МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Вебдодаток «Щоденник мандрівника»»

_____

Здобувачки групи ІТ-02 Ніколенко Софії Олексіївни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Софія НІКОЛЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри інформаційних технологій, ктн., доц., Юлія ПАРФЕНЕНКО

(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) _____ (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Ніколенко Софія Олексіївна

1 Тема роботи *Вебдодаток «Щоденник мандрівника»*

керівник роботи *Парфененко Юлія Вікторівна, к.т.н., доцент*,

затверджені наказом по університету від «*07*» травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи перелік вимог по розробці вебдодатку «Щоденник мандрівника», графічні матеріали для наповнення вебдодатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування вебдодатку, програмна реалізація вебдодатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета, задачі, функціональні вимоги, аналіз аналогів, порівняльна таблиця аналогів, контекстна діаграма ведення щоденника IDEF0, діаграма варіантів використання, приклади документів бази даних, діаграма послідовностей, засоби реалізації, демонстрація вебдодатку, висновки, апробація.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	11.04.2024 – 12.04.2024	виконано
2	Планування робіт	13.04.2024 - 18.04.2024	виконано
3	Аналіз продуктів-аналогів	19.04.2024 - 20.04.2024	виконано
4	Складання технічних завдань	21.04.2024 - 23.04.2024	виконано
5	Модельована та проектування вебдодатку	25.04.2024 - 01.05.2024	виконано
6	Розробка вебдодатку	02.05.2024 - 21.05.2024	виконано
7	Тестування вебдодатку	22.05.2024	виконано
8	Оформлення пояснювальної записки	23.05.2024 - 25.05.2024	виконано

Студент _____ Софія НІКОЛЕНКО
(підпис)

Керівник роботи _____ к.т.н., доц. Юлія ПАРФЕНЕНКО
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Вебдодаток «Щоденник мандівника»».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 128 сторінок, у тому числі 112 сторінок основного тексту, 2 сторінки списку використаних джерел, 6 сторінок додатків.

Актуальність роботи полягає в створені зручного вебдодатку для планування та документування подорожей, обміну враженнями та досвідом між користувачами додатку, пошук різноманітних варіантів для організації подорожі та доступ до інформації про місце подорожі.

Мета роботи: розроблення вебдодатку «Щоденник мандрівника» для планування та збереження інформації про подорож.

Проведено аналіз публікацій і джерел в області туризму, існуючих продуктів-аналогів, визначено мету і задачі роботи.

Виконано моделювання і проектування вебдодатку. Побудовано контекстну діаграму процесу ведення щоденника у нотації IDEF0 та її декомпозицію, UML діаграму варіантів використання та діаграму послідовності.

Описано архітектуру вебдодатку та розроблено сам вебдодаток «Щоденник мандрівника». У роботі реалізовано базу даних, здійснено реалізацію серверної та клієнтської частини вебдодатку. Результатом проведеної роботи є вебдодаток «Щоденник мандрівника», який має адаптивний дизайн.

Ключові слова: вебдодаток, мандрівник, користувач, подорожі, нотатки, React, MongoDB.

ЗМІСТ

ВСТУП

У сучасному світі, насиченому прагненням до подорожей та відкриття чогось нового, досить важливо правильно спланувати та продумати мандрівку, тримати під рукою усі нотатки та мати можливість зберегти інформацію про маршрут.

Зараз спостерігається тенденція, коли люди все більше подорожують і відкривають для себе нові куточки нашої планети. Деякі туристи обирають вже готову екскурсію і насолоджуються чітко спланованою поїздкою. Але є і ті, хто хоче сам спланувати свою подорож, створити свій власний маршрут та отримати незабутні емоції від мандрівки. Беззаперечно, цей варіант наддасть більше волі у своїх діях, але інколи може виникнути проблема з плануванням подорожей. Нерідко мандрівник може забути необхідні речі або взяти те, що зовсім не знадобиться. Ця проблема виникає через неправильне планування чи його відсутність та малу кількість інформації про місце подорожі. Також важливо не тільки отримати емоції, а й зберегти їх, занотувати чи зробити фото.

Об’єкт – інформаційна підтримка ведення щоденника мандрівником. Предмет – технології розроблення вебдодатків для мандрівників.

Метою проекту є розроблення вебдодатку «Щоденник мандрівника» для планування та збереження інформації про подорож.

Для досягнення мети проекту необхідно виконати наступні задачі:

- визначити актуальність роботи та провести аналіз предметної області;
- виконати аналіз аналогічних вебдодатків та визначити функціональні вимоги для вебдодатку «Щоденник мандрівника»;
- виконати моделювання та проектування вебдодатку «Щоденник мандрівника»;
- розробити вебдодаток «Щоденник мандрівника» та провести його тестування.

Практичним значенням роботи є створення зручного вебдодатку для планування, організації та збереження подорожей, який також сприяє соціальній взаємодії між мандрівниками.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У сучасному світі, з розвитком технологій, тема подорожей та нотування стає все більш актуальною. Дослідження доводять, що люди все частіше віддають перевагу індивідуальним мандрівкам, власному досвіду та враженням, а не тур пакетам, запропонованим туристичними агентствами [1-3].

Існують два основних типи інформаційних систем для мандрівників: мобільні додатки та вебдодатки. Мобільні додатки зазвичай надають зручний доступ до інформації про подорожі безпосередньо на мобільному пристрої, що дозволяє користувачам отримувати необхідну інформацію в будь-який час та місце. Однак вони можуть бути обмежені в функціоналі та вимагати значних обсягів пам'яті на пристрої. З іншого боку, вебдодатки доступні через браузер та не вимагають встановлення, що робить їх доступними на будь-якому пристрої з Інтернет-з'єднанням. Також веб-додатки зазвичай мають більший потенціал для складнішого функціоналу та можуть надавати більше можливостей для інтерактивності та спілкування з іншими користувачами. Проте вони можуть бути менш зручними для використання на мобільних пристроях через обмежені розміри екрану та можуть вимагати постійного Інтернет-з'єднання для роботи. Вибір між ними залежить від конкретних потреб та уподобань кожного користувача.

Розумні туристичні технології значно впливають на лояльність туристів. Їхній вплив розглядається через кілька ключових аспектів. Довіра до туристичних соціальних медіа та веб-джерел туристичної інформації грає важливу роль. Ці медіа надають мандрівникам доступ до персоналізованих рекомендацій та відгуків, сприяючи формуванню їхніх уподобань і вибору місць відпочинку [4-6].

Ключова роль є вивчення туристичної інформації через мобільні пристрої та вебдодатки, що стає все більш важливим для подорожуючих. Проект Image, наприклад, досліджує ефективність та вплив такого способу отримання інформації на задоволення та зручність мандрівників [7].

У рамках дослідження цієї теми було виявлено, що багато існуючих вебдодатків, спрямованих на подорожі, зазвичай зосереджені на побудові маршрутів або підборі вже готових турів [8]. Вони пропонують користувачам широкий вибір туристичних об'єктів та маршрутів, але надають обмежені можливості для організації особистих вражень та досвіду [9]. Такі додатки, як правило, не забезпечують можливості збереження особистих нотаток та фотографій, а також відсутність можливості ділитися своїми враженнями з іншими користувачами.

Ця проблема стає все більш актуальною, оскільки люди шукають способи більш особистого та інтерактивного досвіду подорожей [10]. Окрім цього, в роботі [11] розглядається проблема планування маршруту людьми, які ніколи не стикалися з цим та прагнуть вдосконалити свої навички у плануванні мандрівок. У зв’язку з цим виникає потреба у новому вебдодатку, який би забезпечував користувачам можливість планування маршрутів, нотування вражень та спілкування з іншими мандрівниками.

Такий застосунок може стати ідеальним інструментом для подорожуючих, де вони зможуть зберігати свої особисті нотатки, фотографії та відео, а також ділитися своїми враженнями та порадами з іншими користувачами. Такий додаток може створити спільноту мандрівників, де кожен зможе знайти натхнення, поради та нові ідеї для подорожей.

Отже, дослідження у цій області підтверджує актуальність проблеми та необхідність створення нового мобільного застосунку «Щоденник мандрівника», який би задовольняв потреби сучасних мандрівників і допомагав їм зробити свої подорожі більш особистими та пам’ятними.

1.2 Аналіз програмних продуктів - аналогів

На ринку існує значна кількість вебдодатків, спрямованих на надання допомоги під час подорожі, охоплюючи різні аспекти, такі як бронювання готелів, вибір ресторанів, інформація про пам’ятки, тощо. Кожен вебдодаток для планування подорожей характеризується власним набором переваг і недоліків, власним функціоналом та особливостями. Вивчення та розуміння цих особливостей є ключовим для вибору оптимального інструменту для подорожі залежно від конкретних потреб та вимог користувача.

Для порівняння вебдодатків, які вирішують проблему планування подорожей, було обрано наступні сервіси:

- «Travel.sygic» [12];
- «TripIt» [13];
- «PlanYourTrip» [14].

Першим для аналізу було розглянуто сервіс «Travel.sygic».

«Sygic» є компанією, спеціалізованою на розробці навігаційних та картографічних продуктів. Одним із її продуктів є вебдодаток «Travel.sygic» (рис. 1.1).

У вебдодатку присутня можливість реєстрації та авторизації. Перевагами є зручний інтерфейс, представлений у вигляді карти з відзначеними на ній пам’ятками, кафе, готелями, магазинами тощо, що робить його корисним для подорожуючих. Крім того, вебдодаток надає можливість переглядати інформацію про готелі, тури та оренду автомобілів. Навігатор відзначається зручністю використання.

Проте слід відзначити значну кількість недоліків. Більшість функціоналу доступна лише за умови платної підписки. Також відзначається обмежена кількість інформації про різноманітні місця, готелі та інші об’єкти. Відсутність можливості планування маршруту від одного місця до іншого, а також додавання особистих заміток чи фотографій, визначається як суттєвий недолік. Деякі посилання у вебдодатку також можуть виявитися потенційно небезпечними. Окрім всього, у вебдодатку немає можливості ділитися маршрутами чи замітками.

Загалом вебдодаток надає змогу отримати інформацію про розташування більшості пам’яток міста, скласти маршрут до конкретної локації та переглянути інформацію про неї. Однак розширення функціоналу доступно лише на платній основі або взагалі відсутнє для безкоштовної версії.

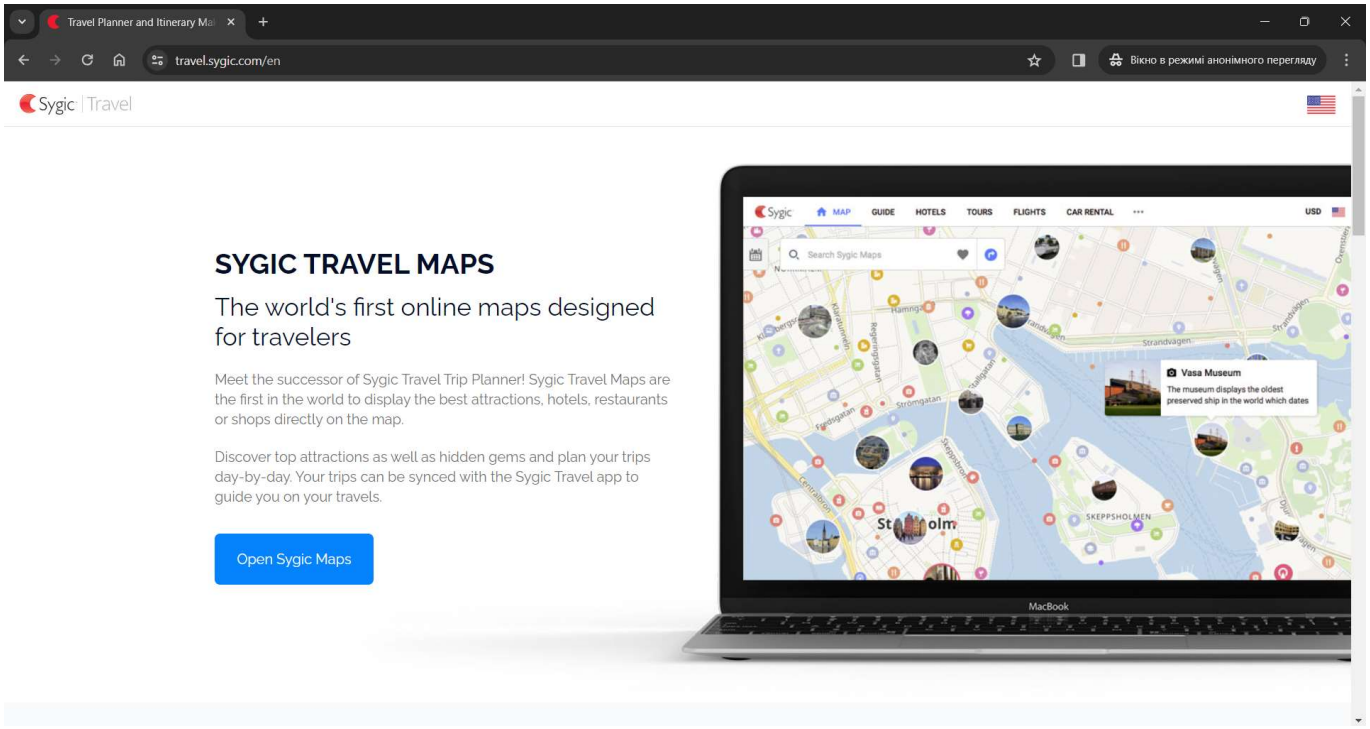

Рисунок 1.1 – Початкова сторінка вебсервісу «Travel.sygic»

Переглянемо вебдодаток «TripIt» (рис. 1.2).

Вебдодаток призначений для систематизації та узагальнення вражень від подорожей, проте є і недоліки. Зараз інтерфейс не надає можливості додавання фотографій, що є суттєвою складовою збагачення звіту про подорож. Наявність застарілих полів для заповнення не дає можливість простого доповнення нотатки про подорож (рис. 1.3).

Крім того відсутність функціоналу для додавання нотаток перед та після подорожі ускладнює можливість повноцінного відображення подій. Важливою функцією також є відстеження часу, але наразі система не надає зручних засобів для цього, що може призводити до невірного хронологічного відображення інформації.

Також, слід зазначити, що у вебдодатку відсутня можливість поширювати свою подорож з іншими користувачами.

Окрім цього, у вебдодатку відсутня українська мова.

Узагальнюючи, вебдодаток «TripIt» для фіксації подорожей відзначається обмеженим функціоналом, включаючи відсутність можливості завантаження фотографій та додавання нотаток після здійснення подорожі.

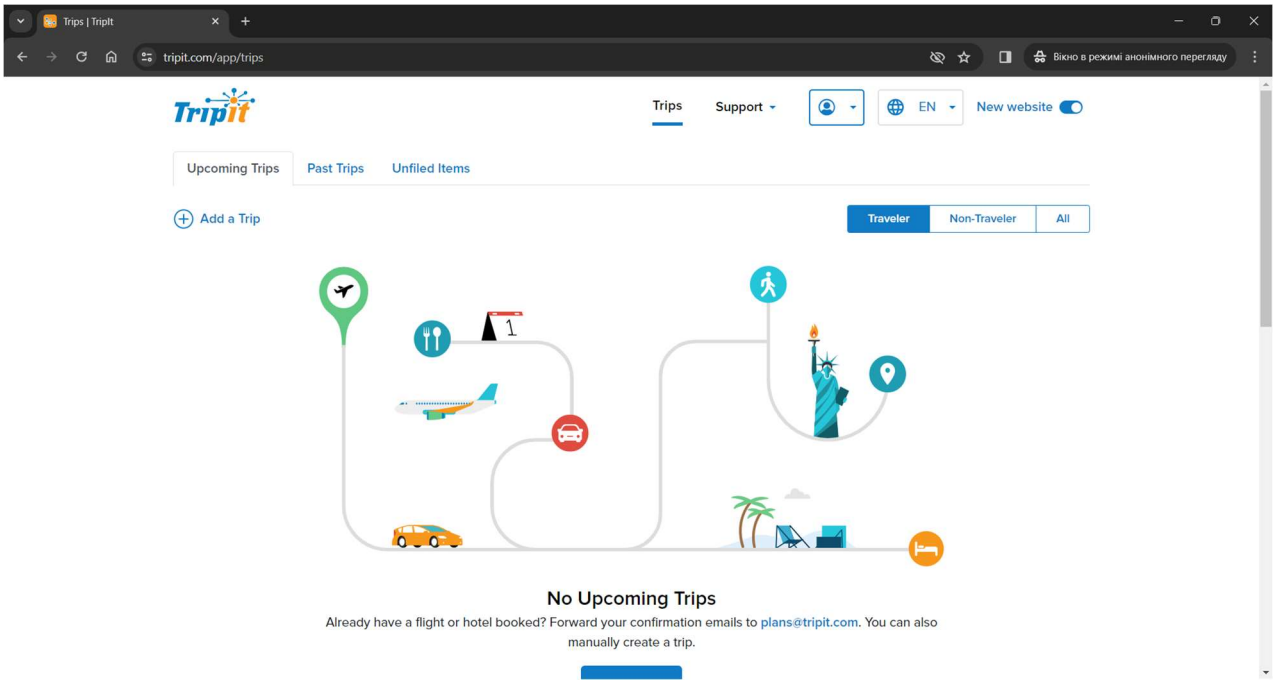

Рисунок 1.2 – Початкова сторінка для зареєстрованого користувача вебдодаток «TripIt»

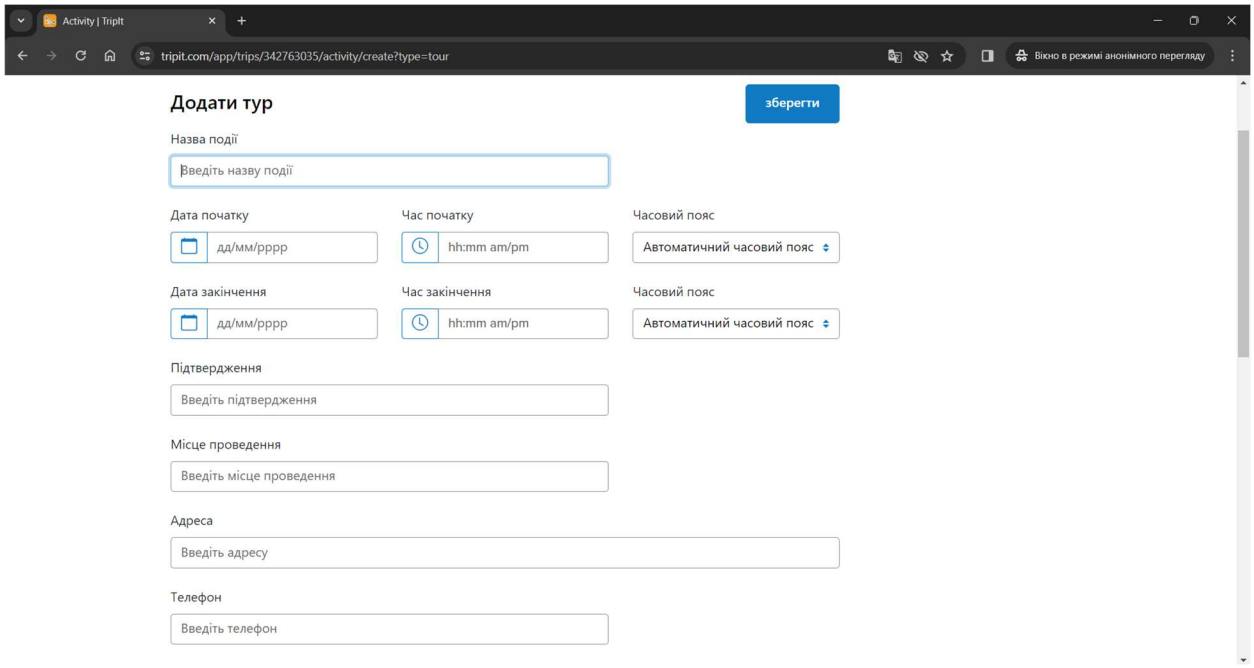

Рисунок 1.3 – Перелік полів для заповнення

Третім вебдодатком для проведення аналізу буде «PlanYourTrip».

Дизайн вебдодатку інтуїтивно зрозумілий та містить інформацію про розробників та загальну концепцію платформи (рис. 1.4). Однією з ключових переваг є можливість редагування подорожей як перед поїздкою, так і під час неї, а також спланувати нову поїздку на основі вже здійснених.

Вебдодаток надає користувачам можливість вибору бажаного стилю подорожі, тривалості, природного ландшафту та знайомства з культурою, не вимагаючи обов’язкової реєстрації для складання маршруту. Після визначення критеріїв, вебдодаток автоматично генерує маршрут та детально описує його (рис. 1.5).

Серед інших переваг варто відзначити наявність великої кількості інформації про пам’ятки та можливість обрати заздалегідь підготований маршрут для подорожі. Також, користувач може додавати кілька подорожей.

Проте, серед недоліків слід зазначити малу кількість країн для подорожей. Відсутність можливості додавання нотаток та фотографій. Крім того, можливість редагування обмежується лише згенерованим вебдодатком планом, не надаючи можливості самостійно створювати поїздки.

У підсумку, вебдодаток відзначається зручним та інноваційним підходом до планування подорожей, але йому бракує можливостей для додавання особистих нотаток та фотографій, а також для самостійного формування маршрутів.

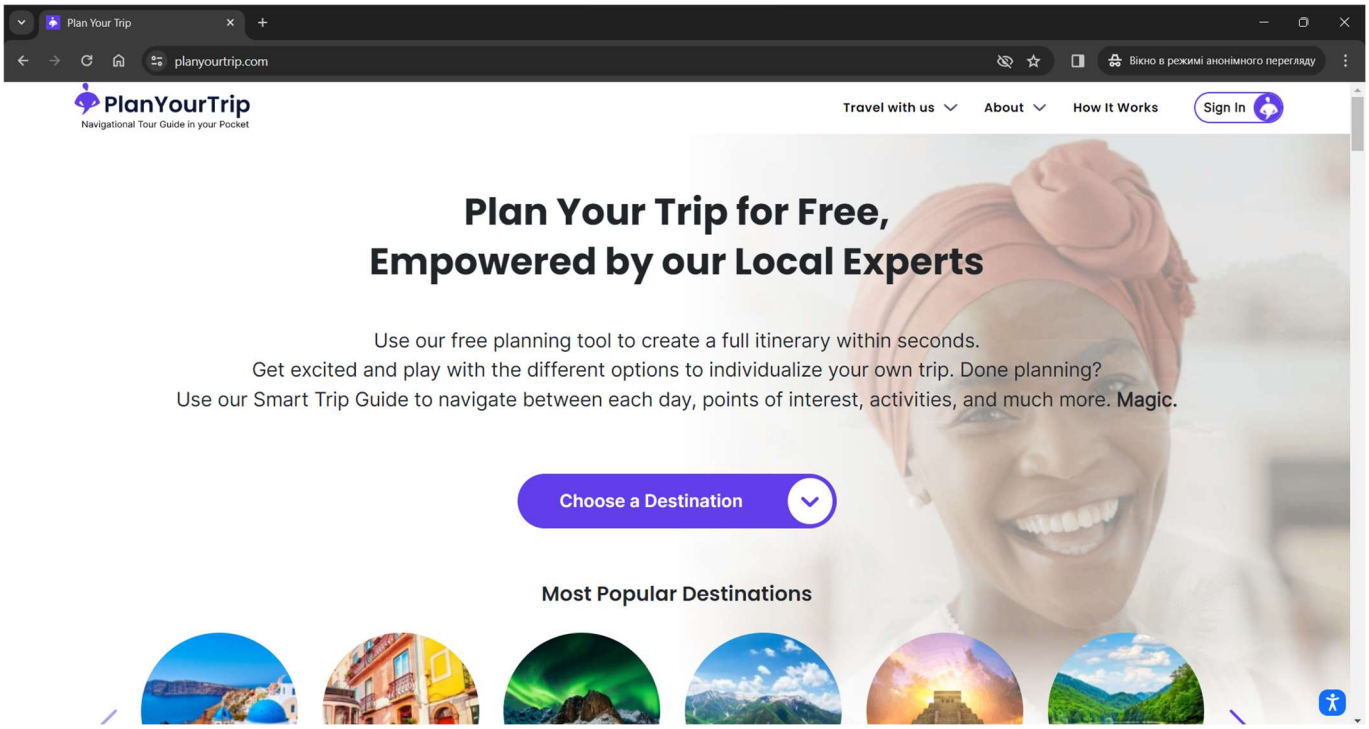

Рисунок 1.4 – Початкова сторінка вебдодатку «PlanYourTrip»

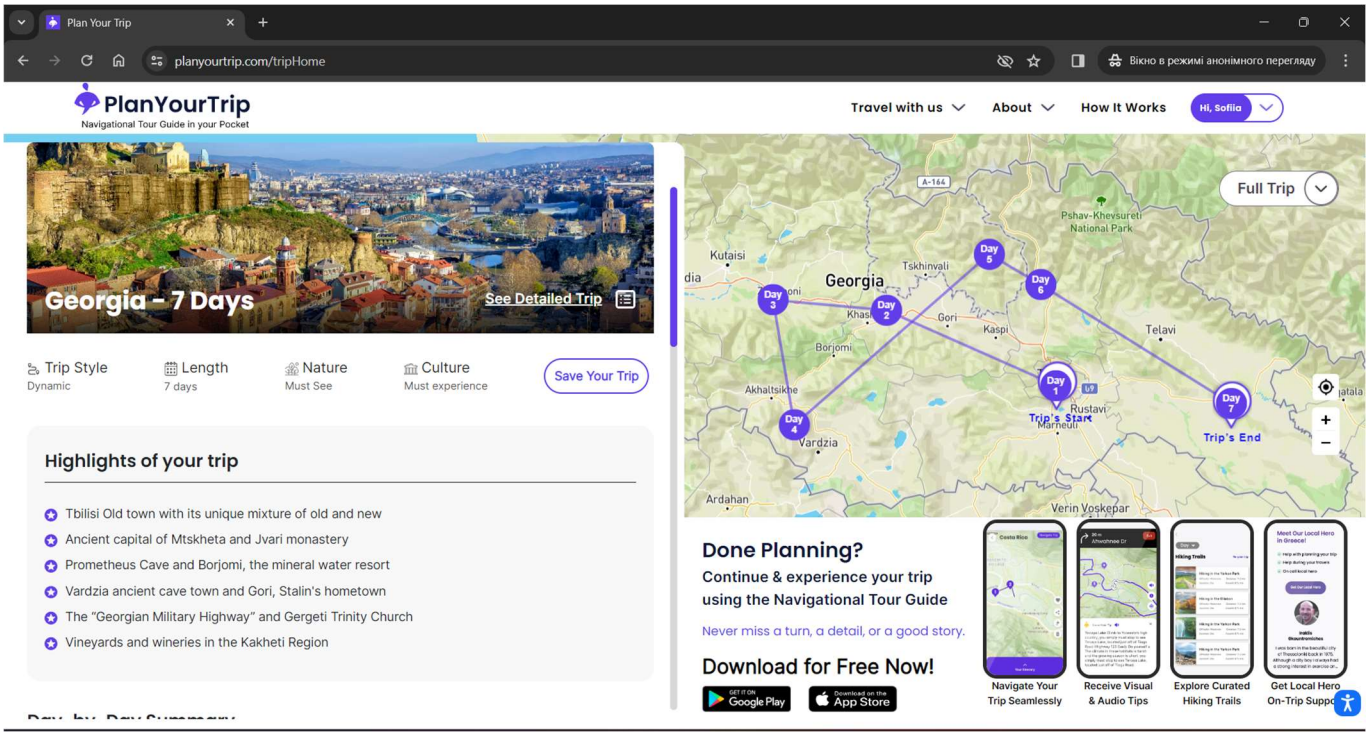

Рисунок 1.5 – Запропонована подорож вебдодатку «PlanYourTrip»

Після детального аналізу аналогів вебдодатків для планування подорожей, було визначено їх переваги та недоліки. Його результати представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів вебдодатків

	Назва вебдодатку		
Характеристика	«Travel.sygic»	«TripIt»	«PlanYourTrip»
Реєстрація/авторизація	+	+	+
Додавання нотаток	-	+	-
Завантаження фото	-	-	-
Поділ нотатками з іншими користувачами платформи	-	-	-
Планування кількох подорожей	-	+	+
Самостійне створення поїздок	+	+	-
Створення нотаток після подорожі	-	-	-

У результаті аналізу та порівняння різних ведодатків було ретельно враховано їх сильні та слабкі сторони. Це дозволить ефективно використовувати набуті знання у подальшій роботі над проектом, сприяти вдосконаленню функціоналу та надавати користувачам найбільш зручний та функціональний інструмент для планування подорожей та зберігання вражень після них. Аналіз сильних сторін дозволить максимально використати їх переваги, тоді як урахування недоліків спрямоване на їх усунення та вдосконалення загального користувацького досвіду.

1.3 Мета та задачі дослідження

Метою проекту є розроблення вебдодатку «Щоденник мандрівника» для планування та збереження інформації про подорож. Він буде служити зручним та інтуїтивно зрозумілим інструментом для планування та збереження інформації та вражень про подорожі. Очікується, що інтерфейс вебдодатку буде виражений у єдиному стилі, надаючи користувачам зрозумілий доступ до функціоналу. Проект спрямований на надання користувачам зручного та функціонального інструменту для детального відображення своїх подорожей, а також на забезпечення можливості ефективного планування та відстеження історії своїх подорожей.

Основні вимоги до створюваного програмного продукту є наступними:

- можливість планування та редагування нотаток;
- можливість створення списків з відповідними категоріями;
- можливість додавання фотографій та, за потреби, підписів до них;
- віджет погоди у вибраному місці;
- можливість пошуку подорожі по певним міткам;
- можливість поширення нотатками з іншими користувачами;
- можливість додавання декількох подорожей;
- зручний та сучасний інтерфейс.

Для досягнення мети проекту необхідно розробити детальну структуру веб-додатку, яка буде включати в себе усі необхідні сторінки та функціонал. Це охоплює реєстрацію та авторизацію користувачів, функціонал додавання нотаток та можливість поділитися ними з іншими користувачами, а також можливість пошуку нотаток за хештегами та інші функції, що вказані у додатку Б.

Щодо вибору нереляційної бази даних MongoDB для реалізації серверної частини, це зроблено з метою забезпечення гнучкості системи та зручного подальшого розвитку вебдодатку. MongoDB є документ-орієнтованою базою даних, яка дозволяє зберігати дані у вигляді документів, що відповідає структурі даних у

вебдодатках, тому вона ідеально підходить для забезпечення гнучкості та ефективності у роботі веб-додатку.

У реалізації клієнтської частини використовуються такі технології як HTML, CSS, та JavaScript разом із фреймворком React. HTML та CSS використовуються для створення структури та оформлення вебсторінок, а JavaScript дозволяє додавати інтерактивність до сторінок. Фреймворк React використовується для швидкого та ефективного створення користувацьких інтерфейсів, що забезпечує зручність та швидкість розробки вебдодатку.

Для досягнення мети проекту також необхідна серверну частину вебдодатку, яка реалізується за допомогою серверного середовища Node.js. Серверна частина відповідає за обробку запитів користувачів, зберігання та обробку даних у базі даних, а також забезпечення взаємодії між клієнтською частиною та базою даних.

Більш детальні вимоги до проекту описані у технічному завданні для розробки проекту (додаток А). Планування робіт представлено у додатку Б.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Моделювання процесу ведення щоденника мандрівника за допомогою вебдодатку

Для моделювання процесу ведення щоденника мандрівника за допомогою вебдодатку використовувалася IDEF0 методологія структурно-функціонального моделювання. На контекстній діаграмі, яка подана на рисунку 2.1, показано, як система взаємодіє з вхідними даними, елементами керування, вихідними даними та внутрішніми механізмами. Сама діаграма побудована з точки зору мандрівника, який займається веденням щоденника. На вхід подаються дані для реєстрації, дані для авторизації та дані про мандрівку. Процес ведення щоденника відбувається за участі вебдодатку, бази даних, вебсервера, апаратного забезпечення та користувача. Сам процес обмежений правилами валідації даних, інструкціями по заповненню щоденника та вимогами до розміщення інформації в інтернеті. На виході отримуємо запис мандрівки до бази даних та інформацію про мандрівку у публічному доступі.

Рисунок 2.1 – Контекстна діаграма процесу ведення щоденника мандрівника в нотації IDEF0

На рис. 2.2 представлено діаграму декомпозиції першого рівня, на якій деталізовано показано розбиття процесу ведення щоденника. У моделі визначено наступні підпроцеси:

– реєстрація;
– авторизація;
– заповнення форми для мандрівки;
– розміщення публічного контенту.

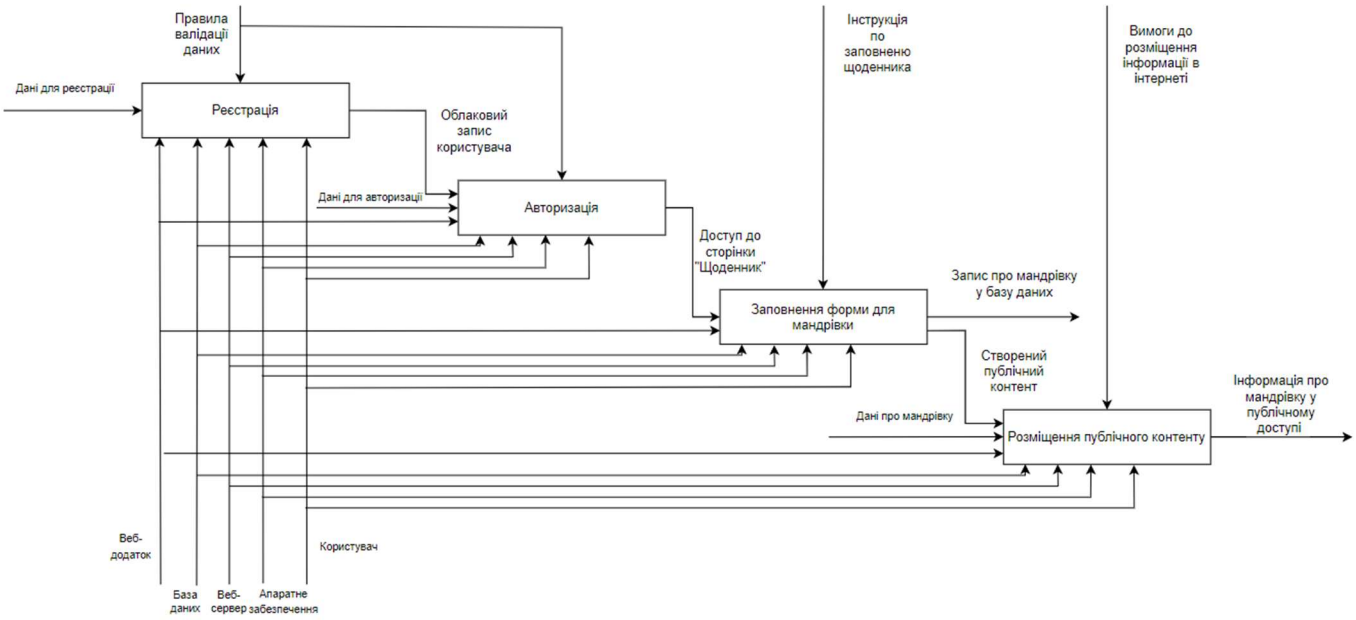

Рисунок 2.2 – Діаграма декомпозиції процесу ведення щоденника мандрівника

Ведення щоденника здійснюється за наступним принципом: користувач проходить процес реєстрації та надає данні для реєстрації, далі користувач надає дані для авторизації. Ці два процеси контролюються правилами валідації даних. Після того, як користувач отримав доступ до сторінки «Щоденник» він заповнює форму для додання мандрівки і після відправки форми відбувається запис до бази даних. Далі користувач може відкрити цю мандрівку для публічного доступу. Цей процес контролюється вимогами до розміщення інформації в інтернеті та на виході отримуємо інформацію про мандрівку у публічному доступі.

2.2 Моделювання варіантів використання вебдодатку

Діаграма варіантів використання вебдодатку «Щоденник мандрівника» у нотації UML представлена на рисунку 2.3. Діаграма варіантів використання допомагає виявити, які фактори впливають на вебдодаток та використовуються при розробці. Вона визначає, які дії можуть виконувати актори (користувачі, інші системи) та як система реагує на їхні запити.

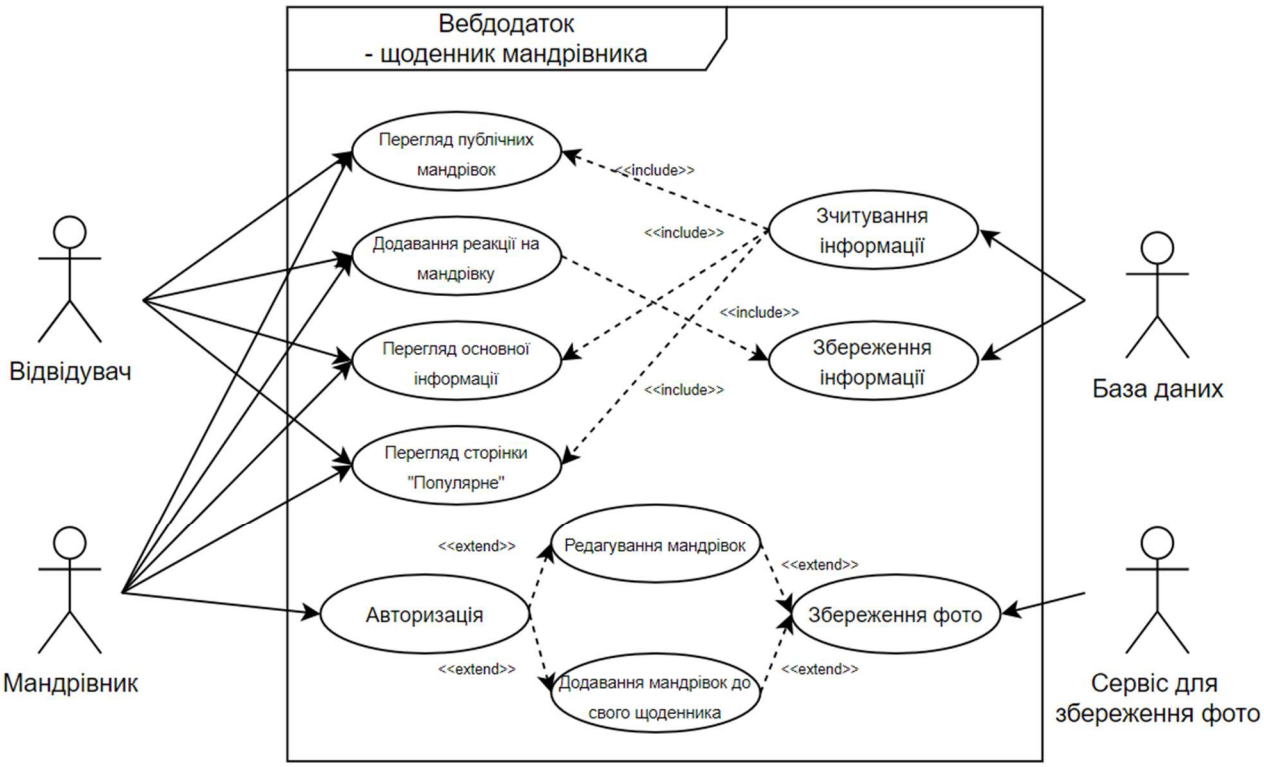

Рисунок 2.3 – Діаграма варіантів використання для вебдодатку «Щоденник мандрівника»

Акторами вебдодатку є:

– відвідувач – звичайний незареєстрований користувач вебдодатку;
– мандрівник – користувач, який після авторизації може додавати як публічний так і приватний контент до вебдодатку;

– база даних – представляє собою нереляційну MongoDB;
– сервіс для збереження фото – хмарне середовище для збереження фотографій користувача.

Усі наведені на діаграмі варіанти використання відповідні основним функціям вебдодатку.

На рис. 2.4 представлено діаграму послідовностей процесу ведення щоденника. Діаграма дає чітке розуміння взаємодії між об’єктами вебдодатку.

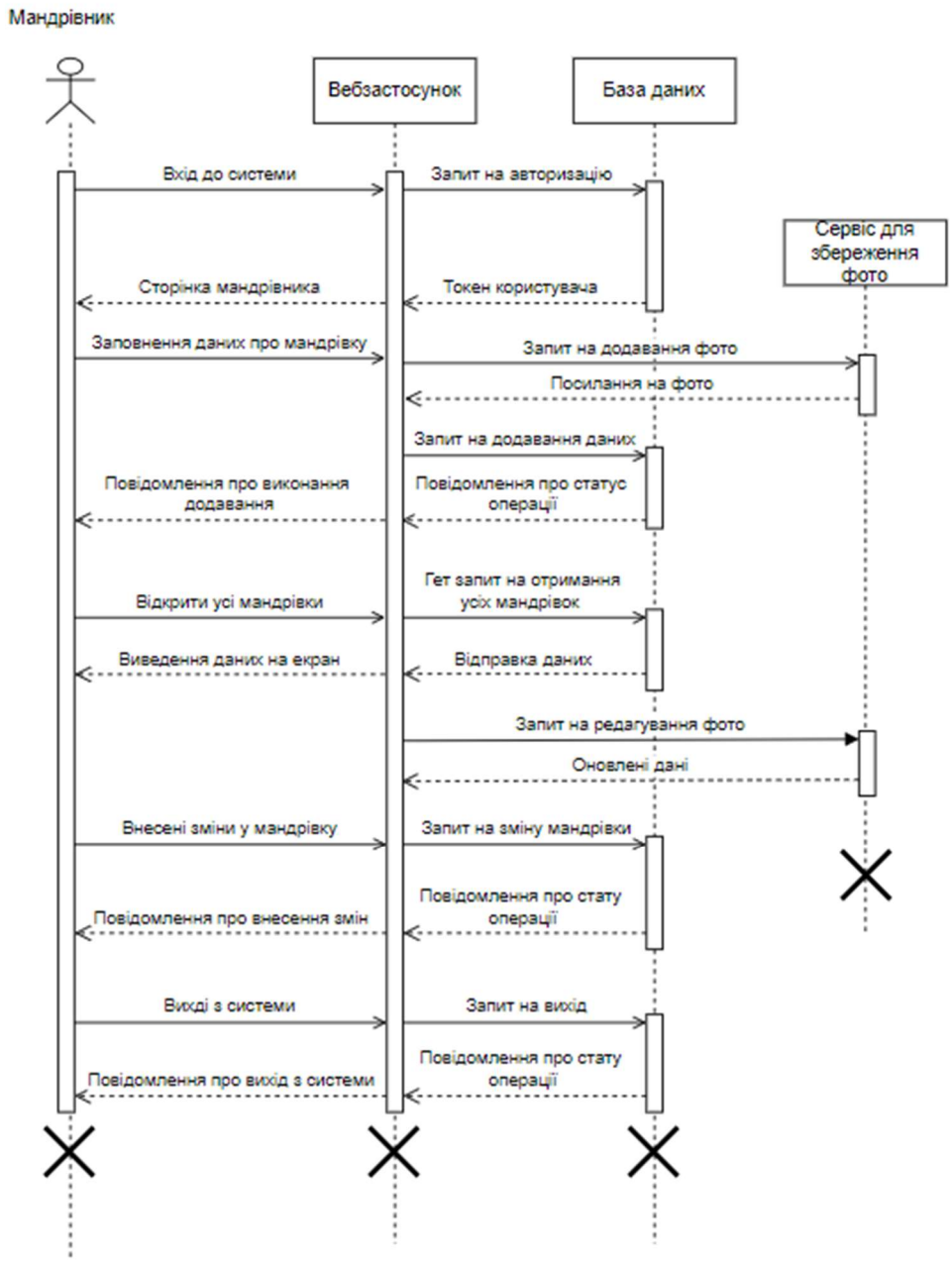

Рисунок 2.4 – Діаграма послідовностей процесу ведення щоденника

З діаграми очевидно, що взаємодія відбувається між користувачем і веб-застосунком, а також між веб-застосунком, базою даних і сервісом для збереження фотографій. В залежності від потреб користувача веб-застосунок виконує запити до бази даних або сервісу для збереження фотографій і повертає відповідь або повідомлення про стан операції.

2.3 Проектування бази даних

Для проектування нереляційної бази даних потрібно визначити перелік колекцій та полів які будуть включати ці колекції. Для вебдодатку розроблено дві колекції –це колекція «user» для збереження усієї інформації про користувача та колекція «trips» для збереження усієї інформації про мандрівку.

Колекція «user» має наступні поля:

- _id – унікальний індикатор користувача;
- email – електронна пошта;
- password – пароль;
- verify – прапорець для вказання, чи пройшов користувач верифікацію;
- verificationToken – токен для верифікації електронної пошти;
- createdAt – дата створення запису;
- updatedAt – дата оновлення запису;
- token – поле для збереження токену.

Колекція «trips» має наступні поля:

- _id – унікальний індикатор мондрівки;
- owner – id користувача, якому належить мандрівки;
- title – назва мандрівки;
- description – опис мандрівки;
- categories – список категорій;
- nameCategory – назва категорії;
- todoList – список нотаток;
- todo – нотатка;
- publicList – прапорець для вказання чи публічна ця категорія;
- isPublic – прапорець для вказання чи публічна мандрівка;
- photos – масив фотографій;
- cdnUrl – посилання на фотографію;

- uuid – унікальний індикатор фотографії;
- likes – кількість вподобайок;
- createdAt – дата створення запису;
- updatedAt – дата оновлення запису;

Документ «trips» пов’язаний з документом «user» через поле «owner».

Структура документів нереляційної бази даних MongoDB наведено на рис. 2.5.

User document

```
{
        _id:<ObjectID>
        email: String,
        password:String,
        verifu: Boolean,
        verificationToken: null,
        createdAt: Date,
        updatedAt: Date,
        token: String
}
```

Trip document

```
{
        _id: <ObjectID>,
        owner: <ObjectID>,
        title: String,
        description: String,
        categories: [ Array
            {
                nameCategory: String,
                todoList: [ Array
                    {
                        todo: String
                    },
                ],
                publicList: Boolean,
            },
        ],
        isPublic: Boolean,
        photos: [ Array
            {
                cdnUrl: String,
                uuid: String,
            },

        ],
        likes: Int32,
        createdAt: Date,
        updatedAt: Date,
}
```

Рисунок 2.5 – Документи нереляційної бази даних

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура вебдодатку

Архітектура вебдодатку «Щоденник мандрівника» (рис. 3.1) побудована з урахуванням вимог щодо гнучкості, масштабованості та високої продуктивності, що є критичними для забезпечення ефективної роботи та зручності використання. Додаток складається з двох основних частин: клієнтська та серверна, які взаємодіють між собою через чітко визначені API кінцеві пункти.

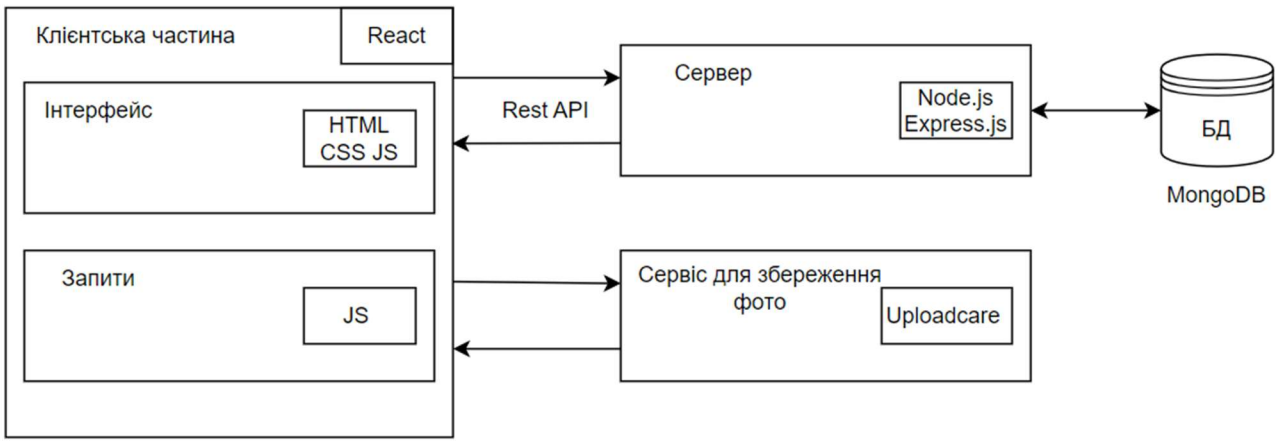

Рисунок 3.1 – Архітектура вебдодатку

Клієнтська частина відповідає за взаємодію користувача з додатком, надаючи зручний та інтуїтивно зрозумілий інтерфейс. Вона реалізована з використанням сучасних веб-технологій, таких як HTML, CSS і JavaScript, а також фреймворків React для компонентного підходу до побудови інтерфейсу. Це забезпечує високу реактивність і динамічність користувацького інтерфейсу, дозволяючи швидко оновлювати контент без перезавантаження сторінки. Сервіс Uploadcare використовується для завантаження, зберігання та обробки зображень, що додаються користувачами до їх записів.

Серверна частина відповідальна за обробку запитів від клієнтської, автентифікацію користувачів, бізнес-логіку та взаємодію з базою даних. Серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js, який спрощує створення та управління сервером. Для автентифікації користувачів використовуються JSON Web Tokens (JWT), що забезпечує безпечну передачу інформації між клієнтом і сервером.

Для зберігання даних використовується нереляційна база даних MongoDB, яка відома своєю гнучкістю і високою продуктивністю. MongoDB дозволяє зберігати дані у вигляді документів BSON, що забезпечує можливість легкої зміни структури даних і масштабування. Це особливо важливо для додатків, які повинні обробляти великі обсяги різнорідної інформації.

Взаємодія між клієнтською і серверною частинами здійснюється через REST API. Кожен кінцевий пункт API призначений для виконання конкретних операцій, таких як створення нової мандрівки, отримання списку мандрівок мандрівника, оновлення або видалення запису. Це дозволяє чітко розділити функціональні обов'язки між клієнтською та серверною частинами додатку, що сприяє спрощенню розробки, тестування та підтримки системи.

Загалом вебдодаток «Щоденник мандрівника» можна поділити на дві основні частини: для авторизованого користувача та для відвідувача сайту. Такий підхід забезпечує зручність використання, безпеку і чітке розмежування функціональності для зареєстрованих та незареєстрованих користувачів.

Для відвідувачів сайту доступні наступні компоненти та функціональні можливості:

- Головна сторінка, на якій розміщена інформація про додаток та короткий опис функціоналу та переваг.
- Навігація на якій розміщені усі доступні для відвідувача сторінки вебдодатку.
- Сторінка реєстрації, на якій розміщена форма для створення нового облікового запису. Після реєстрації користувач повинен підтвердити свою реєстрацію перейшовши за посиланням, яке надходить електронним

листом на вказану при реєстрації пошту. Без підтвердження користувач не може авторизуватися у вебдодатку.

- Сторінка авторизації, де розміщена форма для авторизації вже існуючого користувача.
- Сторінка «Стрічка». Будь-який користувач може переглянути вміст цієї сторінки та залишити реакцію на вподобану ним мандрівку. Після натискання кнопки «Вподобайка», мандрівка піднімається у рейтингу і з’являється на сторінці «Популярні мандрівки».
- На сторінці «Популярні мандрівки» представлені найбільш популярні подорожі. Мандрівки розташовані в порядку спадання популярності, тобто ті, які мають більше вподобань, розміщуються вище на сторінці.

Для зареєстрованих користувачів доступний увесь функціонал що і для звичайного відвідувача додатку. Окрім цього, вони мають доступ до сторінки «Щоденник». Візуально сторінка поділена на 3 частини:

- «Додання мандрівки». В цій частині доступна форма для внесення нових подорожей. Форма включає такі поля: назва, опис, категорії (з можливістю додавання та видалення), публічність категорії та мандрівки (чек-бокси), нотатка та кнопки для управління формою: "Додати категорію", "Видалити категорію", "Додати нотатку", "Видалити нотатку", "Додати фото", "Відправити".
- «Усі твої мандрівки». Ця частина містить перелік усіх мандрівок користувача.
- «Редагування мандрівки». Частина містить перелік всіх мандрівок, які можна редагувати. Під час вибору конкретної мандрівки відкривається форма, що містить всі поля форми для додання мандрівки, а також додаткові поля «Видалити фото» та «Видалити мандрівку».

3.2 Серверна частина вебдодатку

Вебдодаток «Щоденник мандрівника» має обширну архітектуру, що включає як клієнтську, так і серверну частини. Серверна частина побудована з використанням Node.js з фреймворком Express.js, що надає швидку та ефективну обробку запитів користувачів. База даних реалізована з використанням MongoDB, а в якості хмарного середовища для зберігання фотографій обраний сервіс Uploadcare.

Серверна сторона вебдодатку має два основних роутера: один для користувачів, а інший для мандрівок. Роутери – це частини серверного коду, які відповідають за обробку запитів користувачів на певні URL-адреси. Кожен роутер містить набір ендпоінтів, які представляють конкретні дії або операції, які можна виконати на сервері. Роутер для користувачів містить різні ендпоінти, кожен з яких включає перевірку та валідацію вхідних даних. Роутер для користувачів містить наступні ендпоінти:

- /register: POST-запит для реєстрації нового користувача. Вхідні дані перевіряються на відповідність схемі registerSchema, щоб гарантувати правильність даних перед їхнім збереженням у базі.
- /verify/:verificationToken: GET-запит для підтвердження адреси електронної пошти під час реєстрації.
- /verify: POST-запит для повторної відправки листа для підтвердження адреси електронної пошти.
- /login: POST-запит для авторизації користувача. Вхідні дані перевіряються на відповідність схемі loginSchema.
- /current: GET-запит для отримання інформації про поточного авторизованого користувача. Для доступу до цього ендпоінту необхідна аутентифікація.
- /logout: POST-запит для виходу з облікового запису. Для доступу до цього ендпоінту також необхідна аутентифікація.

Роутер для мандрівок включає ендпоінти для створення, редагування, видалення та отримання мандрівок. Ці ендпоінти також реалізовані з урахуванням безпеки та валідації вхідних даних. Роутер для мандрівок містить наступні ендпоінти:

- /keys: GET-запит для отримання ключів доступу користувача, що необхідні для ідентифікації при видаленні фотографій з хмарного середовища.
- /allpublic: GET-запит для отримання всіх публічних мандрівок без необхідності аутентифікації.
- /trips-with-likes: GET-запит для отримання мандрівок разом з кількістю лайків.
- /: GET-запит для отримання всіх мандрівок користувача. Для доступу до цього ендпоінту необхідна аутентифікація.
- /:tripId: GET-запит для отримання конкретної мандрівки за її унікальним ідентифікатором. Для доступу до цього ендпоінту також потрібна аутентифікація.
- /: POST-запит для додавання нової мандрівки. Для доступу до цього ендпоінту також необхідна аутентифікація.
- /:tripId: DELETE-запит для видалення конкретної мандрівки за її унікальним ідентифікатором. Для доступу до цього ендпоінту також потрібна аутентифікація.
- /:tripId: PATCH-запит для оновлення даних про конкретну мандрівку. Для доступу до цього ендпоінту також необхідна аутентифікація та валідація вхідних даних.

Для забезпечення безпеки та правильності обробки даних використовуються мідлвари. Це функції, які обробляють запити користувачів до сервера перед тим, як вони потрапляють до обробників маршрутів Наприклад, мідлвар validateBody використовується для перевірки вхідних даних згідно певної схеми, щоб гарантувати їхню відповідність вимогам.

В ендпоінтах, які відповідають за користувачів, /register або /login, контролери виконують функції реєстрації або авторизації користувачів. Коли користувач надсилає запит на реєстрацію, дані валідуються за допомогою схеми валідації registerSchema. Якщо дані відповідають правилам, вони передаються контролеру, який зберігає нового користувача в базі даних. Аналогічно, при авторизації користувача дані перевіряються за схемою loginSchema, після чого контролер перевіряє правильність введених даних та генерує токен авторизації.

Для мандрівок /trips, контролери виконують функції створення, редагування та видалення мандрівок. Кожен запит на створення або редагування мандрівки перед виконанням перевіряється за допомогою відповідної схеми валідації. Наприклад, схема addSchema визначає правила для валідації даних, які надходять на ендпоінт для створення мандрівок. Якщо дані відповідають правилам схеми, контролер зберігає або оновлює мандрівку в базі даних.

Для перевірки працездатності серверної частини вебдодатка використовувалися такі інструменти, як Postman для тестування запитів до бази даних та MongoDBCompass для зручного управління даними у MongoDB. Для підтвердження електронної адреси користувачів використовувався сервіс mailtrap.io, який створює віртуальну поштову скриньку для перехоплення та перегляду електронних листів без відправлення їх реальним адресатам.

Структура серверної частини вебдодатку наведено на рисунку 3.2.

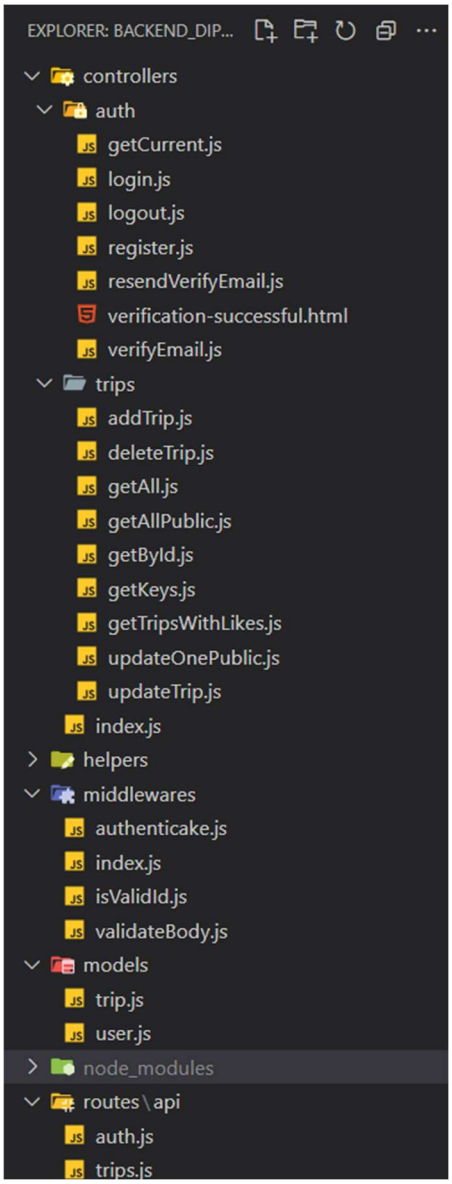

Рисунок 3.2 – Структура серверної частини

3.3 Клієнтська частина вебдодатку

Клієнтська частина вебдодатку відповідає за взаємодію користувача з додатком, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс. Для зручності взаємодії з серверною частиною використовується бібліотека Axios, яка забезпечує ефективну взаємодію з API та виконання HTTP-запитів до сервера. Для стилізації компонентів використовуються бібліотеки Ant Design, Vaterial UI і Styled Components, що

дозволяють застосовувати CSS-in-JS підхід та створювати красивий та масштабований дизайн.

Додаток використовує різні компоненти, такі як іконки, форми, списки та спіннери завантаження, для надання різноманітної функціональності користувачеві. Кожна сторінка має свій власний компонент, що дозволяє легко організувати та управляти кодом. Також використовуються різні бібліотеки, які надають додаткові можливості, такі як робота з файлами через Uploadcare або відображення сповіщень за допомогою бібліотеки React Toastify.

При реєстрації та авторизації користувачів на стороні клієнта виконується перевірка введених даних на відповідність певним правилам за допомогою валідації. Якщо дані введено невірно або не відповідають вимогам, користувачеві відображається повідомлення про помилку, яке може містити конкретну причину невдалої спроби.

Маршрутизація виконана за допомогою бібліотеки react-router-dom, що дозволяє створювати різні маршрути та керувати переходами між сторінками в додатку. Усі посилання та маршрути розміщені у компоненті Layout, що спрощує організацію навігації для користувача та підтримує єдиний стиль веб-додатку.

Після успішної авторизації користувача відбувається перенаправлення на сторінку «Щоденник», де він може переглядати та редагувати свої мандрівки.

При переключенні між сторінками з'являється лоадер, який відображається під час завантаження нової сторінки або виконання запитів до сервера.

Структура клієнтської частини вебдодатку наведено на рисунку 3.3.

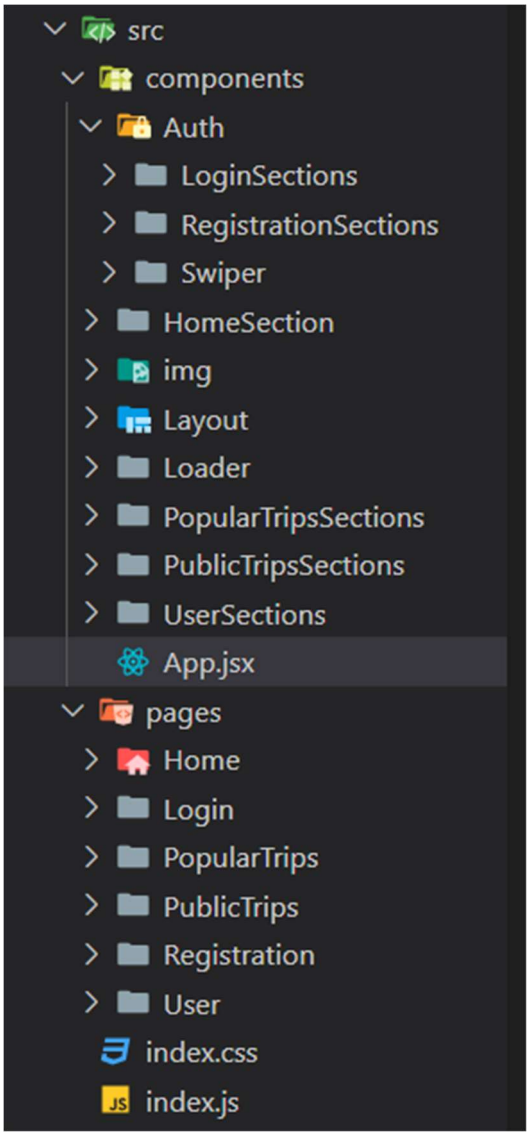

Рисунок 3.3 – Структура клієнтської частини

3.4 Використання вебдодатку

Після запуску додатку користувач потрапляє на Головну сторінку, яка призначена для візуального представлення основної інформації про додаток. На цій сторінці розміщені заголовок, опис додатку та переваги використання програми. Навігаційне меню містить посилання на інші важливі розділи додатку, такі як «Стрічка», «Популярні мандрівки», «Авторизація», «Реєстрація». Кожне посилання

веде на відповідну сторінку або компонент, де користувач зможе здійснювати потрібні дії або отримувати необхідну інформацію (рис. 3.4).

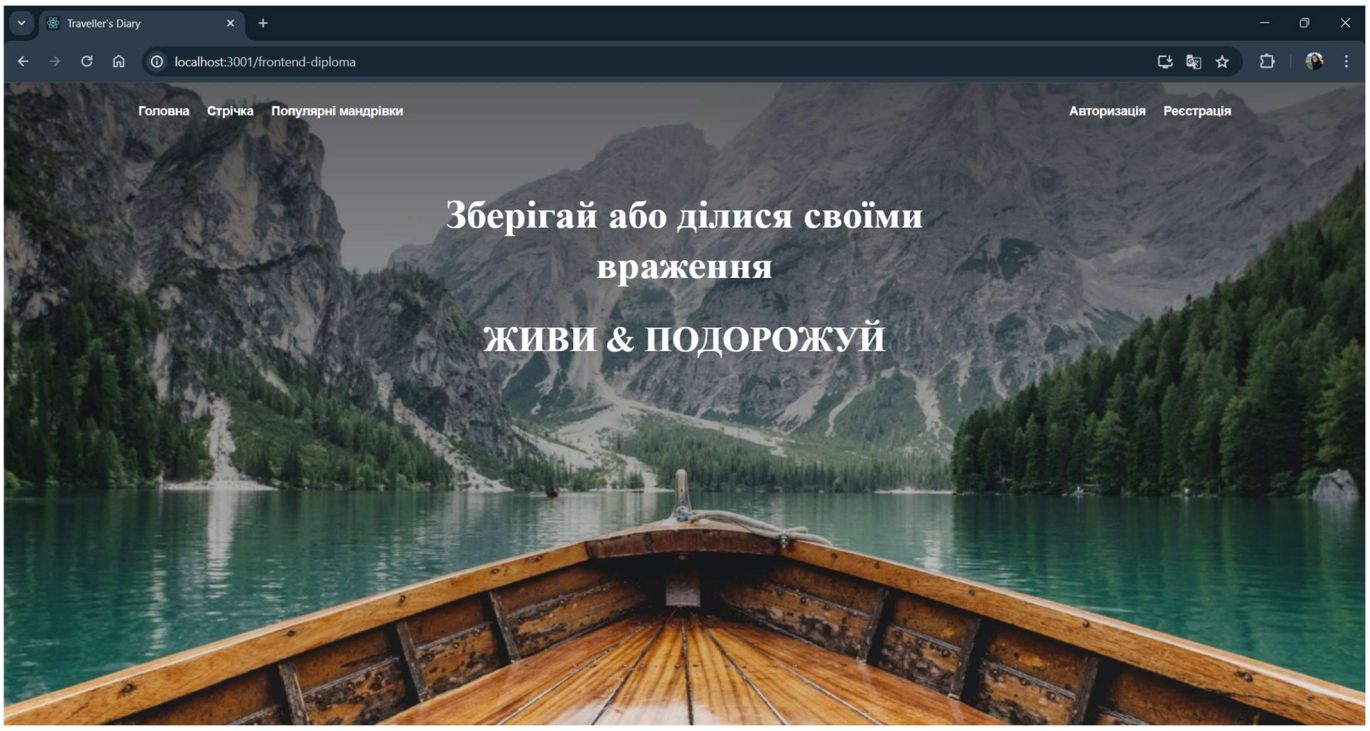

Рисунок 3.4 – Головна сторінка вебдодатку

На сторінці «Реєстрація» розміщується слайдер з можливими фотографіями з додатку, що допомагають візуально привернути увагу користувача та передати атмосферу додатку. Слайдер також використовується і на сторінці «Авторизація». Біля слайдеру знаходиться форма реєстрації, де користувач повинен ввести свої особисті дані, такі як електронну пошту та пароль (рис. 3.5). У разі невірного введення даних або помилки під час реєстрації з’являється повідомлення для користувача з інструкціями або підказками щодо виправлення помилок. Після введення коректних даних користувач повинен підтвердити свою електронну пошту через лист, який надходить йому на пошту (рис. 3.6).

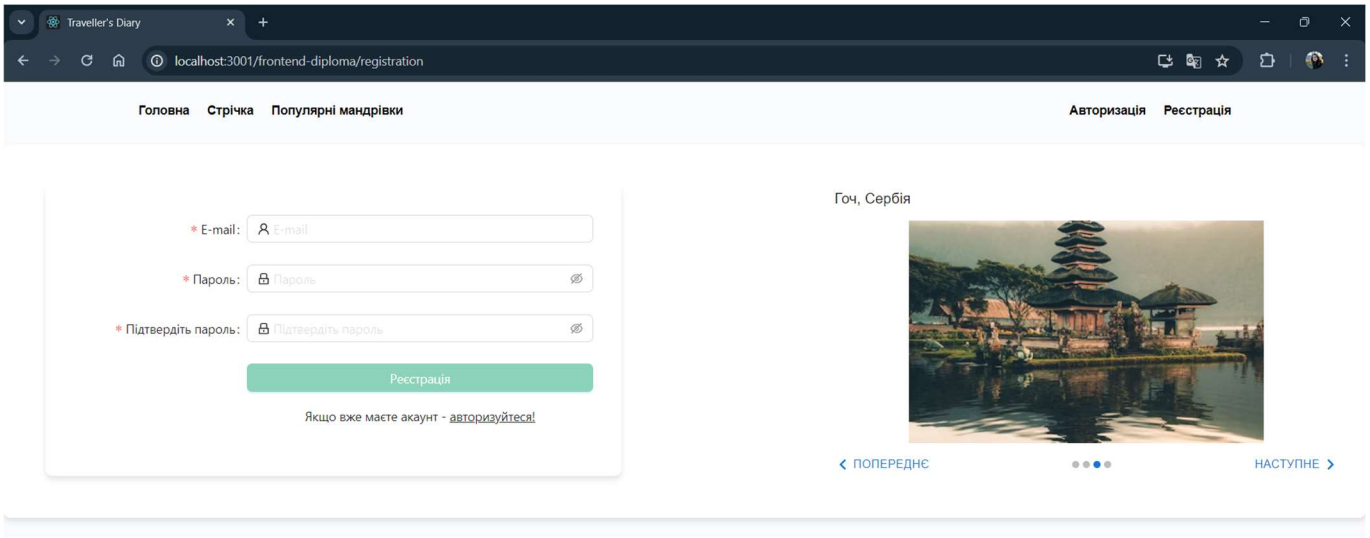

Рисунок 3.5 – Сторінка реєстрації вебдодатку

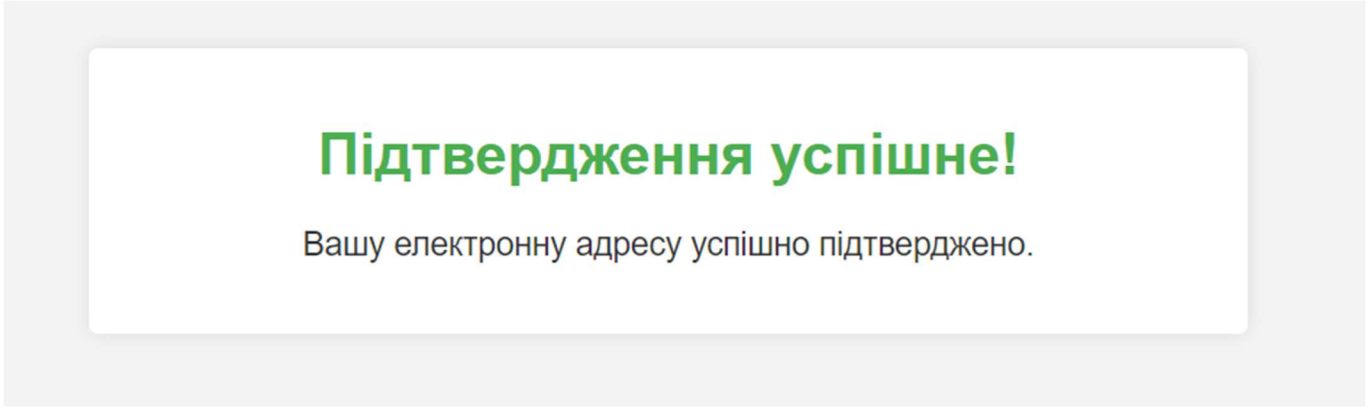

Рисунок 3.6 – Повідомлення про успішне підтвердження електронної пошти

Так, для авторизації в системі користувачу необхідно ввести відповідні облікові дані, такі як електронна адреса та пароль. Після введення цих даних і натискання на відповідну кнопку, система перевіряє їх на відповідність збереженим у базі даних. Якщо вони вірні, користувач отримує доступ до особистого облікового запису та одразу потрапляє на сторінку «Щоденник»(рис. 3.7).

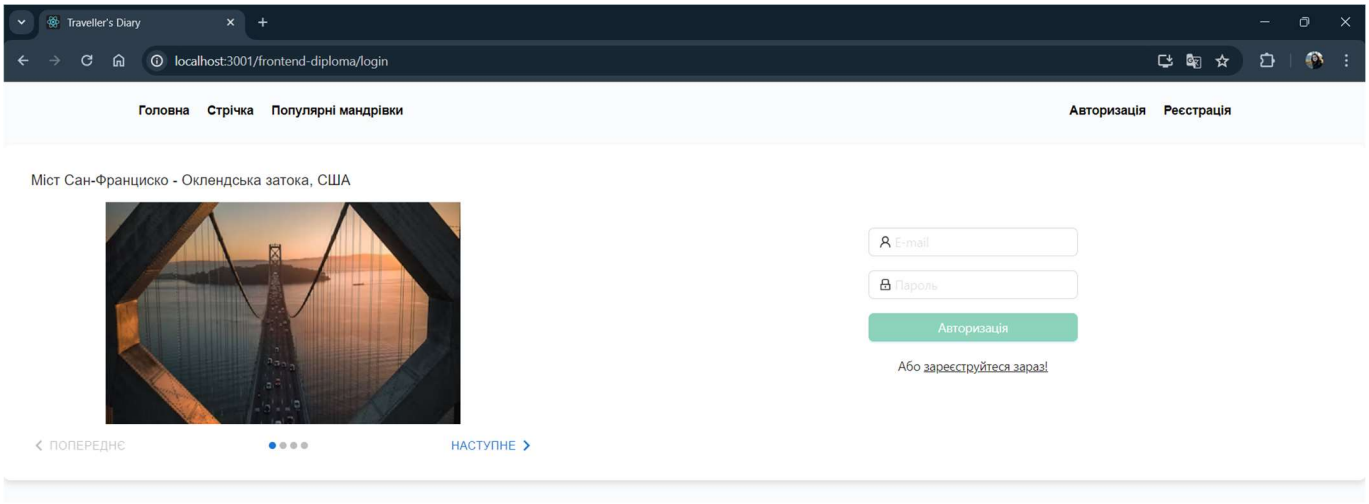

Рисунок 3.7 – Сторінка авторизації вебдодатку

Так, після успішної реєстрації нові дані користувача зберігаються у базі даних. Ці дані включають інформацію, яка була введена користувачем під час реєстрації, такі як електронна адреса та пароль. Вони зберігаються у відповідних полях бази даних, при цьому пароль зберігається у зашифрованому вигляді з метою безпеки (рис. 3.8).

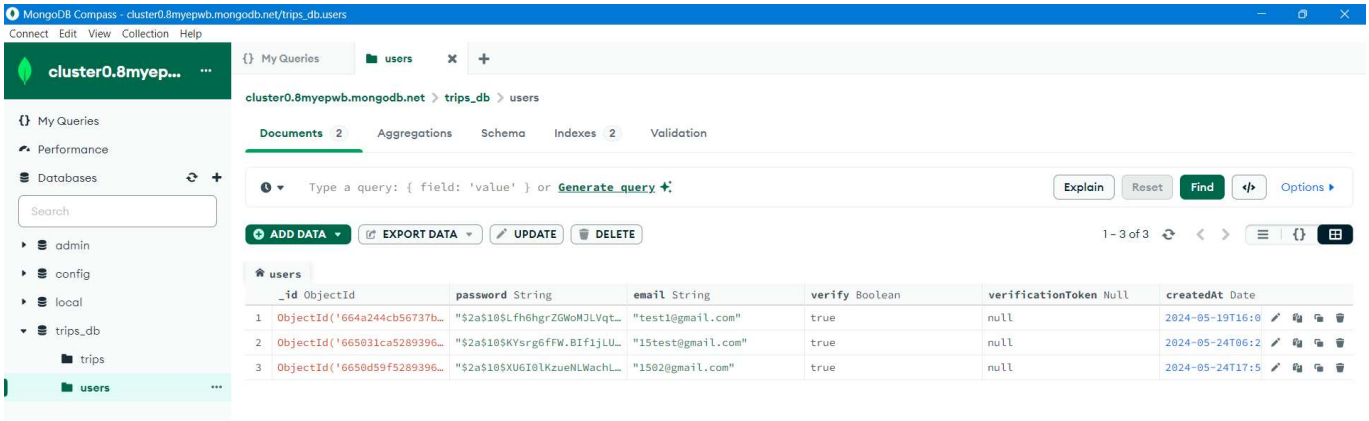

Рисунок 3.8 – Вигляд колекції «users» бази даних

На сторінці «Щоденник» користувач має можливість взаємодіяти зі своїми мандрівками у трьох основних розділах. Перший розділ «Додавання мандрівки» містить форму, де користувач може ввести всі необхідні дані про нову мандрівку, такі як назва, опис, категорії та фотографії (рис. 3.9). Після натискання кнопки «Зберегти» дані про мандрівку додаються до бази даних для подальшого використання (рис. 3.10). У другому розділі «Усі твої мандрівки» користувач може переглянути список усіх

своїх мандрівок у вигляді карток або списку з деталями. Кожна мандрівка відображається разом з назвою, описом та іншими важливими даними (рис. 3.11). У третьому розділі «Редагування мандрівок» користувач може відредагувати або видалити існуючі мандрівки. Він може змінити будь-яку інформацію про мандрівку, таку як назва, опис, категорії або фотографії, а також видалити мандрівку, яка більше не потрібна (рис. 3.12). Ці розділи забезпечують зручний інтерфейс для управління мандрівками та сприяють кращому досвіду користувача.

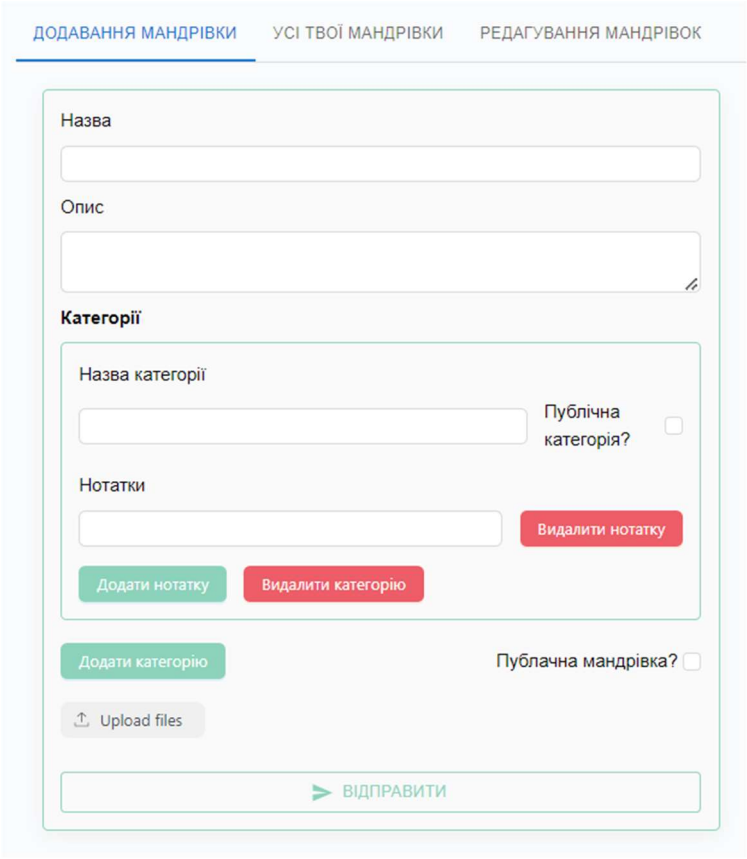

Рисунок 3.9 – Форма для додавання мандрівки

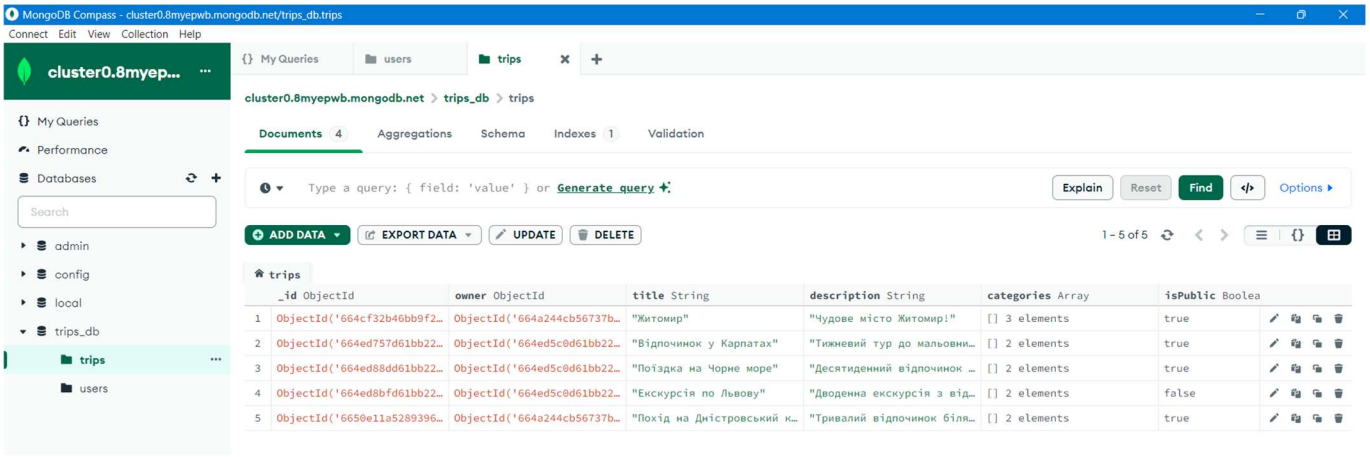

Рисунок 3.10 – Вигляд колекції «trips» бази даних

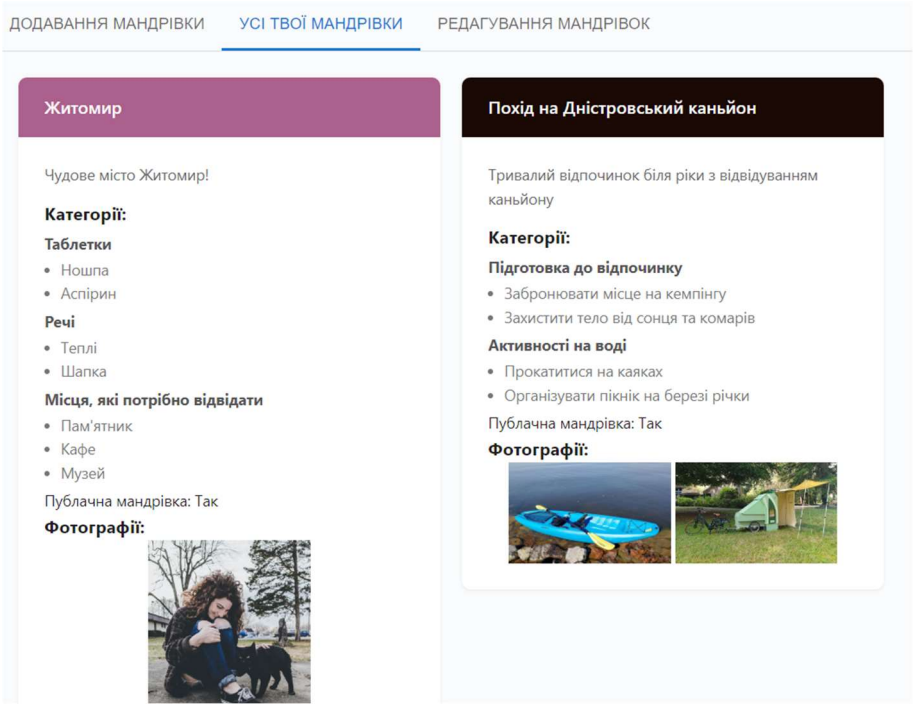

Рисунок 3.11 – Вигляд списку усіх мандрівок користувача

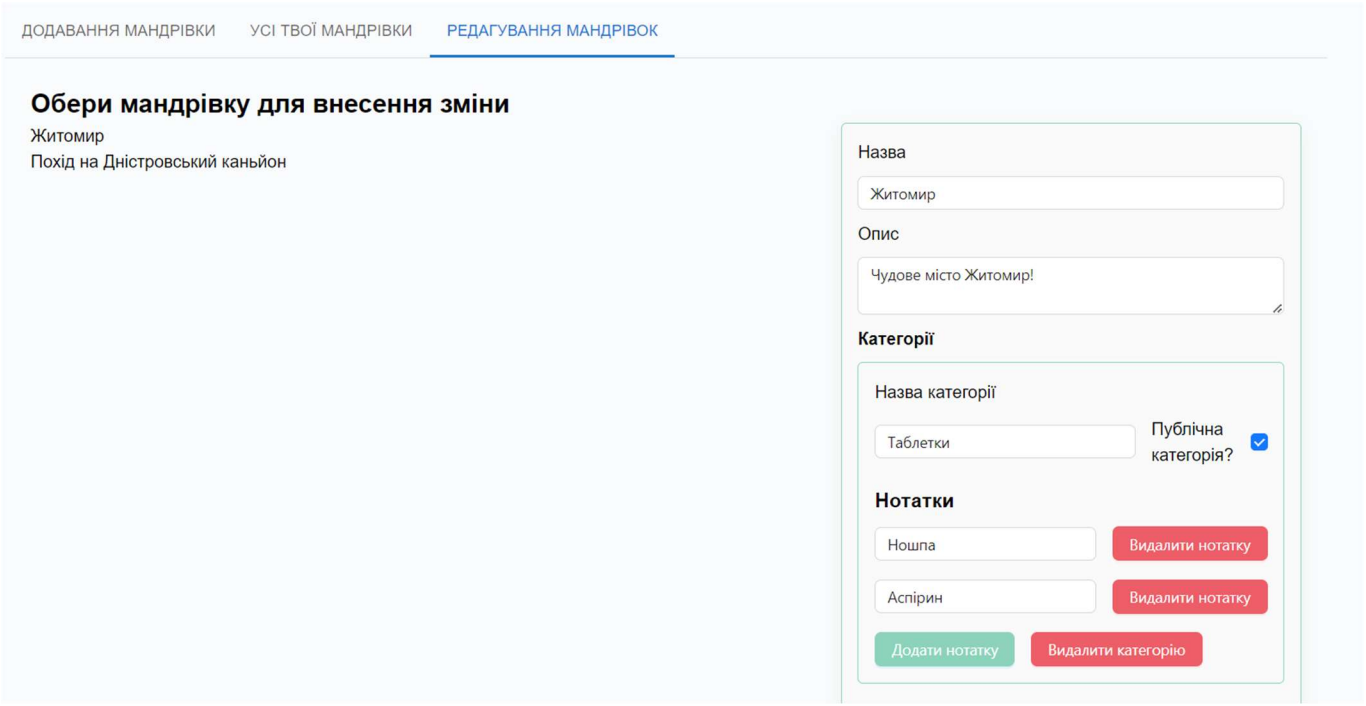

Рисунок 3.12 – Вигляд частини для редагування мандрівок

Інші частини вебдодатку доступні для всіх користувачів незалежно від того, зареєстровані вони чи ні. Сторінка «Стрічка» представляє собою перелік усіх публічних мандрівок, створених користувачами (рис. 3.13). Тут можна побачити різноманітні мандрівки, кожна з яких відображає основні деталі, такі як назва, опис, категорії та фотографії. Кожна мандрівка може містити декілька категорій, наприклад, «пам’ятки», «ресторан», «проживання» тощо. Якщо категорія мандрівки позначена як публічна, то це означає, що будь-який користувач, переглядаючи публічну мандрівку, зможе побачити і деталі цієї категорії, включаючи списки справ, що стосуються відповідної категорії. Окрім цього, користувач може залишити «Вподобайку» на будь-яку мандрівку. Чим більше вподобайок на мандрівці, тим вище вона піднімається у рейтингу на сторінці «Популярні мандрівки» (рис. 3.14). Це забезпечує динамічний та актуальний рейтинг, де користувачі можуть побачити найбільш популярні та цікаві мандрівки.

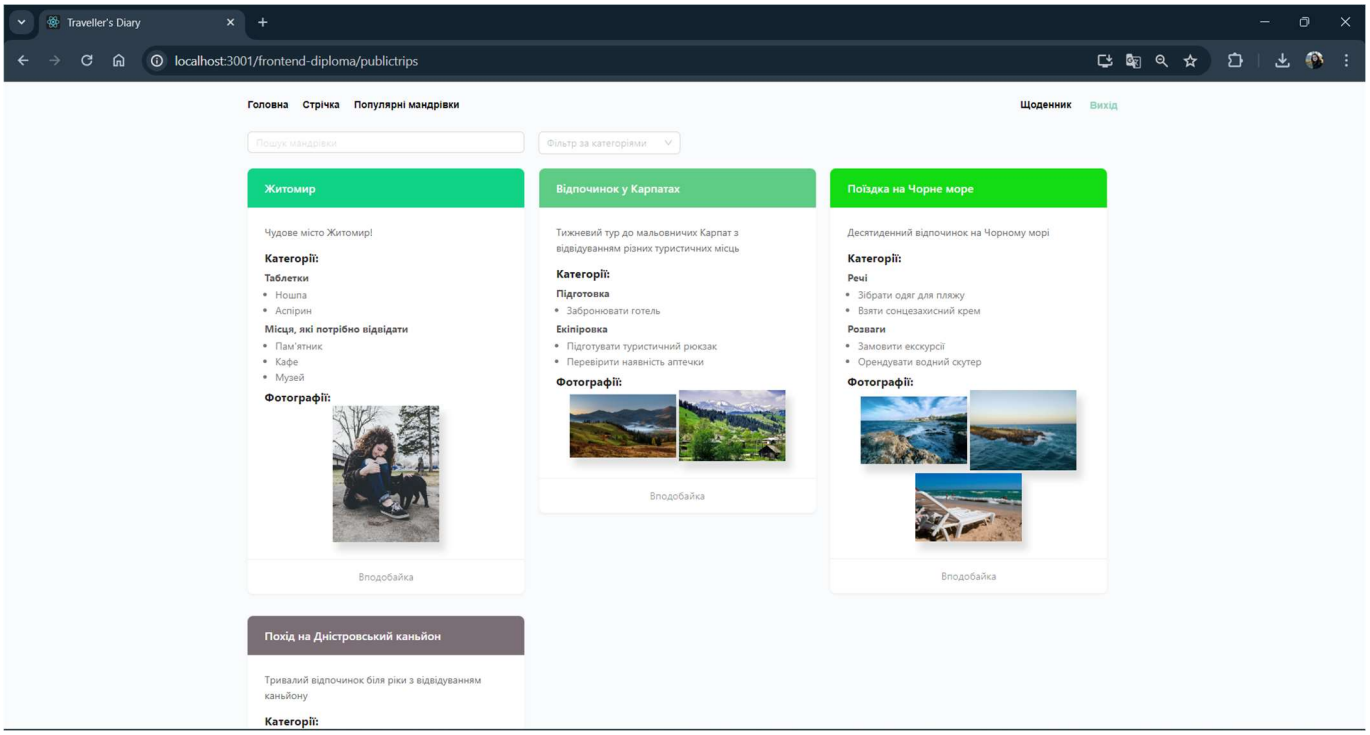

Рисунок 3.13 – Вигляд сторінки «Стрічка»

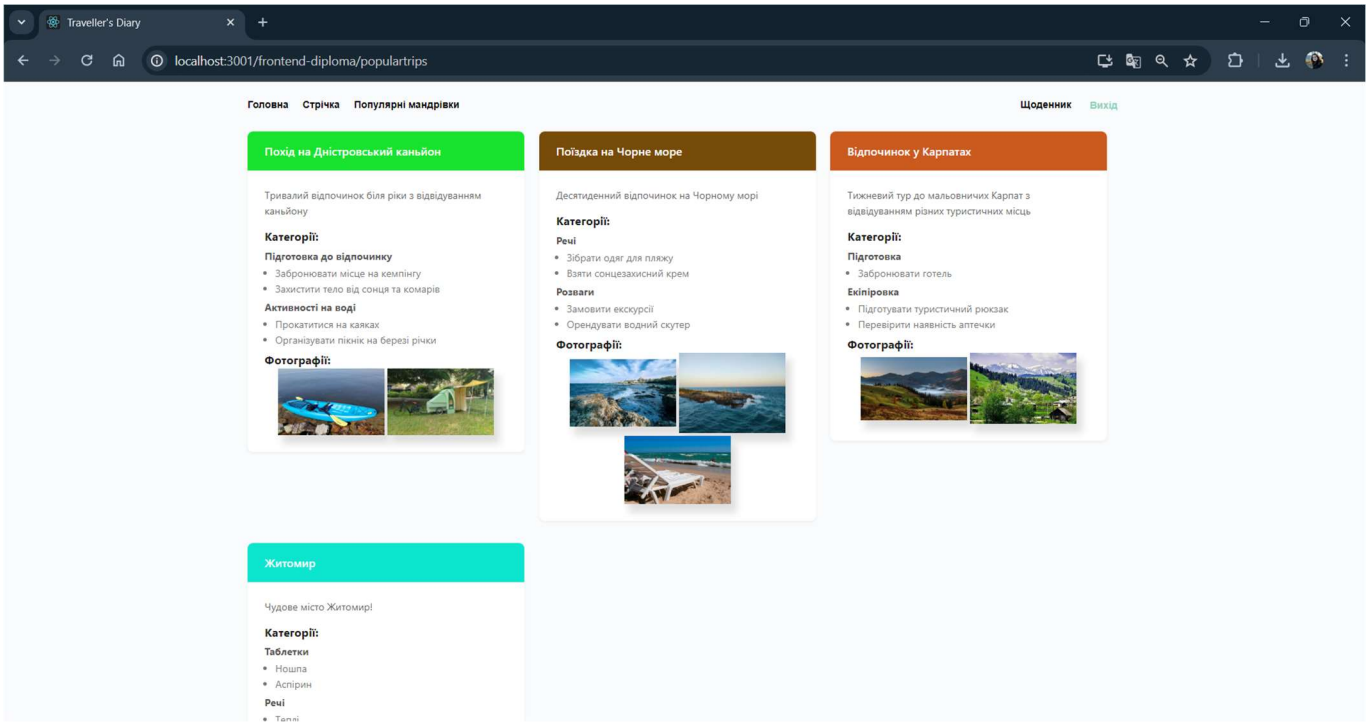

Рисунок 3.14 – Вигляд сторінки «Популярні мандрівки»

На сторінці «Стрічка» також реалізовано пошук по назвах мандрівок та фільтрацію по категоріях (рис. 3.15). При натисканні на будь-яке фото відкривається модальне вікно зі збільшеним виглядом зображення (рис. 3.16).

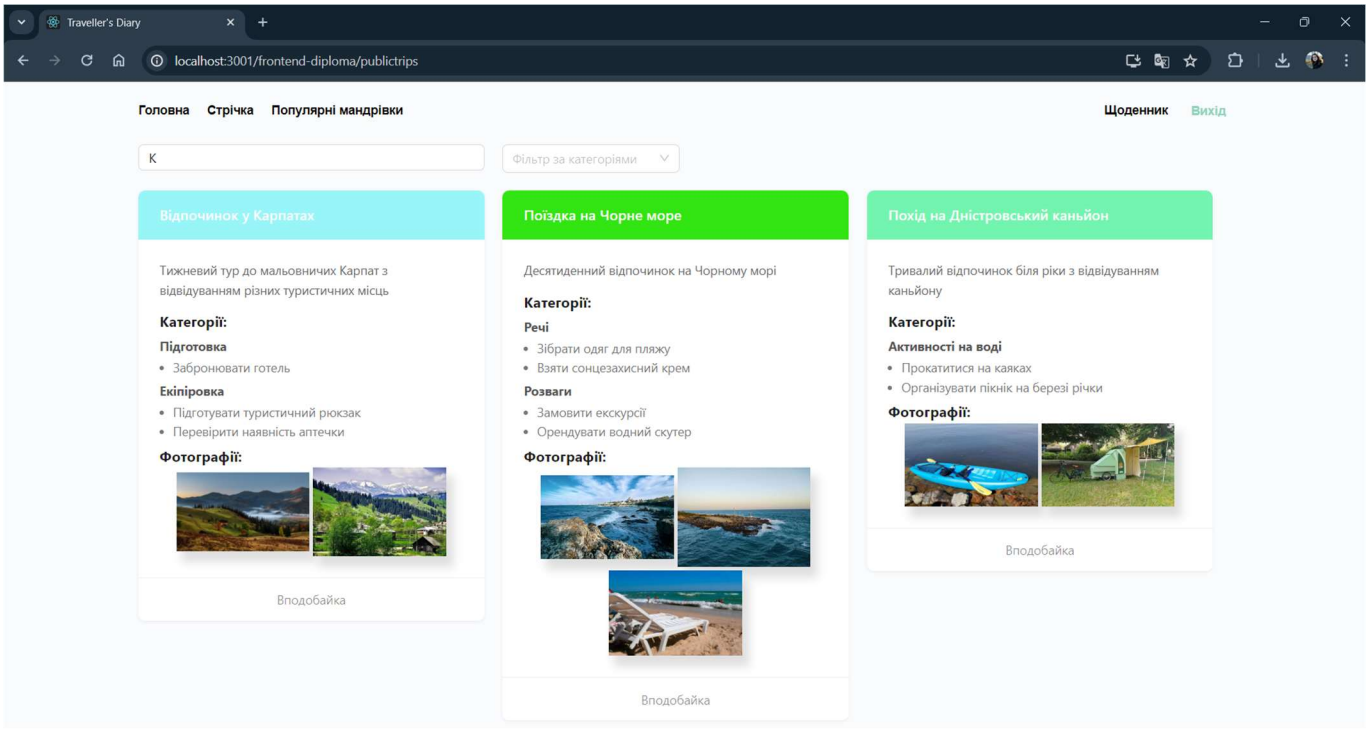

Рисунок 3.15 – Приклад роботи фільтрації на сторінці «Стрічка»

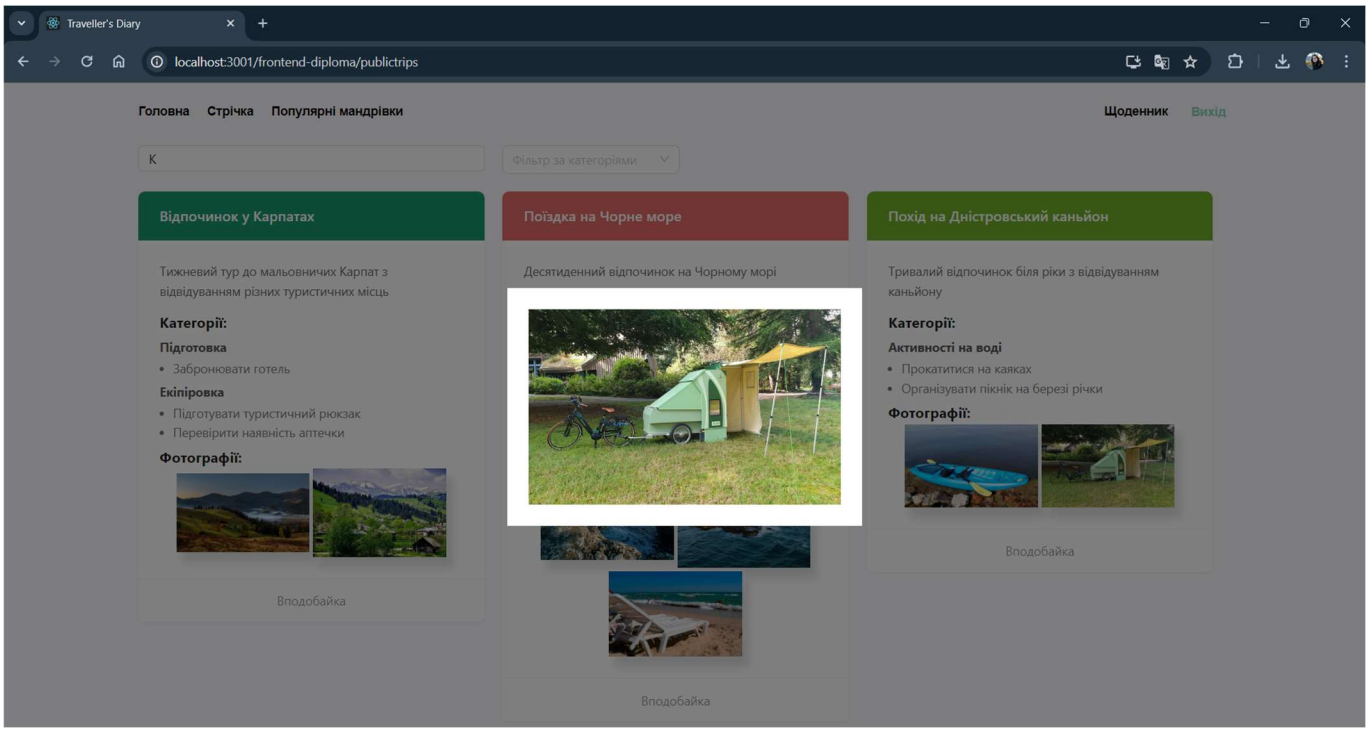

Рисунок 3.16 – Приклад вигляду модального вікна на сторінці «Стрічка»

Для наповнення бази даних використовувалися вигадані дані, які слугували для тестування та демонстрації функціоналу додатку. Картинки були взяті з особистого архіву.

Також для зручності використання вебдодатку з різних пристроїв був розроблений адаптивний дизайн для усіх сторінок додатку (рис. 3.17).

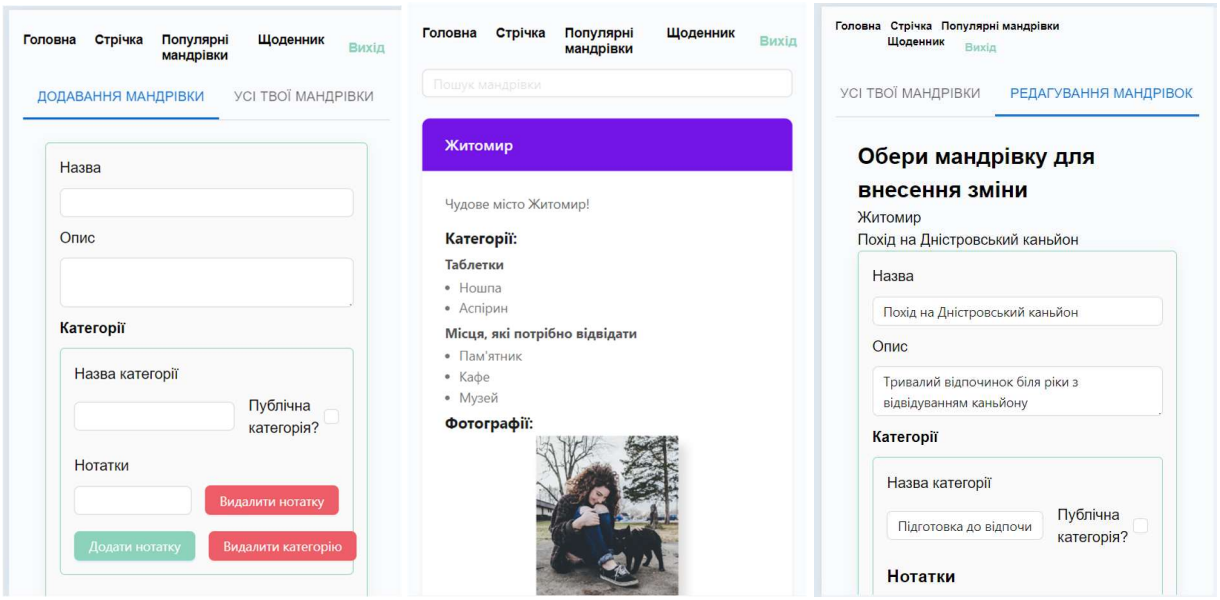

Рисунок 3.17 – Вигляд адаптивного дизайну для різних сторінок

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи бакалавра є розроблений вебдодаток «Щоденник мандрівника» для планування та збереження інформації про подорожі.

Під час роботи над кваліфікаційною роботою було вивчено останні статті та публікації, які стосуються теми розроблення вебдодатків, а також використання інформаційних технологій для нотування вражень від подорожі. Було проведено порівняльний аналіз програм аналогів і встановлено ключові критерії для вебдодатку з планування мандрівок.

У рамках роботи була сформульована постановка задачі, побудована структура розподілу робіт та розроблені основні концепції для додатку «Щоденник мандрівника». Було проведено проектування та моделювання додатку та побудовано діаграми для графічного представлення роботи вебдодатку.

У результаті виконання кваліфікаційної роботи був розроблений вебдодаток «Щоденник мандрівника». Була розроблена серверна частина, яка забезпечує зв’язок між клієнтом і базою даних. Клієнтська частина складається з сторінок авторизації та реєстрації, сторінки користувача для додавання, редагування та перегляду приватних мандрівок, а також сторінки для перегляду публічного контенту та популярних мандрівок. Для кожної сторінки та компонента був розроблений адаптивний дизайн.

Результати роботи були представлені на Міжнародній науковій конференції ІМА 2024 у Сумському державному університеті (додаток В). Лістинг програмних модулів вебдодатку міститься в додатках Г, Д.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Castrillon C. Why The Digital Nomad Lifestyle Is On The Rise. Forbes. URL: https://www.forbes.com/sites/carolinecastrillon/2022/07/17/why-the-digital-nomad-lifestyle-is-on-the-rise/?sh=3c45dcda4934 (date of access: 19.04.2024).

2. Lin Z., Rasoolimanesh S. M. Sharing tourism experiences in social media: a systematic review. Anatolia. 2022. P. 1–15. URL: https://doi.org/10.1080/13032917.2022.2120029 (date of access: 20.04.2024).

3. Sanjoy Kumar Acharjee, Tanvir Ahmed. The impact of social media on tourists' decision-making process: An empirical study based on Bangladesh. Journal of Social Sciences and Management Studies. 2023. Vol. 3, no. 1. P. 56–71. URL: https://doi.org/10.56556/jssms.v3i1.685 (date of access: 19.04.2024).

4. How smart tourism technologies affect tourist destination loyalty / N. Azis et al. Journal of Hospitality and Tourism Technology. 2020. Vol. 11, no. 4. P. 603–625. URL: https://doi.org/10.1108/jhtt-01-2020-0005 (date of access: 22.04.2024).

5. Munar A. M., Jacobsen J. K. S. Trust and Involvement in Tourism Social Media and Web-Based Travel Information Sources. Scandinavian Journal of Hospitality and Tourism. 2013. Vol. 13, no. 1. P. 1–19. URL: https://doi.org/10.1080/15022250.2013.764511 (date of access: 20.04.2024).

6. Rashaad J. Ask Skift: What Are the Major Trends in Travel Tech?. Skift. URL: https://skift.com/2023/12/11/ask-skift-what-are-the-major-trends-in-travel-tech/ (date of access: 20.04.2024).

7. Tourist Information Delivered Through Mobile Devices: Findings from the Image Project / S. J. Edwards et al. Information Technology & Tourism. 2006. Vol. 8, no. 1. P. 31–46. URL: https://doi.org/10.3727/109830506778193887 (date of access: 28.04.2024).

8. Web application for recommending personalised mobile tourist routes / D. Gavalas et al. IET Software. 2012. Vol. 6, no. 4. P. 313. URL: https://doi.org/10.1049/iet-sen.2011.0156 (date of access: 24.04.2024).

9. CoolPath: An Application for Recommending Pedestrian Routes with Reduced Heatstroke Risk / T. Xia et al. Web and Wireless Geographical Information Systems. Cham, 2020. P. 14–23. URL: https://doi.org/10.1007/978-3-030-60952-8_2 (date of access: 24.04.2024).

10. Ho C.-I., Lin M.-H., Chen H.-M. Web users' behavioural patterns of tourism information search: From online to offline. Tourism Management. 2012. Vol. 33, no. 6. P. 1468–1482. URL: https://doi.org/10.1016/j.tourman.2012.01.016 (date of access: 23.04.2024).

11. On the tour planning problem / C. Zhu et al. Annals of Operations Research. 2010. Vol. 192, no. 1. P. 67–86. URL: https://doi.org/10.1007/s10479-010-0763-5 (date of access: 20.04.2024).

12. Travel.sygic. Travel.sygic. URL: https://www.sygic.com (date of access: 22.12.2023).

13. TripIt | Online travel itinerary and trip planner. TripIt | Online travel itinerary and trip planner. URL: https://www.tripit.com/web (date of access: 22.12.2023).

14. PlanYourTrip. PlanYourTrip. URL: https://planyourtrip.com/ (date of access: 22.12.2023).

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інформаційної системи
«Вебдодаток “Щоденник мандрівника”»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій
_____ Парфененко Ю.В.

Студентка групи ІТ-02
_____ Ніколенко С.О.

Суми 2024

1. Призначення й мета створення вебдодатку

1.1 Призначення вебдодатку

Вебдодаток «Щоденник мандрівника» призначений для зручного ведення особистого журналу подорожей, зберігання фотографій, вражень та планування майбутніх подорожей.

1.2 Мета створення вебдодатку

Створення зручного інструменту для подорожуючих, сприяючи їхній мотивації вести журнал подорожей, ділитися враженнями та стимулювати подорожувати до нових місць.

1.3 Цільова аудиторія

До цільової аудиторії вебдодатку входять подорожуючі будь-якого віку та статі, які зацікавлені у зберіганні вражень від подорожей, плануванні майбутніх подорожей та обміном досвідом з іншими мандрівниками.

2 Вимоги до вебдодатку

2.1 Вимоги до вебдодатку в цілому

2.1.1 Вимоги до структури й функціонування вебдодатку

Вебдодаток має використовувати клієнт-серверну архітектуру, де клієнтська частина розроблена на React.js з використанням адаптивного дизайну та маршрутизації через react-router-dom, а серверна частина – на Node.js з Express.js та нереляційною базою даних MongoDB. Функціонально вебдодаток повинен забезпечувати реєстрацію та авторизацію користувачів, створення, перегляд, редагування та видалення мандрівок, а також інтерактивні елементи, такі як сповіщення та лоадери, для покращення користувацького досвіду.

2.1.2 Вимоги до персоналу

Від персоналу не вимагається глибоких технічних знань для підтримки та експлуатації вебдодатку на базі React JS. Основними вимогами є базові навички роботи з інтерфейсом користувача та стандартними функціями веб-браузера, а також розуміння HTML, CSS, JavaScript та Node.js.

2.1.3 Вимоги до збереження інформації

Уся інформація вебдодатку буде зберігатися у нереляційній базі даних MongoDB, забезпечуючи надійність, безпеку та ефективність зберігання даних.

2.1.4 Вимоги до розмежування доступу

Розроблюваний вебдодаток має бути відкритим для усіх користувачів.

Відвідувачі сайту мають можливість переглядати розміщені повідомлення та шукати відповідний запис за міткою.

Зареєстровані користувачі мають можливість створювати нотатки з подорожі та ділитися цими нотатками з іншими, публікуючи їх для загального доступу.

2.2 Структура вебдодатку

2.2.1 Загальна інформація про структуру вебдодатку

Вебдодатку складається з різних сторінок, які одночасно є пунктами головного меню.

Такими пунктами є:

Головна – головне меню та інформація про сам додаток.

Популярні мандрівки – останні популярні пости чи напрямки серед користувачів.

Стрічка – перелік постів від користувачів.

Щоденник – особистий кабінет користувача, де він може працювати над нотатками.

2.2.2 Навігація

Навігація у вебдодатку буде реалізована у верхній частині сайту, в рамках хедера. Користувачі матимуть можливість переходити з будь-якої сторінки на інші шляхом використання меню навігації. Меню буде постійно відображатися на всіх сторінках для зручності користувачів, дозволяючи їм швидко переміщатися між різними частинами вебдодатку.

2.2.3 Наповнення вебдодатку (контент)

Управління контентом головної сторінки та стрічки буде здійснювати адміністратор сайту.

Наповнення стрічки буде залежати від контенту, який публікують користувачі. Вміст стрічки буде формуватися на основі постів, які додають користувачі. Кожен новий пост, який додається до стрічки, буде відображатися в порядку їхньої публікації.

Контент сторінки «Популярні мандрівки» буде залежати від кількості взаємодій з певним постом. Пости, які найбільш активно взаємодіються користувачами, будуть автоматично додаватися на сторінку «Популярні мандрівки» зі стрічки.

2.2.4 Дизайн та структура додатку

Стиль вебдодатку має бути сучасним, приємним для сприйняття, у якості основних кольорів було запропоновано використати зелені та білі відтінки.

Основою мають бути фотографії гарної якості, вебдодаток має бути інтуїтивно зрозумілим для використання.

Розташування елементів на головній сторінці вебдодатку схематично показано на рисунку А.1.

Рисунок А.1 – Схема головної сторінки

Макети сторінок вебдодатку наведені нижче на рисунку А.2 – А.6

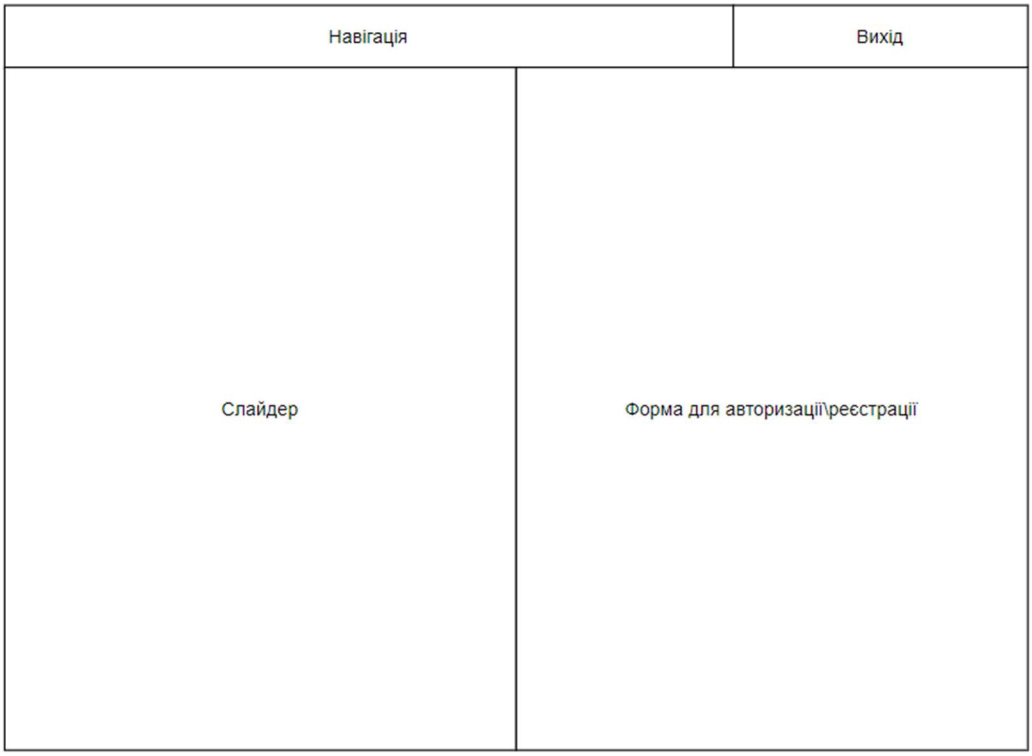

Рисунок А.2 – Схема сторінки реєстрації чи авторизації

Рисунок А.3 – Схема сторінки користувача

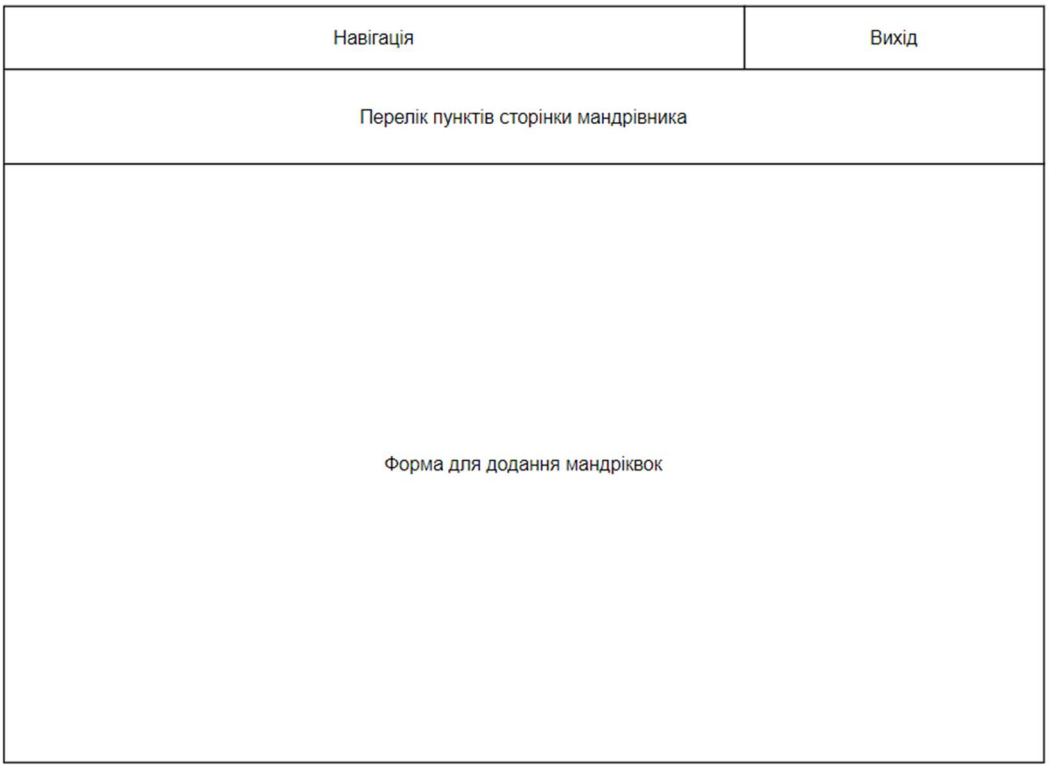

Рисунок А.4 – Схема сторінки користувача (додання мандрівок)

Рисунок А.4 – Схема сторінки «Стрічка»

Рисунок А.5 – Схема сторінки «Популярні мандрівки»

2.2.5 Система навігації (карта вебдодатку)

Карта вебдодатку зображена на рисунку А.2.

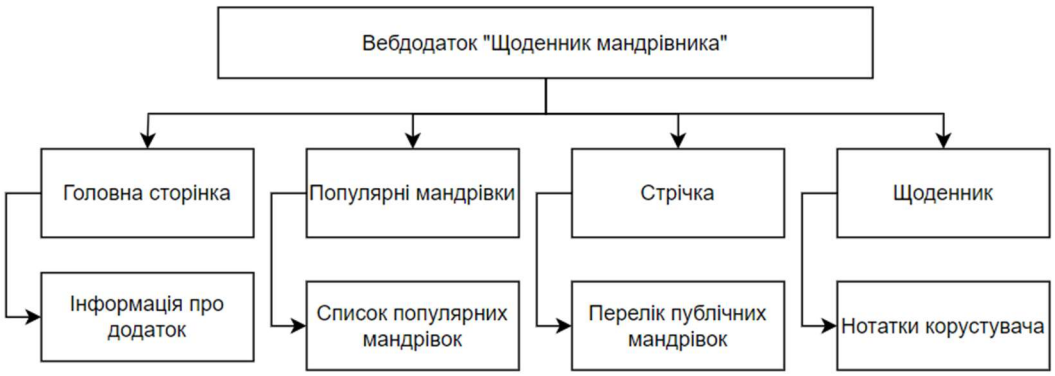

Рисунок А.2 – Карта вебдодатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Перегляд головної сторінки та її контенту	Користувач, відвідувач
UN-02	Можливість реєстрації	Відвідувач
UN-03	Можливість авторизації	Користувач
UN-04	Перегляд інформації про додаток.	Користувач, відвідувач
UN-05	Можливість додавати нотатки	Користувач
UN-06	Можливість редагувати нотаток	Користувач
UN-07	Можливість редагувати особисту інформацію	Користувач

2.3.2 Функціональні вимоги

Проаналізувавши потреби користувачів були визначені наступні функціональні вимоги:

- наявність реєстрації та авторизації клієнтів;
- підтвердження реєстрації через пошту;
- пошук постів по міткам;

– наявність інформації про застосунок;
– розмежування контенту на приватний і публічний;
– можливість адміністрування власної сторінки;
– можливість додавання та редагування заміток про подорож;
– можливість додавання фотографій та, за потреби, підписів до них;
– можливість додавання заміток про декілька подорожей.

2.3.3 Системні вимоги

Проаналізувавши потреби користувачів вебдодатку «Щоденник мандрівника», визначено наступні вимоги:

- наявність сторінки з інформацією про вебдодаток та навігаційного менна Головній сторінці;
- можливість реєстрації та авторизації користувачів з валідацією даних і сповіщеннями про помилки;
- доступ авторизованих користувачів до сторінки «Щоденник», що включає функціонал для додавання, редагування та перегляду мандрівок;
- наявність сторінки «Стрічка» для перегляду всіх публічних мандрівок з можливістю пошуку та фільтрації за категоріями;
- можливість користувачів залишати вподобайки на мандрівках, що підвищує їх рейтинг на сторінці «Популярні мандрівки»;
- забезпечення адаптивного дизайну для зручного використання додатку з різних пристроїв;
- інтеграція лоадерів при переключенні між сторінками для індикації процесу завантаження;
- відображення сповіщень для користувачів у вигляді тостів при успішних або невдалих запитах;
- абезпечення захищеного зберігання даних користувачів та їх мандрівок в базі даних MongoDB;

- можливість додавання, редагування та видалення мандрівок із записом змін в базі даних.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Вебдодаток реалізується за допомогою таких технологій:

- HTML
- CSS
- JS
- React
- Node.js
- MongoDB

2.4.2 Вимоги до лінгвістичного забезпечення

Вебдодаток буде розроблений з урахуванням української мови, а також може містити елементи англомовного інтерфейсу або контенту в окремих випадках.

2.4.3 Вимоги до програмного забезпечення

Клієнтська частина програмного забезпечення має бути сумісна з такими веб-браузерами:

- Веб-браузер: Chrome 2 і вище, або Safari 3.2.1 і вище, або Opera 9.5 і вище, або Internet Explorer 7.0 і вище, або Firefox 3.5 і вище.

3 Склад і зміст робіт зі створення вебдодатку

Докладний опис етапів роботи зі створення вебдодатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення вебдодатку

№	**Склад і зміст робіт**	**Строк розробки**
1	Дослідження предметної області	2 дні
2	Визначення актуальності роботи	1 день
3	Проведення аналіз аналогів вебдодатків	4 дні
4	Визначення функціональних вимог	3 дні
5	Розробка структури додатку	2 днів
6	Визначення функціоналу додатку	5 днів
7	Планування задач та організація робочого процесу	3 дні
8	Створення дизайну та шаблону сторінок	7 днів
9	Верстка сторінок	15 днів
10	Наповнення контентом сторінки	5 днів
11	Розробка бази даних	9 днів
12	Розробка авторизації та реєстрації	8 днів
13	Розробка системи додавання інформації	37 днів
14	Розробка панелі адміністратора	5 днів
15	Тестування	6 днів
16	Тестування системи на стійкість	5 днів
17	Написання супровідної документації	5 день
18	Розміщення застосунку на хостингу	1 днів

Продовження табл. А.3

19	Проведення релізу	1 день
20	Збір відгуків про додаток	1 день
21	Виправлення помилок у роботі додатку	3 дні
	Загальна тривалість робіт	128 днів

4 Вимоги до складу й змісту робіт із введення вебдодатку в експлуатацію

Щоб зробити вебдодаток доступним для клієнтів та відвідувачів, потрібно розмістити його в мережі Інтернет. Для цього необхідно придбати доменне ім'я та простір на хостингу. Вміст бази даних буде перенесено на хостинг так само, як і вебдодаток, де він буде розвиватися далі. Для успішного перенесення на хостинг важливо, щоб параметри хостингу відповідали вимогам, вказаним у ТЗ.

ДОДАТОК Б

Планування робіт

Деталізація мети проєкту методом SMART. Продуктом дипломного проєкту є вебдодаток «Щоденник мандрівника».

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити вебдодаток «Щоденник мандрівника», який дозволить користувачам планувати та зберігати враження від подорожей, а також ділитися своїми нотатками з іншими користувачами.
Measurable (вимірювана)	Завершена та впроваджена розробка.
Achievable (досяжна)	Розробка вебдодатку буде виконана за допомогою технологій HTML, CSS, JS, React, Node.js та бази даних MongoDB. Для написання коду буде використана середовище розробки Visual Studio Code.
Relevant (реалістична)	Створення «Щоденника мандрівника» відповідає потребам сучасних подорожуючих та користувачів, які шукають зручний інструмент для планування та документування своїх подорожей.
Time-framed (обмежена у часі)	Термін встановлений – до 1 червня 2024 року.

Планування змісту структури робіт. Основним інструментом для планування структури проекту є діаграма WBS (Work Breakdown Structure), яка представляє графічну ієрархію згрупованих елементів проекту у вигляді пакету робіт. Ці роботи взаємопов'язані ієрархічно, що дозволяє зв'язати їх з кінцевим продуктом проекту. Побудова WBS діаграми допомагає детально описати роботи, які необхідно виконати

на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту, розбиваючи їх на менші, більш керовані завдання. Діаграма WBS зображена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Після створення WBS, необхідно розробити організаційну структуру виконавців (OBS). OBS фокусується виключно на внутрішній організаційній структурі проекту, не враховуючи відносини між проектними групами чи учасниками і їхніми материнськими організаціями. Діаграма OBS представлена на рис. Б.2. У табл. Б.2 наведено список виконавців, які беруть участь у проекті.

Таблиця Б.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Ніколенко С.О.	Виконує front-end та back-end розробку.
Проектувальник	Ніколенко С.О.	Виконує проєктування бази даних та розробляє структуру вебдодатку.
Тестувальник	Ніколенко С.О.	Відповідає за тестування функціоналу та дизайну вебдодатку.
Косультант проекту	Парфененко Ю.В.	Формує завдання на розробку проекту.
Менеджер проекту	Ніколенко С.О.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

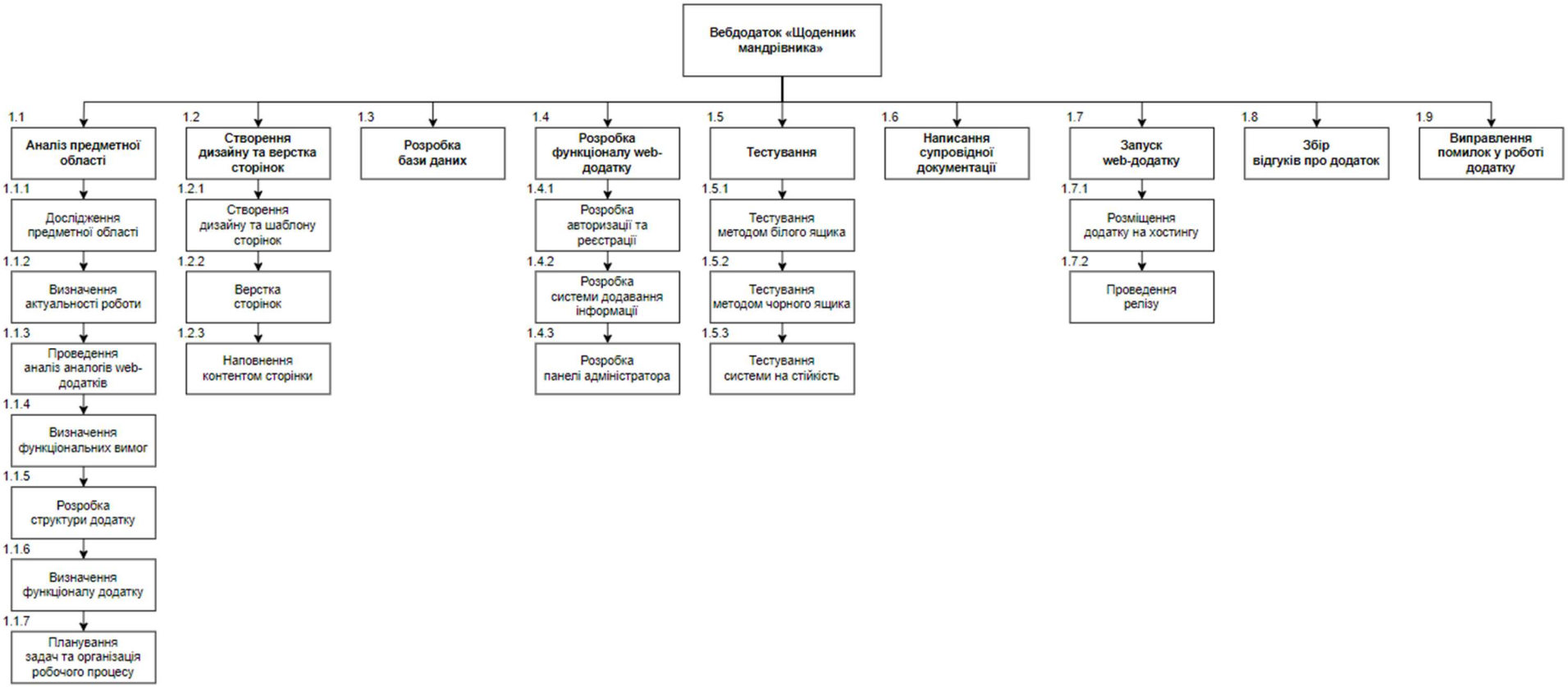

Рисунок Б.1 − WBS. Структура робіт проекту

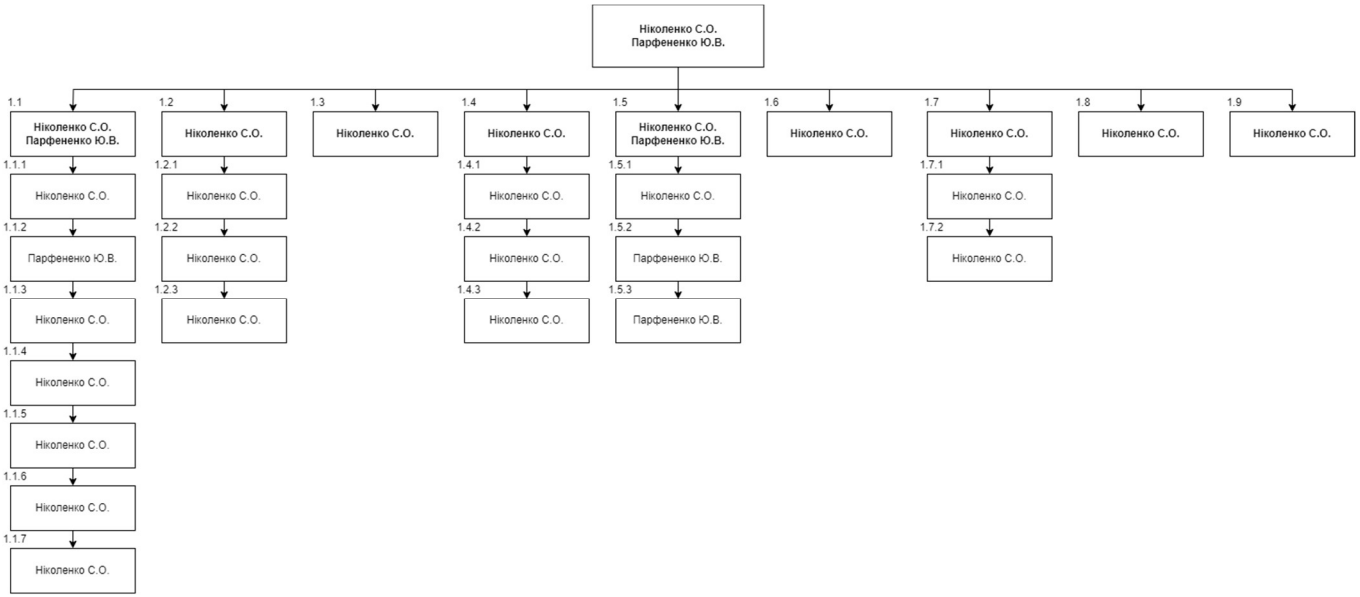

Рисунок Б.2 − Організаційна структура проекту (OBS)

Діаграма Ганта. Наступним кроком є створення календарного плану виконання дипломного проекту. Найбільш популярним форматом графіка у будь-якій галузі є діаграма Ганта. Цей графік дозволяє менеджерам проекту та всій команді розробників візуалізувати часові графіки та взаємозв'язки між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт вказується у днях, але фактична тривалість робочого дня становить приблизно 8 годин. Для отримання реального уявлення про тривалість виконання робіт з урахуванням обмежених ресурсів, а також вихідних і святкових днів, створюється календарний графік. Діаграма Ганта та перелік завдань, відображених на діаграмі, представлені на рис. Б.3-Б.5.

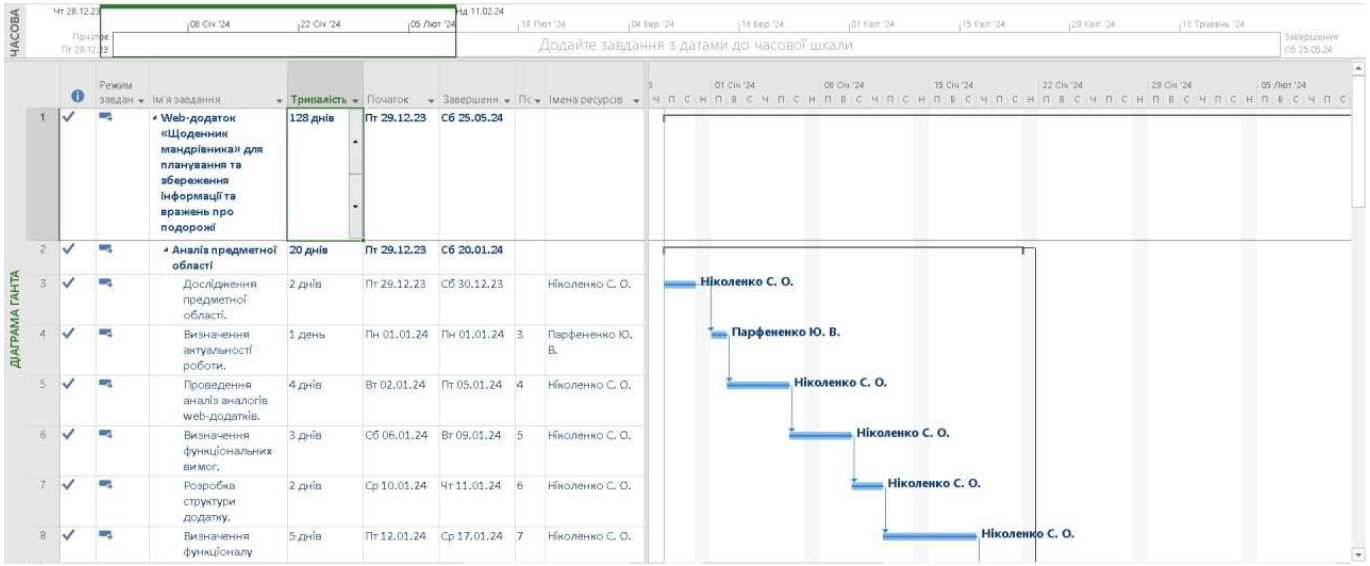

Рисунок Б.3 − Діаграма Ганта

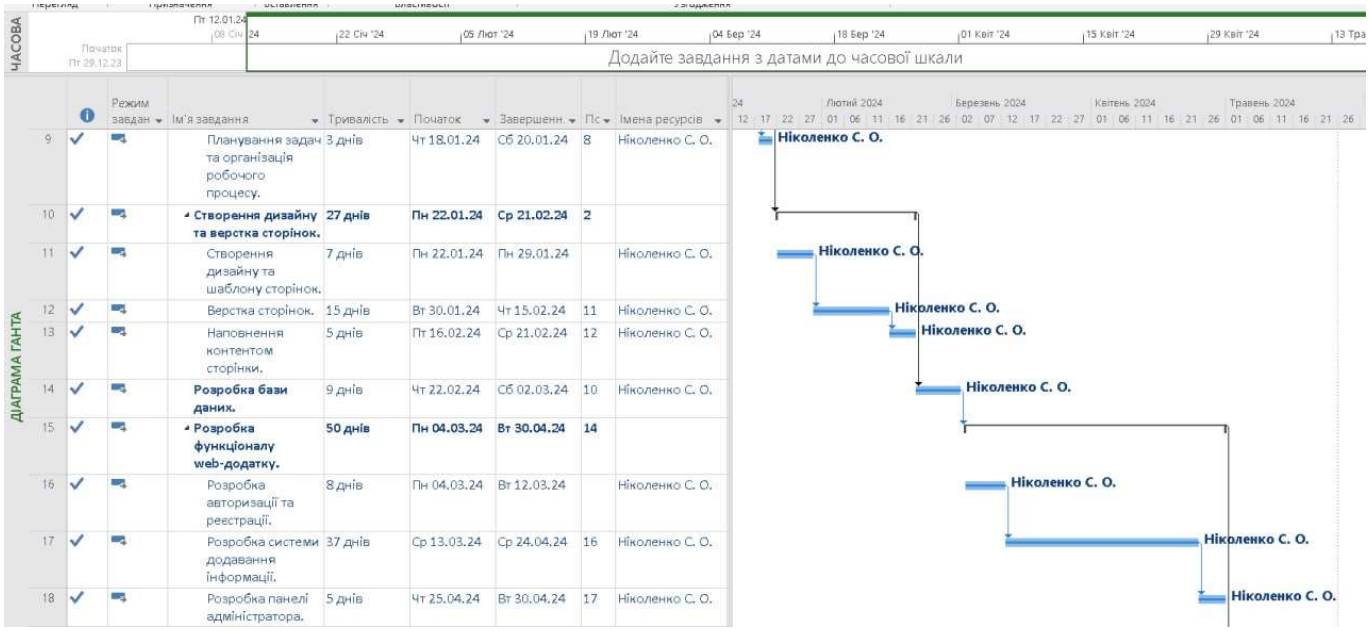

Рисунок Б.4 − Продовження діаграми Ганта

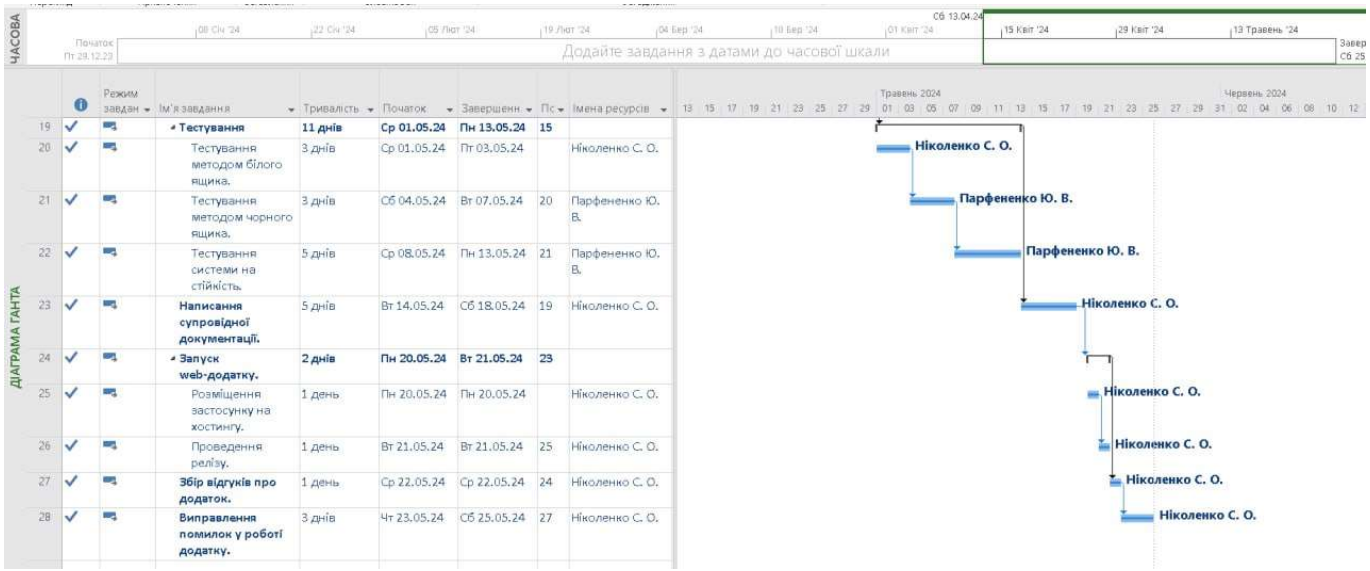

Рисунок Б.5 − Продовження діаграми Ганта

Аналіз ризиків. Здійснимо якісну та кількісну оцінку ризиків проекту. Під час якісної оцінки ідентифікуємо ризики, що потребують негайного реагування. Ця оцінка визначає ступінь важливості кожного ризику і допомагає вибрати відповідний спосіб реагування. Кількісна оцінка ризиків виконується для детальнішої ідентифікації ризиків та оцінки ступеня їх впливу на проект. Обидва типи оцінки можуть застосовуватися окремо або разом, залежно від наявного часу, бюджету і

необхідності. У табл. Б.5 наведена класифікація ризиків за показниками ймовірності їх виникнення та величини втрат.

Далі проведемо планування заходів реагування на ризики, яке включає розробку методів і стратегій зниження їх негативного впливу на проект. Визначимо ефективність заходів реагування і оцінюватимемо, чи будуть наслідки впливу ризиків позитивними або негативними. Оцінка ризиків здійснюється за показниками, що наведені в табл. Б.3. На основі цієї оцінки будуємо матрицю ймовірності виникнення ризиків та їх впливу, яка зображена на рис. Б.6.

Таблиця Б.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

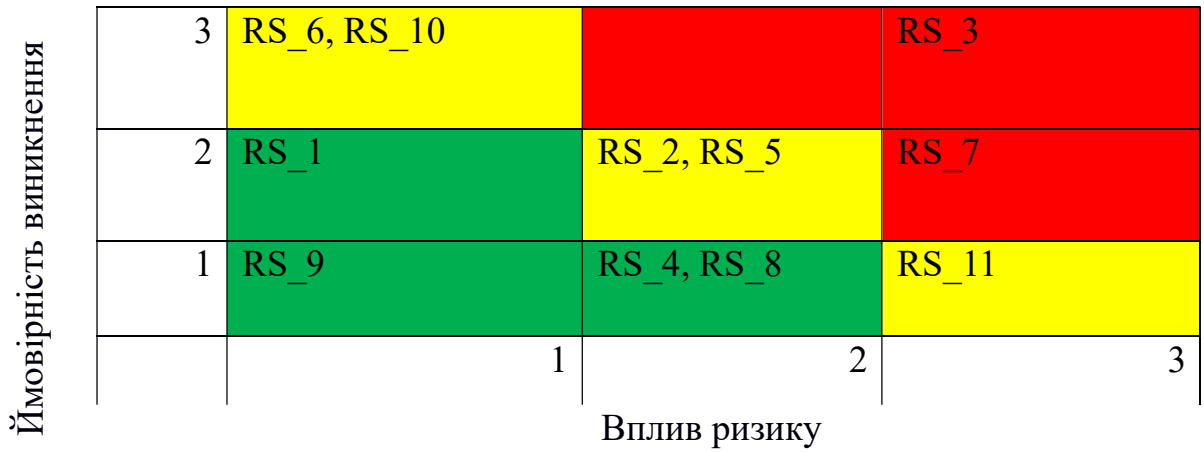

Рисунок Б.6 − Матриця ймовірності виникнення ризиків та впливу ризику

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця Б.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1,8,9,10
2	Виправдані	3≤R≤4	2,4,5,6,11
3	Недопустимі	6≤R≤9	3,7

Таблиця Б.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Несумісність з різними браузерами	Низька	Середній	2	1. Обговорення функціоналу для адаптації під різні браузери.	Пом'якшення	Розробка функціоналу для уникнення проблем з різними браузерами.
RS_2	Відкритий	Поява продукту аналогу	Низька	Високий	4	1. Здійснити попереднє вивчення продуктів аналогів. 2. Обрати унікальну підхід для створення продукту.	Прийняття	
RS_3	Відкритий	Неправильна оцінка обсягу робіт	Середня	Високий	6	1. Збільшення часу виконання робіт. Якщо на якомусь етапі залишилася менша кількість часу, можна використати цей час для наступного етапу.	Пом'якшення	Постійний моніторинг виконання робіт відповідно до заданого плану.
RS_4	Відкритий	Проблеми з використанням середовища програмування	Низька	Середній	3	1. Використання перевіреного середовища програмування. 2. Використання систем контролю версій.	Пом'якшення	Робота з альтернативним програмним забезпеченням.
RS_5	Відкритий	Часте внесення змін у ТЗ	Низька	Середній	4	1. Додаткове обговорення вимог. 2. Чітко описати вимоги до проекту. 3. Затвердити кінцеві вимоги	Перенос	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_5	Відкритий	Часте внесення змін у ТЗ	Низька	Середній	4	3. Додаткове обговорення вимог. 4. Чітко описати вимоги до проекту. 5. Затвердити кінцеві вимоги	Перенос	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.
RS_6	Відкритий	Не реалізація додаткового функціоналу	Середня	Низький	4	1. Відділення додаткового часу на реалізацію. 2. Використання готових рішень.	Пом’якшення	Обговорення внесення змін до проекту.
RS_7	Відкритий	Зміни в роботі зовнішніх сервісів	Низька	Високий	6	1. Складання списку альтернативних сервісів. 2. Обговорення особливостей роботи з альтернативними сервісами.	Попередження	Рефакторинг додатку під інші сервіси.
RS_8	Відкритий	Відсутність процесу копіювання даних	Низька	Середній	2	1. Налаштувати автоматичне резервне збереження даних. 2. Використання системи контролю версій.	Попередження	Налаштувати автоматичне збереження даних на різних етапах розробки проекту.
RS_9	Відкритий	Несуттєві зміни у дизайні	Низька	Низький	1	Обговорення можливості внесення несуттєвих змін у дизайн.	Використання	Виділення часу для внесення змін.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_10	Відкритий	Неактуальні знання та навички розробника	Низька	Середній	2	1. Використання вже вивчених технологій. 2. Обговорення використання альтернативних технологій	Перенос	Здійснювати моніторинг проекту замовником. Надання проміжних результатів виконання проекту після кожного етапу.
RS_11	Відкритий	Неповний або неефективний процес тестування	Низька	Високий	4	1. Застосування різних методів тестування. 2. Виділення більше часу на тестування	Пом’якшення	Збільшення часу на тестування та перенос релізу продукту для проведення більш детального тестування

ДОДАТОК В

Апробація результатів дослідження

Web-додаток «Щоденник мандрівника»

Ніколенко С. О., *студент ІТ-02*, Парфененко Ю. В., *доцент*

Сумський державний університет, м. Суми, Україна

У сучасному світі подорожей важливою є можливість зручного та ефективного планування та збереження вражень. Процес ведення щоденника подорожі та планування мандрівок вимагає значних зусиль від подорожуючих. З урахуванням цих факторів, актуальність розроблення додатку «Щоденник мандрівника» надається платформою для планування подорожі, створення нотаток та поширення вражень від мандрівок.

Мета створення web-додатку «Щоденник мандрівника» – надати користувачам зручну платформу для планування та документування подорожей. Цей додаток допомагає подорожуючим створювати нотатки, ділитися вибраними нотатками з іншими користувачами, зберігати враження від подорожей та ефективно організовувати свої мандрівки. Платформа «Щоденник мандрівника» розроблена з використанням web-інструментів і має специфічний набір функцій, таких як реєстрація та вхід в систему; особистий простір для мандрівника, де він може планувати та вести записи про свої подорожі; головна сторінка, де доступні публічні записи мандрівників, і можливість здійснювати пошук за категоріями, такими як хештеги.

Дизайн платформи адаптивний і підходить для перегляду на різних пристроях та веб-браузерах. Для створення цього проекту використовувалися такі інструменти, як HTML, CSS, JS, React JS та MongoDB. Користувачі web-додатку можуть належати до різних категорій: мандрівники, які хочуть зберігати всю необхідну інформацію про свої подорожі в одному місці та ділитися враженнями; особи, що цікавляться знаходженням достовірної інформації про конкретні місця та плануванням подорожей; адміністратори, які керують публічним контентом.

Отже, розроблений web-додаток «Щоденник мандрівника» є незамінним інструментом для подорожуючих, надаючи їм зручний і організований спосіб планування та документування своїх подорожей. Він забезпечує можливість створювати нотатки, зберігати враження від мандрівок та ділитися ними з іншими користувачами.

ДОДАТОК Г

Серверна частина вебдодатку

Код з файлу server.js

```
require("dotenv").config();

const mongoose = require("mongoose");

const app = require("./app");

const DB_URL = process.env.DB_URL;

mongoose
  .connect(DB_URL)
  .then(
    app.listen(3000, () => {
      console.log("Database connection successful");
    })
  )
  .catch((error) => {
    console.error(error.message);
    process.exit(1);
  });
```

Код з файлу app.js

```
const express = require("express");
const logger = require("morgan");
const cors = require("cors");

const usersRouter = require("./routes/api/auth");
const tripsRouter = require("./routes/api/trips");

const app = express();

const formatsLogger = app.get("env") === "development" ? "dev" :
"short";

app.use(logger(formatsLogger));
app.use(cors());
app.use(express.json());
app.use(express.static("public"));

app.use("/users", usersRouter);
app.use("/api/trips", tripsRouter);

app.use((req, res) => {
  res.status(404).json({ message: "Not found" });
```

```
});

app.use((err, req, res, next) => {
  const { status = 500, message = "Server error" } = err;
  res.status(status).json({ message });
});

module.exports = app;
```

Код з файлу routes\api\auth.js

```
const express = require("express");
const { validateBody, authenticake } = require("../../middlewares");
const { schemas } = require("../../models/user");
const {
  ctrlRegister,
  ctrlLogin,
  ctrlGetCurrent,
  ctrlLogout,
  ctrlVerifyEmail,
  ctrlResendVerifyEmail,
} = require("../../controllers");

const router = express.Router();

router.post("/register", validateBody(schemas.registerSchema),
ctrlRegister);

router.get("/verify/:verificationToken", ctrlVerifyEmail);

router.post(
  "/verify",
  validateBody(schemas.emailSchema),
  ctrlResendVerifyEmail
);

router.post("/login", validateBody(schemas.loginSchema), ctrlLogin);

router.get("/current", authenticake, ctrlGetCurrent);

router.post("/logout", authenticake, ctrlLogout);

module.exports = router;
```

Код з файлу routes\api\trips.js

```
const express = require("express");
const { validateBody, isValidId, authenticake } =
require("../../middlewares");
const { schemas } = require("../../models/trip");
const {
  ctrlGetAll,
  ctrlGetById,
```

```
  ctrlAddTrip,
  ctrlDeleteTrip,
  ctrlUpdateTrip,
} = require("../../controllers");
const getAllPublic = require("../../controllers/trips/getAllPublic");
const updateOnePublic =
require("../../controllers/trips/updateOnePublic");
const getTripsWithLikes =
require("../../controllers/trips/getTripsWithLikes");
const getKeys = require("../../controllers/trips/getKeys");

const router = express.Router();

router.get("/keys", authenticake, getKeys);

router.get("/allpublic", getAllPublic);

router.patch("/:tripId/likes", isValidId, updateOnePublic);

router.get("/trips-with-likes", getTripsWithLikes);

router.get("/", authenticake, ctrlGetAll);

router.get("/:tripId", authenticake, isValidId, ctrlGetById);

router.post("/", authenticake, validateBody(schemas.addSchema),
ctrlAddTrip);

router.delete("/:tripId", authenticake, isValidId, ctrlDeleteTrip);

router.patch(
  "/:tripId",
  authenticake,
  isValidId,
  validateBody(schemas.addSchema),
  ctrlUpdateTrip
);

module.exports = router;
```

Код з файлу \models\trip.js

```
const { Schema, model } = require("mongoose");
const Joi = require("joi");

const { handleMongooseError } = require("../helpers");

const tripSchema = new Schema(
  {
    owner: {
      type: Schema.Types.ObjectId,
      ref: "user",
    },
```

```
    title: {
      type: String,
      required: [true, "Set title for trip"],
    },
    description: {
      type: String,
    },
    categories: [
      {
        nameCategory: {
          type: String,
          required: true,
        },
        todoList: [
          {
            todo: {
              type: String,
              required: true,
            },
          },
        ],
        publicList: {
          type: Boolean,
          default: false,
        },
      },
    ],
    isPublic: {
      type: Boolean,
      default: false,
    },
    photos: [
      {
        cdnUrl: {
          type: String,
          required: true,
        },
        uuid: {
          type: String,
          required: true,
        },
      },
    ],
    likes: {
      type: Number,
      default: 0,
    },
  },
  { versionKey: false, timestamps: true }
);

tripSchema.post("save", handleMongooseError);
```

```
const addSchema = Joi.object({
  title: Joi.string().required().messages({
    "string.base": "title must be a string",
    "string.empty": "title cannot be empty",
    "any.required": "title is a required field",
  }),
  description: Joi.string(),
  categories: Joi.array().items(
    Joi.object({
      nameCategory: Joi.string().required(),
      todoList: Joi.array().items(
        Joi.object({
          todo: Joi.string().required(),
        })
      ),
      publicList: Joi.boolean(),
    })
  ),
  isPublic: Joi.boolean(),
  photos: Joi.array().items(
    Joi.object({
      cdnUrl: Joi.string().uri().required(),
      uuid: Joi.string()
        .guid({ version: ["uuidv4"] })
        .required(),
    })
  ),
});

const schemas = {
  addSchema,
};

const Trip = model("trip", tripSchema);

module.exports = { Trip, schemas };
```

Код з файлу \models\user.js

```
const { Schema, model } = require("mongoose");
const Joi = require("joi");

const { handleMongooseError } = require("../helpers");

const tripSchema = new Schema(
  {
    owner: {
      type: Schema.Types.ObjectId,
      ref: "user",
    },
    title: {
      type: String,
```

```
      required: [true, "Set title for trip"],
    },
    description: {
      type: String,
    },
    categories: [
      {
        nameCategory: {
          type: String,
          required: true,
        },
        todoList: [
          {
            todo: {
              type: String,
              required: true,
            },
          },
        ],
        publicList: {
          type: Boolean,
          default: false,
        },
      },
    ],
    isPublic: {
      type: Boolean,
      default: false,
    },
    photos: [
      {
        cdnUrl: {
          type: String,
          required: true,
        },
        uuid: {
          type: String,
          required: true,
        },
      },
    ],
    likes: {
      type: Number,
      default: 0,
    },
  },
  { versionKey: false, timestamps: true }
);

tripSchema.post("save", handleMongooseError);

const addSchema = Joi.object({
  title: Joi.string().required().messages({
```

```
    "string.base": "title must be a string",
    "string.empty": "title cannot be empty",
    "any.required": "title is a required field",
  }),
  description: Joi.string(),
  categories: Joi.array().items(
    Joi.object({
      nameCategory: Joi.string().required(),
      todoList: Joi.array().items(
        Joi.object({
          todo: Joi.string().required(),
        })
      ),
      publicList: Joi.boolean(),
    })
  ),
  isPublic: Joi.boolean(),
  photos: Joi.array().items(
    Joi.object({
      cdnUrl: Joi.string().uri().required(),
      uuid: Joi.string()
        .guid({ version: ["uuidv4"] })
        .required(),
    })
  ),
});

const schemas = {
  addSchema,
};

const Trip = model("trip", tripSchema);

module.exports = { Trip, schemas };
```

Код з файлу middlewares\validateBody.js

```
const{ HttpError } = require("../helpers");

const validateBody = (schema) => {
  const func = (req, res, next) => {
    if (req.method === "PUT" && Object.keys(req.body).length === 0) {
      return next(HttpError(400, "missing fields"));
    }

    if (req.method === "PATCH" && Object.keys(req.body).length === 0)
{
      return next(HttpError(400, "missing field"));
    }

    const { error } = schema.validate(req.body);
    if (error) {
```

```
      const missingField = error.details[0].context.label;
      const errorMessage = `missing required ${missingField} field`;
      return next(HttpError(400, errorMessage));
    }
    next();
  };

  return func;
};

module.exports = validateBody;
```

Код з файлу middlewares\ isValidId.js

```
const { isValidObjectId } = require("mongoose");

const { HttpError } = require("../helpers");

const isValidId = (req, res, next) => {
  const { tripId } = req.params;
  if (!isValidObjectId(tripId)) {
    next(HttpError(400, `${tripId} is not valid id`));
  }
  next();
};

module.exports = isValidId;
```

Код з файлу middlewares\ authenticake.js

```
const { User } = require("../models/user");
const jwt = require("jsonwebtoken");
const { HttpError } = require("../helpers");

const { SECRET_KEY } = process.env;

const authenticake = async (req, res, next) => {
  const { authorization = "" } = req.headers;
  const [bearer, token] = authorization.split(" ");
  if (bearer !== "Bearer") {
    next(HttpError(401, "Not authorized"));
  }

  try {
    const { id } = jwt.verify(token, SECRET_KEY);
    const user = await User.findOne({ _id: id });

    if (!user || !user.token || user.token !== token) {
      next(HttpError(401, "Not authorized"));
    }
```

```
    req.user = user;
    next();
  } catch {
    next(HttpError(401, "Not authorized"));
  }
};

module.exports = authenticake;
```

Код з файлу controllers\trips\addTrip.js

```
const { Trip } = require("../../models/trip");

const addTrip = async (req, res, next) => {
  const { _id: owner } = req.user;
  const addTrip = await Trip.create({ ...req.body, owner });
  res.status(201).json(addTrip);
};

module.exports = addTrip;
```

Код з файлу controllers\trips\ deleteTrip.js

```
const { Trip } = require("../../models/trip");
const { HttpError } = require("../../helpers");

const deleteTrip = async (req, res, next) => {
  const { tripId } = req.params;
  const tripForRemove = await Trip.findByIdAndRemove(tripId);
  if (!tripForRemove) {
    throw HttpError(404, "Not found");
  }
  res.json({ message: "Trip deleted" });
};

module.exports = deleteTrip;
```

Код з файлу controllers\trips\ getAll.js

```
const { Trip } = require("../../models/trip");

const getAll = async (req, res, next) => {
  const { _id: owner } = req.user;
  const { page = 1, limit = 20 } = req.query;
  const skip = (page - 1) * limit;
  const allTrips = await Trip.find({ owner }, "-createdAt -updatedAt",
{
    skip,
    limit,
  });
```

```
  res.json(allTrips);
};

module.exports = getAll;
```

Код з файлу controllers\trips\ getAllPublic.js

```
const { Trip } = require("../../models/trip");

const getAllPublic = async (req, res, next) => {
  const { page = 1, limit = 20 } = req.query;
  const skip = (page - 1) * limit;

  try {
    const countPromise = Trip.countDocuments({ public: true });
    const tripsPromise = Trip.find({ isPublic: true }, "-createdAt -
updatedAt")
      .skip(skip)
      .limit(limit);

    const [count, trips] = await Promise.all([countPromise,
tripsPromise]);

    res.json({
      totalTrips: count,
      trips,
    });
  } catch (error) {
    next(error);
  }
};

module.exports = getAllPublic;
```

Код з файлу controllers\trips\ getById.js

```
const { Trip } = require("../../models/trip");
const { HttpError } = require("../../helpers");

const getById = async (req, res, next) => {
  const { tripId } = req.params;
  const tripById = await Trip.findById(tripId);
  if (!tripById) {
    throw HttpError(404, "Not found");
  }
  res.json(tripById);
};

module.exports = getById;
```

Код з файлу controllers\trips\ getTripsWithLikes.js

```
const { Trip } = require("../../models/trip");

const getTripsWithLikes = async (req, res, next) => {
  try {
    const trips = await Trip.find({ likes: { $gt: 0 } }).sort({ likes:
-1 });
    res.status(200).json({ success: true, trips });
  } catch (err) {
    console.error("Error fetching trips with likes:", err);
    res
      .status(500)
      .json({ success: false, message: "Failed to fetch trips with
likes" });
  }
};

module.exports = getTripsWithLikes;
```

Код з файлу controllers\trips\ updateOnePublic.js

```
const { Trip } = require("../../models/trip");

const updateOnePublic = async (req, res, next) => {
  const { tripId } = req.params;

  try {
    await Trip.findByIdAndUpdate({ _id: tripId }, { $inc: { likes: 1 }
});
    res
      .status(200)
      .json({ success: true, message: "Likes updated successfully" });
  } catch (err) {
    res.status(500).json({ success: false, message: "Failed to update
likes" });
  }
};

module.exports = updateOnePublic;
```

Код з файлу controllers\trips\ updateTrip.js

```
const { Trip } = require("../../models/trip");
const { HttpError } = require("../../helpers");

const updateTrip = async (req, res, next) => {
  const { tripId } = req.params;
  const updateTrip = await Trip.findByIdAndUpdate(tripId, req.body, {
    new: true,
  });
  if (!updateTrip) {
```

```
    throw HttpError(404, "Not found");
  }
  res.json(updateTrip);
};

module.exports = updateTrip;
```

Код з файлу controllers\auth \ getCurrent.js

```
const getCurrent = async (req, res, next) => {
  const { email } = req.user;

  res.status(200).json({ email });
};

module.exports = getCurrent;
```

Код з файлу controllers\auth \ login.js

```
const { User } = require("../../models/user");
const { HttpError } = require("../../helpers");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

const { SECRET_KEY } = process.env;

const login = async (req, res, next) => {
  const { email, password } = req.body;

  const user = await User.findOne({ email });

  if (!user) {
    throw HttpError(401, "Email or password is wrong");
  }

  if (!user.verify) {
    throw HttpError(401, "Email is not verified");
  }

  const passwordCompare = await bcrypt.compare(password,
user.password);

  if (!passwordCompare) {
    throw HttpError(401, "Email or password is wrong");
  }

  const payload = { id: user._id };

  const token = jwt.sign(payload, SECRET_KEY, { expiresIn: "23h" });
  await User.findByIdAndUpdate(user._id, { token });
```

```
  const response = {
    token: token,
    user: {
      email: user.email,
      subscription: user.subscription,
    },
  };

  res.status(200).json(response);
};

module.exports = login;
```

Код з файлу controllers\auth \ logout.js

```
const { User } = require("../../models/user");

const logout = async (req, res, next) => {
  const { _id } = req.user;
  await User.findByIdAndUpdate(_id, { token: " " });

  res.status(204).json({ message: "No Content" });
};

module.exports = logout;
```

Код з файлу controllers\auth \ register.js

```
const { User } = require("../../models/user");
const { HttpError, sendEmail } = require("../../helpers");
const bcrypt = require("bcryptjs");
const gravatar = require("gravatar");
const crypto = require("node:crypto");

const { BASE_URL } = process.env;

const register = async (req, res, next) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });

  if (user) {
    throw HttpError(409, "Email in use");
  }

  const hashPassword = await bcrypt.hash(password, 10);
  const avatarURL = gravatar.url(email);
  const verificationToken = crypto.randomUUID();

  const verifyEmail = {
    to: email,
    subject: "Verify email",
    html: `
```

```
    <div style="font-family: Arial, sans-serif; padding: 20px;">
      <p>Hello!</p>
      <p>Please click the button below to verify your email:</p>
      <a style="background-color: #8dd3bb; color: #fff; text-
decoration: none; padding: 10px 20px; border-radius: 5px; display:
inline-block;"
         href="${BASE_URL}/users/verify/${verificationToken}"
target="_blank">Verify Email</a>
      <p>If the button above doesn't work, you can also copy and paste
the following link into your browser:</p>
      <p><a href="${BASE_URL}/users/verify/${verificationToken}"
target="_blank">${BASE_URL}/users/verify/${verificationToken}</a></p>
    </div>
  `,
  };

  await sendEmail(verifyEmail);

  const newUser = await User.create({
    ...req.body,
    password: hashPassword,
    avatarURL,
    verificationToken,
  });

  const response = {
    user: {
      email: newUser.email,
      subscription: newUser.subscription,
    },
  };

  res.status(201).json(response);
};

module.exports = register;
```

Код з файлу controllers\auth \ resendVerifyEmail.js

```
const { User } = require("../../models/user");
const { HttpError, sendEmail } = require("../../helpers");

const { BASE_URL } = process.env;

const resendVerifyEmail = async (req, res, next) => {
  const { email } = req.body;
  const user = await User.findOne({ email });
  if (!user) {
    throw HttpError(404, "User not found");
  }
  if (user.verify) {
    throw HttpError(400, "Verification has already been passed");
```

```
  }

  const verifyEmail = {
    to: email,
    subject: "Resend Verification Email",
    html: `
    <div style="font-family: Arial, sans-serif; padding: 20px;">
      <p>Hello!</p>
      <p>We noticed that you haven't verified your email yet. Please
click the button below to resend the verification email:</p>
      <a style="background-color: #8dd3bb; color: #fff; text-
decoration: none; padding: 10px 20px; border-radius: 5px; display:
inline-block;"
         href="${BASE_URL}/users/resend-
verification/${user.verificationToken}" target="_blank">Resend
Verification Email</a>
      <p>If the button above doesn't work, you can also copy and paste
the following link into your browser:</p>
      <p><a href="${BASE_URL}/users/resend-
verification/${user.verificationToken}"
target="_blank">${BASE_URL}/users/resend-
verification/${user.verificationToken}</a></p>
    </div>
  `,
  };

  await sendEmail(verifyEmail);

  res.status(200).json({
    message: "Verification email sent",
  });
};

module.exports = resendVerifyEmail;
```

Код з файлу controllers\auth \ verification-successful.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Email Verification Successful</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
        background-color: #f4f4f4;
      }
```

```
      .container {
        max-width: 600px;
        margin: 50px auto;
        padding: 20px;
        background-color: #fff;
        border-radius: 5px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      }

      .success-message {
        text-align: center;
      }

      .success-message h1 {
        color: #4caf50;
      }

      .success-message p {
        font-size: 18px;
        color: #333;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="success-message">
        <h1>Підтвердження успішне!</h1>
        <p>Вашу електронну адресу успішно підтверджено.</p>
      </div>
    </div>
  </body>
</html>
```

Код з файлу controllers\auth \ verifyEmail.js

```
const { User } = require("../../models/user");
const { HttpError } = require("../../helpers");
const path = require("path");
const fs = require("fs");

const verifyEmail = async (req, res, next) => {
  const { verificationToken } = req.params;
  console.log(verificationToken);
  const user = await User.findOne({ verificationToken }).exec();
  console.log(user);

  if (!user) {
    throw HttpError(404, "User not found");
  }
  await User.findByIdAndUpdate(user._id, {
    verify: true,
    verificationToken: null,
```

```
  });

  const htmlFilePath = path.join(__dirname, "verification-
successful.html");
  const htmlContent = fs.readFileSync(htmlFilePath, "utf8");

  res.status(200).send(htmlContent);
};

module.exports = verifyEmail;
```

Код з файлу helpers\sendEmail.js

```
const ctrlWrapper = (ctrl) => {
  const func = async (req, res, next) => {
    try {
      await ctrl(req, res, next);
    } catch (error) {
      next(error);
    }
  };
  return func;
};

module.exports = ctrlWrapper;
```

Код з файлу helpers\ handleMongooseError.js

```
const handleMongooseError = (error, data, next) => {
  const { name, code } = error;
  const status = name === "MongoServerError" && code === 11000 ? 409 :
400;

  error.status = status;
  next();
};

module.exports = handleMongooseError;
```

Код з файлу helpers\ HttpError.js

```
const HttpError = (status, message) => {
  const error = new Error(message);
  error.status = status;
  return error;
};

module.exports = HttpError;
```

Код з файлу helpers\ sendEmail.js

```
const nodemailer = require("nodemailer");

const transport = nodemailer.createTransport({
  host: "sandbox.smtp.mailtrap.io",
  port: 2525,
  auth: {
    user: process.env.MAILTRAP_USER,
    pass: process.env.MAILTRAP_PASSWORD,
  },
});

function sendEmail(message) {
  return transport.sendMail(message);
}

module.exports = sendEmail;
```

ДОДАТОК Д

Клієнтська частина вебдодатку

Код з файлу index.js

```
import React from 'react';
import { BrowserRouter } from 'react-router-dom';
import ReactDOM from 'react-dom/client';
import { App } from 'components/App';
import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter basename="/frontend-diploma">
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Код з файлу app.js

```
import Layout from './Layout';

const Home = lazy(() => import('../pages/Home'));
const PublicTrips = lazy(() => import('../pages/PublicTrips'));
const PopularTrips = lazy(() => import('../pages/PopularTrips'));
const User = lazy(() => import('../pages/User'));
const Login = lazy(() => import('../pages/Login'));
const Registration = lazy(() => import('../pages/Registration'));

export const App = () => {
  return (
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="publictrips" element={<PublicTrips />} />
        <Route path="populartrips" element={<PopularTrips />} />
        <Route path="user" element={<User />} />
        <Route path="login" element={<Login />} />
        <Route path="registration" element={<Registration />} />
        <Route path="*" element={<Home />} />
      </Route>
    </Routes>
  );
};
```

Код з файлу \pages\Home\Home.jsx

```
import Hero from '../../components/HomeSection/Hero';

const Home = () => {
  return (
    <>
      <Hero></Hero>
    </>
  );
};

export default Home;
```

Код з файлу \pages\ Login \ Login.jsx

```
import LoginForm from
'components/Auth/LoginSections/LoginForm/LoginForm';
import SwipeableTextMobileStepper from
'../../components/Auth/Swiper/Swiper';

import { ContainerForm } from './Login.style';

const Login = () => {
  return (
      <ContainerForm>
        <SwipeableTextMobileStepper></SwipeableTextMobileStepper>
        <LoginForm></LoginForm>
      </ContainerForm>
  );
};

export default Login;
```

Код з файлу \pages\ PopularTrips \ PopularTrips.jsx

```
import AllPopularTrips from
'../../components/PopularTripsSections/AllPopularTrips';
import {Container} from "./PopularTrips.style"

const PopularTrips = () => {
  return (
    <Container>
      <AllPopularTrips></AllPopularTrips>
    </Container>
  );
};
```

```
export default PopularTrips;
```

Код з файлу \pages\ PublicTrips \ PublicTrips.jsx

```
import AllPublicTrips from
'../../components/PublicTripsSections/AllPublicTrips';

import { Container } from './PublicTrips.style';

const PublicTrips = () => {
  return (
    <Container>
      <AllPublicTrips></AllPublicTrips>
    </Container>
  );
};

export default PublicTrips;
```

Код з файлу \pages\ Registration \ Registration.jsx

```
import RegistrationForm from
'components/Auth/RegistrationSections/RegistrationForm/RegistrationFor
m';
import SwipeableTextMobileStepper from
'../../components/Auth/Swiper/Swiper';
import { ContainerForm } from './Registration.style';

const Registration = () => {
  return (
    <ContainerForm>
      <RegistrationForm></RegistrationForm>
      <SwipeableTextMobileStepper></SwipeableTextMobileStepper>
    </ContainerForm>
  );
};

export default Registration;
```

Код з файлу \pages\ User \ User.jsx

```
import React from 'react';
import Box from '@mui/material/Box';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import PropTypes from 'prop-types';
import AddTripForm from
'../../components/UserSections/AddTripForm/AddTripForm';
```

```
import AllUserTrips from
'../../components/UserSections/AllUserTrips/AllUserTrips';
import UpdateTrip from
'../../components/UserSections/UpdateTrip/UpdateTrip';

import { Container } from './User.style';

function CustomTabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && <Box sx={{ p: 3 }}>{children}</Box>}
    </div>
  );
}

CustomTabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.number.isRequired,
  value: PropTypes.number.isRequired,
};

function a11yProps(index) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  };
}

const User = () => {
  const [value, setValue] = React.useState(0);

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };

  return (
    <Container>
      <Box sx={{ width: '100%' }}>
        <Box sx={{ borderBottom: 1, borderColor: 'divider' }}>
          <Tabs
            variant="scrollable"
            scrollButtons="auto"
            aria-label="scrollable auto tabs example"
            value={value}
            onChange={handleChange}
```

```
          >
            <Tab label="Додавання мандрівки" {...a11yProps(0)} />
            <Tab label="Усі твої мандрівки" {...a11yProps(1)} />
            <Tab label="Редагування мандрівок" {...a11yProps(2)} />
          </Tabs>
        </Box>
        <CustomTabPanel value={value} index={0}>
          <AddTripForm />
        </CustomTabPanel>
        <CustomTabPanel value={value} index={1}>
          <AllUserTrips />
        </CustomTabPanel>
        <CustomTabPanel value={value} index={2}>
          <UpdateTrip />
        </CustomTabPanel>
      </Box>
    </Container>
  );
};

export default User;
```

Код з файлу \ components\HomeSection\Hero.jsx

```
import { HeroSection, Container, TitleOne, TitleTwo } from
'./Hero.style';

const Hero = () => {
  return (
    <HeroSection>
      <Container>
        <TitleOne>Зберігай або ділися своїми враження</TitleOne>
        <TitleTwo>ЖИВИ & ПОДОРОЖУЙ</TitleTwo>
      </Container>
    </HeroSection>
  );
};

export default Hero;
```

Код з файлу \ components\HomeSection\Hero.style.jsx

```
import styled from 'styled-components';
import HeroImg from '../img/Hero-img.png';

export const HeroSection = styled.section`
  color: rgba(255, 255, 255, 1);
  height: 100vh;
  background-image: linear-gradient(
      0deg,
      rgba(0, 0, 0, 0) 0%,
```

```
      rgba(0, 0, 0, 0.6) 100%
    ),
    url(${HeroImg});
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  margin-bottom: 40px;
  @media (max-width: 425px) {
    height: auto;
    min-height: 100vh;
  }
`;

export const Container = styled.div`
  font-weight: 700;
  padding: 120px 20px;
  max-width: 754px;
  margin: 0 auto;
  text-align: center;
  @media (max-width: 425px) {
    padding: 80px 20px;
  }
`;

export const TitleOne = styled.h2`
  font-family: 'RalewayRegular';
  font-size: 45px;
  line-height: 57px;
  margin-bottom: 4px;
  @media (max-width: 425px) {
    font-size: 24px;
    line-height: 32px;
  }
`;

export const TitleTwo = styled.h1`
  font-family: 'RalewayRegular';
  font-size: 40px;
  line-height: 101px;
  margin-bottom: 16px;
  @media (max-width: 425px) {
    font-size: 20px;
    line-height: 40px;
  }
`;
```

Код з файлу \ components\ Layout \ Layout.jsx

```
import { NavLink, Outlet, useNavigate, useLocation } from 'react-
router-dom';
import { useEffect, useState, Suspense } from 'react';
import axios from 'axios';
```

```
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import Loader from 'components/Loader/Loader';

import {
  Header,
  Container,
  ContainerDiv,
  Nav,
  Button,
} from './Layout.style';

const Layout = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  useEffect(() => {
    const token = localStorage.getItem('token');
    if (token) {
      setIsAuthenticated(true);
    }
  }, []);

  const handleLogout = () => {
    const token = localStorage.getItem('token');

    axios
      .post('http://localhost:3000/users/logout', null, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then(response => {
        localStorage.removeItem('token');
        setIsAuthenticated(false);
        toast.success('Успішний вихід!');
        navigate('/');
      })
      .catch(error => {
        localStorage.removeItem('token');
        toast.error('Щось пішло не так під час виходу! Спробуйте ще
раз.');
        setIsAuthenticated(false);
        navigate('/');
      });
  };

  const navStyle = {
    color: location.pathname === '/' ? 'white' : 'inherit',
  };

  return (
```

```
    <>
      <ToastContainer />
      <Header>
        <Container>
          <Nav>
            <ContainerDiv>
              <NavLink to="/" style={navStyle}>
                Головна
              </NavLink>
              <NavLink to="/publictrips" style={navStyle}>
                Стрічка
              </NavLink>
              <NavLink to="/populartrips" style={navStyle}>
                Популярні мандрівки
              </NavLink>
            </ContainerDiv>
            <ContainerDiv>
              {isAuthenticated && (
                <>
                  <NavLink to="/user" style={navStyle}>
                    Щоденник
                  </NavLink>
                  <Button onClick={handleLogout}>Вихід</Button>
                </>
              )}
              {!isAuthenticated && (
                <>
                  <NavLink to="/login" style={navStyle}>
                    Авторизація
                  </NavLink>
                  <NavLink to="/registration" style={navStyle}>
                    Реєстрація
                  </NavLink>
                </>
              )}
            </ContainerDiv>
          </Nav>
        </Container>
      </Header>
      <main>
        <Suspense fallback={<Loader />}>
          <Outlet />
        </Suspense>
      </main>
    </>
  );
};

export default Layout;
```

Код з файлу \ components\ Layout \ Layout.style.jsx

```
import styled from 'styled-components';

export const Header = styled.header`
  font-size: 14px;
  font-weight: 600;
  line-height: 17px;
  position: absolute;
  width: 100%;
  top: 0;
  left: 0;
  color: black;
  @media (max-width: 426px) {
    font-size: 12px;
  }
`;

export const Container = styled.div`
  padding-top: 24px;
  padding-bottom: 24px;
  max-width: 1262px;
  margin: 0 auto;
  padding-left: 15px;
  padding-right: 15px;
  @media (max-width: 426px) {
    padding-top: 12px;
    padding-bottom: 12px;
    display: flex;
  }
`;

export const ContainerDiv = styled.div`
  display: flex;
  gap: 20px;
  @media (max-width: 426px) {
    gap: 10px;
  }
`;

export const Nav = styled.nav`
  display: flex;
  justify-content: space-between;
  @media (max-width: 426px) {
    flex-direction: column;
    align-items: center;
  }
`;

export const Button = styled.button`
  color: rgb(141, 211, 187);
  @media (max-width: 426px) {
```

```
    padding: 6px 12px;
  }
`;

export const Footer = styled.footer`
  background-color: rgb(141, 211, 187);
  color: rgb(255, 255, 255);
  padding: 10px;
  text-align: center;
  margin-top: 20px;
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
`;
```

Код з файлу \ components\ PublicTripsSections\ AllPublicTrips.jsx

```
/* eslint-disable jsx-a11y/img-redundant-alt */

import { useState, useEffect } from 'react';
import axios from 'axios';
import { Card, Col, Input, Select } from 'antd';
import Modal from './Modal/Modal';
import { toast, ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

import {
  SearchDiv,
  Cards,
  TripDescription,
  Category,
  CategoryName,
  TodoList,
  TodoItem,
  PhotoList,
  Photo,
} from './AllPublicTrips.style';

const { Option } = Select;

const AllPublicTrips = () => {
  const [trips, setTrips] = useState([]);
  const [searchQuery, setSearchQuery] = useState('');
  const [selectedCategory, setSelectedCategory] = useState('');
  const [open, setOpen] = useState(false);
  const [selectedPhoto, setSelectedPhoto] = useState(null);

  useEffect(() => {
    const fetchTrips = async () => {
      try {
        const response = await axios.get(
```

```
          'http://localhost:3000/api/trips/allpublic'
        );
        setTrips(response.data.trips);
      } catch (error) {
        console.log(error);
      }
    };

    fetchTrips();
  }, []);

  const handleLike = async tripId => {
    try {
      await
axios.patch(`http://localhost:3000/api/trips/${tripId}/likes`);
    } catch (error) {
      console.error('Error liking trip:', error);
    }
  };

  const handleSearch = event => {
    setSearchQuery(event.target.value);
  };

  const handleCategoryChange = value => {
    setSelectedCategory(value);
  };

  const handlePhotoClick = photo => {
    setSelectedPhoto(photo);
    setOpen(true);
  };

  const filteredTrips = trips.filter(trip => {
    const matchesSearchQuery = trip.title
      .toLowerCase()
      .includes(searchQuery.toLowerCase());
    const matchesCategory = selectedCategory
      ? trip.categories.some(
          category => category.nameCategory === selectedCategory
        )
      : true;
    return matchesSearchQuery && matchesCategory;
  });

  useEffect(() => {
    if (searchQuery || selectedCategory) {
      if (filteredTrips.length === 0) {
        toast.info('Нажаль не знайдено жодної мандрівки');
      }
    }
  }, [filteredTrips.length, searchQuery, selectedCategory]);
```

```
  const uniqueCategories = [
    ...new Set(
      trips.flatMap(trip =>
        trip.categories.map(category => category.nameCategory)
      )
    ),
  ];

  const getRandomColor = () => {
    const letters = '0123456789ABCDEF';
    let color = '#';
    for (let i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
  };

  return (
    <div>
      <ToastContainer />
      <SearchDiv>
        <Input
          placeholder="Пошук мандрівки"
          value={searchQuery}
          onChange={handleSearch}
          style={{ marginRight: '20px', width: '390px', height:
'30px'}}
        />
        <Select
          placeholder="Фільтр за категоріями"
          style={{ width: '200px', marginBottom: '20px' }}
          onChange={handleCategoryChange}
          allowClear
        >
          {uniqueCategories.map(category => (
            <Option key={category} value={category}>
              {category}
            </Option>
          ))}
        </Select>
      </SearchDiv>
      <Cards>
        {/* <Row gutter={[16, 16]} justify="center"> */}
        {filteredTrips.map(trip => (
          <Col span={8} key={trip._id}>
            <Card
              style={{ marginBottom: '15px', width: '390px' }}
              title={trip.title}
              bordered={false}
              headStyle={{
                backgroundColor: getRandomColor(),
                color: '#fafbfc',
              }}
```

```
              actions={[
                <button key="like" onClick={() =>
handleLike(trip._id)}>
                  Вподобайка
                </button>,
              ]}
            >
              <TripDescription>{trip.description}</TripDescription>
              <Category>Категорії:</Category>
              {trip.categories
                .filter(category => category.publicList)
                .map(category => (
                  <div key={category._id}>

<CategoryName>{category.nameCategory}</CategoryName>
                    <TodoList>
                      {category.todoList.map(todo => (
                        <TodoItem
key={todo._id}>{todo.todo}</TodoItem>
                      ))}
                    </TodoList>
                  </div>
                ))}
              <div>
                <h3>Фотографії:</h3>
                <PhotoList>
                  {trip.photos.map((photo, index) => (
                    <Photo key={index}>
                      <img
                        src={photo.cdnUrl}
                        alt={`Photo ${index + 1}`}
                        onClick={() => handlePhotoClick(photo.cdnUrl)}
                        style={{
                          cursor: 'pointer',
                          boxShadow: '7px 11px 10px 1px rgba(223, 223,
223, 1)',
                        }}
                      />
                    </Photo>
                  ))}
                </PhotoList>
              </div>
            </Card>
          </Col>
        ))}
        {/* </Row> */}
      </Cards>
      {selectedPhoto && (
        <Modal
          open={open}
          onClose={() => setOpen(false)}
          photo={selectedPhoto}
        />
```

```
      )}
    </div>
  );
};

export default AllPublicTrips;
```

Код з файлу \ components\ PublicTripsSections\ AllPublicTrips.style.jsx

```
import styled from 'styled-components';

export const SearchDiv = styled.div`
  display: flex;
  flex-direction: row;
  row-gap: 16px;

  @media (max-width: 426px) {
    flex-direction: column;
  }
`;

export const Cards = styled.ul`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  row-gap: 16px;

  @media (max-width: 426px) {
    flex-direction: column;
    flex-wrap: wrap;
  }
`;

export const TripDescription = styled.p`
  color: #666;
  font-size: 14px;
  margin-bottom: 12px;

  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const Category = styled.h3`
  margin-bottom: 4px;

  @media (max-width: 425px) {
    font-size: 16px;
  }
`;
```

```
export const CategoryName = styled.h4`
  color: #555;
  margin-bottom: 2px;

  @media (max-width: 425px) {
    font-size: 14px;
  }
`;

export const TodoList = styled.ul`
  list-style-type: disc;
  margin-bottom: 4px;
  margin-left: 15px;

  @media (max-width: 425px) {
    font-size: 12px;
    margin-left: 10px;
  }
`;

export const TodoItem = styled.li`
  color: #777;

  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const PhotoList = styled.ul`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 4px;

  @media (max-width: 425px) {
    gap: 2px;
  }
`;

export const Photo = styled.li`
  width: 150px;
`;
```

Код з файлу \ components\ PublicTripsSections\ Modal\ Modal.jsx

```
import * as React from 'react';
import PropTypes from 'prop-types';
import clsx from 'clsx';
import { styled, css } from '@mui/system';
import { Modal as BaseModal } from '@mui/base/Modal';
```

```
export default function ModalUnstyled({ open, onClose, photo }) {
  return (
    <Modal
      aria-labelledby="unstyled-modal-title"
      aria-describedby="unstyled-modal-description"
      open={open}
      onClose={onClose}
      slots={{ backdrop: StyledBackdrop }}
    >
      <ModalContent sx={{ width: 400 }}>
        <img src={photo} alt="Selected" style={{ width: '100%' }} />
      </ModalContent>
    </Modal>
  );
}

ModalUnstyled.propTypes = {
  open: PropTypes.bool.isRequired,
  onClose: PropTypes.func.isRequired,
  photo: PropTypes.string.isRequired,
};

const Backdrop = React.forwardRef((props, ref) => {
  const { open, className, ...other } = props;
  return (
    <div
      className={clsx({ 'base-Backdrop-open': open }, className)}
      ref={ref}
      {...other}
    />
  );
});

Backdrop.propTypes = {
  className: PropTypes.string.isRequired,
  open: PropTypes.bool,
};

const grey = {
  50: '#F3F6F9',
  100: '#E5EAF2',
  200: '#DAE2ED',
  300: '#C7D0DD',
  400: '#B0B8C4',
  500: '#9DA8B7',
  600: '#6B7A90',
  700: '#434D5B',
  800: '#303740',
  900: '#1C2025',
};

const Modal = styled(BaseModal)`
```

```
  position: fixed;
  z-index: 1300;
  inset: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  border: none;
`;

const StyledBackdrop = styled(Backdrop)`
  z-index: -1;
  position: fixed;
  inset: 0;
  background-color: rgb(0 0 0 / 0.5);
  -webkit-tap-highlight-color: transparent;
`;

const ModalContent = styled('div')(
  ({ theme }) => css`
    position: relative;
    overflow: hidden;
    background-color: ${theme.palette.mode === 'dark' ? grey[900] :
'#fff'};
    border: none;
    padding: 24px;

    @media (max-width: 425px) {
      width: 90%;
      padding: 16px;
    }
  `
);
```

Код з файлу \ components\ PopularTripsSections\ AllPopularTrips.jsx

```
/* eslint-disable jsx-a11y/img-redundant-alt */
import { useState, useEffect } from 'react';
import axios from 'axios';
import { Card, Col} from 'antd';
import Modal from '../PublicTripsSections/Modal/Modal';

import {
  Cards,
  TripDescription,
  Category,
  CategoryName,
  TodoList,
  TodoItem,
  PhotoList,
  Photo,
} from './AllPopularTrips.style';

const AllPopularTrips = () => {
```

```
const [popularTrips, setPopularTrips] = useState([]);
const [open, setOpen] = useState(false);
const [selectedPhoto, setSelectedPhoto] = useState(null);

useEffect(() => {
  const fetchPopularTrips = async () => {
    try {
      const response = await axios.get(
        'http://localhost:3000/api/trips/trips-with-likes'
      );
      setPopularTrips(response.data.trips);
    } catch (error) {
      console.error('Error fetching popular trips:', error);
    }
  };

  fetchPopularTrips();
}, []);

const getRandomColor = () => {
  const letters = '0123456789ABCDEF';
  let color = '#';
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
};

const handlePhotoClick = photo => {
  setSelectedPhoto(photo);
  setOpen(true);
};

return (
  <>
    <Cards>
      {popularTrips.map(trip => (
        <Col span={8} key={trip._id}>
          <Card
            style={{ marginBottom: '15px', width: '390px' }}
            title={trip.title}
            bordered={false}
            headStyle={{
              backgroundColor: getRandomColor(),
              color: '#fafbfc',
            }}
          >
            <TripDescription>{trip.description}</TripDescription>
            <div>
              <Category>Категорії:</Category>
              <ul>
                {trip.categories.map(category => (
                  <li key={category._id}>
```

```
<CategoryName>{category.nameCategory}</CategoryName>
                      <TodoList>
                        {category.todoList.map(todo => (
                          <TodoItem
key={todo._id}>{todo.todo}</TodoItem>
                        ))}
                      </TodoList>
                    </li>
                  ))}
                </ul>
              </div>
              <div>
                <h3>Фотографії:</h3>
                <PhotoList>
                  {trip.photos.map(photo => (
                    <Photo key={photo.uuid}>
                      <img
                        src={photo.cdnUrl}
                        alt={`Trip photo`}
                        onClick={() => handlePhotoClick(photo.cdnUrl)}
                        style={{
                          width: '150px',
                          cursor: 'pointer',
                          boxShadow: '7px 11px 10px 1px rgba(223, 223,
223, 1)',
                        }}
                      />
                    </Photo>
                  ))}
                </PhotoList>
              </div>
            </Card>
          </Col>
        ))}
      </Cards>
      {selectedPhoto && (
        <Modal
          open={open}
          onClose={() => setOpen(false)}
          photo={selectedPhoto}
        />
      )}
    </>
  );
};

export default AllPopularTrips;
```

Код з файлу \ components\ PopularTripsSections\ AllPopularTrips.style.jsx

```
import styled from 'styled-components';
```

```
export const Cards = styled.ul`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  row-gap: 16px;

  @media (max-width: 426px) {
    flex-direction: column;
    flex-wrap: wrap;
  }
`;

export const TripDescription = styled.p`
  color: #666;
  font-size: 14px;
  margin-bottom: 12px;

  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const Category = styled.h3`
  margin-bottom: 4px;

  @media (max-width: 425px) {
    font-size: 16px;
  }
`;

export const CategoryName = styled.h4`
  color: #555;
  margin-bottom: 2px;

  @media (max-width: 425px) {
    font-size: 14px;
  }
`;

export const TodoList = styled.ul`
  list-style-type: disc;
  margin-bottom: 4px;
  margin-left: 15px;

  @media (max-width: 425px) {
    font-size: 12px;
    margin-left: 10px;
  }
`;

export const TodoItem = styled.li`
  color: #777;
```

```
  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const PhotoList = styled.ul`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 4px;

  @media (max-width: 425px) {
    gap: 2px;
  }
`;

export const Photo = styled.li`
  width: 150px;
`;
```

Код з файлу \ components\ Auth\LoginSections\LoginForm\LoginForm.jsx

```
import { LockOutlined, UserOutlined } from '@ant-design/icons';
import { Button, Form, Input } from 'antd';
import { useNavigate, Link } from 'react-router-dom';
import axios from 'axios';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import css from './LoginForm.module.css';

const LoginForm = () => {
  const navigate = useNavigate();

  const onFinish = values => {
    axios
      .post('http://localhost:3000/users/login', {
        email: values.username,
        password: values.password,
      })
      .then(response => {
        const { token } = response.data;
        localStorage.setItem('token', token);
        toast.success('Успішний вхід!');
        navigate('/');
        window.location.reload();
      })
      .catch(error => {
        console.error('There was an error!', error);
        if (error.response && error.response.status === 401) {
```

```
        toast.error('Неправильний email або пароль!');
      } else {
        toast.error('Щось пішло не так! Спробуйте ще раз.');
      }
    });
};

return (
  <div className={css.loginForm}>
    <ToastContainer />
    <Form
      name="normal_login"
      initialValues={{
        remember: true,
      }}
      onFinish={onFinish}
    >
      <Form.Item
        name="username"
        rules={[
          {
            type: 'email',
            message: 'Невірно введено E-mail!',
          },
          {
            required: true,
            message: 'Будь ласка, введіть свій E-mail!',
          },
        ]}
        className={css.loginFormItem}
      >
        <Input
          prefix={<UserOutlined className="site-form-item-icon" />}
          placeholder="E-mail"
        />
      </Form.Item>
      <Form.Item
        name="password"
        rules={[
          {
            required: true,
            message: 'Будь ласка, введіть свій Пароль!',
          },
        ]}
        className={css.loginFormItem}
      >
        <Input
          prefix={<LockOutlined className="site-form-item-icon" />}
          type="password"
          placeholder="Пароль"
        />
      </Form.Item>
      <Form.Item>
```

```
          <Button
            type="primary"
            htmlType="submit"
            className={css.loginFormButton}
          >
            Авторизація
          </Button>
          <div className={css.link}>
            Або <Link to="/registration">зареєструйтеся зараз!</Link>
          </div>
        </Form.Item>
      </Form>
    </div>
  );
};

export default LoginForm;
```

Код з файлу \ components\ Auth\LoginSections\LoginForm\ LoginForm.module.css

```
.loginFormButton {
  background-color: rgba(141, 211, 187, 1);
  color: rgba(255, 255, 255, 1);
}

.loginFormButton:hover {
  background-color: rgb(185, 152, 44);
}

.loginForm {
  max-width: 600px;
  width: 100%;
  margin: 0 auto;
  padding: 2rem;
}

@media (min-width: 426px) {
  .loginForm {
    width: 300px;
  }
}

.loginFormItem {
  margin-bottom: 1rem;
}

.loginFormButton {
  width: 100%;
}

.link {
```

```
  display: block;
  text-align: center;
  margin-top: 1rem;
}

.link a {
  text-decoration: underline;
}
```

Код з файлу \ components\ Auth\ RegistrationSections\RegistrationForm\RegistrationForm.jsx

```
import { Button, Form, Input } from 'antd';
import { LockOutlined, UserOutlined } from '@ant-design/icons';
import axios from 'axios';
import { useNavigate, Link } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import css from './RegistrationForm.module.css';

const formItemLayout = {
  labelCol: {
    xs: {
      span: 24,
    },
    sm: {
      span: 8,
    },
  },
  wrapperCol: {
    xs: {
      span: 24,
    },
    sm: {
      span: 16,
    },
  },
};

const tailFormItemLayout = {
  wrapperCol: {
    xs: {
      span: 24,
      offset: 0,
    },
    sm: {
      span: 16,
      offset: 8,
    },
  },
};
```

```
const RegistrationForm = () => {
  const [form] = Form.useForm();
  const navigate = useNavigate();

  const onFinish = values => {
    axios
      .post('http://localhost:3000/users/register', {
        email: values.email,
        password: values.password,
      })
      .then(response => {
        form.resetFields();
        toast.success('Перевір пошту та пройди верифікацію!!');
        navigate('/login');
      })
      .catch(error => {
        if (error.response && error.response.status === 409) {
          toast.error('Email вже використовується!');
        } else {
          toast.error('Щось пішло не так! Спробуйте ще раз.');
        }
        console.error('There was an error!', error);
      });
  };

  return (
    <div className={css.container}>
      <ToastContainer />
      <div className={css.formContainer}>
        <Form
          {...formItemLayout}
          form={form}
          name="register"
          onFinish={onFinish}
          className={css.form}
          initialValues={{
            residence: ['zhejiang', 'hangzhou', 'xihu'],
            prefix: '86',
          }}
          scrollToFirstError
        >
          <Form.Item
            name="email"
            label="E-mail"
            rules={[
              {
                type: 'email',
                message: 'Введено невірний E-mail!',
              },
              {
                required: true,
                message: 'Будь ласка, введіть свій E-mail!',
```

```
            },
          ]}
        >
          <Input
            prefix={<UserOutlined className="site-form-item-icon"
/>}
            placeholder="E-mail"
          />
        </Form.Item>

        <Form.Item
          name="password"
          label="Пароль"
          rules={[
            {
              required: true,
              message: 'Будь ласка, введіть свій пароль!',
            },
          ]}
          hasFeedback
        >
          <Input.Password
            prefix={<LockOutlined className="site-form-item-icon"
/>}
            placeholder="Пароль"
          />
        </Form.Item>

        <Form.Item
          name="confirm"
          label="Підтвердіть пароль"
          dependencies={['password']}
          hasFeedback
          rules={[
            {
              required: true,
              message: 'Будь ласка, підтвердіть свій пароль!',
            },
            ({ getFieldValue }) => ({
              validator(_, value) {
                if (!value || getFieldValue('password') === value) {
                  return Promise.resolve();
                }
                return Promise.reject(
                  new Error('Новий пароль, який ви ввели, не
збігається!')
                );
              },
            }),
          ]}
        >
          <Input.Password
```

```
              prefix={<LockOutlined className="site-form-item-icon"
/>}
              placeholder="Підтвердіть пароль"
            />
          </Form.Item>
          <Form.Item {...tailFormItemLayout}>
            <Button
              type="primary"
              htmlType="submit"
              className={css.registrationFormButton}
            >
              Реєстрація
            </Button>
            <div className={css.link}>
              Якщо вже маєте акаунт - <Link to="/login">
авторизуйтеся!</Link>
            </div>
          </Form.Item>
        </Form>
      </div>
    </div>
  );
};

export default RegistrationForm;
```

Код з файлу \ components\ Auth\ RegistrationSections\RegistrationForm\ RegistrationForm.module.css

```
export const Photo = styled.li`
  width: 150px;
`;
.registrationFormButton {
  background-color: rgba(141, 211, 187, 1);
  color: rgba(255, 255, 255, 1);
}

.container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  padding: 2rem;
}

.formContainer {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  min-width: 650px;
```

```
}

@media (max-width: 426px) {
  .formContainer {
    min-width: 300px;
  }
}

.form {
  width: 100%;
}

.registrationFormButton {
  width: 100%;
}

.link {
  display: block;
  text-align: center;
  margin-top: 1rem;
}

.link a {
  text-decoration: underline;
}
```

Код з файлу \ components\ Auth\Swiper\Swiper.jsx

```
import * as React from 'react';
import { useTheme } from '@mui/material/styles';
import Box from '@mui/material/Box';
import MobileStepper from '@mui/material/MobileStepper';
import Paper from '@mui/material/Paper';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-
material/KeyboardArrowRight';
import SwipeableViews from 'react-swipeable-views';
import { autoPlay } from 'react-swipeable-views-utils';
import styled from 'styled-components';

const AutoPlaySwipeableViews = autoPlay(SwipeableViews);

const images = [
  {
    label: 'Міст Сан-Франциско - Оклендська затока, США',
    imgPath:
      'https://images.unsplash.com/photo-1537944434965-
cf4679d1a598?auto=format&fit=crop&w=400&h=250&q=60',
  },
```

```
  {
    label: 'Пташки',
    imgPath:
      'https://images.unsplash.com/photo-1538032746644-
0212e812a9e7?auto=format&fit=crop&w=400&h=250&q=60',
  },
  {
    label: 'Балі, Індонезія',
    imgPath:
      'https://images.unsplash.com/photo-1537996194471-
e657df975ab4?auto=format&fit=crop&w=400&h=250',
  },
  {
    label: 'Гоч, Сербія',
    imgPath:
      'https://images.unsplash.com/photo-1512341689857-
198e7e2f3ca8?auto=format&fit=crop&w=400&h=250&q=60',
  },
];

function SwipeableTextMobileStepper() {
  const theme = useTheme();
  const [activeStep, setActiveStep] = React.useState(0);
  const maxSteps = images.length;

  const handleNext = () => {
    setActiveStep(prevActiveStep => prevActiveStep + 1);
  };

  const handleBack = () => {
    setActiveStep(prevActiveStep => prevActiveStep - 1);
  };

  const handleStepChange = step => {
    setActiveStep(step);
  };

  return (
    <StyledBox sx={{ maxWidth: 600, flexGrow: 1 }}>
      <Paper
        square
        elevation={0}
        sx={{
          display: 'flex',
          alignItems: 'center',
          height: 50,
          pl: 2,
          bgcolor: 'background.default',
        }}
      >
        <Typography>{images[activeStep].label}</Typography>
      </Paper>
      <AutoPlaySwipeableViews
```

```
        axis={theme.direction === 'rtl' ? 'x-reverse' : 'x'}
        index={activeStep}
        onChangeIndex={handleStepChange}
        enableMouseEvents
      >
        {images.map((step, index) => (
          <StyledImageContainer key={step.label}>
            <StyledImage component="img" src={step.imgPath}
alt={step.label} />
          </StyledImageContainer>
        ))}
      </AutoPlaySwipeableViews>
      <MobileStepper
        steps={maxSteps}
        position="static"
        activeStep={activeStep}
        nextButton={
          <Button
            size="small"
            onClick={handleNext}
            disabled={activeStep === maxSteps - 1}
          >
            Наступне
            {theme.direction === 'rtl' ? (
              <KeyboardArrowLeft />
            ) : (
              <KeyboardArrowRight />
            )}
          </Button>
        }
        backButton={
          <Button size="small" onClick={handleBack}
disabled={activeStep === 0}>
            {theme.direction === 'rtl' ? (
              <KeyboardArrowRight />
            ) : (
              <KeyboardArrowLeft />
            )}
            Попереднє
          </Button>
        }
      />
    </StyledBox>
  );
}

export default SwipeableTextMobileStepper;

const StyledBox = styled(Box)`
  width: 100%;
`;

const StyledImageContainer = styled.div`
```

```
  display: flex;
  justify-content: center;
`;

const StyledImage = styled.img`
  max-width: 100%;
  width: auto;
  height: auto;
  max-height: 455px;
  object-fit: cover;

  @media (max-width: 425px) {
    max-height: 250px;
  }
`;
```

Код з файлу \ components\ Loader\Loader.jsx

```
import { ColorRing } from 'react-loader-spinner';
import { LoaderWrapper } from './Loader.style';

const Loader = () => {
  return (
    <>
      <LoaderWrapper>
        <ColorRing
          visible={true}
          height="120"
          width="120"
          ariaLabel="blocks-loading"
          wrapperStyle={{}}
          wrapperClass="blocks-wrapper"
          colors={['#e15b64', '#f47e60', '#f8b26a', '#abbd81',
'#849b87']}
        />
      </LoaderWrapper>
    </>
  );
};

export default Loader;
```

Код з файлу \ components\ Loader\ Loader.style.jsx

```
import styled from 'styled-components';

export const LoaderWrapper = styled.div`
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: 70px;
`;
```

Код з файлу \ components\ UserSections\AddTripForm\AddTripForm.jsx

```
import { useState, useEffect } from 'react';
import AddPhotos from './AddPhotos/AddPhotos';
import { Input, Checkbox, Button } from 'antd';
import ButtonSend from '@mui/material/Button';
import SendIcon from '@mui/icons-material/Send';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

import {
  Form,
  CategoriesDiv,
  PublicCheckboxDiv,
  TodoDiv,
  ChangeDiv,
  BottomDiv,
} from './AddTripForm.style';

const { TextArea } = Input;

const AddTripForm = () => {
  const initialState = {
    title: '',
    description: '',
    categories: [
      {
        nameCategory: '',
        todoList: [
          {
            todo: '',
          },
        ],
        publicList: false,
      },
    ],
    isPublic: false,
    photos: [],
  };

  const [data, setData] = useState(initialState);
  const [cdnUrls, setCdnUrls] = useState([]);

  useEffect(() => {
    const savedData = JSON.parse(localStorage.getItem('data'));

    if (savedData) {
      setData(savedData);
    }
  }, []);

  useEffect(() => {
```

```
  localStorage.setItem('data', JSON.stringify(data));
}, [data]);

const handleChange = event => {
  const { name, value } = event.target;

  setData(prevData => ({
    ...prevData,
    [name]: value,
  }));
};

const handleCategoryChange = (event, index) => {
  const { name, value } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories[index] = {
      ...newCategories[index],
      [name]: value,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const handleTodoChange = (event, categoryIndex, todoIndex) => {
  const { value } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    const newTodoList = [...newCategories[categoryIndex].todoList];
    newTodoList[todoIndex] = {
      todo: value,
    };
    newCategories[categoryIndex] = {
      ...newCategories[categoryIndex],
      todoList: newTodoList,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const handlePublicChange = event => {
  const { checked } = event.target;
```

```
  setData(prevData => ({
    ...prevData,
    isPublic: checked,
  }));
};

const handleCategoryPublicChange = (event, categoryIndex) => {
  const { checked } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories[categoryIndex] = {
      ...newCategories[categoryIndex],
      publicList: checked,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const addCategory = () => {
  setData(prevData => ({
    ...prevData,
    categories: [
      ...prevData.categories,
      {
        nameCategory: '',
        todoList: [
          {
            todo: '',
          },
        ],
        publicList: false,
      },
    ],
  }));
};

const addTodo = categoryIndex => {
  const updatedCategories = [...data.categories];
  updatedCategories[categoryIndex].todoList.push({
    todo: '',
  });
  setData(prevData => ({
    ...prevData,
    categories: updatedCategories,
  }));
};

const deleteCategory = categoryIndex => {
```

```
  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories.splice(categoryIndex, 1);

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const deleteTodo = (categoryIndex, todoIndex) => {
  const updatedCategories = [...data.categories];
  updatedCategories[categoryIndex].todoList.splice(todoIndex, 1);
  setData(prevData => ({
    ...prevData,
    categories: updatedCategories,
  }));
};

const handleCdnUrlsChange = urls => {
  setCdnUrls(urls);
};

const handleSubmit = event => {
  event.preventDefault();

  const token = localStorage.getItem('token');
  const url = 'http://localhost:3000/api/trips';

  const updatedData = {
    ...data,
    photos: cdnUrls,
  };

  fetch(url, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify(updatedData),
  })
    .then(response => response.json())
    .then(responseData => {
      toast.success('Мандрівка успішно додано!');
      setData(initialState);
      localStorage.removeItem('data');
      setCdnUrls([]);
      // window.location.reload();
    })
    .catch(error => {
      console.error('Error:', error);
```

```
        toast.error('Упс... Щось пішло не так!');
      });
  };

  return (
    <>
      <ToastContainer></ToastContainer>
      <Form onSubmit={handleSubmit}>
        {/* Title */}
        <label htmlFor="title">Назва</label>
        <Input
          type="text"
          id="title"
          name="title"
          value={data.title}
          onChange={handleChange}
        />

        {/* Description */}
        <label htmlFor="description">Опис</label>
        <TextArea
          type="text"
          id="description"
          name="description"
          value={data.description}
          onChange={handleChange}
        />

        {/* Categories */}
        <h4>Категорії</h4>
        {data.categories.map((category, categoryIndex) => (
          <CategoriesDiv key={categoryIndex}>
            <label htmlFor={`category-${categoryIndex}`}>Назва
категорії</label>
            <TodoDiv>
              <Input
                type="text"
                id={`category-${categoryIndex}`}
                name="nameCategory"
                value={category.nameCategory}
                onChange={event => handleCategoryChange(event,
categoryIndex)}
              />

              {/* Category Public */}
              <PublicCheckboxDiv>
                <label htmlFor={`category-public-${categoryIndex}`}>
                  Публічна категорія?
                </label>
                <Checkbox
                  type="checkbox"
                  id={`category-public-${categoryIndex}`}
                  name="publicList"
```

```
                checked={category.publicList}
                onChange={event =>
                  handleCategoryPublicChange(event, categoryIndex)
                }
              />
            </PublicCheckboxDiv>
          </TodoDiv>

          {/* Todo List */}
          <label>Нотатки</label>
          {category.todoList.map((todo, todoIndex) => (
            <div key={todoIndex}>
              {/* <label htmlFor={`todo-${categoryIndex}-
${todoIndex}`}>
                Todo
              </label> */}
              <TodoDiv>
                <Input
                  type="text"
                  id={`todo-${categoryIndex}-${todoIndex}`}
                  name="todo"
                  value={todo.todo}
                  onChange={event =>
                    handleTodoChange(event, categoryIndex,
todoIndex)
                  }
                />

                {/* Delete Todo */}
                <Button
                  type="primary"
                  danger
                  style={{ backgroundColor: '#ED5E68' }}
                  onClick={() => deleteTodo(categoryIndex,
todoIndex)}
                >
                  Видалити нотатку
                </Button>
              </TodoDiv>
            </div>
          ))}
          <ChangeDiv>
            {/* Add Todo */}
            <Button
              type="primary"
              style={{ backgroundColor: '#8DD3BB' }}
              onClick={() => addTodo(categoryIndex)}
            >
              Додати нотатку
            </Button>

            {/* Delete Category */}
            <Button
```

```
                type="primary"
                style={{ backgroundColor: '#ED5E68' }}
                danger
                onClick={() => deleteCategory(categoryIndex)}
              >
                Видалити категорію
              </Button>
            </ChangeDiv>
          </CategoriesDiv>
        ))}

        <BottomDiv>
          {/* Add Category */}
          <Button
            type="primary"
            style={{ backgroundColor: '#8DD3BB' }}
            onClick={addCategory}
          >
            Додати категорію
          </Button>

          {/* Public */}
          <PublicCheckboxDiv>
            <label htmlFor="isPublic">Публачна мандрівка?</label>
            <Checkbox
              type="checkbox"
              id="isPublic"
              name="isPublic"
              checked={data.isPublic}
              onChange={handlePublicChange}
            />
          </PublicCheckboxDiv>
        </BottomDiv>

        <AddPhotos onCdnUrlsChange={handleCdnUrlsChange} />

        {/* Submit */}
        <ButtonSend
          style={{
            marginTop: '10px',
            color: '#8DD3BB',
            borderColor: '#8DD3BB',
          }}
          variant="outlined"
          startIcon={<SendIcon />}
          type="submit"
        >
          Відправити
        </ButtonSend>
      </Form>
    </>
  );
};
```

```
export default AddTripForm;
```

Код з файлу \ components\ UserSections\AddTripForm\ AddTripForm.style.jsx

```
import styled from 'styled-components';

export const Form = styled.form`
  max-width: 600px;
  padding: 15px;
  border: 1px solid #8dd3bb;
  border-radius: 5px;
  background-color: #fafafa;
  display: flex;
  flex-direction: column;
  gap: 10px;
  box-shadow: 0px 4px 16px 0px rgba(17, 34, 17, 0.05);
`;

export const CategoriesDiv = styled.div`
  display: flex;
  flex-direction: column;
  padding: 15px;
  border: 1px solid #8dd3bb;
  border-radius: 5px;
  margin-bottom: 10px;
  gap: 10px;
`;

export const PublicCheckboxDiv = styled.div`
  display: flex;
  gap: 5px;
`;

export const TodoDiv = styled.div`
  display: flex;
  align-items: center;
  gap: 15px;
  margin-bottom: 7px;
`;

export const ChangeDiv = styled.div`
  display: flex;
  align-items: center;
  gap: 15px;
`;

export const BottomDiv = styled.div`
  display: flex;
  justify-content: space-between;
  align-items: center;
```

```
  margin-bottom: 10px;
`;
```

Код з файлу \ components\ UserSections\AddTripForm\ AddPhotos\AddPhotos.jsx

```
/* eslint-disable react-hooks/exhaustive-deps */
import React, { useEffect, useRef, useState } from 'react';
import * as LR from '@uploadcare/blocks';
import '@uploadcare/blocks/web/lr-file-uploader-regular.min.css';

import { ContainerImg } from './AddPhotos.style';

LR.registerBlocks(LR);

const AddPhotos = ({ onCdnUrlsChange }) => {
  const [files, setFiles] = useState([]);
  const [cdnUrls, setCdnUrls] = useState([]);
  const ctxProviderRef = useRef(null);

  useEffect(() => {
    const ctxProvider = ctxProviderRef.current;
    if (!ctxProvider) return;

    const handleChangeEvent = event => {
      const newFiles = event.detail.allEntries.filter(
        file => file.status === 'success'
      );
      setFiles(newFiles);
      setCdnUrls(prevCdnUrls => {
        const newUrls = newFiles.map(file => ({
          cdnUrl: file.cdnUrl,
          uuid: file.uuid,
        }));
        const uniqueUrls = [
          ...prevCdnUrls,
          ...newUrls.filter(
            newUrl => !prevCdnUrls.some(url => url.cdnUrl ===
newUrl.cdnUrl)
          ),
        ];
        return uniqueUrls;
      });
    };

    ctxProvider.addEventListener('change', handleChangeEvent);

    return () => {
      ctxProvider.removeEventListener('change', handleChangeEvent);
    };
  }, []);
```

```
  useEffect(() => {
    onCdnUrlsChange(cdnUrls);
  }, [cdnUrls]);

  return (
    <div>
      <lr-config ctx-name="my-uploader" pubkey="274c6cf9681b13936265"
/>

      <lr-file-uploader-regular ctx-name="my-uploader" />

      <lr-upload-ctx-provider ctx-name="my-uploader"
ref={ctxProviderRef} />

      <ContainerImg>
        {files.map(file => (
          <div key={file.uuid}>
            <img
              style={{ width: '150px' }}
              src={file.cdnUrl}
              alt={file.fileInfo.originalFilename}
            />
          </div>
        ))}
      </ContainerImg>
    </div>
  );
};

export default AddPhotos;
```

Код з файлу \ components\ UserSections\AddTripForm\ AddPhotos\ AddPhotos.style.jsx

```
import styled from 'styled-components';

export const ContainerImg = styled.div`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 8px;
  margin-top: 10px;
`;
```

Код з файлу \ components\ UserSections\ AllUserTrips\AllUserTrips.jsx

```
/* eslint-disable jsx-a11y/img-redundant-alt */

import React, { useState, useEffect } from 'react';
import { Card, Col } from 'antd';
```

```
import {
  Cards,
  TripDescription,
  Category,
  CategoryName,
  TodoList,
  TodoItem,
  PhotoList,
  Photo,
} from './AllUserTrips.style';

const AllUserTrips = () => {
  const [trips, setTrips] = useState([]);
  const token = localStorage.getItem('token');

  useEffect(() => {
    fetch('http://localhost:3000/api/trips', {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
      .then(response => response.json())
      .then(data => setTrips(data))
      .catch(error => console.error('Error fetching trips:', error));
  }, [token]);

  const getRandomColor = () => {
    const letters = '0123456789ABCDEF';
    let color = '#';
    for (let i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
  };

  return (
    <>
      <Cards>
        {trips.map(trip => (
          <Col span={8} key={trip._id}>
            <Card
              style={{ marginBottom: '15px', width: '390px' }}
              title={trip.title}
              bordered={false}
              headStyle={{
                backgroundColor: getRandomColor(),
                color: '#fafbfc',
              }}
            >
              <TripDescription>{trip.description}</TripDescription>
              <Category>Категорії:</Category>
              <ul>
                {trip.categories.map((category, index) => (
```

```
                  <li key={index}>

<CategoryName>{category.nameCategory}</CategoryName>
                    <TodoList>
                      {category.todoList.map((todo, index) => (
                        <TodoItem key={index}>{todo.todo}</TodoItem>
                      ))}
                    </TodoList>
                  </li>
                ))}
              </ul>
              <p>Публачна мандрівка: {trip.isPublic ? 'Так' :
'Ні'}</p>
              {trip.photos.length > 0 && (
                <div>
                  <h3>Фотографії:</h3>
                  <PhotoList>
                    {trip.photos.map((photo, index) => (
                      <Photo key={index}>
                        <img src={photo.cdnUrl} alt={`Photo ${index +
1}`} />
                      </Photo>
                    ))}
                  </PhotoList>
                </div>
              )}
            </Card>
          </Col>
        ))}
      </Cards>
    </>
  );
};

export default AllUserTrips;
```

Код з файлу \ components\ UserSections\ AllUserTrips\ AllUserTrips.style.jsx

```
import styled from 'styled-components';

export const Cards = styled.ul`
  display: flex;
  flex-direction: row;
  gap: 16px;

  @media (max-width: 426px) {
    flex-direction: column;
  }
`;

export const TripDescription = styled.p`
```

```
  color: #666;
  font-size: 14px;
  margin-bottom: 12px;

  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const Category = styled.h3`
  margin-bottom: 4px;

  @media (max-width: 425px) {
    font-size: 16px;
  }
`;

export const CategoryName = styled.h4`
  color: #555;
  margin-bottom: 2px;

  @media (max-width: 425px) {
    font-size: 14px;
  }
`;

export const TodoList = styled.ul`
  list-style-type: disc;
  margin-bottom: 4px;
  margin-left: 15px;

  @media (max-width: 425px) {
    font-size: 12px;
    margin-left: 10px;
  }
`;

export const TodoItem = styled.li`
  color: #777;

  @media (max-width: 425px) {
    font-size: 12px;
  }
`;

export const PhotoList = styled.ul`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 4px;
```

```
  @media (max-width: 425px) {
    gap: 2px;
  }
`;

export const Photo = styled.li`
  width: 150px;
`;
```

Код з файлу \ components\ UserSections\ UpdateTrip\UpdateTrip.jsx

```
/* eslint-disable jsx-a11y/img-redundant-alt */
import { useState, useEffect } from 'react';
import {
  deleteFile,
  deleteFiles,
  UploadcareSimpleAuthSchema,
} from '@uploadcare/rest-client';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Input, Checkbox, Button } from 'antd';
import ButtonSend from '@mui/material/Button';
import SendIcon from '@mui/icons-material/Send';

import AddPhotos from '../AddTripForm/AddPhotos/AddPhotos';

import {
  UpdateDiv,
  Form,
  CategoriesDiv,
  PublicCheckboxDiv,
  TodoDiv,
  ChangeDiv,
  BottomDiv,
  ContainerImg,
} from './UpdateTrip.style';

const { TextArea } = Input;

const UpdateTrip = () => {
  const initialState = {
    title: '',
    description: '',
    categories: [
      {
        nameCategory: '',
        todoList: [
          {
            todo: '',
          },
        ],
        publicList: false,
```

```
    },
  ],
  isPublic: false,
  photos: [],
};

const [data, setData] = useState(initialState);
const [cdnUrls, setCdnUrls] = useState([]);
const [trips, setTrips] = useState([]);
const [selectedTripId, setSelectedTripId] = useState(null);
const [isTripSelected, setIsTripSelected] = useState(false);
const [uploadcareSimpleAuthSchema, setUploadcareSimpleAuthSchema] =
  useState(null);

useEffect(() => {
  const token = localStorage.getItem('token');
  const url = 'http://localhost:3000/api/trips/keys';

  fetch(url, {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
  })
    .then(response => response.json())
    .then(data => {
      setUploadcareSimpleAuthSchema(
        new UploadcareSimpleAuthSchema({
          publicKey: data.publicKey,
          secretKey: data.secretKey,
        })
      );
    })
    .catch(error => {
      console.error('Error fetching keys:', error);
    });
}, []);

useEffect(() => {
  const savedData = JSON.parse(localStorage.getItem('data'));

  if (savedData) {
    setData(savedData);
  }
}, []);

useEffect(() => {
  localStorage.setItem('data', JSON.stringify(data));
}, [data]);

useEffect(() => {
  const token = localStorage.getItem('token');
```

```
  const url = 'http://localhost:3000/api/trips';

  fetch(url, {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
  })
    .then(response => response.json())
    .then(responseData => {
      setTrips(responseData);
    })
    .catch(error => {
      console.error('Error fetching trips:', error);
    });
}, []);

const handleChange = event => {
  const { name, value } = event.target;

  setData(prevData => ({
    ...prevData,
    [name]: value,
  }));
};

const handleCategoryChange = (event, index) => {
  const { name, value } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories[index] = {
      ...newCategories[index],
      [name]: value,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const handleTodoChange = (event, categoryIndex, todoIndex) => {
  const { value } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    const newTodoList = [...newCategories[categoryIndex].todoList];
    newTodoList[todoIndex] = {
      todo: value,
    };
```

```
    newCategories[categoryIndex] = {
      ...newCategories[categoryIndex],
      todoList: newTodoList,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const handlePublicChange = event => {
  const { checked } = event.target;

  setData(prevData => ({
    ...prevData,
    isPublic: checked,
  }));
};

const handleCategoryPublicChange = (event, categoryIndex) => {
  const { checked } = event.target;

  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories[categoryIndex] = {
      ...newCategories[categoryIndex],
      publicList: checked,
    };

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const addCategory = () => {
  setData(prevData => ({
    ...prevData,
    categories: [
      ...prevData.categories,
      {
        nameCategory: '',
        todoList: [
          {
            todo: '',
          },
        ],
        publicList: false,
      },
    ],
```

```
  }));
};

const addTodo = categoryIndex => {
  const updatedCategories = [...data.categories];
  updatedCategories[categoryIndex].todoList.push({
    todo: '',
  });
  setData(prevData => ({
    ...prevData,
    categories: updatedCategories,
  }));
};

const deleteCategory = categoryIndex => {
  setData(prevData => {
    const newCategories = [...prevData.categories];
    newCategories.splice(categoryIndex, 1);

    return {
      ...prevData,
      categories: newCategories,
    };
  });
};

const deleteTodo = (categoryIndex, todoIndex) => {
  const updatedCategories = [...data.categories];
  updatedCategories[categoryIndex].todoList.splice(todoIndex, 1);
  setData(prevData => ({
    ...prevData,
    categories: updatedCategories,
  }));
};

const handleCdnUrlsChange = urls => {
  setCdnUrls(urls);
};

const handlePhotoDelete = index => {
  const updatedPhotos = [...data.photos];
  const deletedPhoto = updatedPhotos.splice(index, 1)[0];

  deleteFile(
    { uuid: deletedPhoto.uuid },
    { authSchema: uploadcareSimpleAuthSchema }
  );

  setData(prevData => ({
    ...prevData,
    photos: updatedPhotos,
  }));
};
```

```
  const handleSubmit = event => {
    event.preventDefault();

    const token = localStorage.getItem('token');
    const url = `http://localhost:3000/api/trips/${selectedTripId}`;

    const updatedPhotos = data.photos.map(photo => {
      const cdnUrl = cdnUrls.find(url => url === photo.cdnUrl);
      return cdnUrl
        ? { cdnUrl, uuid: photo.uuid }
        : { cdnUrl: photo.cdnUrl, uuid: photo.uuid };
    });

    cdnUrls.forEach(newPhoto => {
      if (!data.photos.find(photo => photo.cdnUrl ===
newPhoto.cdnUrl)) {
        updatedPhotos.push({ cdnUrl: newPhoto.cdnUrl, uuid:
newPhoto.uuid });
      }
    });

    const updatedData = {
      ...data,
      categories: data.categories.map(category => ({
        nameCategory: category.nameCategory,
        todoList: category.todoList.map(todo => ({
          todo: todo.todo,
        })),
        publicList: category.publicList,
      })),
      photos: [...updatedPhotos],
    };

    fetch(url, {
      method: 'PATCH',
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`,
      },
      body: JSON.stringify(updatedData),
    })
      .then(response => response.json())
      .then(responseData => {
        toast.success('Мадрівка успішно оновлена!');
        setData(initialState);
        localStorage.removeItem('data');
        setCdnUrls([]);
        window.location.reload();
      })
      .catch(error => {
        console.error('Error:', error);
        toast.error('Упс... Щось пішло не так!');
```

```
    });
};

const handleTripSelect = tripId => {
  const selectedTrip = trips.find(trip => trip._id === tripId);
  setData({
    title: selectedTrip.title,
    description: selectedTrip.description,
    categories: selectedTrip.categories.map(category => ({
      nameCategory: category.nameCategory,
      todoList: category.todoList.map(todo => ({
        todo: todo.todo,
      })),
      publicList: category.publicList,
    })),
    isPublic: selectedTrip.isPublic,
    photos: selectedTrip.photos.map(photo => ({
      cdnUrl: photo.cdnUrl,
      uuid: photo.uuid,
    })),
  });
  setCdnUrls(selectedTrip.photos.map(photo => photo.cdnUrl));
  setSelectedTripId(tripId);
  setIsTripSelected(true);
};

const handleDeleteTrip = () => {
  const token = localStorage.getItem('token');
  const url = `http://localhost:3000/api/trips/${selectedTripId}`;

  const photoUUIDs = data.photos.map(photo => photo.uuid);

  fetch(url, {
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
  })
    .then(response => response.json())
    .then(responseData => {
      toast.success('Мандрівка успішно видалена!');
      setData(initialState);
      localStorage.removeItem('data');
      setCdnUrls([]);
      setIsTripSelected(false);
      setTrips(trips.filter(trip => trip._id !== selectedTripId));
    })
    .catch(error => {
      console.error('Error deleting trip:', error);
      toast.error('Упс... Щось пішло не так!');
    });
```

```
    if (photoUUIDs.length > 0) {
      deleteFiles(
        { uuids: photoUUIDs },
        { authSchema: uploadcareSimpleAuthSchema }
      );
    }
  };

  return (
    <>
      <ToastContainer />
      <h2>Обери мандрівку для внесення зміни</h2>
      <UpdateDiv>
        <ul>
          {trips.map(trip => (
            <li key={trip._id} onClick={() =>
handleTripSelect(trip._id)}>
              {trip.title}
            </li>
          ))}
        </ul>

        {isTripSelected && (
          <>
            <Form onSubmit={handleSubmit}>
              {/* Title */}
              <label htmlFor="title">Назва</label>
              <Input
                type="text"
                id="title"
                name="title"
                value={data.title}
                onChange={handleChange}
              />

              {/* Description */}
              <label htmlFor="description">Опис</label>
              <TextArea
                type="text"
                id="description"
                name="description"
                value={data.description}
                onChange={handleChange}
              />

              {/* Categories */}
              <h4>Категорії</h4>
              {data.categories.map((category, categoryIndex) => (
                <CategoriesDiv key={categoryIndex}>
                  <label htmlFor={`category-${categoryIndex}`}>
                    Назва категорії
                  </label>
                  <TodoDiv>
```

```
                    <Input
                      type="text"
                      id={`category-${categoryIndex}`}
                      name="nameCategory"
                      value={category.nameCategory}
                      onChange={event =>
                        handleCategoryChange(event, categoryIndex)
                      }
                    />

                    {/* Category Public */}
                    <PublicCheckboxDiv>
                      <label htmlFor={`category-public-
${categoryIndex}`}>
                        Публічна категорія?
                      </label>
                      <Checkbox
                        type="checkbox"
                        id={`category-public-${categoryIndex}`}
                        name="publicList"
                        checked={category.publicList}
                        onChange={event =>
                          handleCategoryPublicChange(event,
categoryIndex)
                        }
                      />
                    </PublicCheckboxDiv>
                  </TodoDiv>

                  {/* Todo List */}
                  <h3>Нотатки</h3>
                  {category.todoList.map((todo, todoIndex) => (
                    <TodoDiv key={todoIndex}>
                      {/* <label htmlFor={`todo-${categoryIndex}-
${todoIndex}`}>
                      Todo
                    </label> */}
                      <Input
                        type="text"
                        id={`todo-${categoryIndex}-${todoIndex}`}
                        name="todo"
                        value={todo.todo}
                        onChange={event =>
                          handleTodoChange(event, categoryIndex,
todoIndex)
                        }
                      />

                      {/* Delete Todo */}
                      <Button
                        type="primary"
                        danger
                        style={{ backgroundColor: '#ED5E68' }}
```

```
                        onClick={() => deleteTodo(categoryIndex,
todoIndex)}
                      >
                        Видалити нотатку
                      </Button>
                    </TodoDiv>
                  ))}
                  <ChangeDiv>
                    {/* Add Todo */}
                    <Button
                      type="primary"
                      style={{ backgroundColor: '#8DD3BB' }}
                      onClick={() => addTodo(categoryIndex)}
                    >
                      Додати нотатку
                    </Button>

                    {/* Delete Category */}
                    <Button
                      type="primary"
                      style={{ backgroundColor: '#ED5E68' }}
                      danger
                      onClick={() => deleteCategory(categoryIndex)}
                    >
                      Видалити категорію
                    </Button>
                  </ChangeDiv>
                </CategoriesDiv>
              ))}

              <BottomDiv>
                {/* Add Category */}
                <Button
                  type="primary"
                  style={{ backgroundColor: '#8DD3BB' }}
                  onClick={addCategory}
                >
                  Додати категорію
                </Button>

                {/* Public */}
                <PublicCheckboxDiv>
                  <label htmlFor="isPublic">Публачна
мандрівка?</label>
                  <Checkbox
                    type="checkbox"
                    id="isPublic"
                    name="isPublic"
                    checked={data.isPublic}
                    onChange={handlePublicChange}
                  />
                </PublicCheckboxDiv>
              </BottomDiv>
```

```
            <AddPhotos onCdnUrlsChange={handleCdnUrlsChange} />

            <ContainerImg>
              {data.photos.map((photo, index) => (
                <div key={index}>
                  <img
                    style={{ width: '150px' }}
                    src={photo.cdnUrl}
                    alt={`Trip photo ${index}`}
                  />
                  <Button
                    type="primary"
                    style={{ backgroundColor: '#ED5E68' }}
                    danger
                    onClick={() => handlePhotoDelete(index)}
                  >
                    Видалити
                  </Button>
                </div>
              ))}
            </ContainerImg>

            {/* Submit */}
            <ButtonSend
              style={{
                marginTop: '10px',
                color: '#8DD3BB',
                borderColor: '#8DD3BB',
              }}
              variant="outlined"
              startIcon={<SendIcon />}
              type="submit"
            >
              Відправити
            </ButtonSend>

            {/* Delete Trip */}
            <Button
              type="primary"
              style={{ backgroundColor: '#ED5E68' }}
              danger
              onClick={handleDeleteTrip}
            >
              Видалити мандрівку
            </Button>
          </Form>
        </>
      )}
    </UpdateDiv>
  </>
 );
};
```

```
export default UpdateTrip;
```

Код з файлу \ components\ UserSections\ UpdateTrip\ UpdateTrip.style.jsx

```
import styled from 'styled-components';

export const UpdateDiv = styled.div`
  display: flex;
  justify-content: space-between;

  @media (max-width: 426px) {
    flex-direction: column;
  }
`;

export const Form = styled.form`
  max-width: 600px;
  padding: 15px;
  border: 1px solid #8dd3bb;
  border-radius: 5px;
  background-color: #fafafa;
  display: flex;
  flex-direction: column;
  gap: 10px;
  box-shadow: 0px 4px 16px 0px rgba(17, 34, 17, 0.05);
`;

export const CategoriesDiv = styled.div`
  display: flex;
  flex-direction: column;
  padding: 15px;
  border: 1px solid #8dd3bb;
  border-radius: 5px;
  margin-bottom: 10px;
  gap: 10px;
`;

export const PublicCheckboxDiv = styled.div`
  display: flex;
  gap: 5px;
`;

export const TodoDiv = styled.div`
  display: flex;
  align-items: center;
  gap: 15px;
  margin-bottom: 7px;
`;

export const ChangeDiv = styled.div`
  display: flex;
```

```
  align-items: center;
  gap: 15px;
`;

export const BottomDiv = styled.div`
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 10px;
`;

export const ContainerImg = styled.div`
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 8px;
  margin-top: 10px;
`;
```