кількість балів для кожної категорії на основі вибраних відповідей;
- місця з найбільшою кількістю балів у відповідних категоріях будуть відібрані як рекомендовані;
- рекомендовані місця будуть відсортовані за релевантністю та популярністю і відображені користувачеві.

Рекомендації на основі популярності місць серед усіх користувачів. Окрім персоналізованих рекомендацій на основі квізів, додаток «Tripster» також надаватиме рекомендації популярних місць, які користуються найбільшою популярністю серед усіх користувачів. Алгоритм аналізуватиме дані про відвідування місць, рейтинги та відгуки користувачів, а також кількість додавань місць до списку вподобаних або планів подорожей. На основі цих даних буде формуватися список найпопулярніших місць. Алгоритм працюватиме за наступними кроками:

- збір даних про взаємодію користувачів з місцями (відвідування, рейтинги, відгуки, додавання до вподобаних тощо);
- обчислення показників популярності для кожного місця на основі зібраних даних;
- сортування місць за показниками популярності у порядку спадання;
- відображення найпопулярніших місць користувачеві як рекомендації.

Алгоритм рекомендацій буде реалізований на фронтенді з використанням мови програмування JavaScript. Дані про місця, результати квізів та взаємодію користувачів будуть отримуватися з серверної частини через API.

Для забезпечення ефективності та швидкодії алгоритму рекомендацій, будуть застосовані такі оптимізації:

- кешування даних про місця та популярність на фронтенді для зменшення кількості запитів до сервера;
- застосування техніки ледачого завантаження (lazy loading) для відображення рекомендацій частинами, щоб не завантажувати відразу весь список місць.

Алгоритм рекомендацій буде постійно вдосконалюватися на основі зворотного зв'язку від користувачів та аналізу їхньої поведінки. Це дозволить покращувати якість та релевантність рекомендацій з часом.

Загалом, алгоритм рекомендацій у фронтенд частині додатку «Tripster» допоможе користувачам знаходити цікаві та релевантні місця для відвідування, враховуючи їхні особисті вподобання та популярність місць серед інших користувачів. Це зробить досвід планування подорожей більш персоналізованим та захоплюючим.

3.5 Створення UI/UX дизайну системи

UI/UX дизайн відіграє ключову роль у створенні привабливого, зручного та інтуїтивно зрозумілого користувацького інтерфейсу сайту «Tripster». Процес розробки дизайну буде складатися з кількох етапів, які забезпечать послідовне та ефективне створення візуальної частини додатку.

3.5.1 Дослідження та аналіз

Перед початком розробки користувацького інтерфейсу «Tripster» було проведено ретельне дослідження та аналіз існуючих рішень у галузі планування подорожей. Це дозволило визначити найкращі практики, ключові функції та елементи інтерфейсу, які користувачі очікують від подібних додатків.

Було проаналізовано популярні веб-сайти та додатки для планування подорожей, такі як TripAdvisor, Airbnb, Expedia. Особлива увага приділялася структурі навігації, розміщенню основних функцій, візуальним елементам та загальній естетиці інтерфейсу. Це допомогло зрозуміти, які елементи є звичними та інтуїтивними для користувачів, а також визначити можливості для вдосконалення та диференціації.

Під час дослідження були вивчені останні тенденції в галузі дизайну користувацького інтерфейсу, такі як матеріальний дизайн, адаптивний дизайн, мікровзаємодії та доступність. Також було розглянуто найкращі практики створення інтуїтивних та зручних інтерфейсів, принципи організації контенту та навігації, використання кольору, типографіки та іконок.

Рекомендації та керівні принципи для розробки користувацького інтерфейсу:

- зручна та інтуїтивна навігація – ключове завдання при проектуванні. Структура повинна мати чітку ієрархію розділів та підрозділів, логічне групування функцій та елементів. Використання знайомих іконок та міток допоможе користувачам легко орієнтуватися в інтерфейсі [12];
- зосередження на основних функціях дозволить уникнути переобтяження інтерфейсу зайвими елементами. Найважливіші функції, такі як планування подорожі, перегляд інформації та збереження спогадів, повинні бути розміщені на головній сторінці та мати простий та швидкий доступ;
- персоналізація та налаштування підвищать зручність використання застосунку. Користувачі повинні мати можливість налаштовувати

інтерфейс відповідно до своїх уподобань, а також зберігати налаштування та дані про минулі подорожі для повторного використання;

– зручність введення даних та взаємодії є важливою для забезпечення безперебійного та ефективного процесу планування подорожі. Застосування елементів керування, що автоматично заповнюються, підказок, інтеграція з календарями та іншими зовнішніми інструментами, а також можливість легко імпортувати та експортувати дані сприятимуть покращенню користувацького досвіду.

Ці рекомендації та принципи стали основою для розробки привабливого, зручного та функціонального користувацького інтерфейсу веб-застосунку «Tripster», що відповідає сучасним тенденціям та потребам користувачів у плануванні подорожей.

3.5.2 Інформаційна архітектура та прототипування

Інформаційна архітектура та прототипування є важливими етапами в процесі розробки фронтенд частини «Tripster». Вони допомагають визначити структуру та організацію контенту, навігацію та взаємодію користувача з інтерфейсом.

Інформаційна архітектура (ІА) – це процес організації, структурування та маркування контенту веб-додатку таким чином, щоб користувачі могли легко знаходити потрібну інформацію та виконувати завдання. Для сайту інформаційна архітектура буде розроблена з урахуванням потреб і цілей користувачів, а також бізнес-вимог проекту [13].

Основні принципи інформаційної архітектури, які будуть застосовані при розробці веб-додатку «Tripster»:

– ієрархічна організація. Контент буде організовано у логічну ієрархію, де головні розділи будуть розділені на підрозділи, а підрозділи – на окремі сторінки або компоненти. Це допоможе користувачам легко орієнтуватися та знаходити потрібну інформацію [14];

– інтуїтивна навігація. Навігаційні елементи, такі як меню, посилання та хлібні крихти, будуть розроблені таким чином, щоб користувачі могли

легко переміщатися між різними розділами та сторінками веб-додатку. Навігація буде послідовною та зрозумілою на всіх сторінках.

Прототипування – це процес створення спрощених макетів або інтерактивних моделей веб-додатку перед початком розробки. Прототипи допомагають візуалізувати дизайн, перевірити зручність використання та отримати зворотний зв'язок від зацікавлених сторін на ранніх етапах проекту [15].

Для сайту «Tripster» будуть створені прототипи основних сторінок та компонентів.

Сторінка списку місць (див. рис. 3.4). Прототип сторінки списку місць відображатиме список доступних місць з можливістю фільтрації, сортування та пошуку. Він також міститиме інформацію про кожне місце, таку як назва, фото, рейтинг та короткий опис.

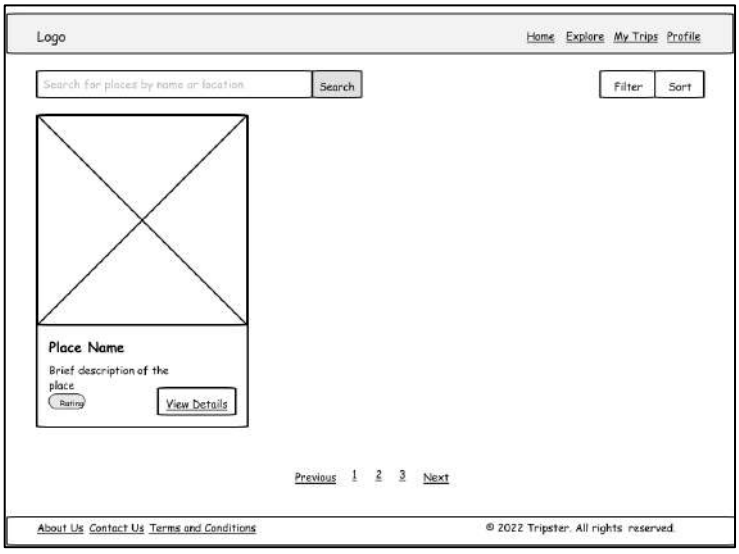


Рисунок 3.4 – Прототип сторінки списку місць (виконано самостійно)

Головна сторінка (див. рис. 3.5). Прототип головної сторінки міститиме основні розділи, такі як пошук місць, популярні напрямки та заклики до дії.

Сторінка деталей місця (див. рис. А.1). Прототип сторінки деталей місця міститиме детальну інформацію про конкретне місце, включаючи фотографії, опис, розташування на карті, відгуки та рекомендації.

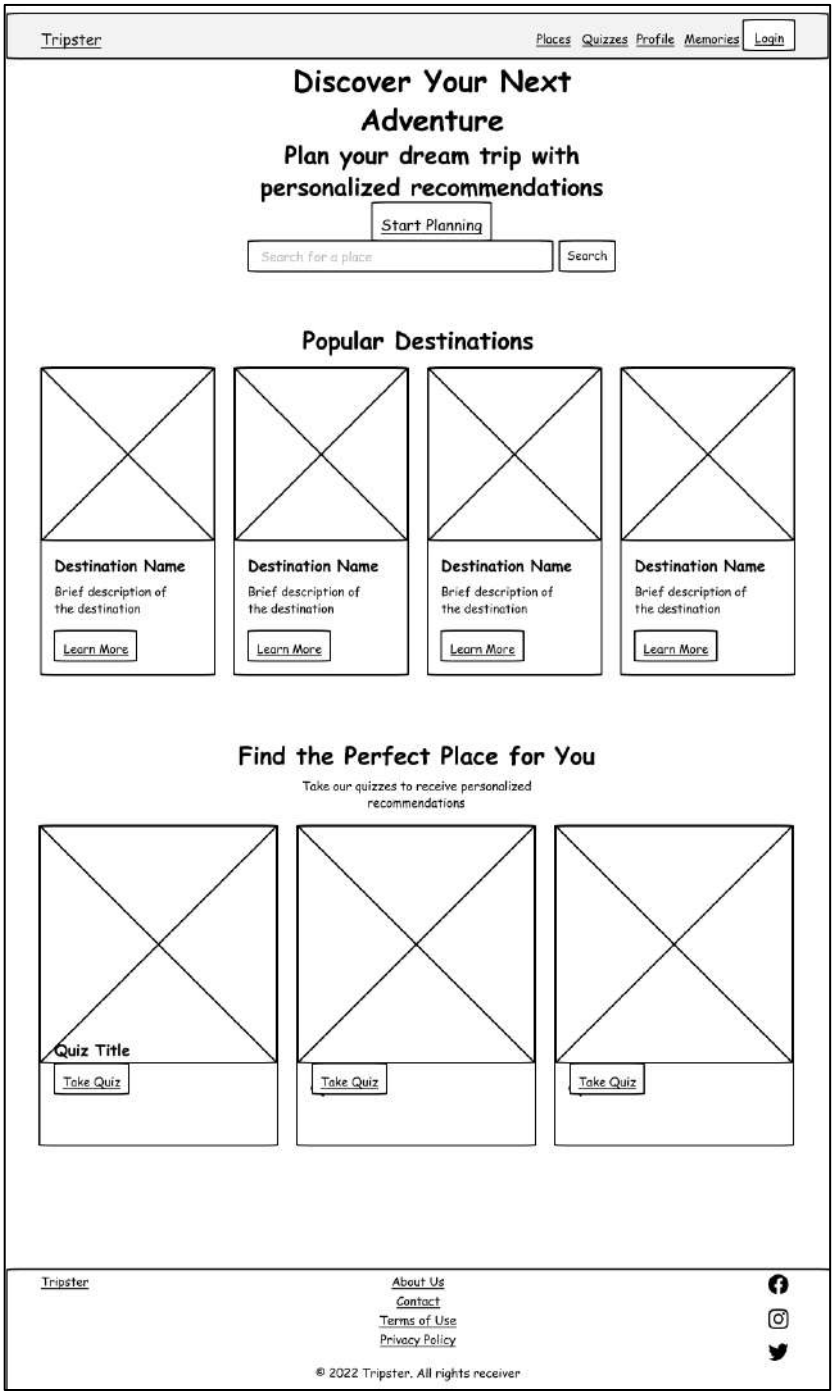


Рисунок 3.5 – Прототип головної сторінки (виконано самостійно)

Прототипи створені з використанням Figma. Правильно розроблена інформаційна архітектура та якісні прототипи закладають міцну основу для успішної реалізації проекту та забезпечення позитивного досвіду користувачів.

4 ОПИС ПРИЙНЯТНИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис структури проєкту

Веб-застосування «Tripster» має модульну та структуровану архітектуру, засновану на принципах компонентного підходу React та серверного рендерингу Next.js. Розглянемо детальніше основні складові цієї архітектури.

Структура проекту та файлова система:

- коренева директорія проекту містить стандартні файли Next.js, такі як next.config.mjs, package.json та конфігураційні файли для інструментів розробки (lint, commit, тощо);
- директорія src містить основний код додатку, поділений на підмодулі та компоненти;
- директорія src/app відповідає за маршрутизацію та містить файли сторінок (page.tsx), згруповані за функціональністю (наприклад, city, place, vacations тощо);
- директорія src/api містить функції для взаємодії з API та отримання даних;
- директорія src/components зберігає повторно використовувані компоненти користувацького інтерфейсу, такі як компоненти навігації, форми, галереї зображень тощо;
- інші директорії, такі як src/hooks, src/types, src/utils, містять допоміжні модулі, типи даних та функції для роботи з різними аспектами додатку.

Принцип роботи Next.js та маршрутизація:

- Next.js використовує підхід «file-based routing», де кожен файл у директорії src/app відповідає окремому маршруту в додатку, наприклад, файл src/app/place/[id]/page.tsx визначає сторінку з деталями певного місця, де [id] є динамічним параметром, що передається через URL;
- Next.js автоматично визначає маршрути на основі файлової структури та надає можливості для серверного рендерингу та статичного генерування сторінок.

Компонентний підхід React:

– додаток «Tripster» дотримується компонентного підходу React, де кожен компонент є незалежним і відповідає за певну частину інтерфейсу або функціональності;
– компоненти можуть бути атомарними (наприклад, кнопка або поле введення) або складеними (наприклад, форма авторизації або карусель зображень) ;
– директорія src/components містить багато повторно використовуваних компонентів, таких як ImageCarousel, GoogleMap, PlaceCarousel тощо.

Інтеграція з бібліотеками та сторонніми інструментами:

– у проекті використовуються різні бібліотеки та інструменти для покращення функціональності та продуктивності додатку;
– для роботи з UI та дизайном використовуються «Material-UI»;
– для роботи з датами та часом використовується бібліотека «dayjs»;
– для керування станом та виконання асинхронних запитів використовується «React Query»;
– для роботи з формами використовується бібліотека «react-hook-form»;
– для інтеграції з Google Maps використовується бібліотека «@react-google-maps/api».

Робота з даними та API:

– додаток взаємодіє з різними API для отримання даних про місця, користувачів, рекомендації тощо;
– у директорії src/api міститься логіка для виконання запитів до API та обробки отриманих даних;

Інші важливі складові:

– директорія src/types містить визначення типів даних для різних сутностей додатку (місця, користувачі, подорожі тощо);
– директорія src/utils містить допоміжні функції та утиліти для роботи з маршрутизацією, датами, зберіганням даних тощо.

Така архітектура забезпечує modular, масштабованість, підтримуваність та ефективність веб-застосування «Tripster», дозволяючи легко додавати нові функції, компоненти та інтегрувати з різними сервісами та API.

4.2 Авторизація та автентифікація

Авторизація та автентифікація є важливими аспектами веб-застосування «Tripster», забезпечуючи безпеку та конфіденційність даних користувачів. Для реалізації цих функцій використовуються бібліотеки «axios» та «react-query», а також хук useAuth, створений спеціально для управління станом авторизації в додатку.

4.2.1 Налаштування axios та обробка токенів

Для виконання запитів до API використовується бібліотека axios. Налаштування axios відбувається в файлі src/api/index.ts. Тут створюється екземпляр axios з базовим URL API та заголовками, включаючи заголовок «Authorization» з токеном доступу, якщо він присутній в cookie.

```
const axios = axiosDefault.create({
  baseURL: process.env.NEXT_PUBLIC_API_URL,
  headers: {
    'Content-Type': 'application/json',
    Accept: 'application/json',
    ...(getCookie(Storage.TOKEN) && {
      Authorization: `Bearer ${getCookie(Storage.TOKEN)}`
    })
  }
});
```

Токен доступу зберігається в cookie після успішної авторизації користувача. Функції для роботи з cookie (getCookie, setCookie, deleteCookie) знаходяться в файлі src/constraints/storage.ts.

4.2.2 Робота з профілем користувача

Функція getProfileInfo з файлу src/api/user.ts виконує GET-запит до /profile для отримання інформації про профіль авторизованого користувача. Отримані дані зберігаються в cookie за допомогою функції setCookie.

```
export const getProfileInfo = async (): Promise<User> => {
  const { data } = await axios.get('/profile');
  setCookie(Storage.USER, JSON.stringify(data));
  return data;
};
```

4.2.3 Реєстрація та авторизація користувачів

Функції registerUser та loginUser з файлу src/api/user.ts відповідають за реєстрацію нового користувача та вхід в систему відповідно.

Для реєстрації нового користувача виконується POST-запит до /register з даними користувача (ім'я, електронна пошта, пароль тощо).

Для авторизації користувача виконується POST-запит до /login з електронною поштою та паролем. У випадку успішної авторизації, сервер повертає токен доступу, який зберігається в cookie за допомогою функції setCookie.

4.2.4 Контекст авторизації (AuthContext)

Для управління станом авторизації в додатку використовується контекст AuthContext, реалізований в файлі src/components/providers/AuthProvider.tsx. Цей контекст надає наступні функції та стани:

- user: об'єкт, що містить дані про авторизованого користувача;
- setUser: функція для оновлення даних про користувача;
- loginMutation: функція для виконання мутації авторизації, яка викликає loginUser з переданими даними (електронна пошта та пароль) ;
- registerMutation: функція для виконання мутації реєстрації, яка викликає registerUser з переданими даними користувача;

– logout: функція для виходу з облікового запису користувача, яка видаляє токен та дані користувача з cookie;
– isLoading: стан, що вказує, чи відбувається в даний момент завантаження даних користувача або авторизація/реєстрація;
– error: об'єкт, що містить інформацію про помилки, які виникли під час авторизації/реєстрації або отримання даних користувача.

Контекст AuthContext використовується в різних частинах додатку, наприклад, на сторінках авторизації та реєстрації, для управління процесом входу/виходу користувача з системи та отримання даних про авторизованого користувача.

Завдяки ретельно продуманій архітектурі та використанню сучасних бібліотек, таких як axios та react-query, веб-застосування «Tripster» забезпечує надійну систему авторизації та автентифікації, захищаючи конфіденційні дані користувачів та забезпечуючи безпечний доступ до функціоналу додатку.

4.2.5 Інтерфейс для авторизації, реєстрації

На рисунку 4.1 зображено екран авторизації, на рисунку 4.2 – екран реєстрації.

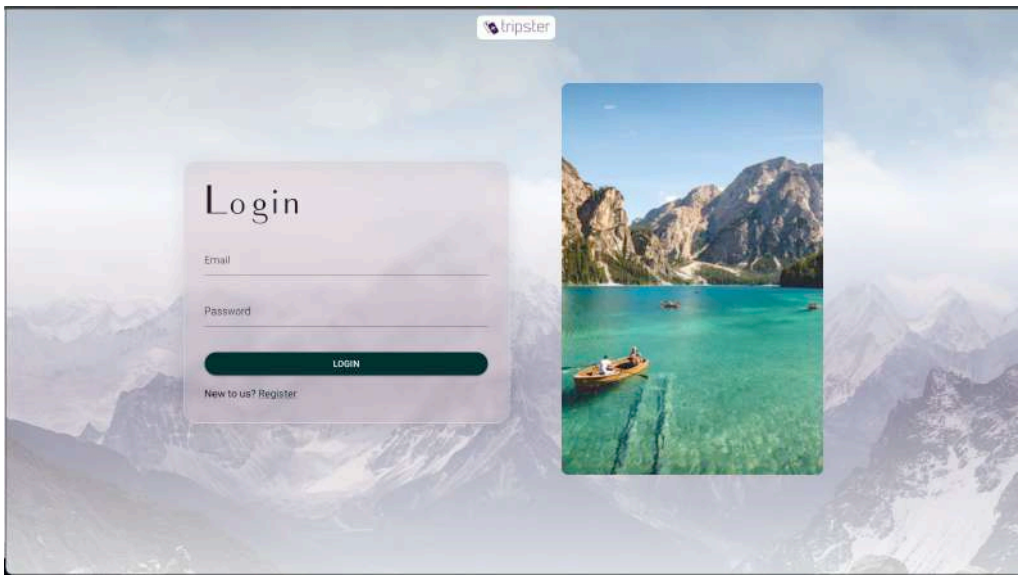


Рисунок 4.1 – Екран авторизації (виконано самостійно)

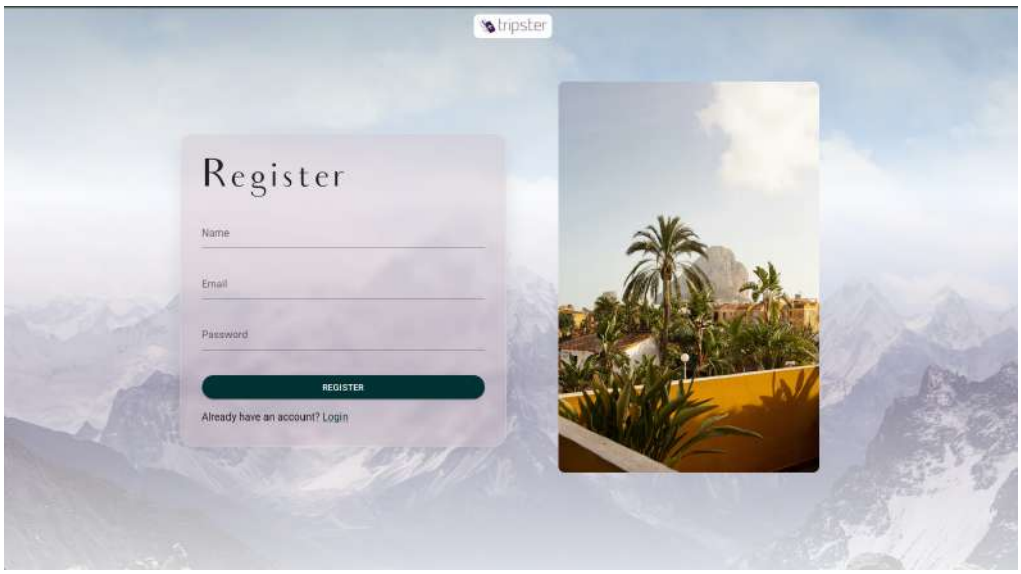


Рисунок 4.2 – Екран реєстрації (виконано самостійно)

Справа було додано UI елемент (слайдер) для привабливості інтерфейсу. Налаштування теми для всіх сторінок сталі, що поєднує всі сторінки в одне ціле.

4.3 Перегляд визначних місць

Після успішної авторизації користувач потрапляє на головну сторінку, яка відображає визначні місця, а також дає змогу шукати конкретне місце, фільтрувати їх за типом (див. рис. 4.3).

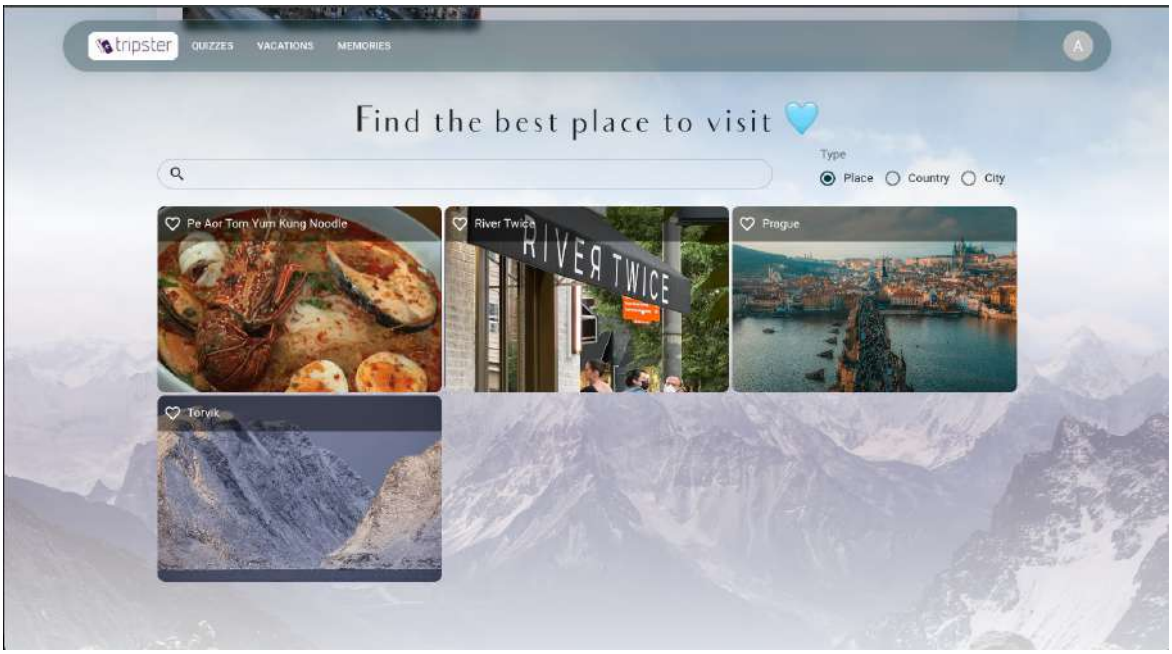


Рисунок 4.3 – Екран визначних місць (виконано самостійно)

Для користувачів, які пройшли квізи, дана сторінка відображає секцію з рекомендованими місцями (див. рис. 4.4).

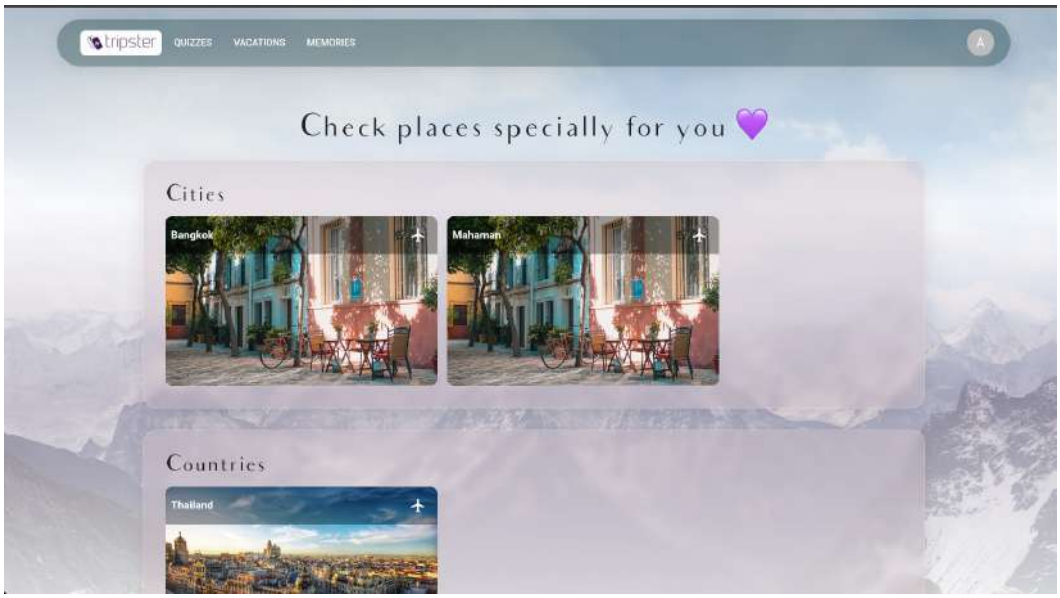


Рисунок 4.4 – Відображення рекомендованих місць (виконано самостійно)

Також реалізовано детальний перегляд кожного з місць і можливість додавання їх до обраних (див. рис. 4.5, 4.6).

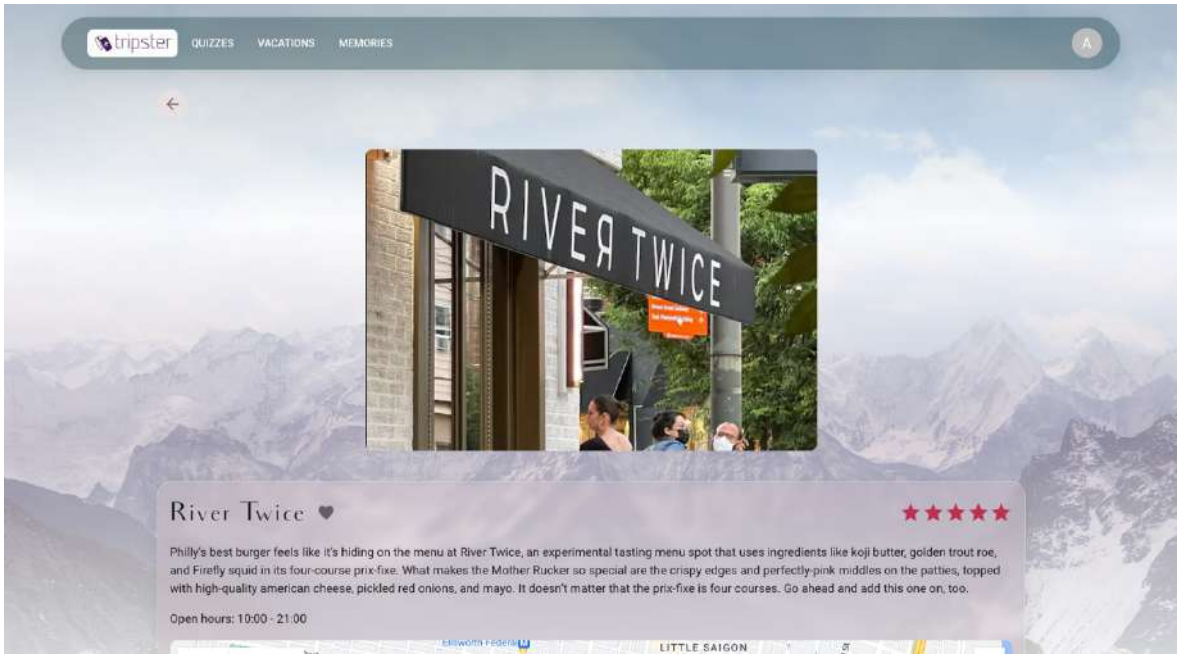


Рисунок 4.5 – Детальна інформація про місце (виконано самостійно)

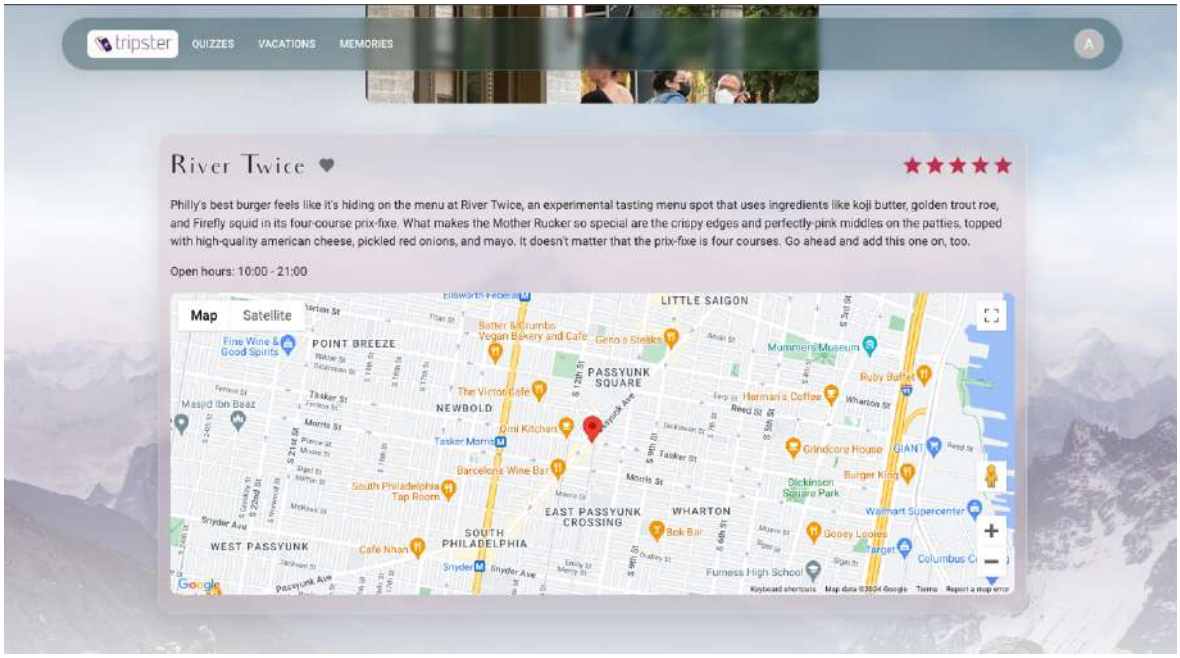


Рисунок 4.6 – Детальна інформація про місце (виконано самостійно)

Використано Google Maps API для підключення карт з локаціями.

Отримання даних з бекенду відбувалось через інтерфейс бібліотеки «react-query». Функція useQuery дозволяє виконувати асинхронні запити та отримувати відповідь, коли вона стане доступною:

```
const {
  data: places,
  isLoading: isLoadingPlaces,
  isError: isErrorPlaces
} = useQuery({
  queryKey: ['places'],
  queryFn: getAllPlaces
});
```

А функція getAllPlaces натомість використовує екземпляр класу axios для звернення до API, в даному випадку виконує GET запит:

```
export const getAllPlaces = async (): Promise<PlaceInCollection[]> =>
{
  const { data } = await axios.get<AllPlacesResponse>('/places');
  return data.placeCollection.items;
};
```

Завдяки ретельно продуманому дизайну та ефективному використанню бібліотек і API, веб-застосування «Tripster» надає зручний та інформативний інтерфейс для перегляду визначних місць, а також персоналізовані рекомендації на основі інтересів користувача.

4.4 Проходження вікторин для визначення особистих інтересів

В шапці, яку бачить авторизований користувач є посилання на сторінку з вікторинами, зображено це на рисунку 4.7.

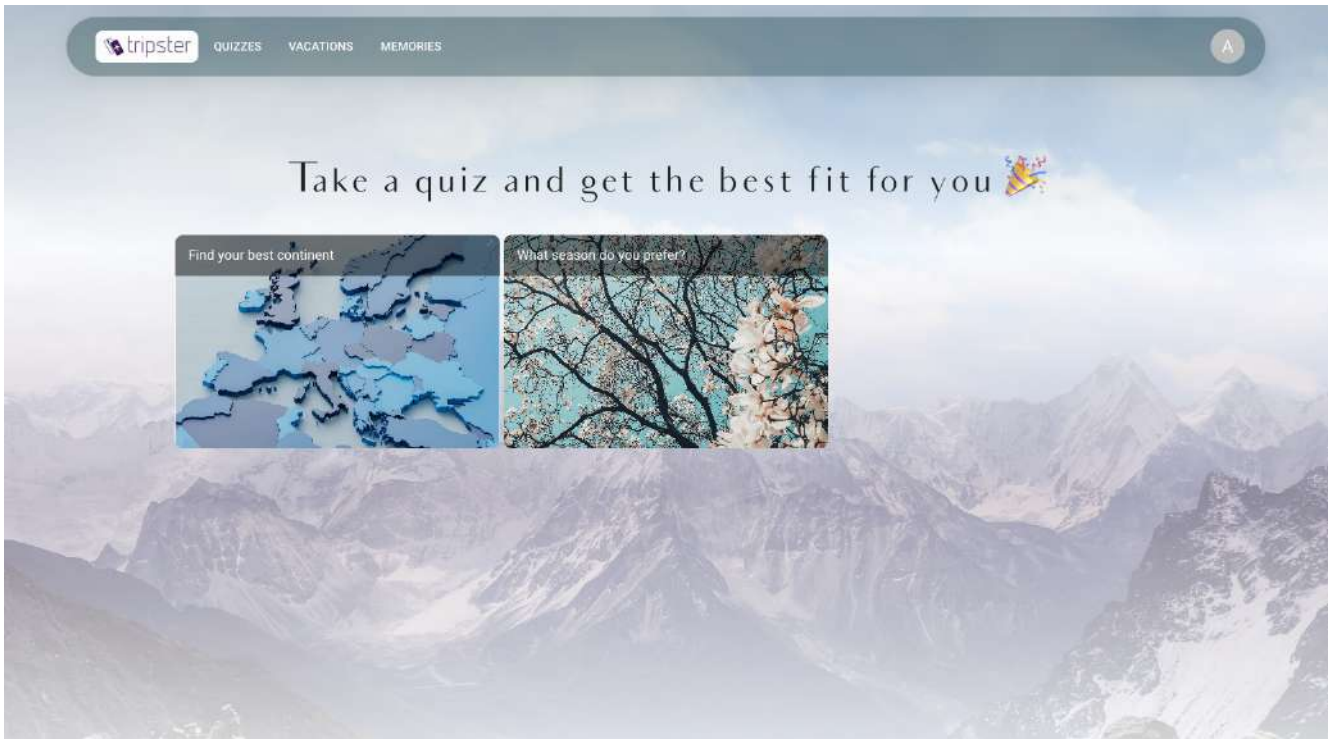


Рисунок 4.7 – Сторінка з вікторинами (виконано самостійно)

Сама вікторина складається з головної сторінки (див. рис. 4.8), запитань (див. рис. 4.9) і результатів (див. рис. 4.10). Ці всі стани регулюються сторінкою quizzes/{id}. Логіка розмітки:

```
return (
  <LayoutAuthorized containerStyles={{ overflow: 'hidden' }}>
    {resultsData ? (
      <QuizResult result={resultsData} hasStartButton />
```

```
    ) : isQuizStart ? (
      <QuizStart
        quiz={quiz}
        quizData={quizData}
        timeToPassQuiz={timeToPassQuiz}
      />
    ) : (
      <QuizStep quizData={quizData} />
    )}
  </LayoutAuthorized>
);
```

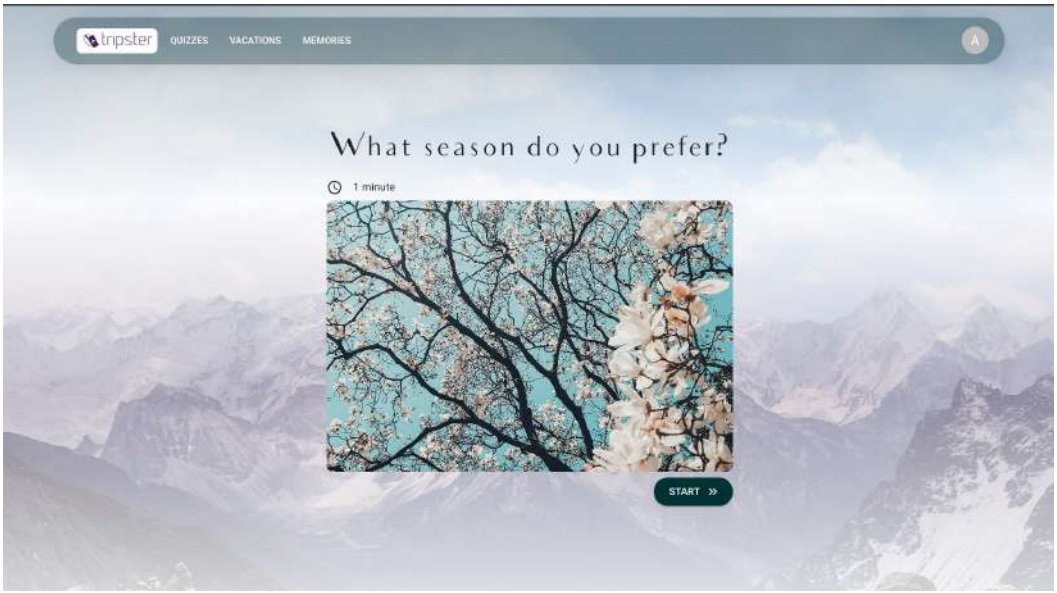


Рисунок 4.8 – Головна сторінка вікторини (виконано самостійно)

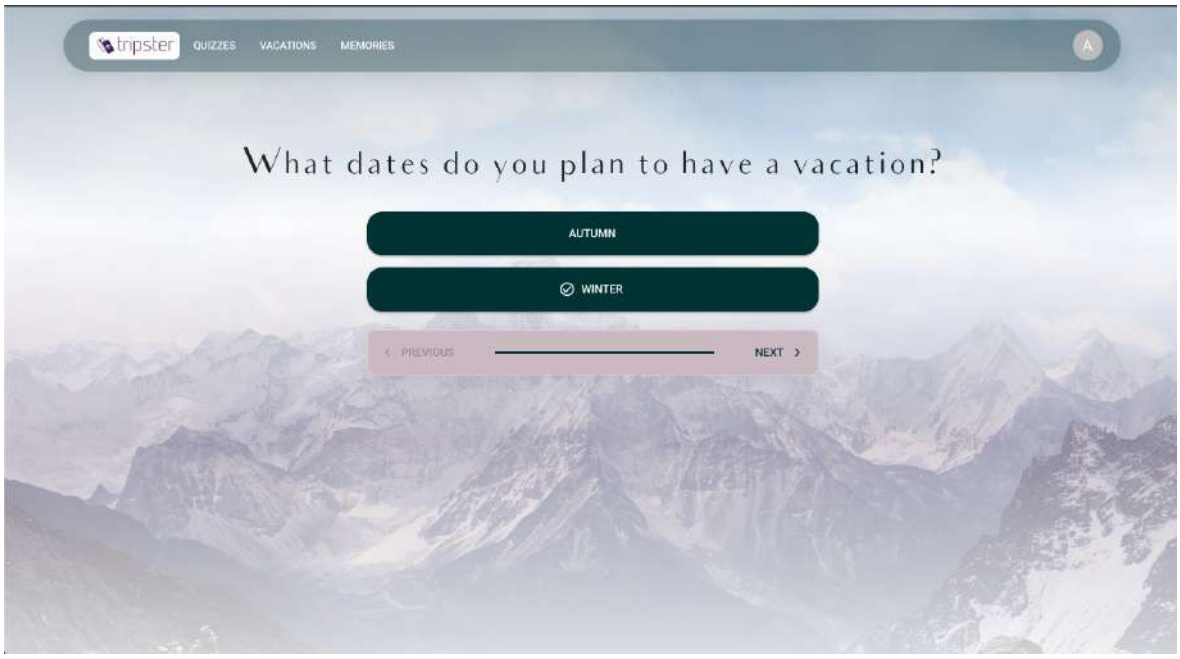


Рисунок 4.9 – Сторінка запитання в вікторині (виконано самостійно)

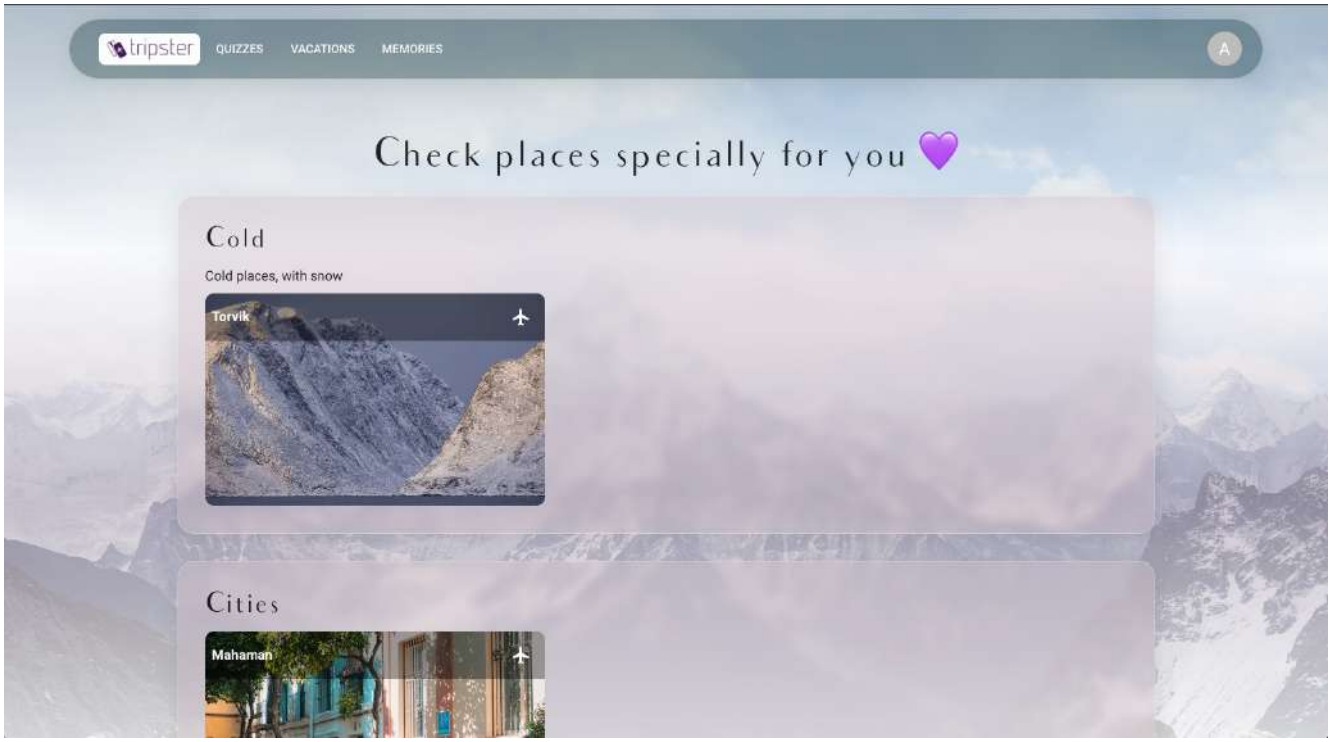


Рисунок 4.10 – Сторінка персоналізованих рекомендацій (виконано самостійно)

Для універсальності тестів було розроблено хук useQuiz, який мав закрити всі потреби базової логіки тестів (навігація по запитаннях, збереження прогресу, визначення результатів тощо). Далі наведено типи, які використовуються хуком як значення що вертається:

```
export type QuizData = {
  currentQuestionIndex: number;
  isQuizStart: boolean;
  isFirstStep: boolean;
  isLastStep: boolean;
  goToQuizStart: () => void;
  goToFirstQuizStep: () => void;
  goToNextQuizStep: () => void;
  goToPrevQuizStep: () => void;
  onQuizStepSubmit: (answerId: string) => void;
  currentStep: Quiz['questionsCollection']['items'][0];
  currentStepState: QuizInStorage[0] | null;
  stepsCount: number;
};

type UseQuizResult = {
  timeToPassQuiz?: string;
  onQuizStepSubmit: (answerId: string) => void;
  quizData?: QuizData;
```

```
  resultsData?: QuizResult;
} & UseQueryResult<Quiz, DefaultError>;
```

Для того, щоб прогрес користувача зберігався, навіть якщо він не встиг пройти весь тест до кінця, при сабміті одного кроку з запитанням, воно зберігається в local storage. Таким чином згодом можна повернутись до тої ж вікторини й продовжити проходження.

Дана концепція є дуже важливою задля персоніфікації й зручності користувачам створювати подорожі, а також розуміти, де саме їм має сподобатись найбільше.

4.5 Мандрівки

Мандрівки є головним функціоналом даної програмної системи, оскільки напрямлені саме на планування подорожей, шляхом додавання всієї необхідної інформації в інтерфейс самої подорожі. На рисунку 4.11 зображено сторінку з переліком всіх мандрівок створених користувачем.

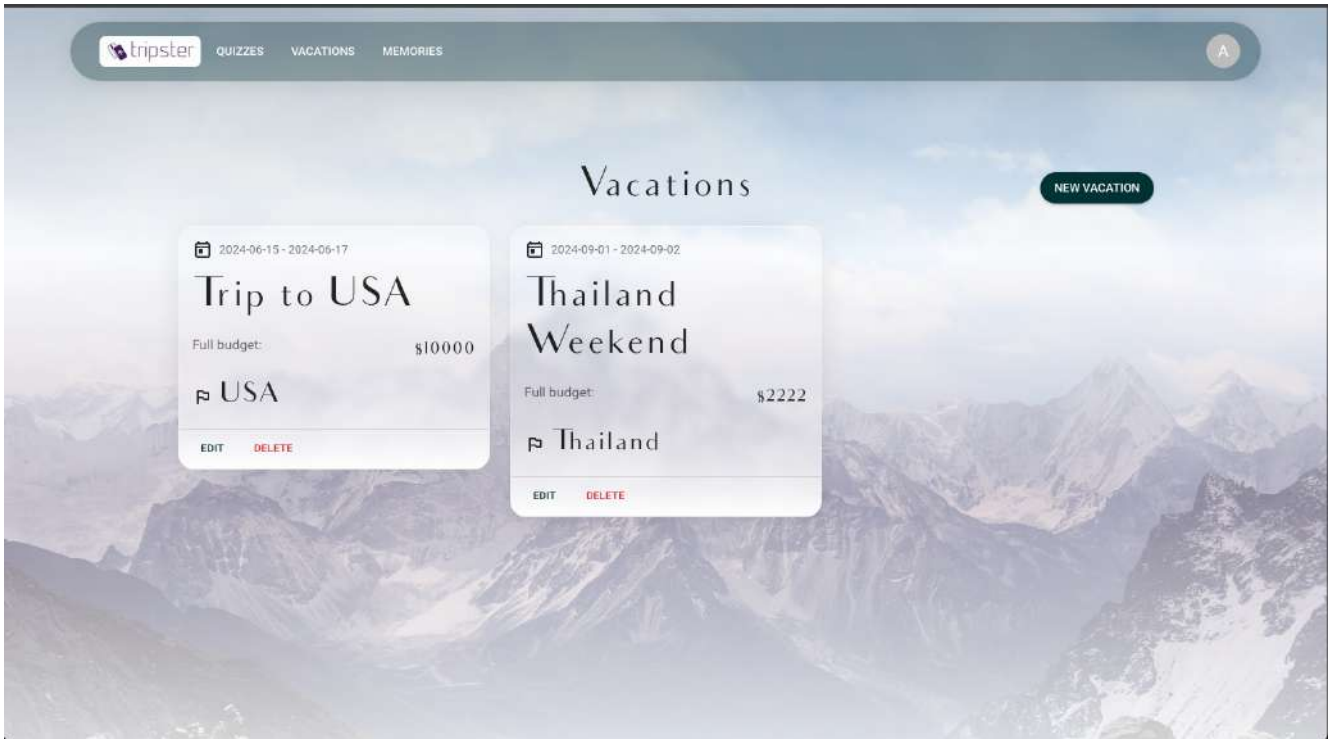


Рисунок 4.11 – Сторінка мандрівок (виконано самостійно)

Є можливість переглянути їх більш детально, що показано на рисунках 4.12, 4.13, 4.14. Було додано карту, яка позначає країну, можливість редагування й видалення.

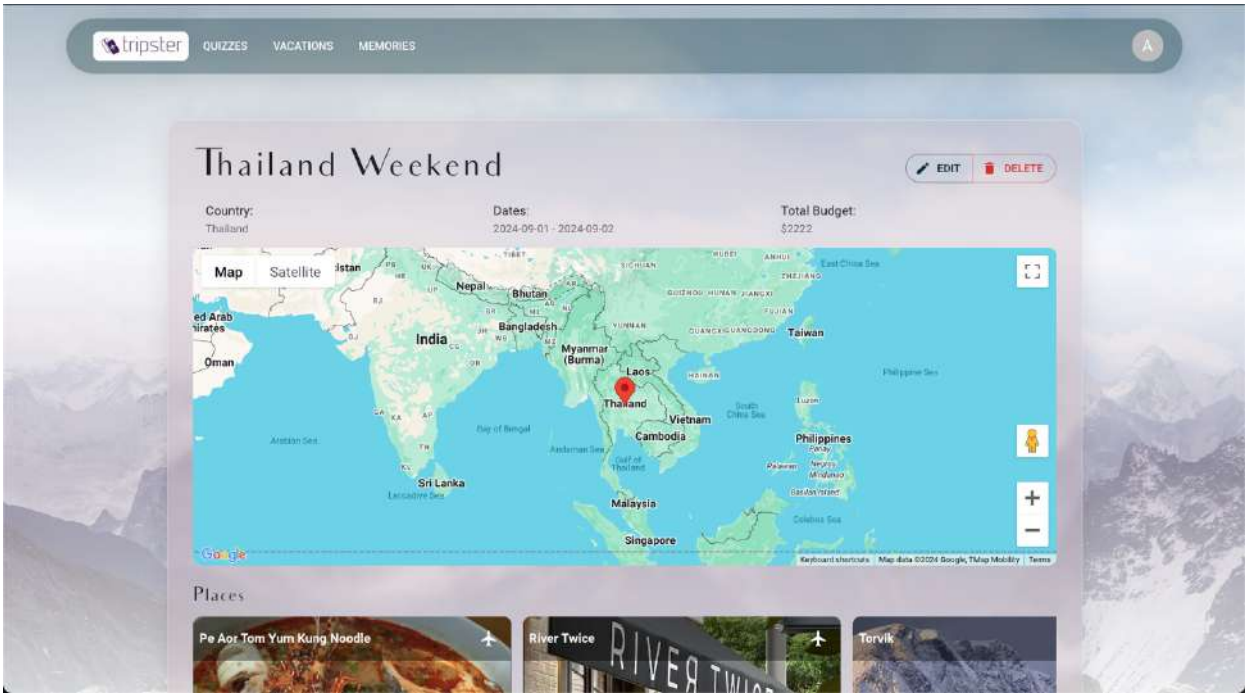


Рисунок 4.12 – Сторінка мандрівки (виконано самостійно)

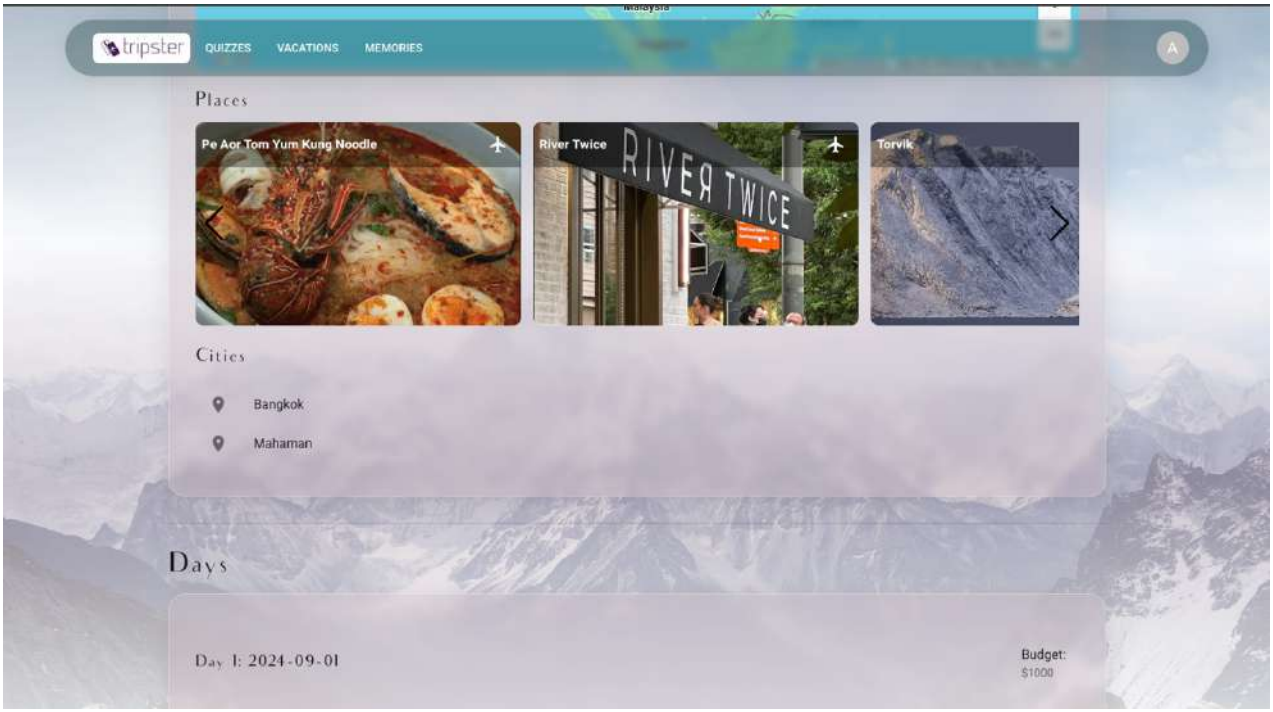


Рисунок 4.13 – Сторінка мандрівки (виконано самостійно)

Також сторінка інформації про мандрівку має карусель визначних місць й міст який турист планує відвідати. Присутні плани на день.

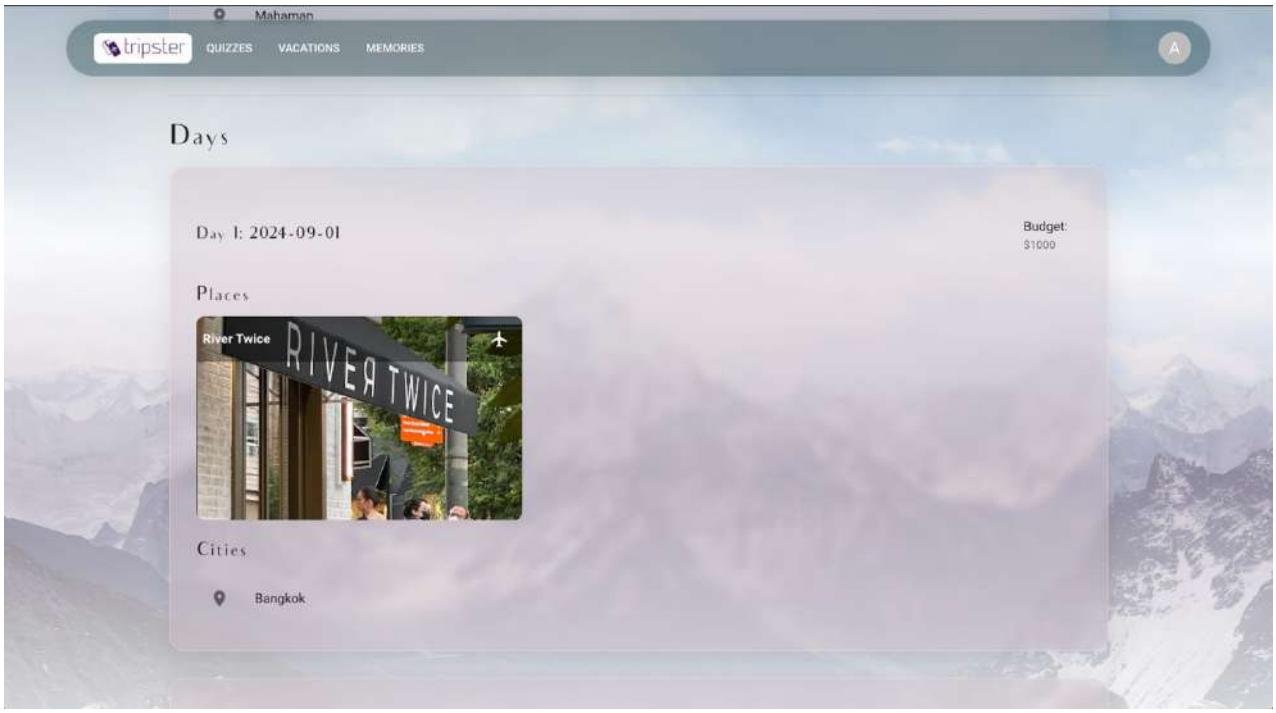


Рисунок 4.14 – Сторінка мандрівки (виконано самостійно)

Форма створення та оновлення подорожі зображена на рисунках 4.15, 4.16.

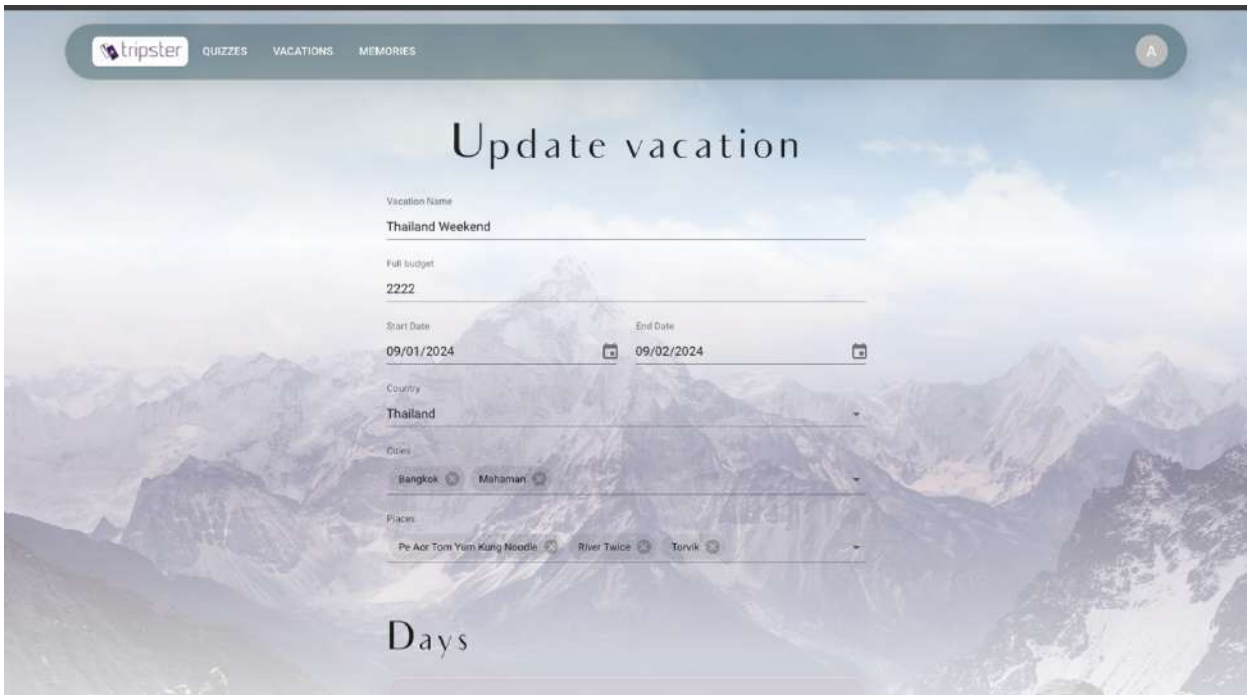


Рисунок 4.15 – Сторінка додавання, редагування мандрівки (виконано самостійно)

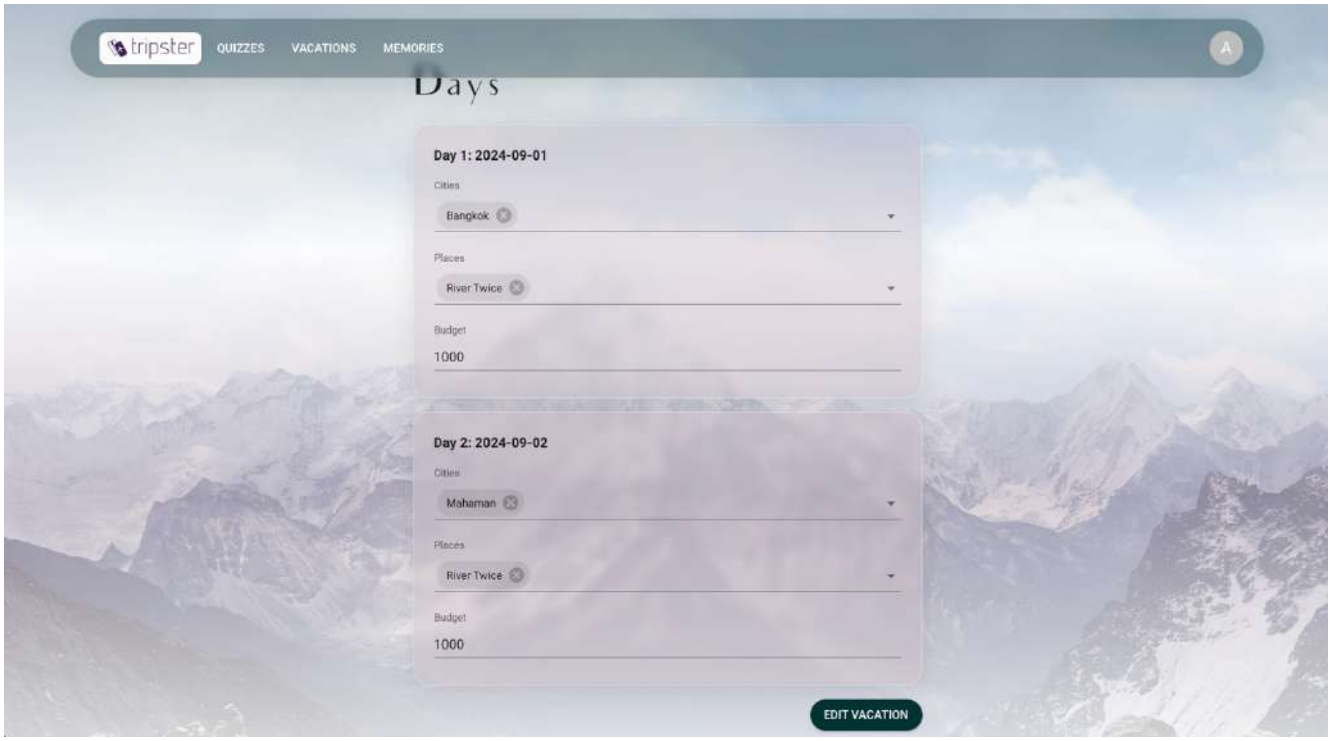


Рисунок 4.16 – Сторінка додавання, редагування мандрівки (виконано самостійно)

Реалізована така форма за допомогою бібліотеки «react-hook-form», яка допомагає контролювати всі поля, їх властивості, валідацію та стан. Використано UI компоненти такі як: Autocomplete, DatePicker для зручного заповнення інформації.

Далі наведено реалізацію компонента для додавання стартової дати подорожі:

```
<Controller
  control={control}
  name='dateDiapazone.startDate'
  rules={{ required: 'Start date is required' }}
  render={({ field: { value, ref, onChange } }) => {
    return (
      <DatePicker
        minDate={dayjs()}
        label='Start Date'
        value={value}
        inputRef={ref}
        onChange={onChange}
        slotProps={{
          textField: {
            error: !!errors.dateDiapazone?.startDate,
            helperText:
              errors.dateDiapazone?.startDate &&
              'Start date is required'
          }
```

```
            }}
            sx={{ flex: 1 }}
          />
        );
      }}
    />
```

Controller – спеціальний компоненти з «react-hook-form», який дозволяє контролювати будь який input компонент. DatePicker – компонент для вибору дати. Стан цього інпуту ми контролюємо самі через пропси value, onChange.

Отже, користувачі можуть ефективно планувати мандрівки прямо на веб-сайті «Tripster».

4.6 Згадки про минулі подорожі

Так як програмна система «Tripster» супроводжує користувача на всьому життєвому циклі мандрівки, необхідним було створити згадки, до яких він може звертатится. Тому на сторінці /memories можна проглянути галереї поїздок, якщо такі були (див. рис. 4.17). Сама галерея зображена на рисунку 4.18.

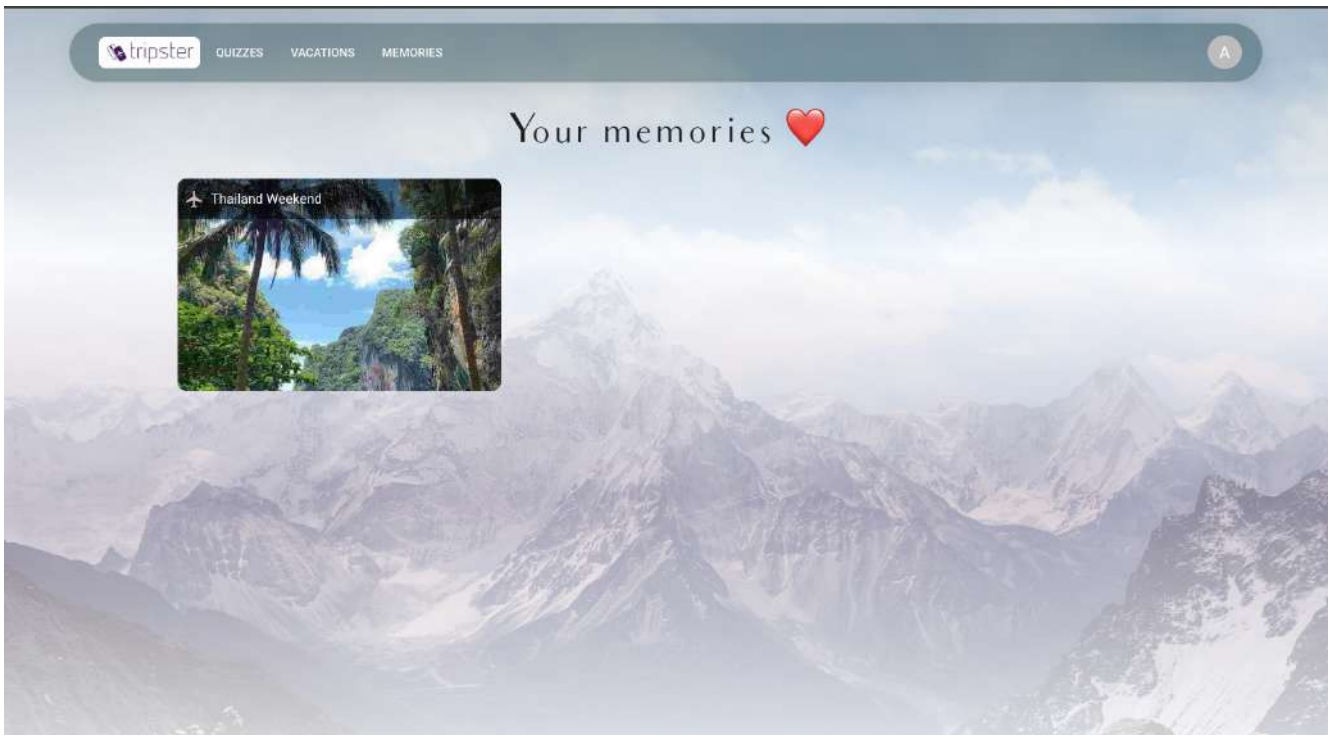


Рисунок 4.17 – Сторінка згадок (виконано самостійно)

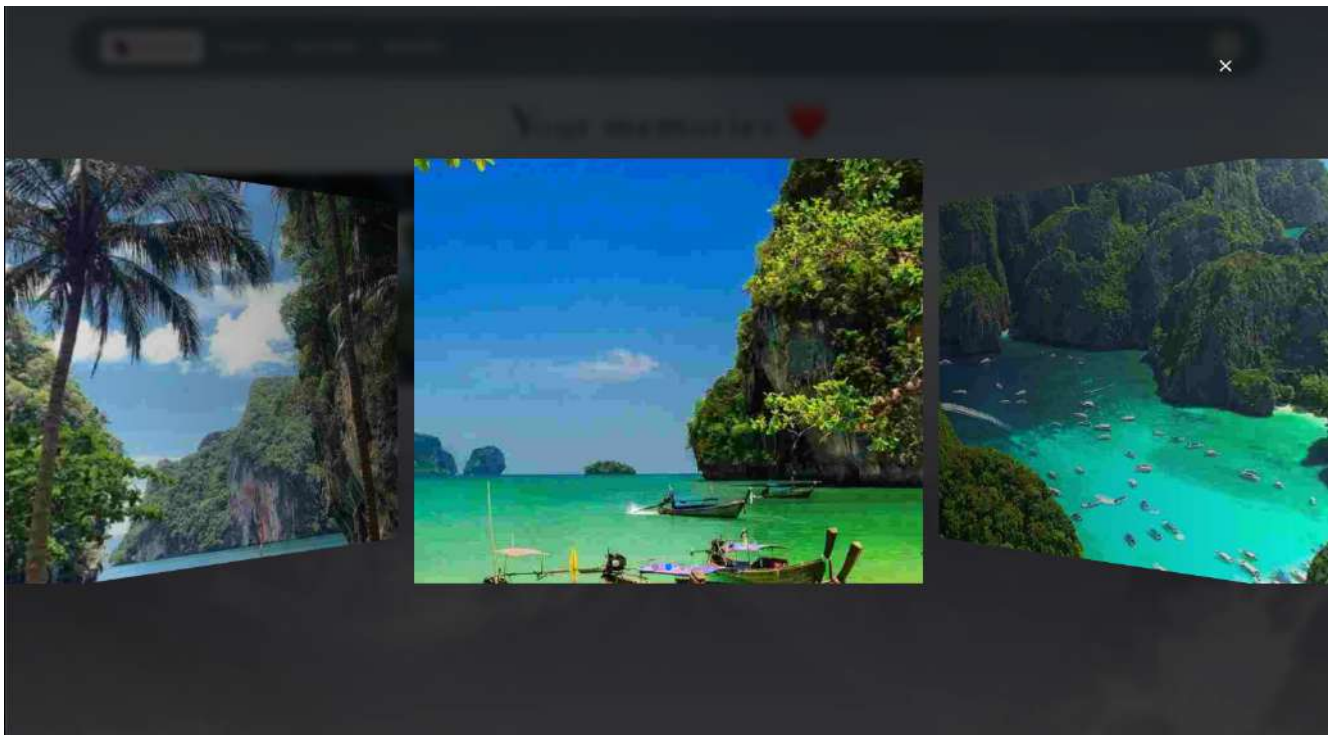


Рисунок 4.18 – Галерея подорожі (виконано самостійно)

Отже, мандрівник в будь який момент часу може звернутися до минулих подорожей та нагадати собі про отримані емоції.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для забезпечення високої якості та надійності клієнтської частини програмної системи «Tripster» було проведено мануальне тестування. Цей підхід дозволив детально перевірити користувацький інтерфейс, інтерактивні елементи та поведінку додатку в різних сценаріях використання.

5.1. Процес мануального тестування

Мануальне тестування веб-додатку «Tripster» проводилося відповідно до розроблених тест-кейсів, які охоплювали різні функціональні аспекти додатку. Тест-кейси містили детальний опис кроків тестування та очікуваних результатів.

Для ефективного відстеження та управління тест-кейсами використовувався спеціалізований інструмент TestRail. Цей інструмент дозволяв зберігати тест-кейси, відстежувати статус їх виконання, призначати їх та генерувати звіти про результати тестування.

5.2. Звітування про тестування

Після створення тест-кейсів, було проведено тестування, результати зображено на рисунку 5.1.

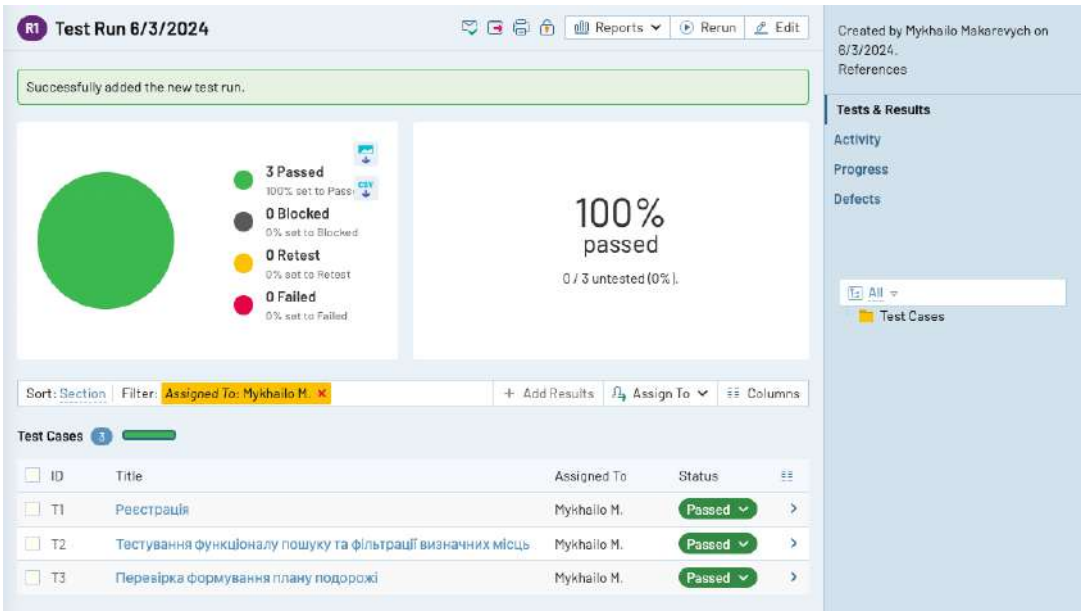


Рисунок 5.1 – Результати проходження тестів (виконано самостійно)

Завдяки ретельному підходу до мануального тестування та використанню різноманітних методів перевірки функціональності, користувацького інтерфейсу, сумісності та продуктивності, було забезпечено високу якість та надійність клієнтської частини програмної системи «Tripster».

Тест-кейс №1 для перевірки успішності реєстрації користувача зображено на рисунку 5.2.

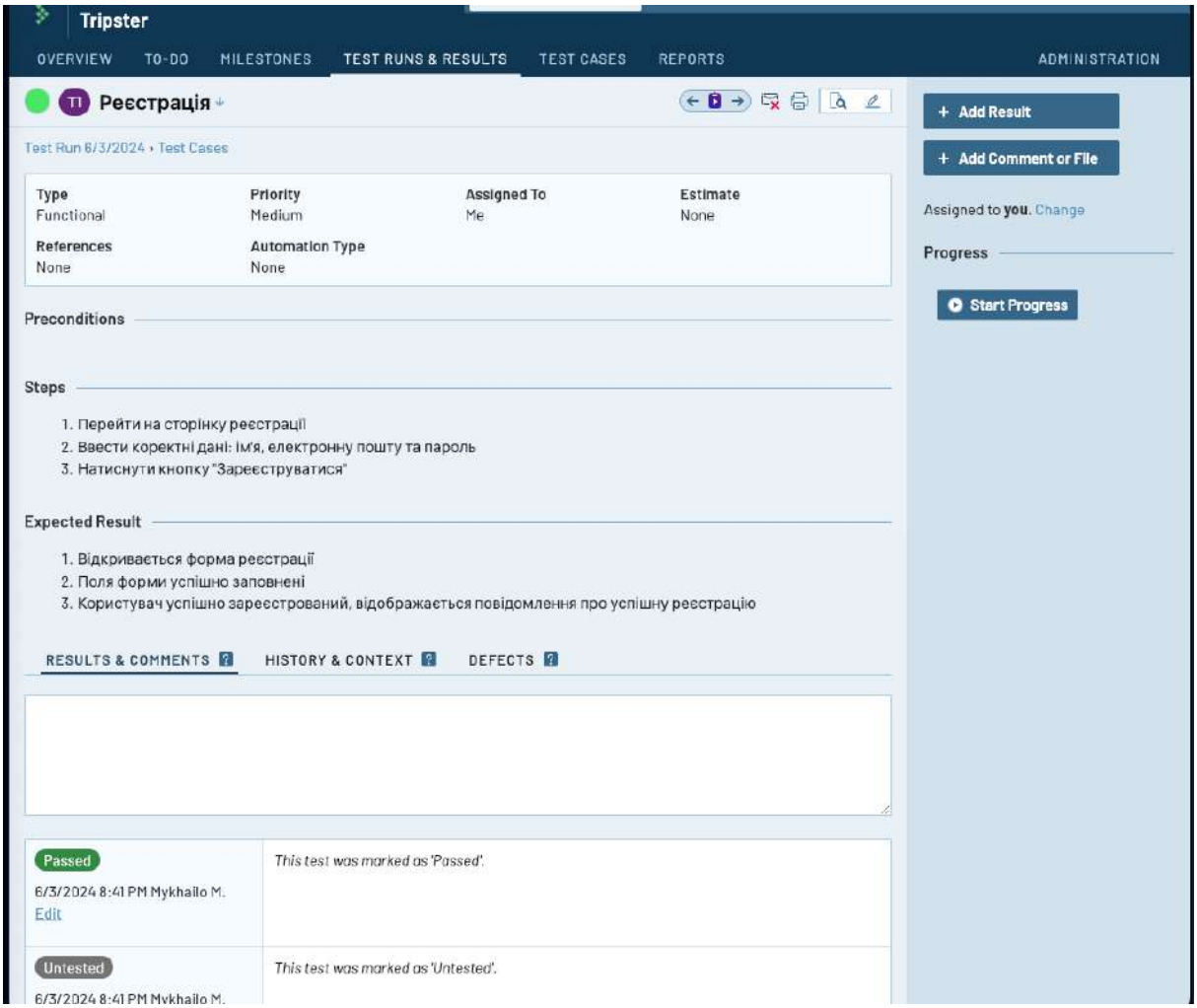


Рисунок 5.2 – Тест-кейс №1 (виконано самостійно)

Тест-кейс №2 «Тестування функціоналу пошуку та фільтрації визначних місць» зображено на рисунку 5.2.

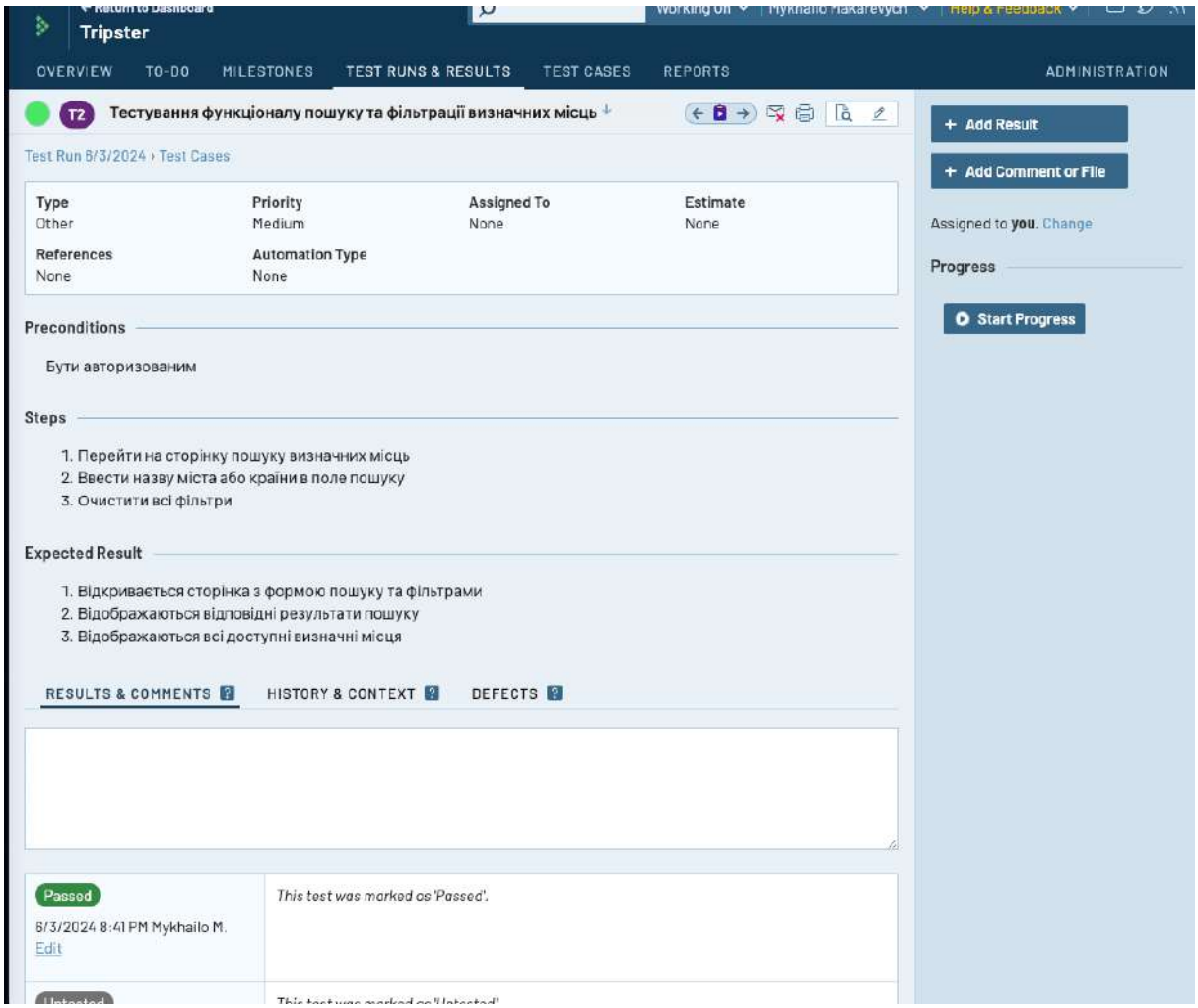


Рисунок 5.3 – Тест-кейс №2 (виконано самостійно)

Отже, всі тести успішно пройдені.

5.3. Переваги та недоліки мануального тестування

Мануальне тестування має як переваги, так і недоліки, які важливо враховувати під час планування та проведення тестування.

Переваги мануального тестування:

– гнучкість: мануальне тестування дозволяє легко адаптуватися до змін вимог або поведінки додатку без необхідності внесення змін у автоматизовані скрипти;

– реалістична імітація дій користувача: тестувальники можуть імітувати реалістичні сценарії використання додатку, враховуючи різноманітні умови та поведінку користувачів;

- перевірка зручності використання: під час мануального тестування можна ефективно оцінити зручність використання додатку, інтуїтивність інтерфейсу та загальний досвід користувача;
- виявлення непередбачуваних проблем: тестувальники можуть виявляти непередбачувані проблеми або помилки, які важко передбачити під час автоматизованого тестування.

Недоліки мануального тестування:

- трудомісткість: мануальне тестування може бути трудомістким процесом, особливо для великих проектів з великою кількістю тест-кейсів;
- людський фактор: результати тестування можуть залежати від уваги, досвіду та уважності тестувальників, що може призвести до помилок або пропущених випадків;
- складність відтворення: іноді може бути складно відтворити певні проблеми або помилки, виявлені під час мануального тестування;
- обмежена масштабованість: для великих проектів з великою кількістю тест-кейсів мануальне тестування може бути неефективним та вимагати значних ресурсів;
- відсутність автоматизації: мануальне тестування не передбачає автоматизацію, що може призвести до повторної роботи під час кожного циклу тестування.

Незважаючи на певні недоліки, мануальне тестування залишається важливим етапом у забезпеченні якості програмного забезпечення. Його часто поєднують з автоматизованим тестуванням для досягнення максимальної ефективності та охоплення різних аспектів тестування.

У випадку веб-додатку «Tripster» мануальне тестування було найкращим вибором на даному етапі розробки, оскільки воно дозволило ретельно перевірити користувацький інтерфейс, функціональність та зручність використання додатку. Однак, у майбутньому, для підвищення ефективності та масштабованості процесу тестування, може бути розглянуто впровадження автоматизованого тестування для певних типів тест-кейсів.

ВИСНОВКИ

У ході розробки кваліфікаційної роботи бакалавра «Програмна система для подорожей визначними місцями. Front-end» було успішно реалізовано фронтенд частину даної програмної системи. Основною метою проєкту було створення зручного та привабливого користувацького інтерфейсу, який спрощує процес планування подорожей та надає користувачам необхідні інструменти для пошуку, збереження та організації інформації про визначні місця.

У результаті виконання проєкту було досягнуто наступних результатів:

- проведено аналіз предметної області та визначено основні вимоги до функціональності та дизайну фронтенд частини «Tripster». Було враховано потреби потенційних користувачів та проаналізовано існуючі рішення на ринку;
- розроблено архітектуру фронтенд частини додатку з використанням сучасних технологій та підходів, таких як React, Next.js та Material UI. Архітектура була побудована з урахуванням принципів модульності, масштабованості та зручності підтримки;
- реалізовано функціональні можливості фронтенд частини веб-додатку, такі як пошук та фільтрація місць, збереження вподобаних місць, планування подорожей, перегляд детальної інформації про місця та інтеграція з серверною частиною через API. Було забезпечено плавну та інтуїтивно зрозумілу взаємодію користувача з додатком;
- впроваджено механізми управління станом додатку за допомогою React Context API, що дозволило ефективно передавати дані між компонентами та забезпечити синхронізацію стану у всьому додатку;
- розроблено алгоритм рекомендацій, який надає користувачам персоналізовані підказки щодо місць, які можуть їх зацікавити, на основі результатів опитувань та аналізу вподобань інших користувачів. Це допомагає користувачам відкривати нові та релевантні місця для подорожей;

- проведено ретельне тестування фронтенд частини веб-додатку, щоб забезпечити її коректну роботу, відповідність вимогам та відсутність критичних помилок. Було виконано юзабіліті-тестування з реальними користувачами для отримання зворотного зв'язку та подальшого вдосконалення користувацького досвіду;
- оптимізовано продуктивність та швидкість завантаження веб-додатку шляхом застосування техніки ледачого завантаження, мінімізації та стиснення ресурсів, а також використання ефективних алгоритмів та структур даних.

Фронтенд частина програмної системи «Tripster» відповідає поставленим вимогам та забезпечує зручний і привабливий користувацький інтерфейс для планування подорожей. Додаток надає користувачам необхідні інструменти для пошуку, збереження та організації інформації про визначні місця, а також персоналізовані рекомендації для відкриття нових напрямків.

Проєкт має потенціал для подальшого розвитку та вдосконалення. Можливі напрямки включають інтеграцію з додатковими джерелами даних про визначні місця, розширення функціональності планування подорожей, додавання можливості створення спільних подорожей для групи користувачів та інтеграцію з соціальними мережами для обміну враженнями та рекомендаціями.

Загалом, розробка сайту «Tripster» була успішно завершена, і проєкт готовий до подальшого розгортання та використання користувачами. Отриманий досвід та знання, здобуті під час роботи над проєктом, будуть корисними для майбутніх проєктів та професійного розвитку в галузі фронтенд розробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Statistics of tourism. UNWTO. URL: https://www.unwto.org/tourism-statistics/ (дата звернення: 09.04.2024).
2. Economic Impact Research. WTTC. URL: https://wttc.org/research/economic-impact/ (дата звернення: 09.04.2024).
3. Tripadvisor. URL: https://www.tripadvisor.com (дата звернення: 09.04.2024).
4. Google Trips. URL: https://www.google.com/travel/ (дата звернення: 09.04.2024).
5. Roadtrippers. URL: https://roadtrippers.com (дата звернення: 09.04.2024).
6. Sygic Travel. URL: https://travel.sygic.com/en (дата звернення: 09.04.2024).
7. Банкс А., Порселло Е. Адаптивний веб-дизайн з використанням Material UI. Одеса : Видавництво «Астропринт», 2018.
8. Грін А. Модульний дизайн з використанням React. Київ : Видавництво «Фоліо», 2017.
9. Стефанов С. React.js: повне керівництво для розробників. Львів : Видавництво Старого Лева, 2020.
10. Бабич Н. П., Жуков І. А. Основи веб-розробки. Київ : Видавництво «Академія», 2018.
11. Фрімен А. Pro React 16. Київ : Видавництво «Діалектика», 2019.
12. Хортон С., Лінч П. Дж. Принципи успішного веб-дизайну. Харків : Видавництво «Vivat», 2020.
13. Браун Д. М. Досвід взаємодії: створення якісних UI/UX дизайнів. Київ : Видавництво «Фоліо», 2017.
14. Кедлек Т. Мікровзаємодії: деталі створюють дизайн. Львів : Видавництво Старого Лева, 2018.
15. Веру Л. Анатомія веб-сайту: ілюстрований посібник з інформаційної архітектури. Київ : Видавництво «Фоліо», 2017.

ДОДАТОК А

Прототипи інтерфейсу користувача

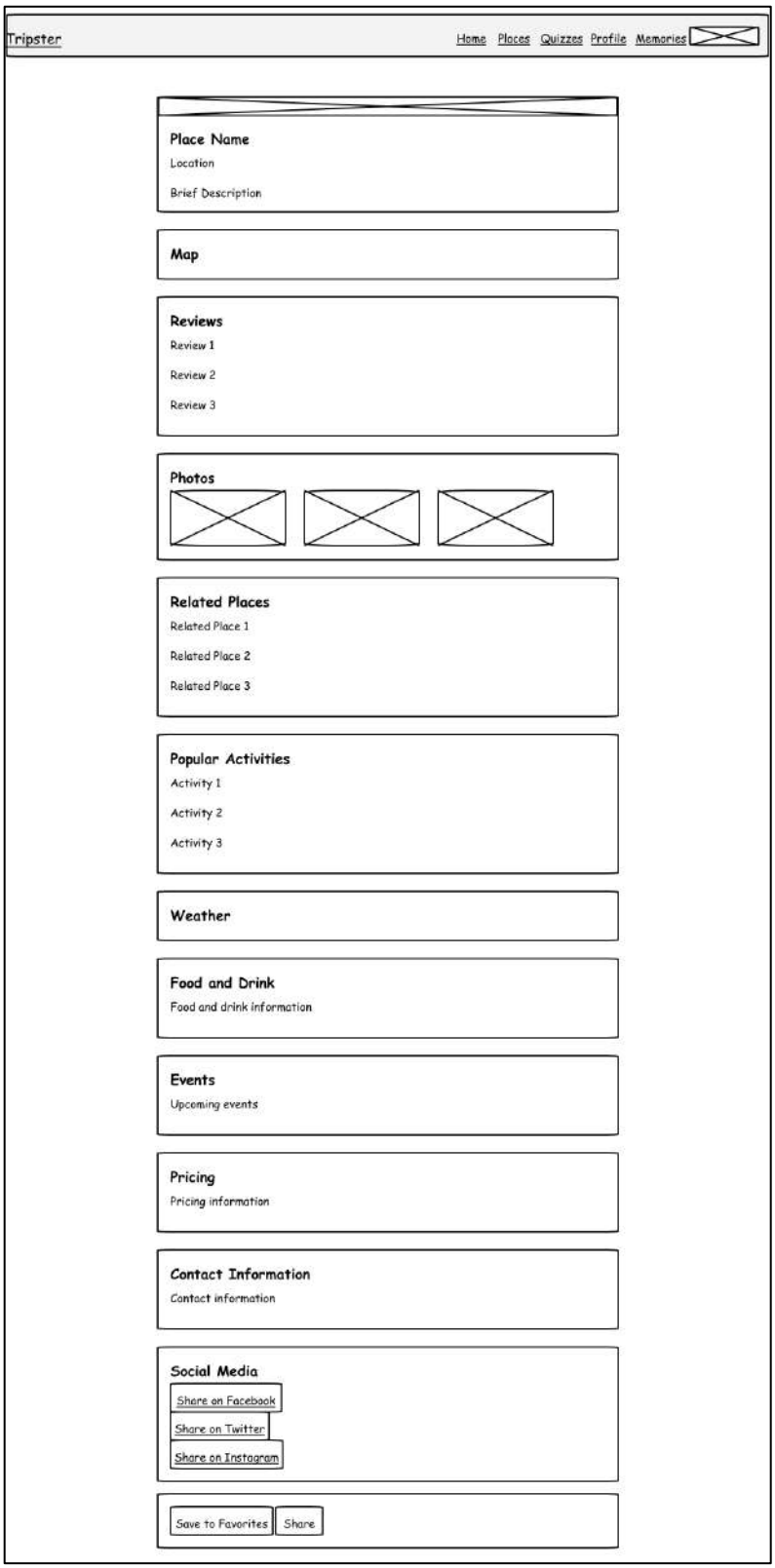


Рисунок А.1 – Прототип сторінки детальної інформації про місце (виконано самостійно)

ДОДАТОК Б

Сертифікат про участь в конференції «Perspectives of contemporary science: theory and practice»

ДОДАТОК В

Текст публікації для конференції «Perspectives of contemporary science: theory and practice»

РОЗРОБКА ФРОНТЕНД ЧАСТИНИ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Макаревич Михайло Павлович
Харківський національний університет радіоелектроніки
студент кафедри Програмної інженерії
Онищенко Костянтин Георгійович
Харківський національний університет радіоелектроніки
старший викладач кафедри Програмної інженерії
м. Харків, Україна

Ключові слова: Мандрівки, react, typescript, next.js

Сучасний світ відкриває нові можливості для мандрівників, проте процес планування подорожі часто стає справжнім викликом. Існуючі рішення не завжди забезпечують зручний та інтуїтивний інтерфейс, що ускладнює ефективну організацію поїздок. Метою даного проєкту є створення фронтенд частини програмної системи для комплексного планування та організації подорожей, що надасть користувачам зручні інструменти та інтерактивний інтерфейс.

Для розробки фронтенду буде використано React [1] у поєднанні з TypeScript [2], що забезпечить типову безпеку та кращу підтримку коду. Фреймворк Next.js [3] дозволить здійснювати рендеринг на стороні сервера, підвищити продуктивність та оптимізувати додаток для пошукових систем. Для керування станом та асинхронними запитами буде застосовано React Query [4], а для стилізації інтерфейсу – бібліотека Material UI [5].

Фронтенд система матиме модульну структуру, побудовану за принципами компонентного підходу та розділення відповідальності між різними частинами додатку. Ілюстрація даного підходу наведена на рисунку 1. Pages – сторінки нашого веб-застосунку, components – компоненти, основні з яких наведено нижче, apі – всі потрібні ендпоінти, налаштування зв'язку з сервером, helpers – допоміжні функції, утиліти, налаштування,

store-налаштування глобального сховища (дані з якого можна отримати в будь-якому компоненті), hooks – хуки, assets – медіафайли (зображення, іконки).

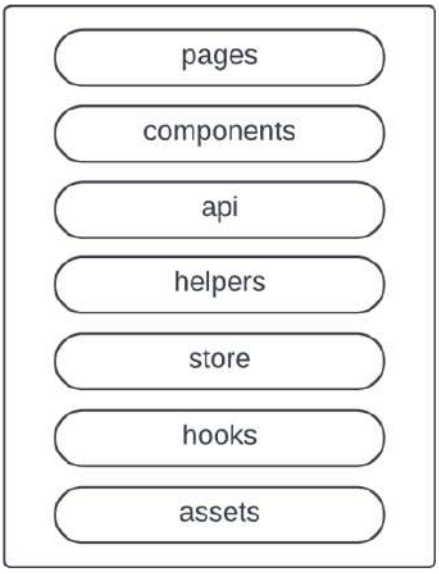


Рис. 1. Базові сутності (директорії) модульної архітектури фронтенду

Основні компоненти системи:

- Авторизація користувачів: цей компонент відповідатиме за реєстрацію, авторизацію та керування обліковими записами користувачів. Він забезпечить безпечний доступ до системи та захист конфіденційних даних користувачів;
- Планування подорожей: центральний компонент, який надаватиме користувачам інструменти для ефективного планування їхніх подорожей. Він включатиме можливості вибору дат, визначних місць, заходів, створення нотаток та планування бюджету;
- Перегляд інформації про визначні пам'ятки та місця: цей компонент забезпечить доступ до детальної інформації про визначні місця, включаючи опис, фотографії, рекомендовані заходи та маршрути.
- Створення спогадів: компонент для створення фотоальбомів та нотаток, що дозволить користувачам зберігати спогади від своїх подорожей. Він може включати можливість прив'язки фотографій та нотаток до місць на карті.

Взаємодія фронтенду з бекенд частиною відбуватиметься через спеціалізовані API (Application Programming Interface). Це забезпечить чітке розділення між клієнтською та серверною частинами додатку та полегшить їх незалежну розробку та підтримку.

Для ефективного керування асинхронними запитами до API та кешування даних буде використано бібліотеку React Query. Вона забезпечить оптимізацію продуктивності додатку, зменшить кількість непотрібних запитів та покращить користувацький досвід.

Веб-додаток матиме такі основні функції:

- Тестування для визначення найбільш підходящих місць відвідування на основі переваг та інтересів користувача;
- Отримання детальної інформації про визначні пам'ятки та місця, включаючи опис, фотографії, рекомендовані заходи та маршрути;
- Формування плану подорожі з можливістю вибору дат, визначних місць, заходів та створення нотаток для кожного дня;
- Планування бюджету подорожі з урахуванням витрат на проживання, транспорт, харчування та розваги;
- Перегляд спогадів від минулих подорожей, включаючи фотографії, нотатки та рефлексії.

Розроблений фронтенд веб-додатку для планування подорожей забезпечить комплексне та зручне рішення для самостійної організації мандрівок. Використання React, TypeScript, Next.js, React Query та Material UI гарантуватиме високу продуктивність, масштабованість та легкість підтримки системи. Ключові функції, такі як визначення найкращих місць, отримання інформації про визначні пам'ятки, планування подорожей та бюджету, а також перегляд спогадів, надаватимуть повний комплекс необхідних інструментів для планування подорожей з інтуїтивним інтерфейсом.

Розроблене рішення відрізнятиметься повнотою функціоналу, зручністю використання та високими показниками безпеки. У майбутньому планується розширення системи, зокрема, додавання можливостей спільного планування

подорожей для груп користувачів, інтеграції з сервісами навігації. Це дозволить створити ще більш комплексний та корисний сервіс для мандрівників усього світу.

СПИСОК ДЖЕРЕЛ

1. React [Електронний ресурс]. URL: https://react.dev/ (дата звернення: 26.03.2024).

2. TypeScript [Електронний ресурс]. URL: https://www.typescriptlang.org (дата звернення: 26.03.2024).

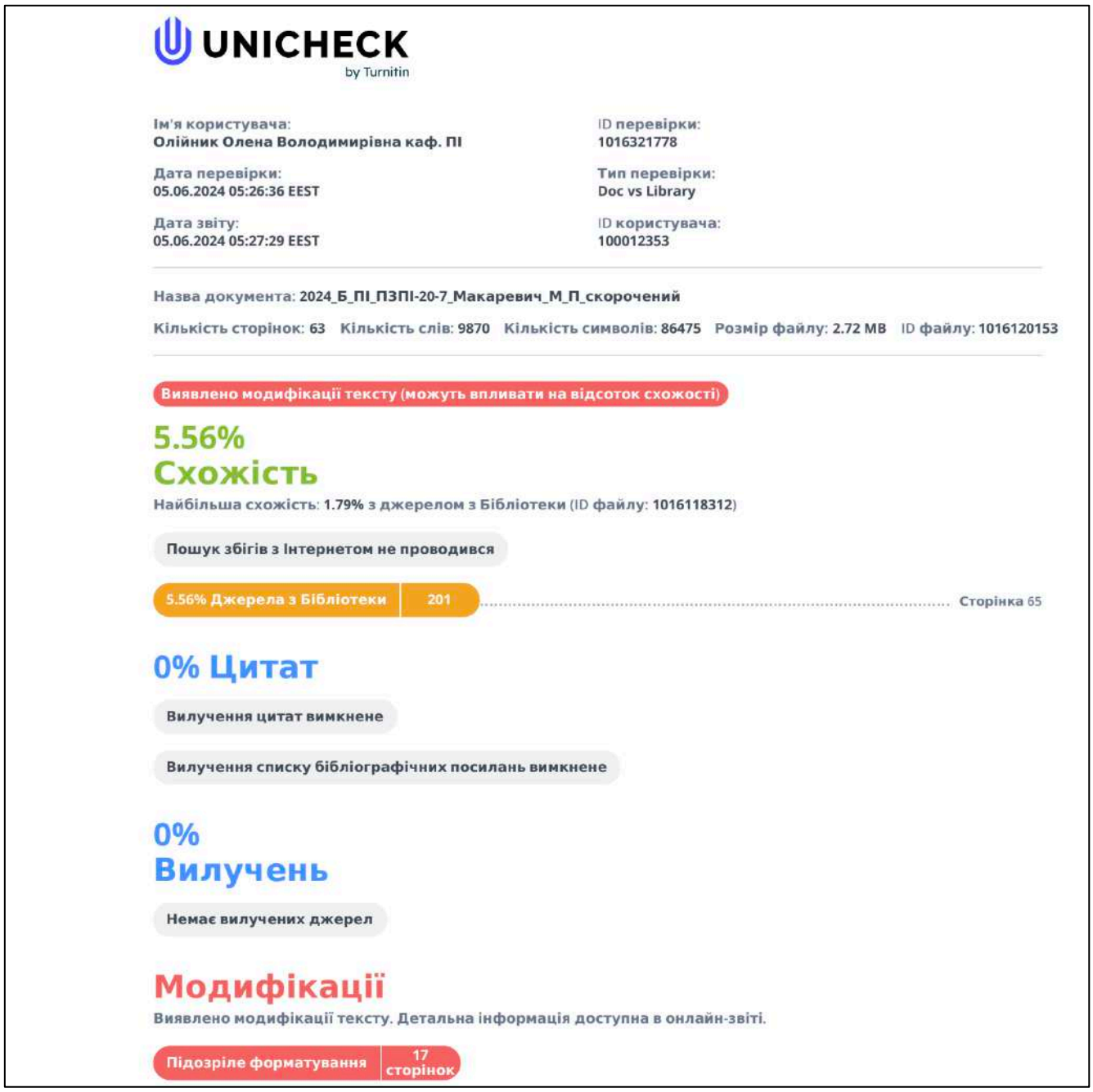
3. Next.js [Електронний ресурс]. URL: https://nextjs.org/ (дата звернення: 26.03.2024).

4. React Query [Електронний ресурс]. URL: https://tanstack.com/query/latest (дата звернення: 26.03.2024).

5. Material UI [Електронний ресурс]. URL: https://mui.com/ (дата звернення: 26.03.2024).

ДОДАТОК Г

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016321778

Дата перевірки:
05.06.2024 05:26:36 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
05.06.2024 05:27:29 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-7_Макаревич_М_П_скорочений

Кількість сторінок: 63 Кількість слів: 9870 Кількість символів: 86475 Розмір файлу: 2.72 MB ID файлу: 1016120153

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.56%
Схожість

Найбільша схожість: 1.79% з джерелом з Бібліотеки (ID файлу: 1016118312)

Пошук збігів з Інтернетом не проводився

5.56% Джерела з Бібліотеки 201 ..... Сторінка 65

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

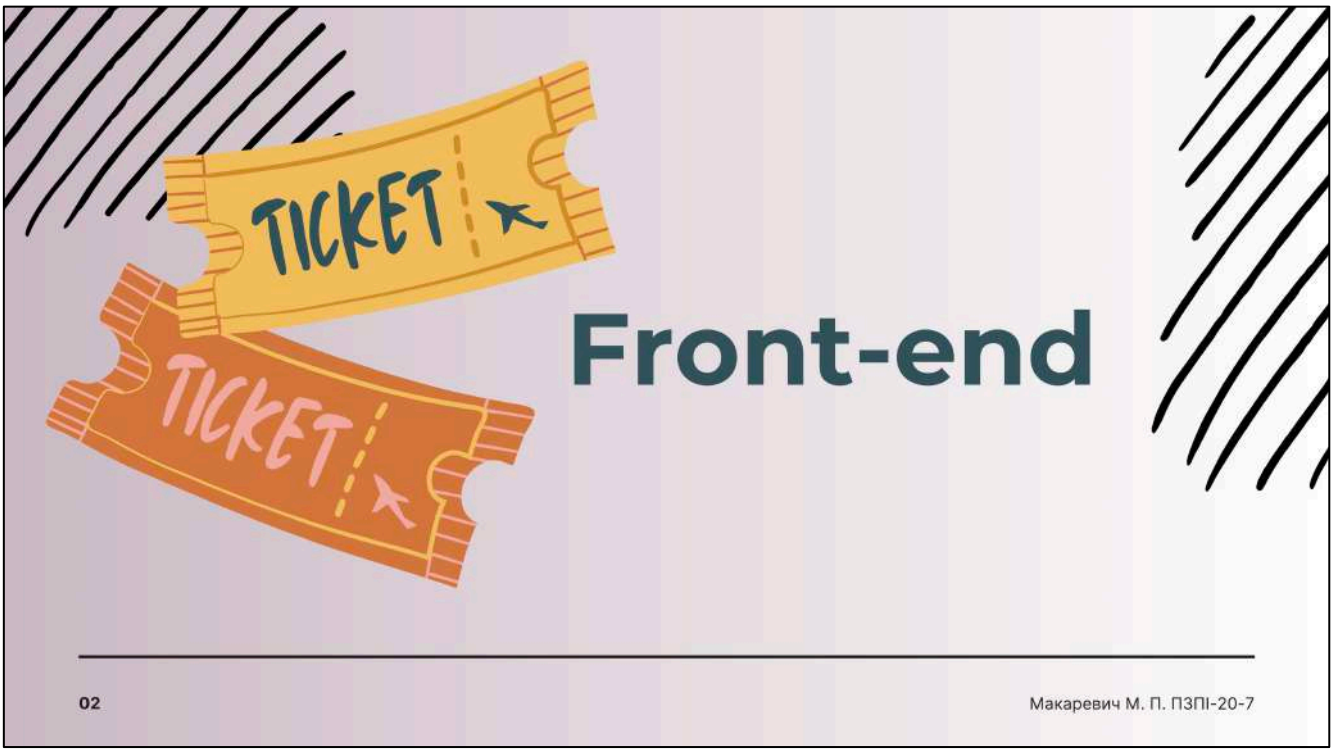
Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

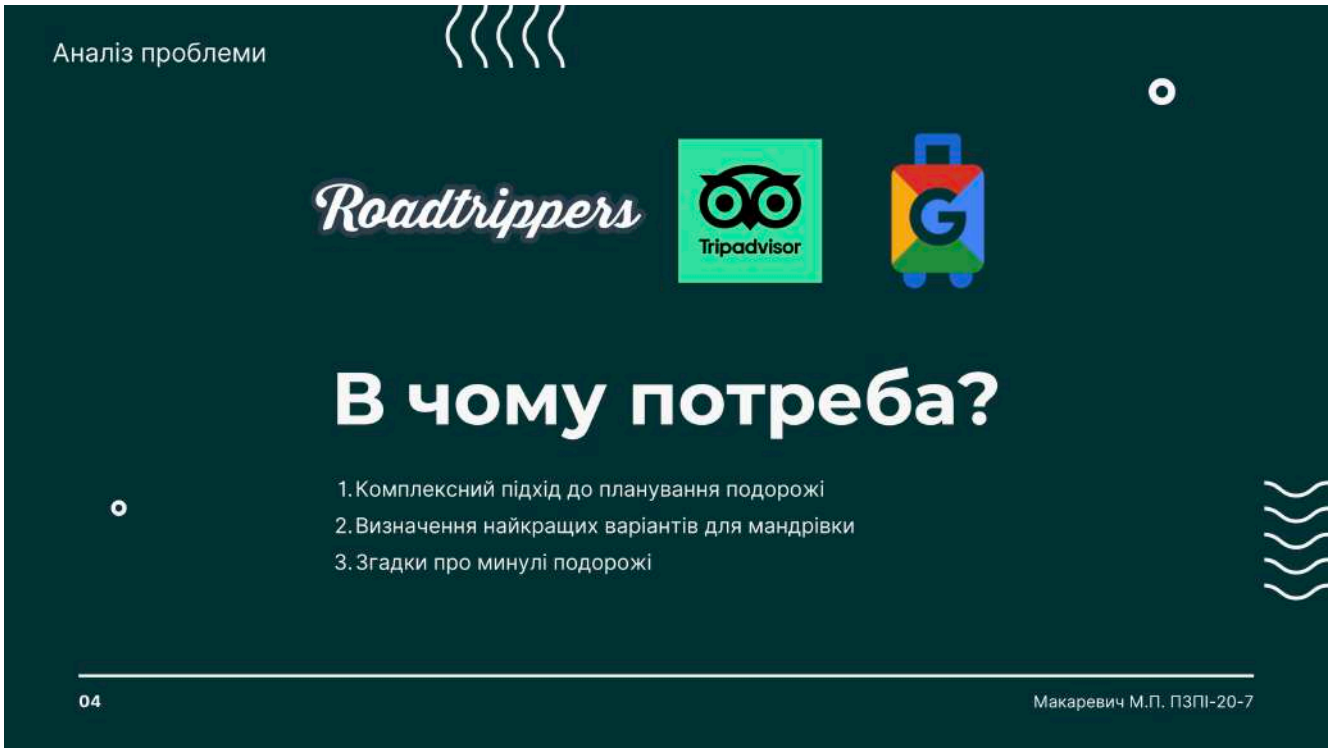
Підозріле форматування 17 сторінок

ДОДАТОК Д

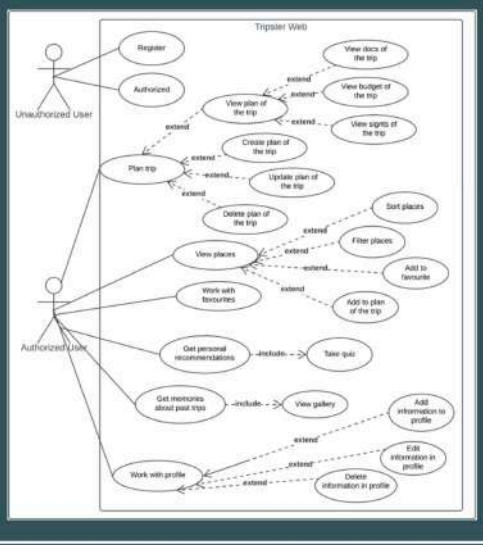
Слайди презентації




DUBAI
BARCELONA
TOKYO
SEOUL
Мета роботи
• Спростити планування подорожей для мандрівників
• Реалізувати інтерфейс для отримання інформації про визначні місця
• Запровадити можливість перегляду згадок про минулі мандри
03
Макаревич М.П. ПЗПІ-20-7

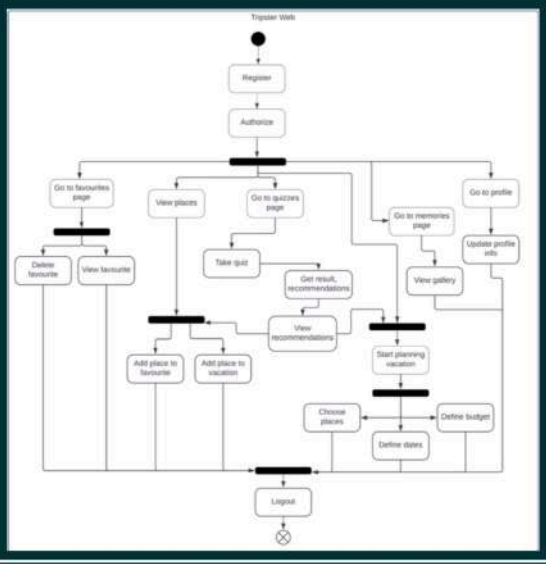

Аналіз проблеми
Roadtrippers
Tripadvisor
В чому потреба?
1. Комплексний підхід до планування подорожі
2. Визначення найкращих варіантів для мандрівки
3. Згадки про минулі подорожі
04
Макаревич М.П. ПЗПІ-20-7

Постановка задачі | Опис системи

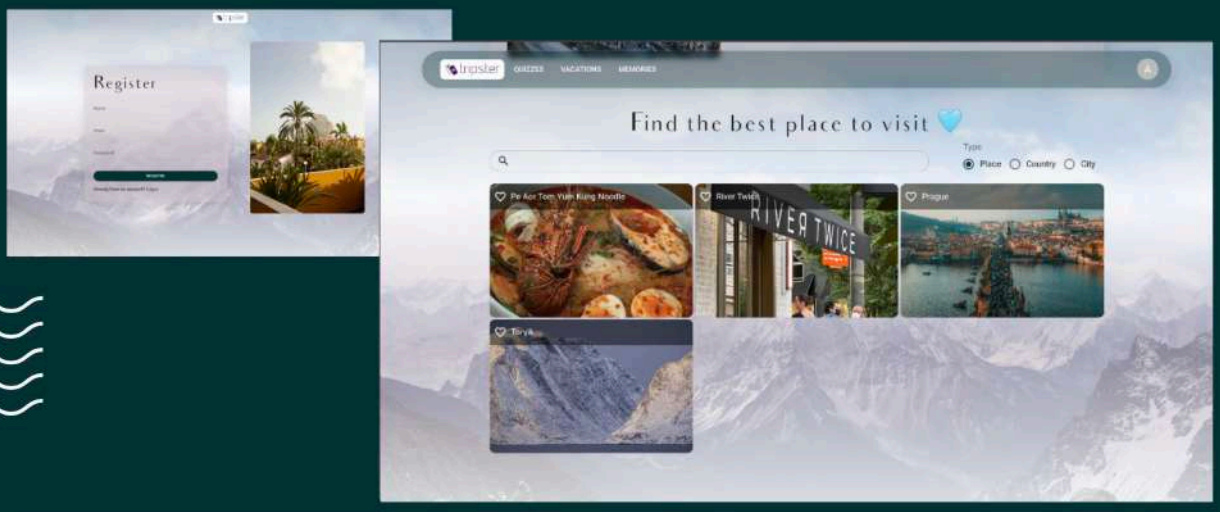


Технології розробки

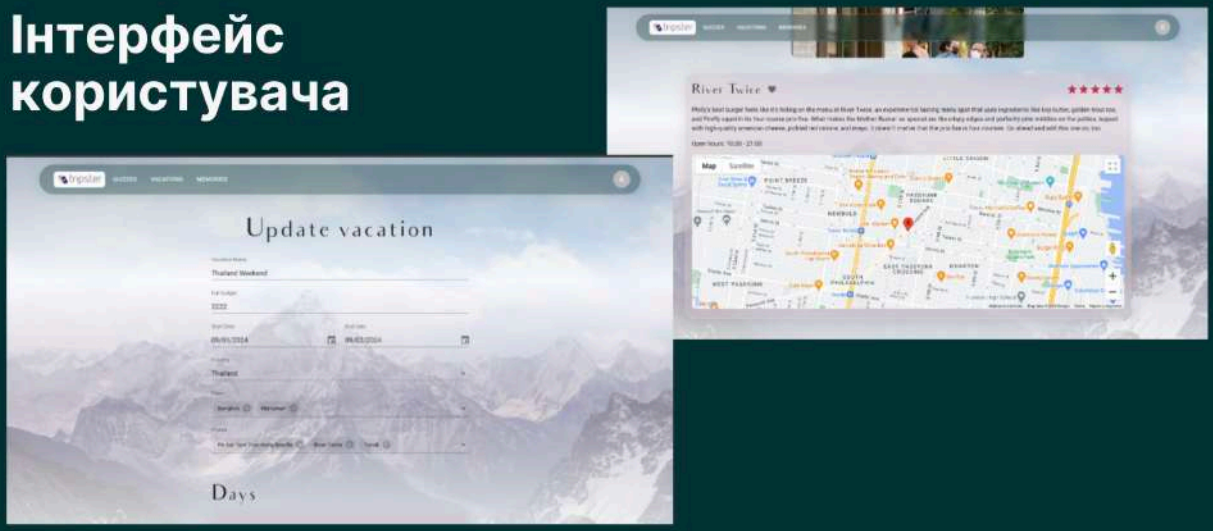
Діаграма діяльності



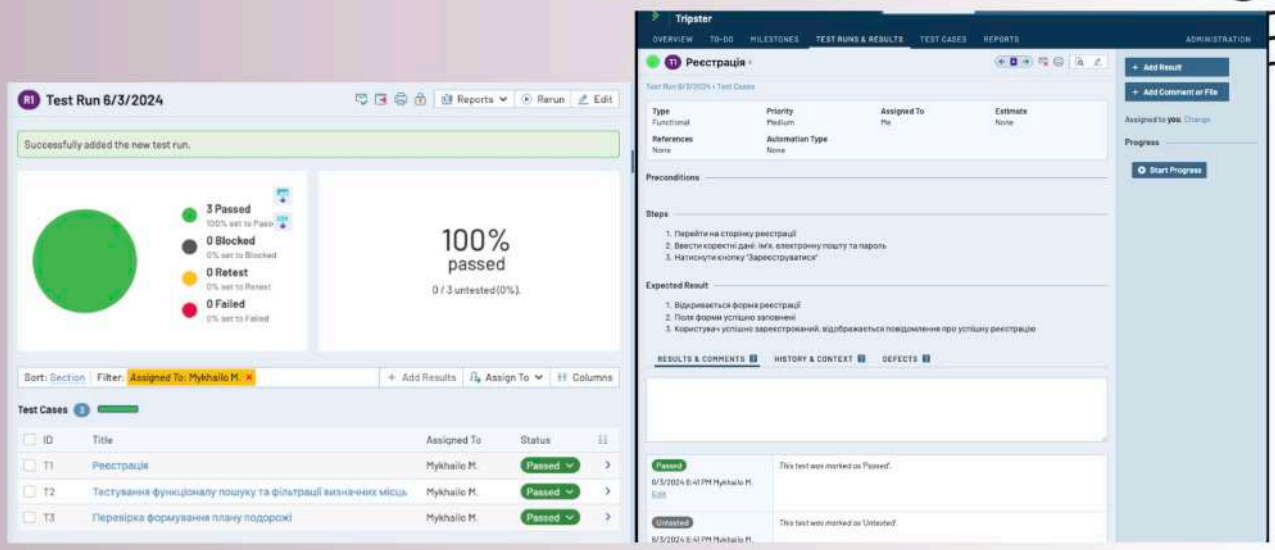
Інтерфейс користувача



Інтерфейс користувача



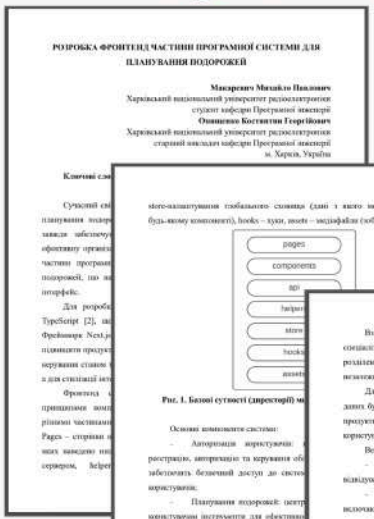
Тестування



Результати

- Проведено аналіз предметної області та визначено основні вимоги до функціональності та дизайну фронтенд частини «Tripster». Було враховано потреби потенційних користувачів та проаналізовано існуючі рішення на ринку;
- Розроблено архітектуру фронтенд частини додатку з використанням сучасних технологій та підходів, таких як React, Next.js та Material UI.
- Реалізовано функціональні можливості фронтенд частини веб-додатку, такі як пошук та фільтрація місць, збереження вподобаних місць, планування подорожей, перегляд детальної інформації про місця та інтеграція з серверною частиною через API. Було забезпечено плавну та інтуїтивно зрозумілу взаємодію користувача з додатком

Публікація


РОЗРОБКА ФРОНТЕНД ЧАСТИНИ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ