МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ПЛАНУВАННЯ ПОДОРОЖЕЙ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22010808

Здобувачка _____А. М. Задніпрянець

підпис

«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____ Є. О. Давиденко

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Є. О. Давиденко

«17» січня 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано здобувачці групи 408 факультету комп'ютерних наук

Задніпрянець Анні Миколаївні

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Вебзастосунок планування подорожей

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «_» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом роботи є вебзастосунок планування подорожей.

4. Перелік питань, що підлягають розробці:
   - аналіз аналогічних застосунків планування подорожей;
   - розробка специфікації вимог;
   - моделювання програмного забезпечення;
   - розробка дизайну вебзастосунку;
   - розробка та тестування програмного забезпечення.

5. Перелік графічних матеріалів

Презентація.

6. Завдання до спеціальної частини

Охорона праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи канд. техн. наук, доцент Давиденко Євген Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Задніпрянець Анна Миколаївна
(прізвище, ім'я, по батькові здобувача)

_____
(підпис)

Дата видачі завдання «17» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок планування подорожей

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	16.01.2024	17.01.2024	виконано
2	Огляд літератури за темою роботи	15.01.2024	19.01.2024	виконано
3	Складання календарного плану КРБ	19.02.2024	20.02.2024	виконано
4	Аналіз предметної області	15.01.2024	02.02.2024	виконано
5	Розробка проєктних рішень	03.02.2024	07.02.2024	виконано
6	Моделювання та конструювання ПЗ	01.03.2024	29.03.2024	виконано
7	Кодування та тестування розробленого ПЗ, розробка керівництва користувача	29.03.2024	14.06.2024	виконано
8	Розробка спеціальної частини з охорони праці	22.05.2024	27.05.2024	виконано
9	Відгук керівника КРБ	13.06.2024	13.06.2024	виконано
10	Оформлення КРБ та презентації	15.05.2024	10.06.2024	виконано
11	Попередній захист	05.06.2024	05.06.2024	виконано
12	Завершення оформлення КРБ та презентації	10.06.2024	13.06.2024	виконано
13	Рецензування	14.06.2024	14.06.2024	виконано
14	Захист кваліфікаційної роботи	25.06.2024	25.06.2024	виконано

Розробила здобувачка Задніпрянець А. М. _____
(*прізвище, ім'я, по батькові)* *(підпис)*

«20» лютого 2024 р.

Керівник роботи канд. техн. наук, доцент Давиденко Є. О. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«20» лютого 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок планування подорожей»

Здобувачка 408 гр.: Задніпрянець Анна Миколаївна

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Кваліфікаційна робота присвячена розробці програмного забезпечення для автоматизації процесу планування подорожей.

Об’єкт роботи: процес планування та організації подорожей.

Предмет роботи: програмні засоби створення вебзастосунку планування подорожей.

Мета роботи: розробка вебзастосунку для автоматизації процесу планування подорожей.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків та переліку джерел посилання.

У вступі визначається актуальність теми, мета роботи, предмет дослідження та об’єкт дослідження.

У першому розділі проаналізовано аналогічні застосунки планування подорожей, виділено їх переваги та недоліки. Описано визначений функціонал та розроблену специфікацію вимог до програмного забезпечення. У другому розділі описано розроблені проєктні рішення, а саме моделювання функцій та інформаційних моделей. У третьому розділі описано виконану роботу з моделювання та проєктування програмного забезпечення, а саме вибір технологій реалізації, мов програмування та вибір компонентів програмного забезпечення. У четвертому розділі надано опис кодування програмного забезпечення, результатів тестування розробленого застосунку та розробленого керівництва користувача. У висновках проводиться аналіз виконаних завдань та отриманих результатів.

КРБ викладена на 62 сторінки, вона містить 4 розділи, 47 ілюстрацій, 11 таблиць, 21 джерело в переліку посилань.

Ключові слова: *створення вебзастосунку, Laravel, планування подорожей, створення щоденних маршрутів, відстеження витрат, спільні подорожі.*

ABSTRACT

of the Bachelor's Thesis

"Trip planning web application"

Student of group 408: Zadniprianets Anna

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor Davydenko Y. O.

The bachelor's thesis is devoted to the development of software for the automation of the trip planning process.

The object of work: the process of planning and organizing trips.

The subject of work: software tools for creating a trip planning web application.

Objective: development of a web application for the automation of the trip planning process.

The bachelor's thesis consists of an introduction, four chapters, conclusions and a list of reference sources. The introduction defines the relevance of the topic, the purpose of the work, the subject of research and the object of research.

In the first section, similar travel planning applications are analyzed, their advantages and disadvantages are highlighted. The defined functionality and the developed specification of software requirements are described.

The second section describes the developed project solutions, namely the modeling of functions and information models.

The third section describes the work done on software modeling and design, namely the selection of implementation technologies, programming languages, and the selection of software components.

The fourth section describes the coding of the software, the test results of the developed application, and the developed user manual. In the conclusions, the performed tasks and the obtained results are analyzed.

The bachelor's thesis is presented on 62 pages, it contains 4 sections, 47 illustrations, 11 tables, 21 sources in the list of references.

Keywords: *creating a web application, Laravel, trip planning, daily itineraries, expense tracking, shared trips*.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
ПЗ	–	програмне забезпечення
СКБД	–	система управління базами даних
ОС	–	операційна система
API	–	application programming interface
CRUD	–	create, add, update, delete
CSS	–	cascading style sheets
IDE	–	integrated development environment
IEEE	–	Institute of Electrical and Electronics Engineers
HTML	–	hypertext markup language
HTTP	–	hypertext transfer protocol
MVC	–	model-view-controller
PHP	–	hypertext preprocessor
SQL	–	structured query language
UML	–	unified modeling language

ВСТУП

Актуальність теми кваліфікаційної роботи бакалавра зумовлена зростанням числа людей, які подорожують по всьому світу, та потребою у зручних інструментах організації та планування подорожей.

На сьогодні, все більше людей використовують застосунки для планування своїх справ та роботи. Онлайн-планувальники дозволяють зберігати власні плани, що є доступними з будь-якого пристрою з підключенням до Інтернету. Вони надають можливість доступу до створених планів у будь-який час і з будь-якого місця, що забезпечує зручність в організації свого часу та планів. Завдяки застосункам-планувальникам можна легко керувати своїми завданнями, що стає у нагоді при сучасному ритмі життя. Застосунки планування подорожей не є винятком.

Для забезпечення зручної організації подорожі важливо мати всі плани зібрані у одному місці. У цьому велику роль відіграють застосунки планування подорожей. Вони дозволяють збирати всю необхідну інформацію про подорож, включаючи мапу маршруту, інформацію про місця для відвідування, щоденні плани, бронювання та інше. Такі застосунки надають інструменти для створення детальних планів подорожей, включаючи розклади, списки, побудову щоденних маршрутів, а також надають можливість встановлювати бюджет та розподіляти витрати. Крім того, користувачі можуть спільно планувати свої подорожі з друзями або родиною, редагувати плани та обмінюватися ідеями, що є перевагою для подорожей групою.

Якщо людина не має часу складати план подорожі з нуля, вебзастосунки планування подорожей знову стають у нагоді, адже можна взяти вже готові шаблони, створені іншими користувачами, для своєї майбутньої подорожі. Такі шаблони можна модифікувати під власні потреби і вподобання, змінювати маршрут, додавати нові місця чи вилучати ті, які не цікавлять.

Також завзяті мандрівники прагнуть знайти спільноту людей, які поділяють їх пристрасть до подорожей. У вебзастосунках планування подорожей вони можуть ділитися враженнями, порадами щодо маршрутів, рекомендаціями про найкращі

місця для відвідування та іншими корисними деталями. Така спільнота надає можливість обміну цінною інформацією, що допомагає кожному спланувати подорож якнайкраще.

Об’єкт роботи: процес планування та організації подорожей.

Предмет роботи: програмні засоби створення вебзастосунку планування подорожей.

Мета роботи: розробка вебзастосунку для автоматизації процесу планування подорожей.

Для досягнення поставленої мети необхідно виконати наступні **завдання**:

– проаналізувати аналогічні застосунки планування подорожей;
– визначити функціонал застосунку, що розробляється;
– розробити специфікацію вимог до програмного забезпечення;
– розробити UML-діаграми, що моделюють роботу програмного забезпечення;
– розробити дизайн застосунку;
– виконати кодування програмного забезпечення;
– провести тестування програмного забезпечення.

Сфера застосування: вебзастосунок планування подорожей можна використовувати як у сфері повсякденного життя, так і у інших сферах. Такі вебзастосунки можуть бути корисні як для індивідуальних мандрівників, так і для груп, родин або бізнес-подорожей. Вони спрощують процес планування та організації подорожей, дозволяють обмінюватися ідеями, а також покращують загальний досвід подорожування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд аналогічних застосунків

Вебзастосунки для планування подорожей є різноманітними, і кожен з них має свої функції та особливості. Обраними застосунками-аналогами є Funliday, Polarsteps та Grouppi.

Funliday [1]

Основним функціоналом вебзастосунку Funliday є створення щоденного маршруту (плану) подорожі, його публікація та перегляд створених планів інших користувачів. До створеної подорожі можна додавати друзів та редагувати маршрут разом з ними. Користувач редагує подорож на спеціальній сторінці (рис. 1.1). До кожного вказаного при створенні подорожі дня користувач створює окремий план (список місць для відвідування). До кожного пункту списку можна додати час, локацію та нотатки.

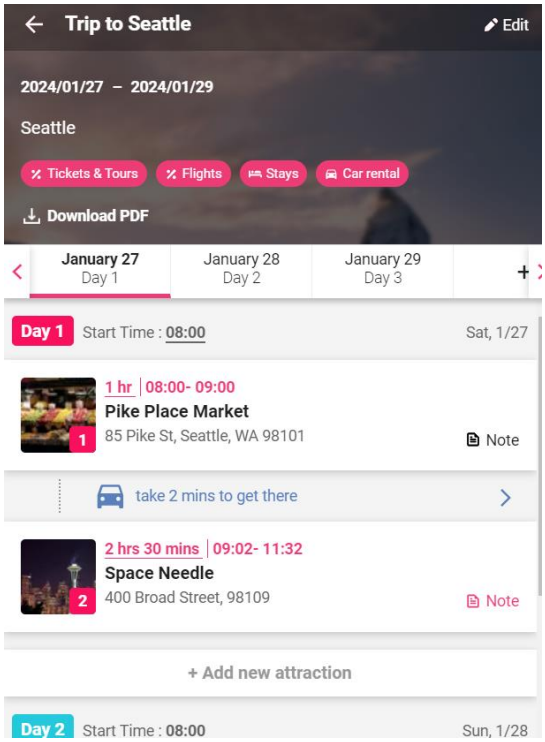

Рисунок 1.1 – Інтерфейс Funliday

Всі створені користувачем подорожі знаходяться на сторінці Trips (рис. 1.2).

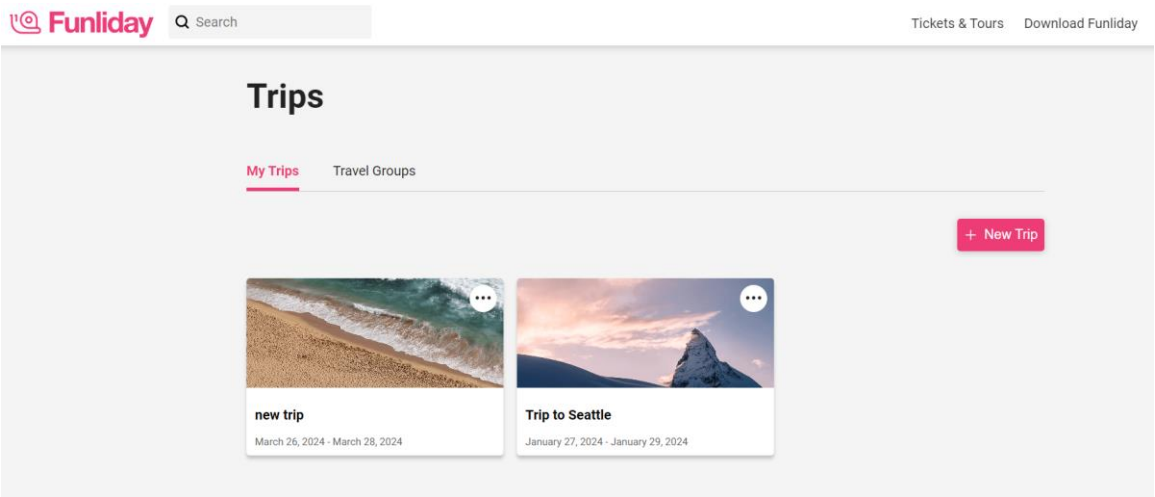

Рисунок 1.2 – Інтерфейс Funliday, продовження

На сторінці «Профіль» наявні стандартні налаштування профілю: зміна введених при реєстрації даних, перегляд своїх подорожей.

На сторінці пошуку можна шукати публічні плани подорожей інших користувачів та шукати самих користувачів відповідно. До публічних подорожей можна додавати коментарі.

Перелік основних функцій, переваг та недоліків застосунку Funliday наведено в табл. 1.1.

Таблиця 1.1 – Характеристика Funliday

Назва	Funliday
Виробник	Компанія Funliday, Inc
Вебсайт	www.funliday.com
Мова реалізації	Javascript
Основні функції	1) побудова щоденного маршруту подорожі; 2) створення подорожей спільно з іншими користувачами; 3) публікація плану подорожі; 4) перегляд опублікованих користувачами планів, написання коментарів.

Кінець таблиці 1.1

Переваги	1)	перегляд створеного маршруту на мапі;
	2)	можливість додавати друзів у подорож.
Недоліки	1)	інтуїтивно незрозумілий інтерфейс у деяких вкладках;
	2)	відсутність можливості відстеження витрат;
	3)	деякі функції є прихованими на сторінках вебзастосунку.

Polarsteps [2]

Наступним аналогом є вебзастосунок Polarsteps. Polarsteps дозволяє планувати подорожі, будуючи щоденні плани (рис. 1.3, 1.5), в яких можна вказати активність, опис до неї та дату. Користувачі вводять всі дані вручну. Також можна додавати фото: ця функція робить Polarsteps не лише звичайним планувальником подорожей, а й трекером подорожей: тобто можна зберігати та відстежувати спогади у вигляді трекеру з фотографіями.

Плани подорожей можна публікувати, тоді їх будуть бачити всі користувачі вебзастосунку, а також до них можна написати коментарі.

Готові подорожі та користувачів вебзастосунку можна шукати (рис. 1.4). Також наявний функціонал слідкування (підписки на користувачів).

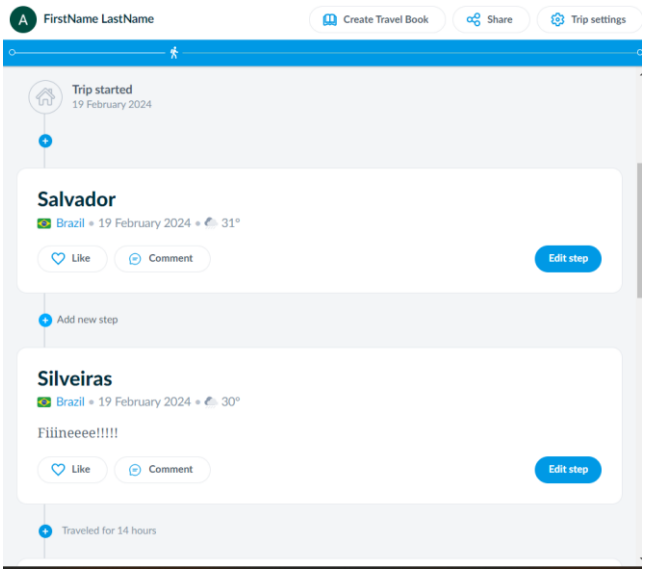

Рисунок 1.3 – Інтерфейс Polarsteps

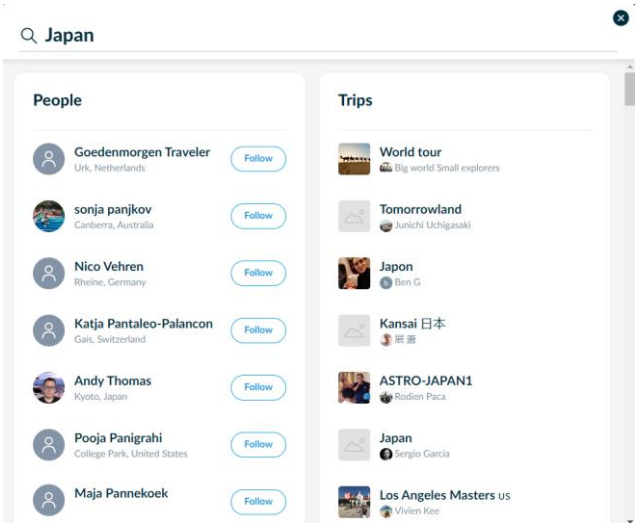

Рисунок 1.4 – Інтерфейс Polarsteps, продовження

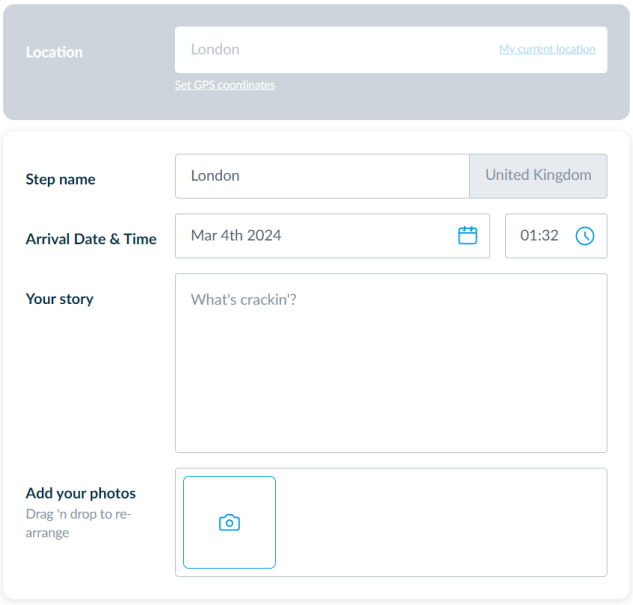

Рисунок 1.5 – Форма додання місця в маршрут

Перелік основних функцій, переваг та недоліків вебзастосунку Polarsteps наведено в табл. 1.2.

Таблиця 1.2 – Характеристика Polarsteps

Назва	Polarsteps
Виробник	Polarsteps
Вебсайт	www.polarsteps.com
Мова реалізації	Javascript

Кінець таблиці 1.2

Основні функції	1) 2) 3) 4)	побудова щоденного маршруту подорожі; публікація плану подорожі; пошук подорожей, користувачів, путівників; перегляд статистики подорожей.
Переваги	1) 2)	зручний та привабливий інтерфейс; наявність офіційних путівників подорожей від редакторів.
Недоліки	1) 2)	відсутність можливості спільного редагування подорожей; відсутність можливості відстеження витрат.

Grouppii [3]

Останнім аналогом можна виділити застосунок Grouppii. На відміну від попередніх аналогів, він є мобільним застосунком. Застосунки планування подорожей розробляються як для мобільних пристроїв, так і для користувачів персональних комп’ютерів, і вони мають подібний функціонал, тому для аналізу предметної області доцільно виділити мобільний застосунок як аналог майбутньої системи. Інтерфейс Grouppi зображено на рис. 1.6. Як і попередні аналоги, застосунок є помічником у плануванні подорожей, де також можна будувати щоденні плани (додавати до днів активності, опис, дати). Користувачі вводять всі дані вручну.

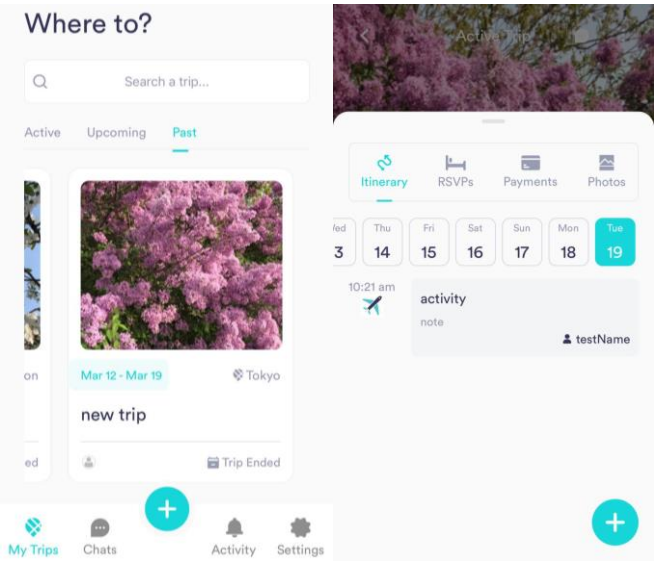

Рисунок 1.6 – Інтерфейс Grouppi

Перелік основних функцій, переваг та недоліків застосунку Grouppii наведено в табл. 1.3.

Таблиця 1.3 – Характеристика Grouppii

Назва	Grouppii
Виробник	Grouppii Group
Вебсайт	www.grouppii.com
Мова реалізації	Java
Основні функції	1) створення маршрутів подорожей; 2) взаємодія з іншими користувачами; 3) вбудований чат у застосунок; 4) додання та відстеження витрат.
Переваги	1) привабливий інтерфейс; 2) можливість спільного редагування подорожей (запрошення друзів).
Недоліки	1) при створенні подорожей та активностей усі поля форми введення є обов'язковими для заповнення, напр. фото; 2) інтуїтивно незрозумілий інтерфейс для функціоналу відстеження витрат.

Проаналізувавши застосунки-аналоги, можна перейти до аналізу системи, що розробляється.

1.2 Аналіз застосунку, що розробляється

Призначенням вебзастосунку планування подорожей, що розробляється, є автоматизація процесу організації та планування подорожей.

Користувачі системи

У системі передбачені наступні ролі користувачів:

1) користувач – роль більшості користувачів системи, основним функціоналом є створення та організація подорожей, їх публікація;

2) адміністратор – може блокувати облікові записи користувачів, видаляти коментарі та подорожі, також повністю успадковує функції користувача.

Оскільки у вебзастосунку наявний функціонал генерації контенту користувачами, доцільно відстежувати опублікований контент та впровадити можливість відправлення скарг на контент, що є публічним.

Функціями системи є:

– авторизація та реєстрація користувачів;

– створення нової подорожі, редагування та видалення існуючих подорожей;

– редагування профілю;

– пошук подорожей (за ключовими словами та/або автором) та їх перегляд;

– пошук користувачів;

– перегляд профілів користувачів;

– написання коментарів до опублікованих подорожей;

– можливість відправити скаргу на опублікований контент.

При редагуванні подорожі:

– створення щоденного маршруту (плану): додання місця, нотаток та витрат (до кожного визначеного дня);

– встановлення видимості подорожі (публікація);

– встановлення бюджету, додання витрат;

– перегляд статистики витрат;

– додання/видалення інших користувачів до/з подорожі;

– встановлення фото подорожі, додання опису.

Для адміністратора:

– блокування облікових записів користувачів;

– видалення коментарів, опублікованих подорожей;

– перегляд загальної кількості зареєстрованих користувачів, створених подорожей, коментарів та скарг в адмін-панелі.

Отже, для користувача системи необхідно розробити такі сторінки:

– реєстрації;

– авторизації;

– головну сторінку;
– сторінку профілю користувача;
– сторінку для редагування даних користувача;
– відновлення паролю;
– сторінка створення подорожі;
– сторінка редагування подорожі;
– сторінка пошуку подорожей користувачів;
– модальні вікна для реалізації деякого функціоналу.

Для адміністратора необхідно створити адмін-панель, через яку він буде мати змогу переглядати статистику використання вебзастосунку (кількість користувачів, кількість створених подорожей та написаних коментарів) та керувати скаргами користувачів та коментарями. Також адміністратор повинен мати доступ до всіх перелічених сторінок вище, адже він успадковує всі функції звичайного користувача.

1.3 Специфікація вимог до програмного забезпечення

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням ПЗ, що розробляється, є автоматизація процесу організації та планування подорожей.

Погодження, що ухвалені в програмній документації

Погоджень не ухвалено.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 14.06.2024р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Дане ПЗ не має обмежень у сферах його застосування, застосунок можна використовувати у повсякденному житті та інших сферах діяльності.

Характеристики користувачів

Основні характеристики користувачів: доступ до мережі Інтернет, наявність ПК або мобільного пристрою.

Загальна структура і склад системи

Основні частини для створення програмного забезпечення: вебзастосунок, сервер та база даних.

Загальні обмеження

Обмеження для роботи з ПЗ – наявність доступу до мережі Інтернет.

ФУНКЦІЇ ВЕБЗАСТОСУНКУ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Функція створення подорожі

Опис функції

Функція створення подорожі дозволяє користувачу створити пустий шаблон майбутньої подорожі, у якому можна планувати активності по дням.

Вхідна і вихідна інформація

Вхідна інформація – назва подорожі, дати початку та завершення, місце відправлення та призначення, імена користувачів (username) для додання до подорожі;

вихідна інформація – сторінка створення та редагування плану подорожі.

Функціональні вимоги

Доступ до мережі Інтернет.

Функція редагування подорожі

Опис функції

Функція редагування подорожі надає користувачу можливість створювати щоденний план подорожі, керувати витратами та запрошувати друзів для спільного редагування.

Вхідна і вихідна інформація

Вхідна інформація – місце, опис, категорія витрат username/email користувача для спільної подорожі (необов’язково);

вихідна інформація – готовий план подорожі.

Функціональні вимоги

База даних з зареєстрованими користувачами та доступ до мережі Інтернет.

Функція пошуку подорожей

Опис функції

Функція пошуку подорожей дозволяє користувачам шукати за ключовими словами та іменами користувачів опубліковані подорожі та переглядати їх, писати до них коментарі. Також користувачі можуть написати до них скаргу.

Вхідна і вихідна інформація

Вхідна інформація – ключові слова для пошуку, текст коментаря/скарги;

вихідна інформація – знайдені подорожі.

Функціональні вимоги

База даних подорожей, створених користувачами, та доступ до мережі Інтернет.

Функція редагування облікового запису

Опис функції

Функція редагування облікового запису дозволяє користувачу змінити дані, внесені при реєстрації.

Вхідна і вихідна інформація

Вхідна інформація – ім’я, ім’я користувача (username), пароль, опис;

вихідна інформація – змінені дані профілю.

Функціональні вимоги

Доступ до мережі Інтернет.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В цьому ПЗ джерелом вхідної інформації є користувач. Користувач має задавати вручну необхідні дані, а саме дати, локації, бюджет, нотатки, ключові слова для пошуку подорожей.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до цього пункту відсутні.

Вимоги до способів організації, збереження та ведення інформації

В якості БД обрано MySQL.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Відсутні жорсткі вимоги до технічного забезпечення. Персональний комп’ютер чи ноутбук з технічними характеристиками, що підтримують будь-який сучасний браузер.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура складається з клієнтської частини, серверної частини та БД.

Системне програмне забезпечення

Застосунок має бути розроблено з використанням фреймворку Laravel. В якості БД для вебзастосунку обрано MySQL.

Мережне програмне забезпечення

Для створення ПЗ використано ОС Windows 11, у якості IDE обрано PhpStorm, для перегляду вебсторінок – браузер Google Chrome.

Програмне забезпечення ведення інформаційної бази

CRUD-операції виконуються через БД MySQL.

Мова і технологія розробки ПЗ

Програмне забезпечення має розроблятися з використанням фреймворку Laravel. Мова розробки – PHP.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс користувача повинен задовольняти всі вимоги дизайну, щоб забезпечити користувачеві максимальну зручність та ефективність в роботі з системою, скорочуючи час, необхідний для ознайомлення з її функціоналом.

Апаратний інтерфейс

Апаратним інтерфейсом є пристрій користувача (ПК чи мобільний пристрій), який буде використано для роботи з вебзастосунком.

Програмний інтерфейс

Laravel – фреймворк для розробки вебзастосунків на мові програмування PHP. Він надає зручні засоби для роботи з маршрутизацією, базами даних, автентифікацією, а також включає безліч розширень для спрощення розробки та підтримки коду.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

ПЗ доступне для використання будь-яким користувачем, який має доступ до вебзастосунку, при наявності апаратного забезпечення і доступу до мережі Інтернет.

Переносимість

Програмне забезпечення може працювати на ОС Windows (версії 10 та вище) та з мобільними пристроями.

Продуктивність

Продуктивність роботи ПЗ залежить від швидкості підключення до мережі Інтернет. Час виконання запитів не має перевищувати 2 секунди.

Надійність

При реєстрації та використанні застосунку, інформація, що надається користувачем, повинна бути конфіденційною, і ПЗ повинне виключити можливість доступу до персональних даних користувача для інших користувачів. Для отримання доступу до власних даних користувач має пройти авторизацію в системі.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи бакалавра проведено аналіз аналогічних застосунків: виділено їх основні функції, переваги та недоліки. На основі проведеного аналізу сформовано вимоги до програмного забезпечення, що розробляється.

Визначено функціонал та користувачів системи, розроблено специфікацію вимог. У специфікації вимог надано характеристику ПЗ, що розробляється. Описано функціональні та нефункціональні вимоги до ПЗ. Надано перелік функцій системи: вхідну та вихідну інформацію, опис та назву функції, функціональні вимоги до неї.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ПЛАНУВАННЯ ПОДОРОЖЕЙ

2.1 Сценарії використання системи

Сценарії використання (use case) – це опис взаємодії користувача з системою для досягнення своєї мети або виконання певної операції, включаючи можливі варіанти успішного та неуспішного завершення.

Написання сценаріїв використання дозволяє чітко визначити, хто буде користувачем системи, які сценарії взаємодії з нею передбачаються та які цілі вона має досягти. Сценарії використання є специфікаціями функціональних та поведінкових вимог до системи, які визначають, які конкретні дії вона повинна виконувати.

Існує три форми сценаріїв використання:

1) коротка – короткий опис одного з основних сценаріїв (зазвичай успішного) роботи системи у вигляді одного абзацу, зазвичай використовується на етапі початкового аналізу вимог до системи;

2) поверхнева – загальний опис у вільній формі всіх сценаріїв (основних та альтернативних) одного з сценаріїв використання, використовується на етапі початкового аналізу вимог до системи;

3) повна – детальний опис всіх кроків та дій, включаючи умови перед і після виконання сценарію використання; зазвичай використовується на етапі вибору невеликої частини важливих сценаріїв (критичних для роботи системи) з повного переліку на коротку та поверхневу форми.

Для вебзастосунку планування подорожей розроблено 3 сценарії використання: по одному у короткій, поверхневій та повній формах відповідно.

Сценарій №1: авторизуватися у системі.

Користувач переходить в застосунок на сторінку логіну. Система відкриває користувачу форму авторизації. Користувач заповнює форму авторизації. Користувач натискає на кнопку «Sign in». Система виконує валідацію введених даних. Система відкриває користувачу сторінку його профілю.

Сценарій №2: створити нову подорож.

Головний сценарій (успішний): користувач натискає кнопку «Створити подорож» на головній сторінці. Система відкриває форму створення нової подорожі. Користувач вводить місце призначення у відповідне поле. Користувач вводить дати початку та закінчення подорожі. Користувач натискає кнопку «Add friends». Система відкриває форму додання інших користувачів до подорожі. Користувач додає інших користувачів. Користувач натискає кнопку «Save». Система відкриває користувачу сторінку для редагування створеної подорожі.

Альтернативні сценарії:

1) користувач залишає обов’язкові поля вводу незаповненими;
2) користувач не авторизований у системі;
3) технічний збій роботи системи, видається повідомлення про помилку.

Сценарій №3: **редагувати існуючу подорож (табл. 2.1)**

Таблиця 2.1 – Сценарій використання «Редагувати створену подорож»

Usecase section	**Comment**
Use Case Name	Редагувати створену подорож
Scope	Вебзастосунок планування подорожей
Level	Мета користувача (user-goal)
Primary Actor	Користувач
Stakeholders and interests	Користувач – зацікавлений у редагуванні створеної подорожі
Preconditions	Користувач створив нову подорож
Success guarantee	1) користувач має використовувати браузер, що підтримує HTML5 та Javascript; 2) користувач має використовувати стабільне підключення до мережі Інтернет.

Кінець таблиці 2.1

Main Success Scenario	1) користувач обрав подорож для редагування; 2) користувач обирає параметр подорожі для редагування: редагування маршруту / бюджет / користувачі у подорожі / витрати / видимість подорожі; 3) користувач редагує обраний параметр; 4) користувач зберігає зміни; 5) система зберігає внесені користувачем зміни; 6) користувач переглядає оновлені деталі подорожі; 7) пункти 2-6 повторюються якщо користувач редагує декілька параметрів; 8) користувач закриває вкладку редагування подорожі.
Extensions	1) користувач не заповнює обов’язкові поля при введені даних; 2) технічний збій роботи системи.
Special Requirements	Система має виконувати запити не довше 2 секунд
Technology and Data Variations List	Немає
Frequency of Occurrence	Система може працювати майже безперервно
Miscellaneous	Немає

Для відображення взаємодії між користувачами у системі, та функцій користувача у ній, доцільно розробити діаграму використання системи (use case diagram).

Use case діаграма – це вид UML-діаграми, яка відображає взаємодію між користувачами системи та самою системою. Вона використовується для візуалізації функціональних вимог до системи та сценаріїв використання. Основна мета use case діаграми – показати, як користувачі (актори) взаємодіють з системою для досягнення певних цілей.

Варіанти використання зазвичай моделюються на ранній стадії (наприклад, під час збору вимог), щоб описати та відобразити, як актор може використовувати

систему. Моделі прецедентів фіксують лише те, як актор може використовувати систему, а не те, як система побудована [4].

Діаграму варіантів використання вебзастосунку планування подорожей показано на рис. 2.1.

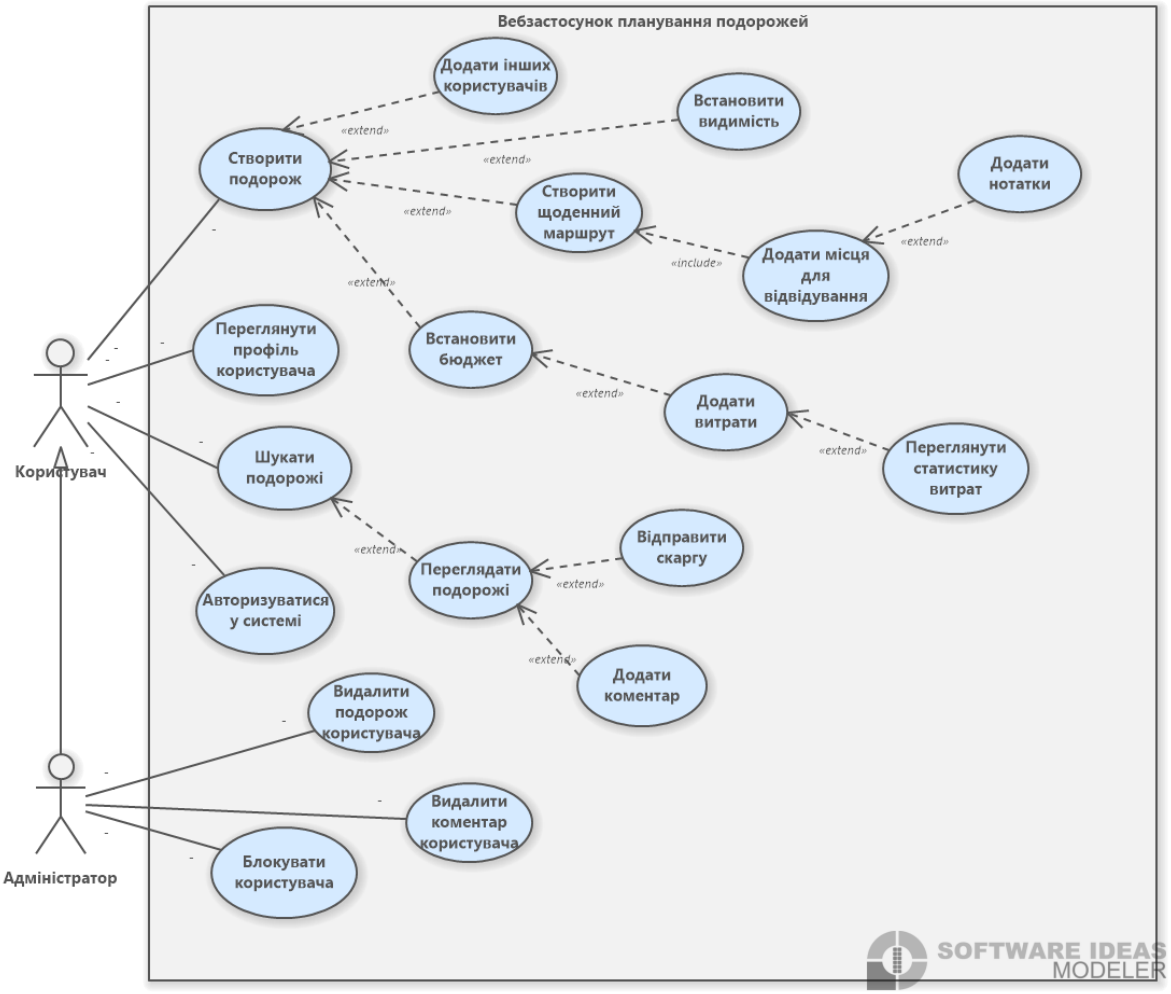

Рисунок 2.1 – Діаграма використання

Use case діаграму розроблено за допомогою інструменту Software Ideas Modeler.

Дане ПЗ має інструменти для діаграм UML, блок-схем, діаграм потоків даних та інших видів діаграм та задач, що допомагають моделювати програмне забезпечення або його ключові частини. Також ПЗ має обширну документацію з прикладами діаграм.

2.2 Алгоритми роботи вебзастосунку

Для відображення алгоритмів роботи системи, необхідно побудувати низку UML-діаграм.

Щоб показати виконання деяких сценаріїв використання, доцільно представити їх у вигляді блок-схем. У UML, альтернативою блок-схем слугує діаграма діяльності.

Діаграми діяльності використовуються для візуалізації послідовності дій або процесів в системі. У цих діаграмах дії або кроки процесу представлені у вигляді прямокутників, а лінії з’єднують ці дії, відображаючи послідовність виконання. Стрілки показують напрямок процесу.

Однією з ключових особливостей діаграм діяльності є їх здатність відобразити умови та рішення.

Умовні лінії можуть бути додані для визначення альтернативних шляхів виконання процесу в залежності від певних умов або критеріїв.

Наприклад, якщо умова виконується, то виконується одна дія, в іншому випадку – інша.

Діаграми діяльності допомагають показати послідовність кроків у процесі.

Створено три діаграми діяльності, які демонструють алгоритм деяких функцій користувача: створення подорожі, її редагування та пошук подорожей користувачів.

Створення нової подорожі (рис. 2.2)

Діаграма складається з таких кроків, як: перевірка на авторизацію, введення даних до форми, вибір: збереження шаблону подорожі (незаповненого плану) чи одразу приступити до редагування.

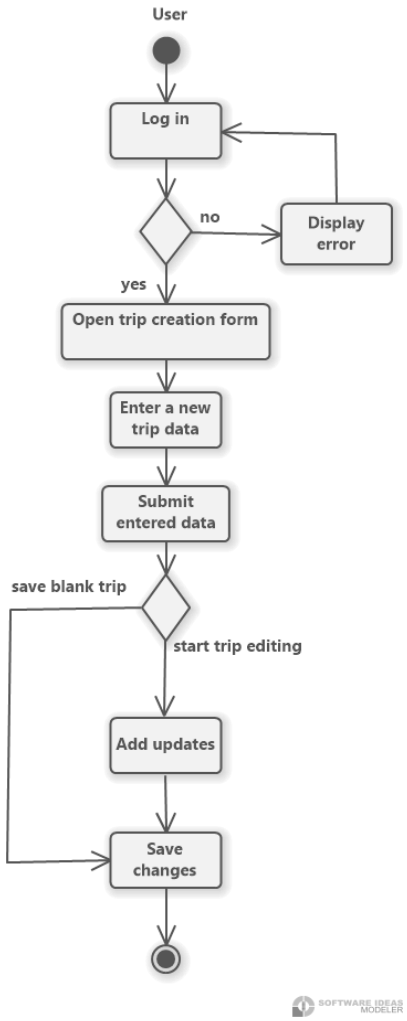

Рисунок 2.2 – Діаграма діяльності створення подорожі

Пошук та перегляд подорожей користувачів (рис. 2.3)

Діаграма складається з таких кроків, як: введення даних до форми, отримання результатів пошуку, перегляд знайденої подорожі. Якщо користувач бажає додати коментар або надіслати скаргу, він може це зробити, авторизувавшись у застосунку. Якщо за ключовими словами подорожі не знайдено, користувач може продовжити пошук до того моменту, поки бажана подорож не буде знайдена. Продемонстровано варіант діаграми діяльності пошуку подорожей для авторизованого та неавторизованого користувача.

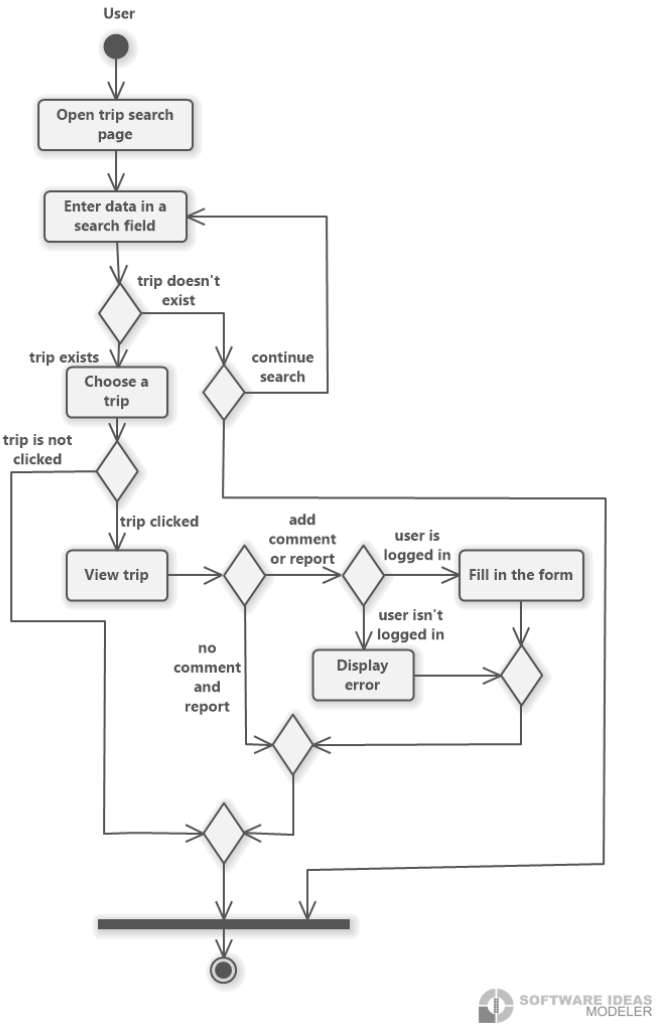

Рисунок 2.3 – Діаграма діяльності пошуку подорожей

Редагування створеної подорожі (рис. 2.4)

У цій діаграмі відсутня перевірка на авторизацію, адже створити подорож може лише авторизований користувач, відповідно редагувати подорож також.

Діаграма показує варіанти редагування подорожі: редагування даних, внесених при створенні подорожі, створення щоденного плану, додання витрат, запрошення інших користувачів до подорожі та їх видалення. До кожного дня можна додавати активності, отже етап «Додання кроку (активності в день)» може повторюватися декілька разів.

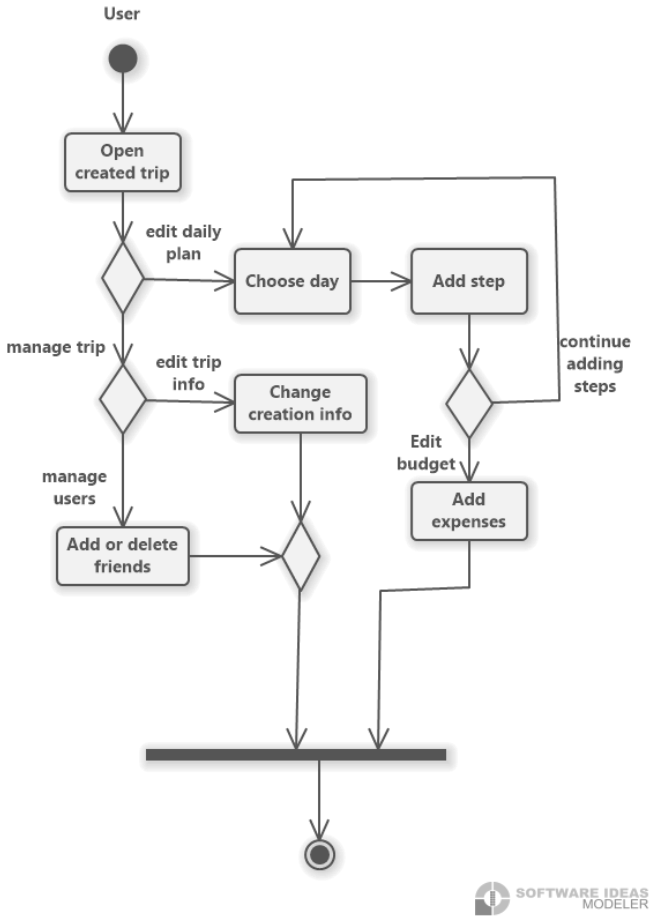

Рисунок 2.4 – Діаграма діяльності редагування подорожі

Діаграми діяльності розроблено за допомогою програмного забезпечення Software Ideas Modeler для демонстрації деяких варіантів роботи користувача з застосунком.

Діаграми взаємодії описують, як об’єкти «спілкуються» один з одним. Діаграма послідовності показує, як послідовність повідомлень надсилається та отримується між набором об’єктів для виконання певної функції. Вони зосереджені на послідовності повідомлень, тобто на тому, як повідомлення надсилаються та отримуються між кількома об’єктами. Діаграма послідовності також показує взаємодію для конкретного сценарію – певну взаємодію між об’єктами, яка відбувається в певний момент часу під час виконання системи (наприклад, коли використовується певна функція) [4]. Діаграма кооперації зосереджена на відносинах між взаємодіючими об’єктами.

На рис. 2.5 зображено загальну діаграму послідовності, що демонструє роботу застосунку. Головним актором є користувач, що взаємодіє з застосунком.

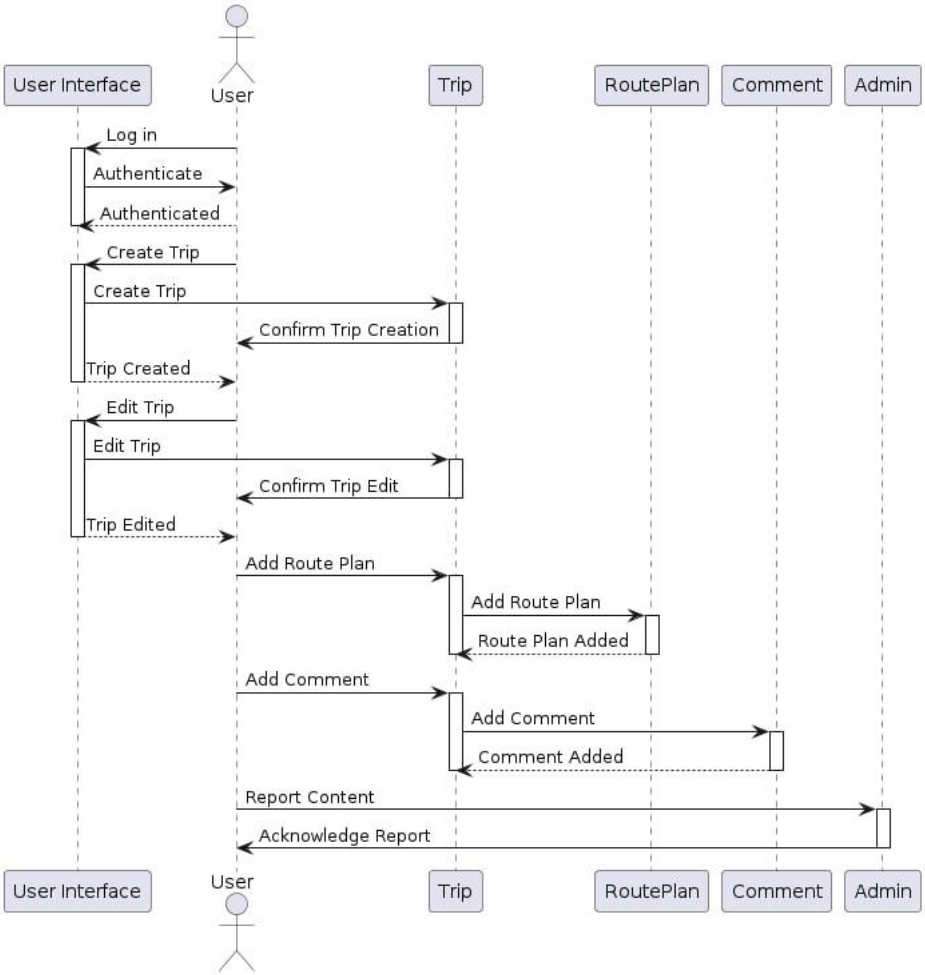

Рисунок 2.5 – Діаграма послідовності

Користувач взаємодіє з інтерфейсом вебзастосунку, через нього вносить необхідні зміни, також актором є адміністратор системи.

2.3 Діаграми станів та переходів

Діаграми станів описують, які стани може мати об’єкт протягом свого життєвого циклу, а також поведінку в цих станах разом із тим, які події викликають зміну стану [4]. На рис. 2.6 зображено загальну діаграму станів, що демонструє роботу всього застосунку, а саме сторінки, по яким може перейти користувач.

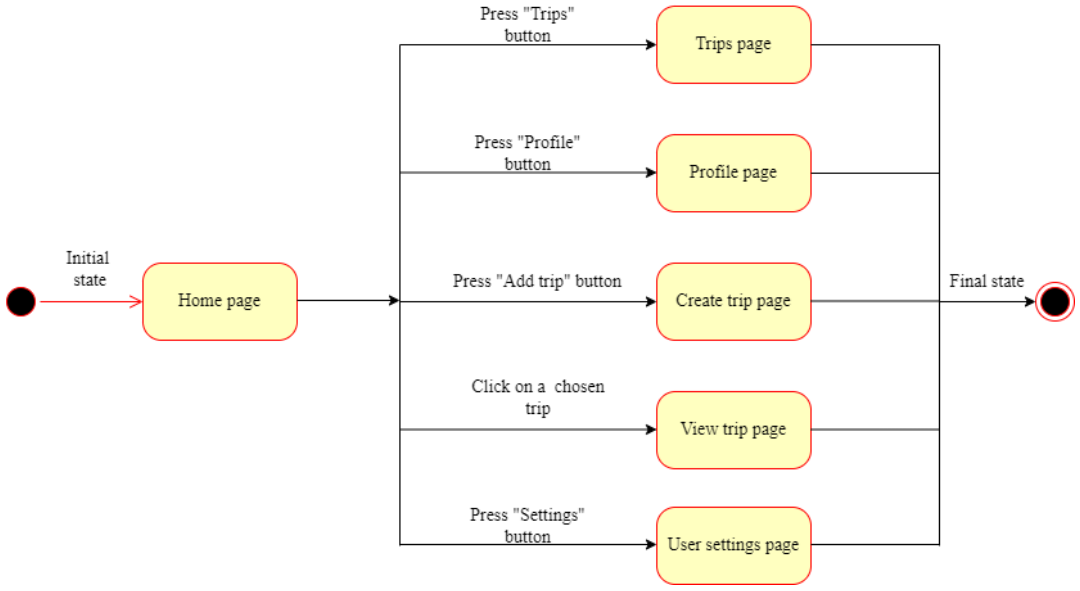

Рисунок 2.6 – Діаграма станів роботи застосунку

На рис. 2.7 зображено діаграму станів створення подорожі. Користувач може зберегти пустий шаблон подорожі, а також відредагувати подорож одразу: додати витрати, створити щоденний план.

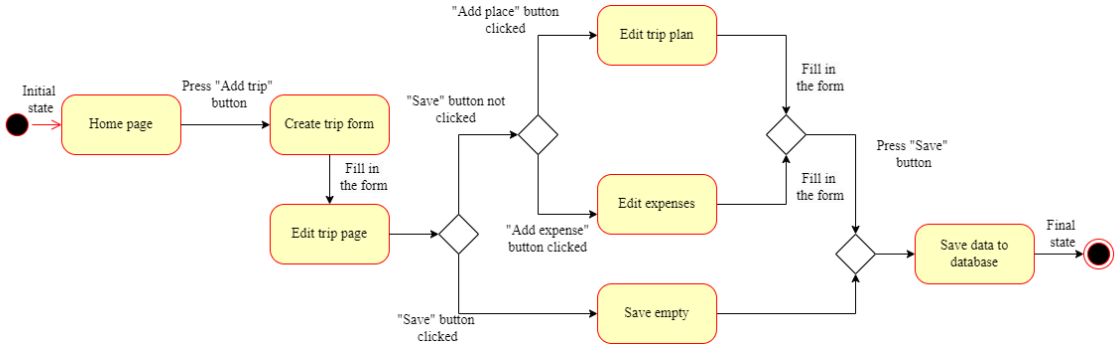

Рисунок 2.7 – Діаграма станів створення подорожі

На рис. 2.8 зображено діаграму станів додання інших користувачів до подорожі. Цю дію можна виконати як під час створення подорожі, так і при редагуванні.

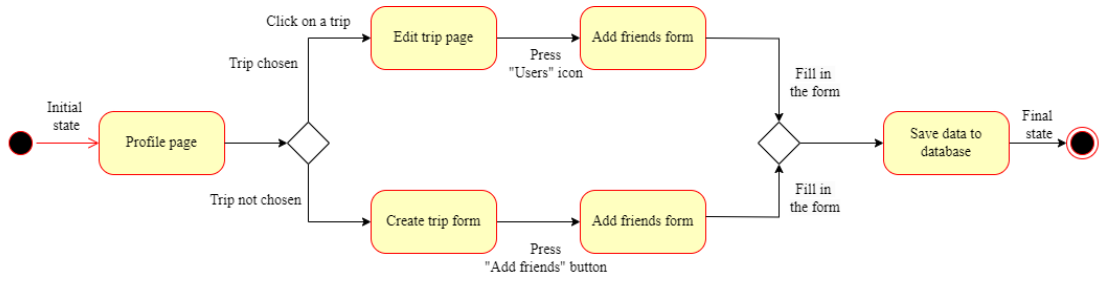

Рисунок 2.8 – Діаграма станів додання друзів до подорожі

Діаграми послідовності та станів розроблено за допомогою програмного забезпечення StarUML. За даними дослідження основних програмних продуктів, які використовуються для навчання моделюванню ПЗ, опублікованому IEEE у 2017 році, StarUML стало третім найбільш використовуваним ПЗ: 23,9% респондентів використовували його на своїх курсах. Дослідження базувалося на міжнародній вибірці зі 150 науковців [5].

2.4 Розробка мокапів вебзастосунку

Для виконання кодування вебзастосунку планування подорожей необхідно спочатку розробити його дизайн у вигляді мокапів (mock-ups).

Мокапи – це візуальні моделі або прототипи, що використовуються для представлення дизайну або функціональності майбутнього програмного забезпечення. Їх створюють на ранніх етапах проєктування для того, щоб продемонструвати, як буде виглядати та працювати кінцевий продукт.

Дизайн вебзастосунку планування подорожей розроблено у середовищі Figma.

Зараз на ринку доступно багато дизайнерських програм, які можна використовувати для вирішення будь-яких творчих завдань. Але Figma є одним із найулюбленіших інструментів багатьох дизайнерів і стає все популярнішою. На це є багато вагомих причин. По-перше, Figma дозволяє дизайнерам та іншим членам команди працювати в режимі реального часу. Завдяки цій потужній функції Figma виділяється серед інших інструментів, оскільки вона покращує не лише роботу над дизайном, але й сам процес командної співпраці. По-друге, Figma виходить за межі розробки дизайну продукту та генерує код CSS для використання розробниками [6].

На рис. 2.9 та 2.10 показано мокап головної сторінки вебзастосунку.

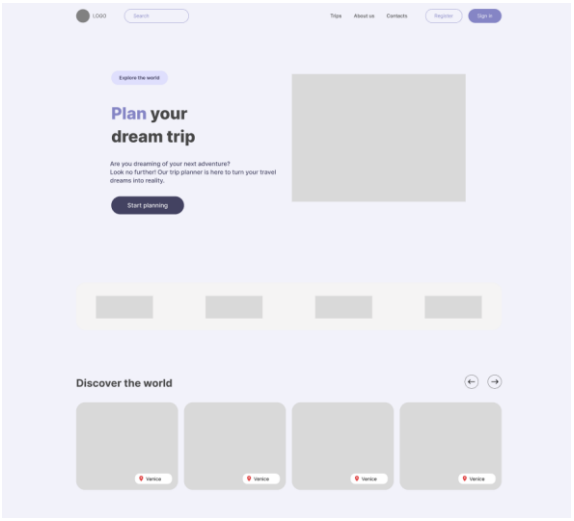

Рисунок 2.9 – Сторінка Home

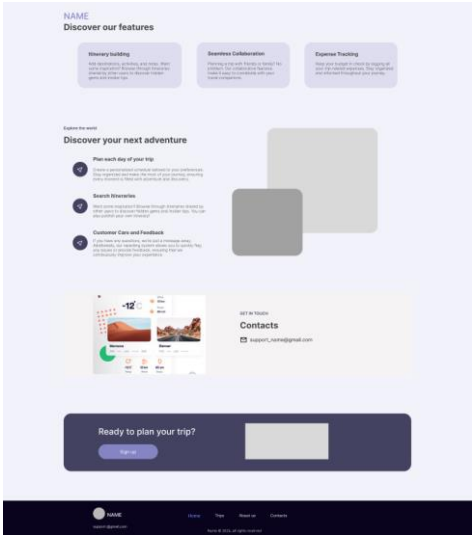

Рисунок 2.10 – Сторінка Home, продовження

Вигляд сторінки профілю користувача показано на рис. 2.11.

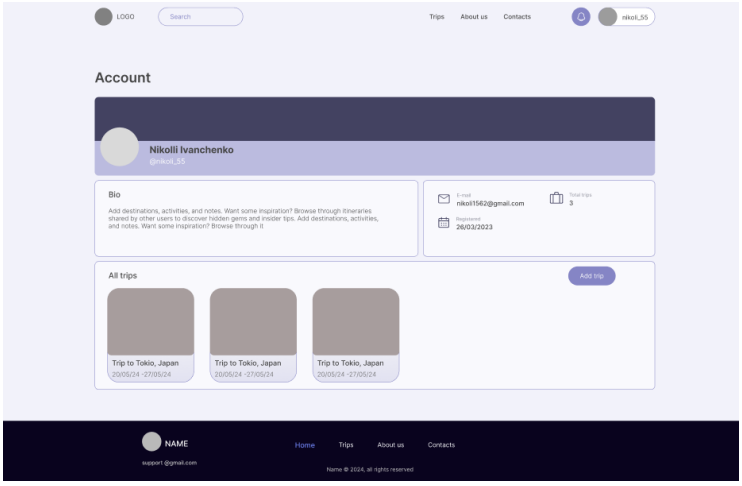

Рисунок 2.11 – Сторінка профілю користувача

Основним функціоналом вебзастосунку є планування подорожей, тому на рис. 2.12 показано дизайн форми редагування подорожі.

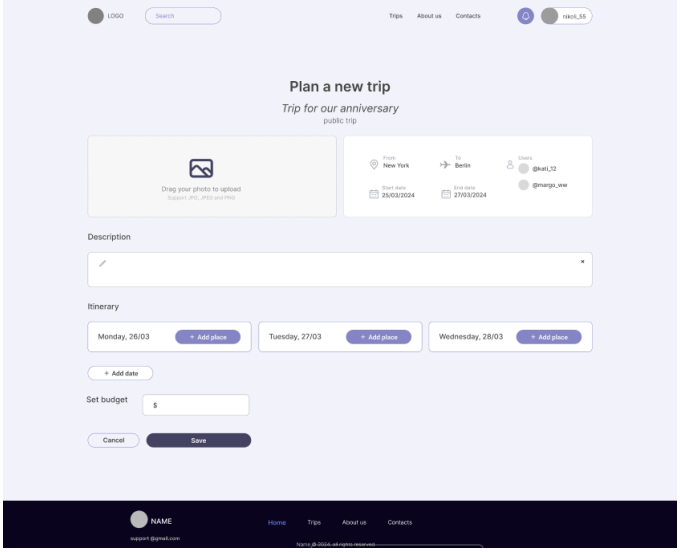

Рисунок 2.12 – Сторінка редагування подорожі

Також розроблено дизайн модальних вікон (рис. 2.13 – 2.15).

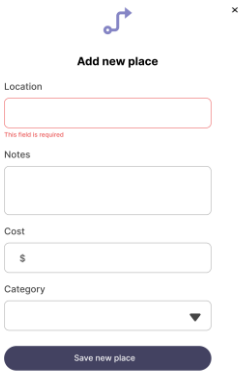

Рисунок 2.13 – Модальне вікно додання активності у денний план

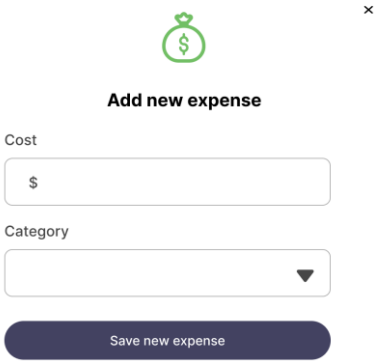

Рисунок 2.14 – Модальне вікно додання витрат

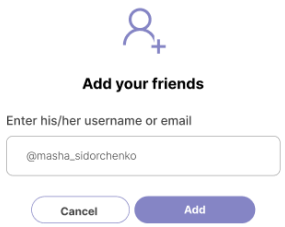

Рисунок 2.15 – Модальне вікно запрошення друзів

Дизайн адмін-панелі показано на рис. 2.16.

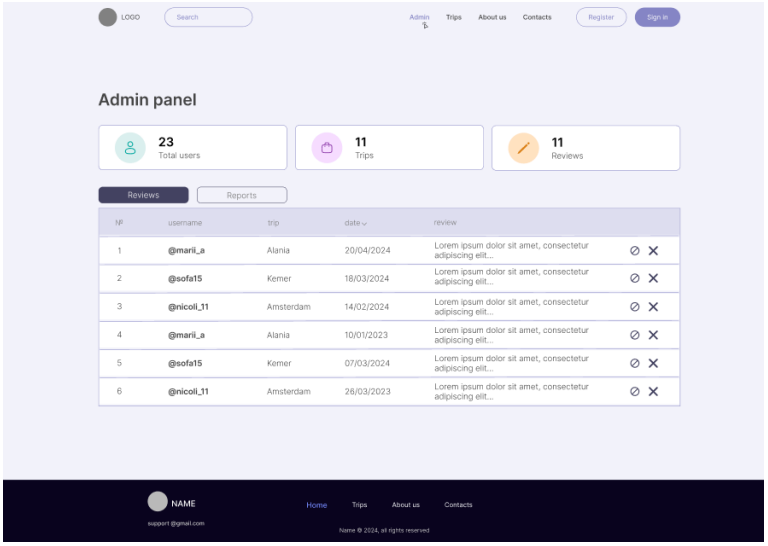

Рисунок 2.16 – Адмін-панель

Отже, продемонстровано основні мокапи вебзастосунку, що розробляється. Деякі аспекти буде розглянуто окремо при розгляді керівництва користувача у розділі 4.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра виконано моделювання вебзастосунку планування подорожей. Після розробки сценаріїв використання у короткій, поверхневій, та повній формах побудовано діаграму варіантів використання системи.

Для демонстрації алгоритмів роботи програмного забезпечення, що розробляється, наведено декілька типів UML-діаграм: діаграми послідовності, станів та діяльності. Також показано дизайн деяких сторінок вебзастосунку у вигляді мокапів.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Для кодування програмного забезпечення треба обрати мови програмування, фреймворки/бібліотеки, БД, що підійдуть найкраще для вирішення поставлених завдань та розроблених вимог до програмного забезпечення, та обґрунтувати їх вибір. Також необхідно побудувати діаграму класів, що покаже структуру та організацію програмного коду.

3.1 Огляд технологій

Мова програмування PHP призначена для веброзробки, і є потужним інструментом для створення динамічних та інтерактивних вебсторінок.

Сама мова є надзвичайно гнучкою. Наприклад, відсутнє обмеження виведенням лише HTML або інших текстових файлів – можна створити будь-який формат документа. PHP має вбудовану підтримку для створення файлів зображень та PDF-файлів. Однією з найважливіших особливостей PHP є широка підтримка баз даних. PHP підтримує всі основні бази даних (включаючи MySQL, PostgreSQL, Oracle тощо). Також підтримуються бази даних моделі NoSQL, наприклад MongoDB. За допомогою PHP створювати вебсторінки з динамічним вмістом із бази даних надзвичайно просто. Також PHP надає бібліотеку коду для виконання типових завдань, таких як абстракція бази даних, обробка помилок тощо [7].

Отже, перевагами мови PHP є:

– сумісність з різними платформами (Windows, Linux, Mac OS X тощо);

– сумісність з майже з усіма серверами, які використовуються сьогодні (Apache тощо);

– підтримка багатьох баз даних;

– простота у вивченні.

Під час розробки за допомогою чистого PHP бізнес-логіка змішується із запитами до бази даних. Через поєднання такого режиму розробки ускладнюється обслуговування та масштабованість програми. Для вирішення цієї проблеми PHP пропонує різні фреймворки для розробки вебзастосунків.

Фреймворки PHP допомагають розробникам створювати вебзастосунки швидше та легше, надаючи базову модель фреймворку, а також повний набір API, бібліотек і розширень, а також допомагають розробникам підвищити продуктивність, зменшивши кількість повторюваного коду в проєкті [8].

Популярними фреймворками PHP є:

– Symfony;
– CodeIgniter;
– Laravel;
– CakePHP;
– Yii тощо.

Кожен із цих фреймворків має свої переваги та недоліки, і вибір найкращого PHP-фреймворку для розробки певного проєкту залежить від вимог до ПЗ.

Відповідно до порівняльного дослідження продуктивності фреймворків PHP [9], в якому порівняно Laravel, Symfony та CodeIgniter, визначено, що при перевірці продуктивності за критерієм кількості оброблених запитів в секунду, Laravel здатний обробляти 3000 запитів на секунду (найвищий результат) порівняно з іншими зазначеними фреймворками.

За критерієм обсягу пам’яті, яка використовується для відображення вебсторінки, визначено, що Laravel використовує приблизно 518 кБ порівняно з CodeIgniter, чий показник є трохи вищим, та останнім йде Symfony з пам’яттю приблизно 1711 кБ.

Також обраховано, що Laravel має найменший час відгуку (це один з найважливіших критеріїв для оцінки продуктивності патерну Model-View-Controller, далі – MVC): 4,46 мілісекунди (далі – мс), порівняно з CodeIgniter з 7,2 мс, а потім Symfony з 12 мс.

За кількістю файлів, необхідних для завантаження вебсторінки, CodeIgniter має найменшу кількість файлів – 22, потім Symfony з 15 файлами, і останнім є Laravel з 26 файлами.

Після оцінки продуктивності трьох фреймворків: Laravel, Symfony та CodeIgniter за такими критеріями, як: кількість оброблених запитів за секунду,

використання пам’яті та час відповіді, отримані результати свідчать проте, що Laravel перевершує інші PHP-фреймворки, що використовують патерн MVC.

За даними Google Trends [10], за останні 12 місяців (до травня 2024 року), пошуковий запит «Laravel» користується найвищою популярністю (по світу) серед наведених вище п’яти PHP-фреймворків (рис. 3.1).

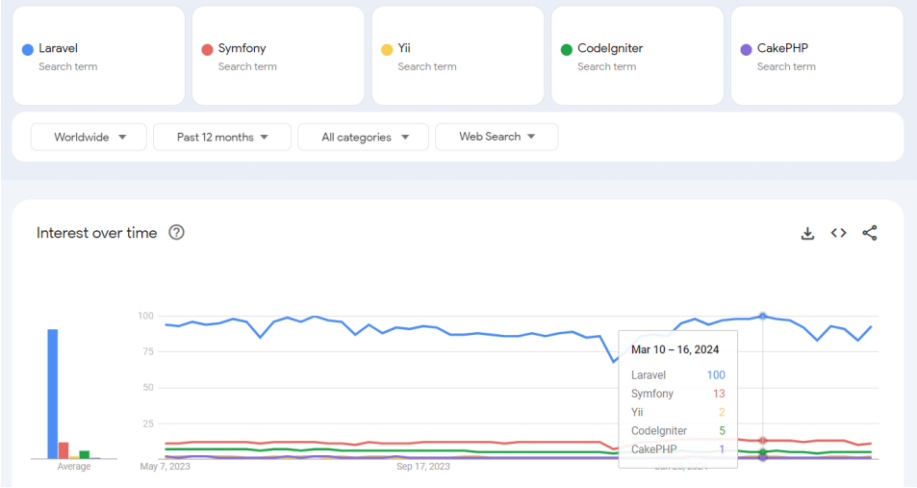

Рисунок 3.1 – Пошукові запити PHP-фреймворків від Google Trends, світ

Схожою є і статистика в Україні (за такими ж критеріями), де Laravel теж переважає кількісно (рис. 3.2).

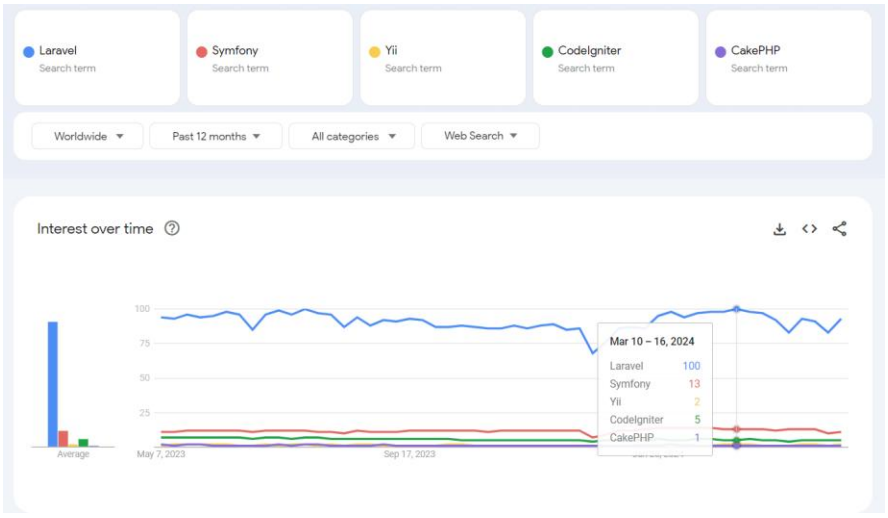

Рисунок 3.2 – Пошукові запити PHP-фреймворків від Google Trends, Україна

Перевагами Laravel є:

– обширна та якісна документація;

– легкість у вивченні;
– відкритий код, фреймворк є повністю безкоштовним.

Отже, серед PHP-фреймворків для розробки обрано Laravel.

Для зберігання інформації вебзастосунку необхідно обрати базу даних.

Традиційні системи БД базуються на реляційній моделі. Вони широко відомі як бази даних SQL. На противагу реляційним БД виступають нереляційні бази даних, що відомі як бази даних NoSQL. Більшість із останніх базується на зберіганні простих пар ключ-значення на передумові, що простота веде до швидкості [11].

Реляційна модель і модель NoSQL (нереляційна) мають переваги та недоліки в певному контексті.

Нереляційні БД непридатні для складних запитів із кількома об’єктами, оскільки більшість не підтримують зв’язки, зовнішні ключі та оператори JOIN. Вони не використовують мову SQL для запитів до бази даних за допомогою об’єктно-орієнтованих API, а останнім часом і мов, схожих на SQL. Бази даних NoSQL призначені для великих розподілених вебзастосунків, які дозволяють керувати та аналізувати величезні обсяги даних із високою швидкістю їх обробки. Реляційні бази даних є зручним інструментом для управління структурованими даними. Вони особливо корисні при створенні застосунків, які ґрунтуються на зв’язках між різними наборами даних. Реляційні БД також допомагають забезпечити узгодженість даних у таблицях [12].

Також, оскільки реляційні БД використовують мову SQL, можна легко змінювати СКБД у разі необхідності, тому що код буде майже однаковим.

Отже, у цьому проєкті вирішено використовувати саме реляційну модель баз даних.

Типовими прикладами реляційних СКБД є MySQL, Microsoft SQL Server, PostgreSQL, MariaDB тощо.

Повертаючись до аналізу пошукових запитів від Google Trends, на рис. 3.3 можна побачити, що за світовою статистикою запит «MySQL» у категорії «System

software» серед подібних реляційних СКБД є найбільш популярним (травень 2023 – травень 2024) [13].

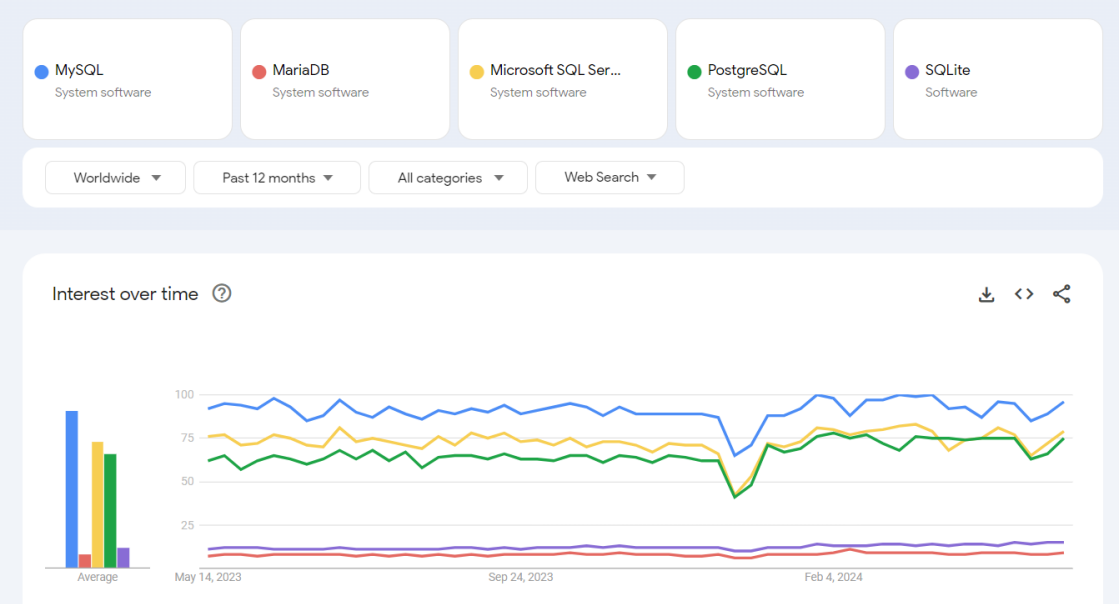

Рисунок 3.3 – Пошукові запити реляційних СКБД від Google Trends, світ

Результати дослідження порівняння продуктивності СКБД MySQL та MariaDB [14] показали, що MySQL має вищу продуктивність, ніж MariaDB, у початковій серії тестів у транзакціях за секунду.

При порівнянні продуктивності MySQL, Microsoft SQL Server та PostgreSQL за допомогою вебзастосунку та фреймворку Laravel [15], результати, отримані для однакових структур БД, відрізнялися залежно від виконуваних операцій з БД. MySQL працює швидше з командами читання, тоді як PostgreSQL краще працює з операціями читання-запису, масивними наборами даних і складними запитами. Результати, досягнуті SQL Server у більшості перевірених операцій значно відрізняються від результатів конкурентів: отримано найгірші показники. Як зазначають автори, причиною цього є технічні показники середовища тестування і можна припустити, що з більш ефективним апаратним забезпеченням у своєму розпорядженні SQL Server отримає результати, більш подібні до результатів своїх конкурентів.

Також, MySQL – суто реляційна база даних, тоді як PostgreSQL – об’єктно-реляційна база даних.

PostgreSQL пропонує більш складні типи даних і дозволяє об’єктам успадковувати властивості [16].

Розробка вебзастосунку планування подорожей не потребує складних налаштувань у БД, тому СКБД PostgreSQL не буде використано для його розробки.

Отже, для розробки вебзастосунку планування подорожей обрано базу даних MySQL.

Для стилізації та адаптивності буде використано CSS-фреймворк Tailwind CSS. За допомогою Tailwind можна стилізувати елементи, застосовуючи попередньо визначені класи в HTML-тегах. Фреймворк пропонує широкі можливості налаштування для адаптації стилів відповідно до конкретних потреб.

На рис. 3.4 зображено стек використаних технологій (логотипи).

Рисунок 3.4 – Стек технологій

Для кодування ПЗ використано IDE PhpStorm.

Редактор коду PhpStorm є потужним помічником для створення коду. Основними функціями редактору є можливість рефакторингу, миттєве виявлення помилок, типізація, швидка та проста навігація за кодовою базою, можливість тестування; також у дане IDE вбудовано помічник зі штучним інтелектом, який може пояснити код, запропонувати рефакторинг, згенерувати юніт-тести, код, конвертувати файл у іншу мову програмування та інші функції. PhpStorm підтримує найпопулярніші фреймворки PHP, включно з Laravel. PhpStorm надає

значну допомогу в кодуванні та підтримку навігації для обраних фреймворків розробки. Окрім PHP, PhpStorm підтримує розробку на JavaScript, TypeScript, jQuery та інших основних frontend-технологіях [17].

3.2 Патерн MVC у Laravel

Model-View-Controller (MVC) – це патерн проєктування, який використовується при розробці вебзастосунків.

Він розділяє програмне забезпечення на три взаємопов'язані частини: рівень даних, рівень презентації та користувача, щоб відокремити внутрішнє представлення інформації від способів, якими інформація подається або приймається користувачем. Центральний компонент, модель (model), складається з даних програми, бізнес-правил, логіки та функцій. View (подання) може бути будь-яким вихідним представленням інформації. MVC архітектуру використовують різні мови програмування у фреймворках: ASP.NET, CakePHP, Laravel, Ruby on Rails, Spring MVC тощо [18].

Основні компоненти архітектури MVC показано на рис. 3.5.

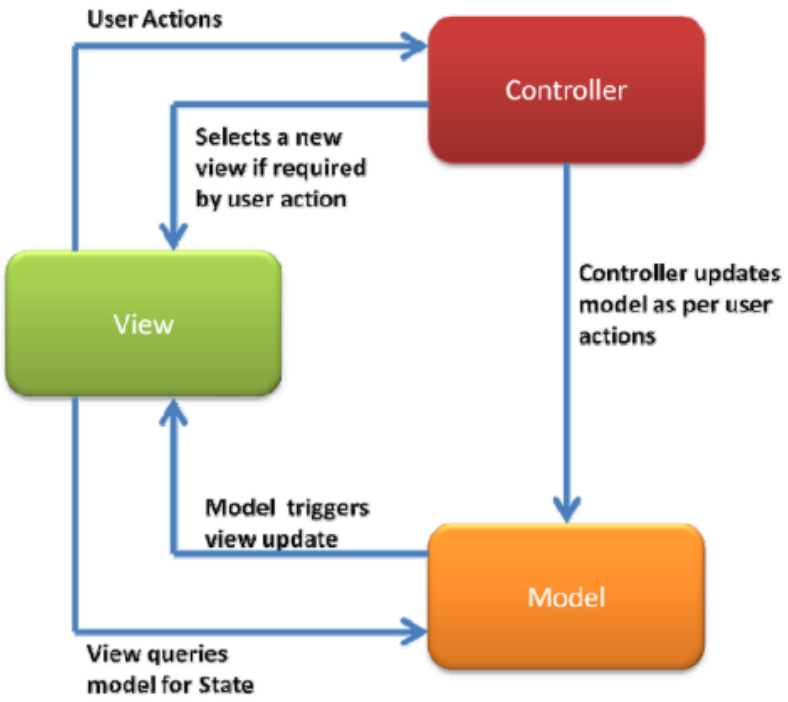

Рисунок 3.5 – Робота архітектури MVC

Отже, кожен з компонентів MVC відповідає за:

– model – це компонент, який взаємодіє з базою даних для обробки даних, логіки та правил;

– view – утворює частину, яка взаємодіє з користувачем, відображаючи вихідні дані та приймаючи вхідні дані в різних формах;

– controller – надсилає команди до моделі для оновлення даних, а також надсилає команди для перегляду для зміни даних, які приймаються або відображаються.

Laravel є широко використовуваним фреймворком MVC, який відповідає сучасним практикам розробки PHP. Він пропонує такі функції, як маршрутизація, ORM (об’єктно-реляційне відображення), кешування та автентифікація. Eloquent ORM, який є частиною Laravel, використовує конструктор запитів і підтримує складні зв’язки таблиць. Laravel має надійну систему маршрутизації; дозволяє застосовувати фільтри та дії до запитів [19].

Під час використання Eloquent кожна таблиця бази даних має відповідну «модель», яка використовується для взаємодії з цією таблицею. Окрім отримання записів із таблиці БД, моделі Eloquent також дозволяють вставляти, оновлювати та видаляти записи з таблиці.

Eloquent спрощує керування зв’язками між таблицями та роботу з ними, і підтримує низку загальних зв’язків:

1) one to one;
2) one to many;
3) many to many;
4) has one through;
5) has many through;
6) one to one (поліморфний);
7) one to many (поліморфний);
8) many to many (поліморфний) [20].

Зв’язки Eloquent визначаються як методи у класах моделей. Контролер у Laravel може мати будь-яку кількість відкритих методів, які відповідатимуть на

вхідні HTTP-запити. Шаблони перегляду (view) зазвичай пишуться з використанням мови шаблонів Blade. Blade – це механізм створення шаблонів, який входить до складу Laravel. На відміну від деяких механізмів створення шаблонів PHP, Blade не обмежує використання простого коду PHP у шаблонах. Файли шаблонів Blade використовують розширення файлу .blade.php.

3.3 Розробка діаграми класів

У будь-якій об’єктно-орієнтованій реалізації код організований у класи. Таким чином, діаграма класів є оглядом структури коду та його внутрішніх зв’язків.

На рис. 3.6 продемонстровано діаграму класів вебзастосунку.

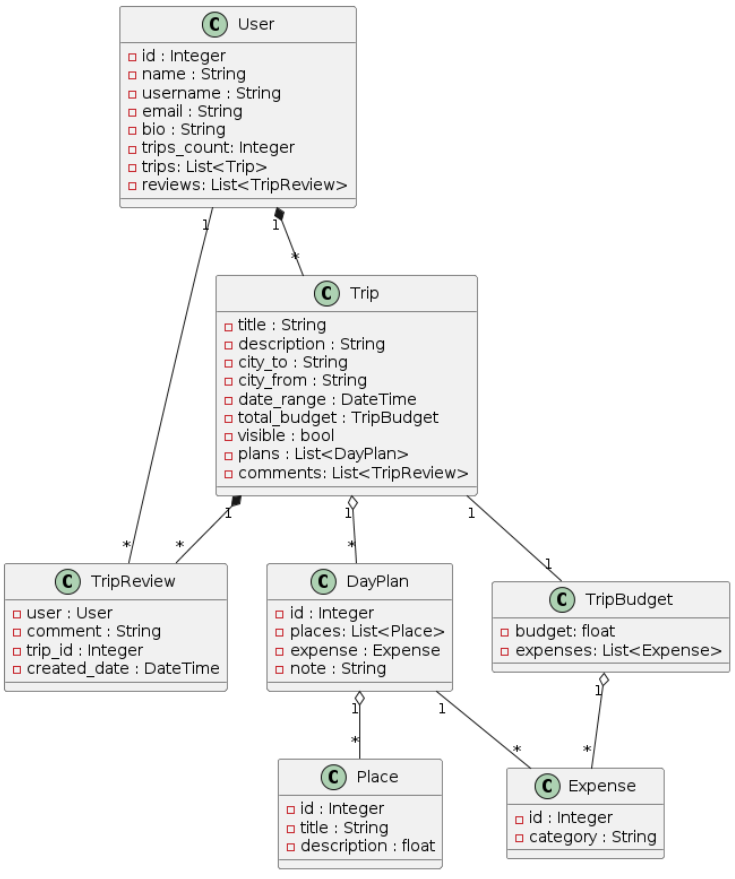

Рисунок 3.6 – Діаграма класів

Діаграма класів часто представляє неповне уявлення про всю систему. Іноді ілюструються лише ті методи та атрибути, які є корисними для представлення цього завдання. Деякі класи або зв’язки можуть бути відсутніми.

Ця діаграма класів демонструє відношення між створеними моделями Eloquent. На діаграмі класів, поданій вище, відсутні стандартні класи контролерів фреймворку Laravel та виключено деякі моделі для кращого розуміння структури коду. На діаграмі також відсутні методи, тому що у Laravel моделі взаємодіють через інші вбудовані елементи фреймворку.

Таблиця 3.1 – Опис діаграми класів

Клас	**Основні атрибути**	**Призначення**
User	username	Клас користувача для створення та редагування подорожей
	name	
	password	
	email	
	trips	
	reviews	
Trip	title	Клас створення та редагування подорожі
	plans	
	comments	
	date_range	
DayPlan	places	Клас редагування щоденного плану
	notes	
	expenses	
	id	
TripReview	user	Клас створення коментаря
	comment	
TripBudget	budget	Клас бюджету подорожі
	expenses	

Класи Place та Expense слугують для формування плану на день (клас DayPlan) та бюджету подорожі (TripBudget).

Взаємодію між наведеними моделями забезпечено контролерами Laravel, що не продемонстровані на діаграмі класів. Також на діаграмі класів відсутні класи, що забезпечують адміністрування системою (через адмін-панель у вебзастосунку).

3.4 Проєктування бази даних

Для вебзастосунку планування подорожей необхідно створити таблиці бази даних, у яких буде зберігатися інформація. Декілька таблиць, що мають бути створені для керування базою даних:

– user;
– trip;
– trip_day;
– trip_review;
– trip_owner;
– trip_day_place;
– trip_budget;
– city;
– place тощо.

Фізична модель бази даних показана на рисунку 3.7.

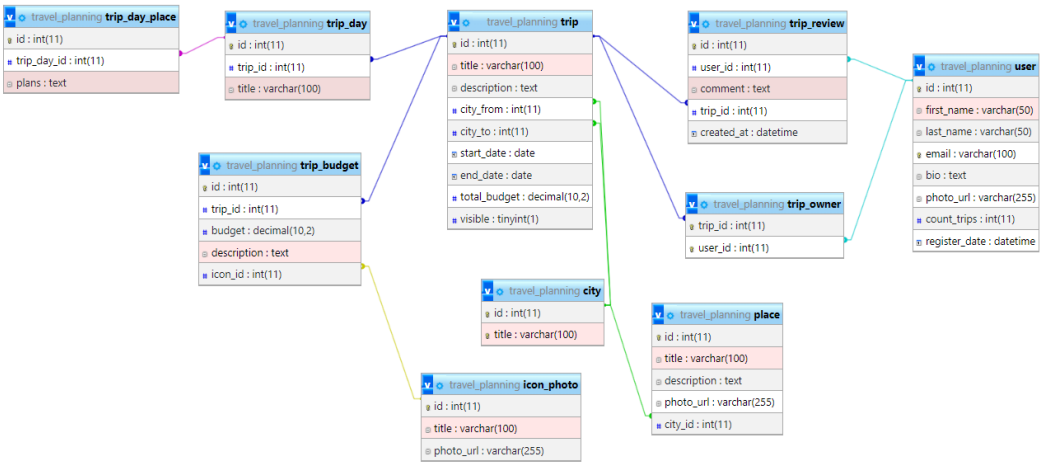

Рисунок 3.7 – Фізична модель бази даних

Таблиця user зберігає інформацію про користувачів, які зареєстровані у вебзастосунку:

– id: унікальний ідентифікатор користувача (primary key);
– first_name: ім’я користувача;
– username;
– email: електронна адреса користувача, використовується для входу;
– bio: біографія користувача, додаткова інформація про нього;
– password: пароль;
– photo_url: посилання до фотографії користувача;
– count_trips: лічильник кількості подорожей, створених користувачем;
– register_date: дата реєстрації користувача.

Таблиця city містить інформацію про міста, які є пунктами відправлення або призначення в подорожах:

– id: унікальний ідентифікатор міста;
– title: назва міста.

Таблиця place зберігає інформацію про місця, які можна відвідати під час подорожі:

– id: унікальний ідентифікатор місця;
– title: назва місця;
– description: опис місця;
– photo_url: посилання до фотографії місця;
– city_id: ідентифікатор міста, де розташоване місце (зв’язаний зовнішнім ключем з таблицею city).

Таблиця trip зберігає інформацію про подорожі, створені користувачами:

– id: унікальний ідентифікатор подорожі;
– title: назва подорожі;
– description: опис подорожі;
– city_from: ідентифікатор міста відправлення (зв’язаний зовнішнім ключем з таблицею city);
– city_to: ідентифікатор міста призначення (зв’язаний зовнішнім ключем з таблицею city);
– start_date: дата початку подорожі;

– end_date: дата завершення подорожі;

– total_budget: загальний бюджет подорожі;

– visible: прапорець видимості подорожі.

Таблиця trip_owner відображає зв’язок між подорожами та їх власниками:

– trip_id: ідентифікатор подорожі (зв’язаний зовнішнім ключем з таблицею trip);

– user_id: ідентифікатор користувача, власника подорожі (зв’язаний зовнішнім ключем з таблицею user).

Таблиця trip_day зберігає інформацію про окремі дні в рамках однієї подорожі.

Таблиця trip_day_place зберігає інформацію про плани на певні місця відвідування у певний день подорожі:

– id: унікальний ідентифікатор запису;

– trip_day_id: ідентифікатор дня подорожі (зв’язаний зовнішнім ключем з таблицею trip_day);

– plans: опис планів або діяльності на цей день.

Таблиця trip_budget зберігає інформацію про бюджет та витрати, пов’язані з подорожжю:

– id: унікальний ідентифікатор запису;

– trip_id: ідентифікатор подорожі (зв’язаний зовнішнім ключем з таблицею trip);

– budget: сума бюджету або витрат;

– description: опис витрат або категорії бюджету;

– icon_id: ідентифікатор іконки (зв’язаний зовнішнім ключем з таблицею icon_photo).

Таблиця trip_review зберігає відгуки користувачів про подорожі:

– id: унікальний ідентифікатор відгуку;

– user_id: ідентифікатор користувача, який залишив відгук (зв’язаний зовнішнім ключем з таблицею user);

– comment: текст відгуку;

– trip_id: ідентифікатор подорожі, до якої залишено відгук (зв’язаний зовнішнім ключем з таблицею trip);

– created_at: дата та час створення відгуку.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи бакалавра виконано проєктування вебзастосунку планування подорожей.

Проведено порівняльний аналіз актуальних та популярних PHP-фреймворків, баз даних, в результаті якого визначено стек технологій для розробки вебзастосунку планування подорожей.

Оглянуто обрані технології для розробки: мову програмування PHP, фреймворк Laravel, CSS-фреймворк Tailwind CSS для забезпечення адаптивності сторінок вебзастосунку та базу даних MySQL для зберігання інформації.

Описано IDE PhpStorm, у якому буде виконано розробку ПЗ. Розглянуто патерн MVC як основу Laravel.

Надано короткий опис системи об’єктно-реляційного відображення Eloquent для роботи з базами даних у Laravel.

Закріплено навички з розробки діаграми класів. До створеної діаграми класів додано її опис.

Показано фізичну модель бази даних, що демонструє перелік розроблених таблиць системи, надано детальний опис деяких таблиць.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Вебзастосунок планування подорожей розроблено для звичайного користувача та адміністратора системи. Користувачу доступні такі сторінки, як: реєстрація, авторизація, профіль, головна, редагування профілю, відновлення паролю, створення подорожі, редагування подорожі, пошук та їх перегляд. Для адміністратора розроблено адмін-панель, також йому доступні всі сторінки, що й користувачу.

4.1 Реалізація програмних компонентів ПЗ

Одним із головних програмних компонентів вебзастосунку є авторизація та реєстрація, що забезпечить керування користувачами. У Laravel проєкті реєстрація та авторизація реалізована відповідними контролерами:

– AuthController;
– RegisterController;
– UserController.

Для зберігання даних користувача у БД створено модель User (рис. 4.1). UserController взаємодіє з моделлю User для збереження користувачів у БД.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $table = 'user';
    protected $fillable = ['name', 'username', 'email', 'bio', 'photo_url','password', 'count_trips',
'register_date'];

    public function trips()
    {
        return $this->hasMany(Trip::class, 'trip_id');
    }

    public function tripReviews()
    {
        return $this->hasMany(TripReview::class, 'user_id');
    }
}
```

Рисунок 4.1 – Модель User

Також, коли гість потрапляє у вебзастосунок, він бачить перед собою головну сторінку.

На головній сторінці сайту має бути розміщено ключову інформацію, яка допоможе користувачам зрозуміти призначення застосунку, знайти необхідну інформацію та контакти власників. На рис. 4.2 зображено код файлу home.blade.php, що є головною сторінкою вебзастосунку.

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    @vite('resources/css/app.css')
    <title>@yield('title')</title>
</head>
<body>
    @include('partials.header')
    @yield('content')

    @include('partials.footer')
    @vite('resources/js/app.js')

</body>
</html>
```

Рисунок 4.2 – Код сторінки Home

Окремими компонентами є footer та header. Вони показані на рис. 4.3 та 4.4 відповідно. Адаптивність у цих компонентах забезпечена класами Tailwind CSS, так само і на інших сторінках вебзастосунку.

```
<header class="flex flex-row justify-between">
    <div class="logo flex flex-row justify-around">
        <img src="{{asset('public/logo.png')}}" alt="">
        <input class="rounded-2xl border border-purple-500" type="text">
    </div>
    <div class="flex flex-row">
        <div class="flex flex-row justify-evenly">
            <a href="{{route('trips')}}">Trips</a>
            <a href="{{route('about')}}">About us</a>
            <a href="{{route('contact')}}">Contacts</a>
        </div>
        <div class="flex flex-row justify-evenly">
            <a class="rounded-2xl border border-purple-500 text-purple-500"
href="">Register</a>
            <a class="rounded-2xl border border-purple-600 bg-purple-500
text-white" href="">Sign in</a>
        </div>
    </div>
</header>
```

Рисунок 4.3 – Код компоненту Header

```
<footer class="bg-[#09031E] text-white">
    <div class="w-4/6 flex flex-row justify-around">
        <div class="flex flex-col justify-between">
            <img src="{{asset('public/logo.png')}}" alt="">
            <p>support_name@gmail.com</p>
        </div>
        <div class="flex flex-col justify-between">
            <div class="flex flex-row justify-evenly">
                <a href="{{route('main')}}">Home</a>
                <a href="{{route('trips')}}">Trips</a>
                <a href="{{route('about')}}">About us</a>
                <a href="{{route('contact')}}">Contacts</a>
            </div>
            <p>Name @ 2024, all rights reserved</p>
        </div>
    </div>
</footer>
```

Рисунок 4.4 – Код компоненту Footer

Оскільки основними функціями вебзастосунку є створення та редагування подорожей, доцільно продемонструвати контролери, що відповідають за цей функціонал. Такими контролерами є:

– TripController;
– PlaceController;
– TripReviewController;
– TripBudgetController;
– TripDayController тощо.

Контролери виконують CRUD-операції, отже мають схожу структуру та методи, тому буде продемонстровано лише контролер щоденного плану подорожі (рис. 4.5).

```
class TripDayController extends Controller
{
    public function index()
    {
        $tripDays = TripDay::all();
        return view('trip_days.index', compact('tripDays'));
    }

    public function create()
    {
        return view('trip_days.create');
    }

    public function store(Request $request)
    {
        TripDay::create($request->all());
        return redirect()->route('trip_days.index');
    }

    public function show(TripDay $tripDay)
    {
        return view('trip_days.show', compact('tripDay'));
    }
```

Рисунок 4.5 – Частина коду TripDayController

Також для підключення додаткових бібліотек та фреймворків у проєкті наявні деякі config-файли, наприклад файл підключення фреймворку Tailwind (рис. 4.6).

```
/** @type {import('tailwindcss').Config} */
export default {
    content: [
        "./resources/**/*.blade.php",
        "./resources/**/*.js",
        "./resources/**/*.vue",
    ],
    theme: {
        extend: {},
    },
    plugins: [],
}
```

Рисунок 4.6 – Налаштування Tailwind

4.2 Розробка керівництва користувача

Гість системи потрапляє у вебзастосунок через головну сторінку – Home. На сторінці можна почати планувати подорож (для цього необхідно авторизуватися) за допомогою кнопки «Start planning» (рис. 4.7).

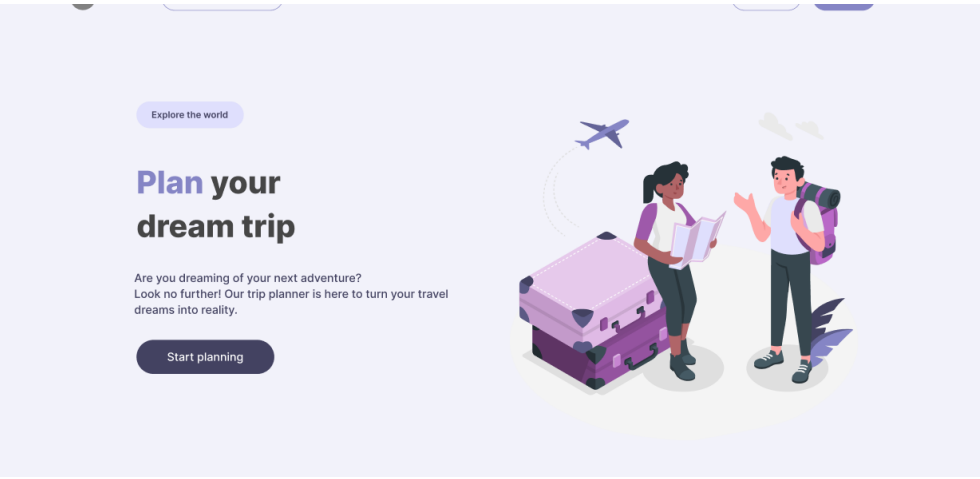

Рисунок 4.7 – Головна сторінка

Також на головній сторінці можна зареєструватися через кнопки у хедері та на сторінці нижче (рис. 4.8).

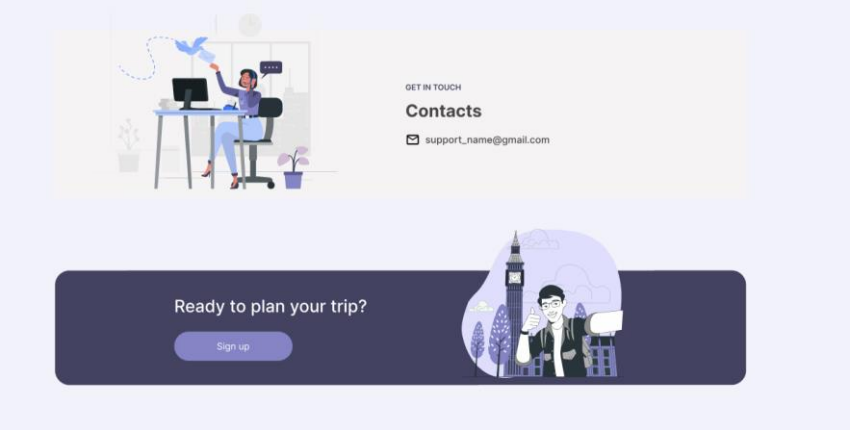

Рисунок 4.8 – Кнопка реєстрації на головній сторінці

Також на сторінці «Home» описано функціонал застосунку, контакти підтримки у разі виникнення запитань: є посилання на ці блоки у хедері: «About us», «Contacts».

На сторінці реєстрації користувачу необхідно ввести такі дані, як: ім’я, ім’я користувача (username), електронну пошту, пароль та підтвердження паролю (рис. 4.9).

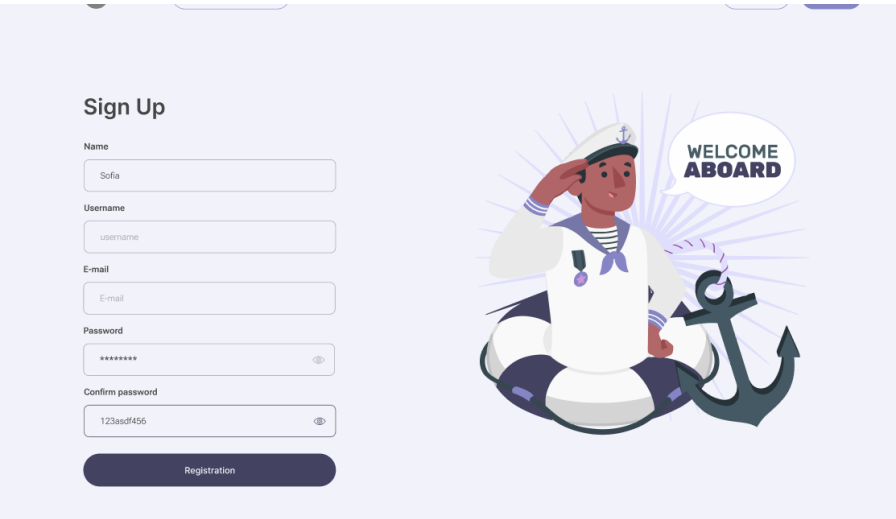

Рисунок 4.9 – Реєстрація

Для зареєстрованого користувача застосунку є форма входу (рис. 4.10), також з посиланням у хедері. У формі авторизації необхідно ввести електронну пошту та пароль.

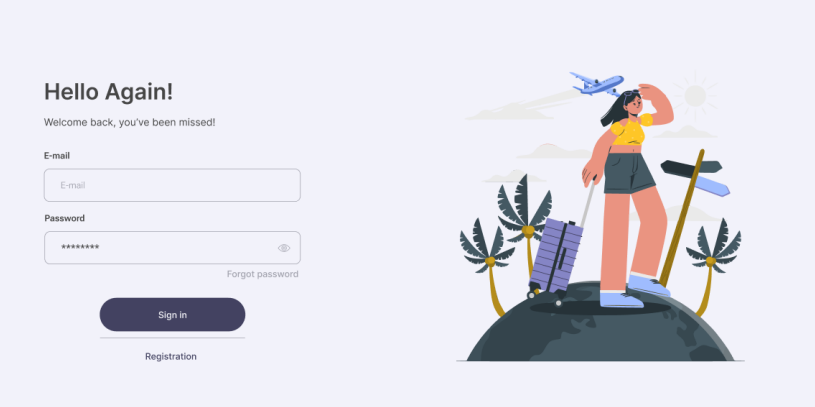

Рисунок 4.10 – Авторизація у системі

Якщо користувач забув пароль, можна натиснути на кнопку «Forgot password», що відкриє форму введення пошти для надсилання інструкцій по відновленню пароля (рис. 4.11).

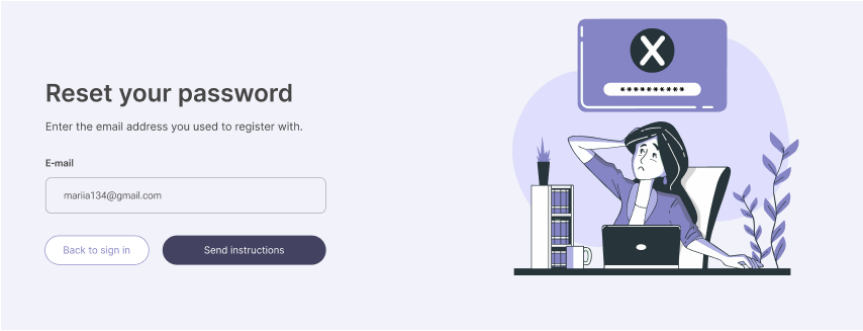

Рисунок 4.11 – Форма «Reset password»

Проте, пароль можна змінити, якщо користувач пам’ятає його: для цього необхідно перейти на сторінку редагування профілю. Після натиснення кнопки «Change password», відкриється відповідна сторінка (рис. 4.12).

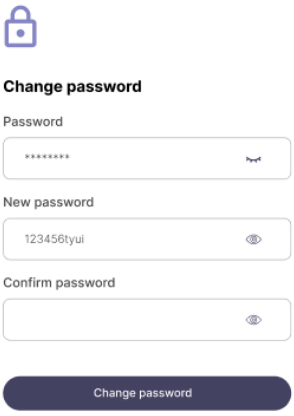

Рисунок 4.12 – Форма «Change password»

Для пошуку подорожей інших користувачів необхідно перейти або за посиланням «Trips» у хедері, або скористатися полем пошуку у ньому ж. Результат знайдених подорожей буде відображено на сторінці «Trips» (рис. 4.13).

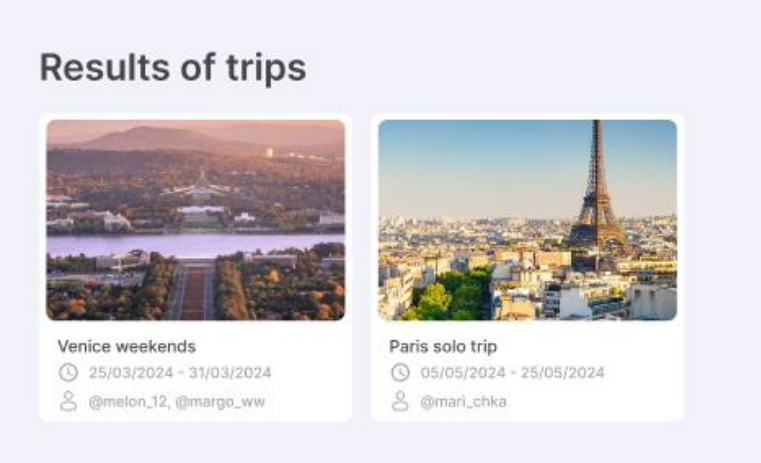

Рисунок 4.13 – Результат пошуку

Для створення нової подорожі необхідно перейти за посиланням «Add Trip» на сторінці профілю користувача, або з головної сторінки. Відкриється форма, показана на рис. 4.14.

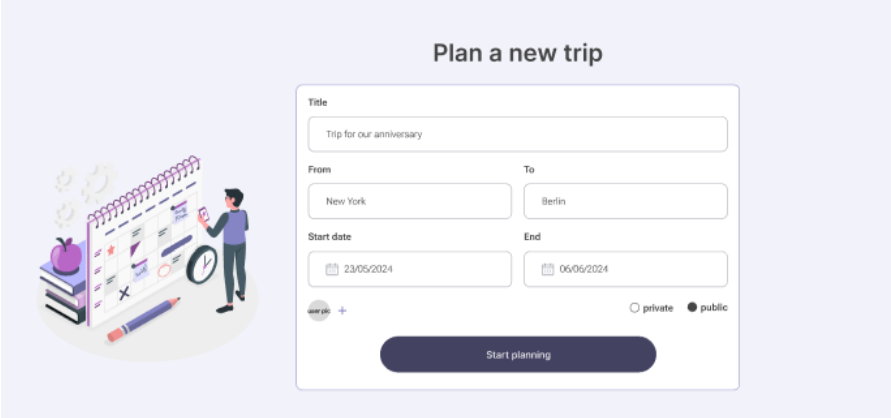

Рисунок 4.14 – Форма створення подорожі

Після натиснення кнопки «Start planning», відкриється наступна сторінка (рис. 4.15).

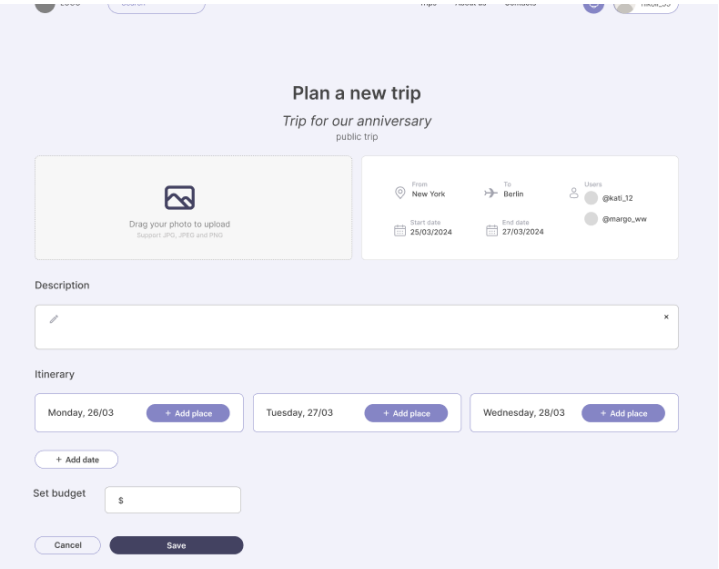

Рисунок 4.15 – Редагування подорожі

Мокап сторінки перегляду відредагованої подорожі виглядає наступним чином (рис. 4.16). Тут можна переглянути плани по дням та статистику витрат. Для користувачів, що не є редакторами подорожі, блок «Expenses» не буде відображено.

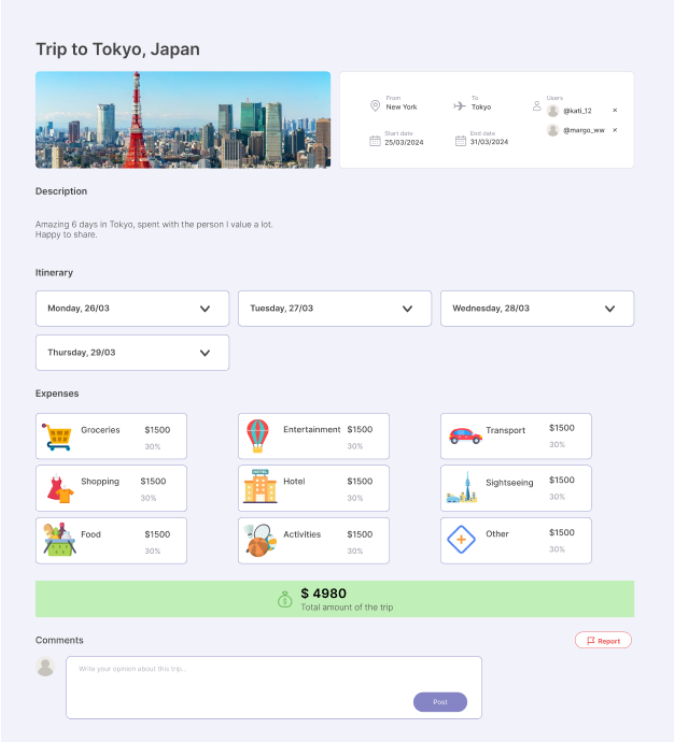

Рисунок 4.16 – Перегляд подорожі

Додані місця відображаються у списках, що випадають (рис. 4.17). Витрати вказувати необов’язково.

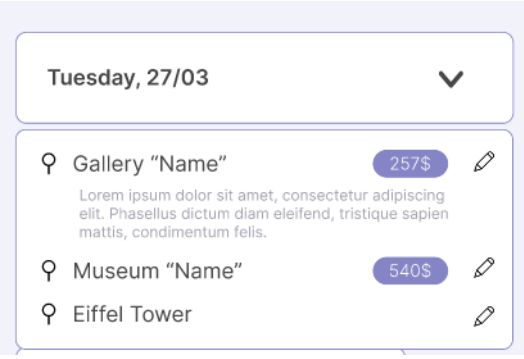

Рисунок 4.17 – План на день

Користувач може переглянути сповіщення у хедері (рис. 4.18).

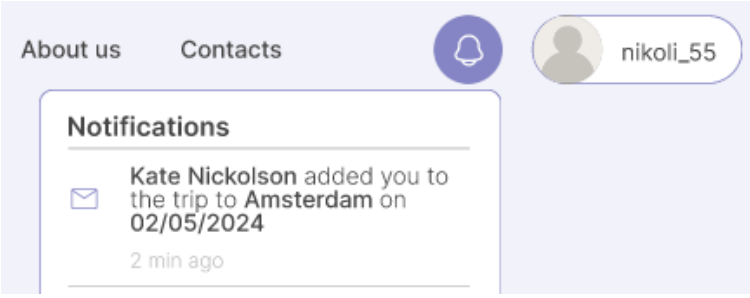

Рисунок 4.18 – Сповіщення

Мокапи деяких модальних вікон наведено у розділі 2.

4.3 Тестування ПЗ

Значну частину часу та зусиль розробки програмних проєктів витрачається на тестування програмного забезпечення. Тестування ПЗ визначається як процес оцінювання програмного забезпечення з метою виявлення недоліків або помилок у розробленому коді. Тестування проводиться для того, щоб гарантувати, що програмне забезпечення правильно виконує свою призначену мету, отримувати доступ, досягати та зберігати якість програмного забезпечення, і таким чином перевіряти, що програмне забезпечення придатне для використання [21].

Для перевірки коректності результату розробки ПЗ «Вебзастосунок планування подорожей» розроблено деякі тестові сценарії, які наведено в табл. 4.1 – 4.6.

Таблиця 4.1 – Сценарій тестування «Реєстрація»

ID	1
Мета	Створити обліковий запис
Передумови	1) користувач відкрив сторінку реєстрації; 2) користувач не авторизований у вебзастосунку.
Успішний сценарій	1) користувач заповнює обов'язкові поля: ім'я, ім'я користувача, пошту, пароль та підтвердження паролю; 2) користувач натискає кнопку «Sign up». 3) система виконує валідацію даних; 4) система відкриває користувачу сторінку його профілю.
Розширення	1) користувач не заповнює обов'язкові поля; 2) система виводить повідомлення про необхідність заповнення обов'язкових полів.
	1) користувач вводить вже існуючі дані у БД; 2) система виводить повідомлення, що такий користувач вже існує.
	1) пароль та підтвердження паролю не збігаються; 2) система виводить повідомлення відповідне повідомлення.
Статус	Виконано

Таблиця 4.2 – Сценарій тестування «Авторизація»

ID	2
Мета	Авторизуватися у вебзастосунку
Передумови	1) користувач відкрив сторінку авторизації; 2) користувач вже має обліковий запис.
Успішний сценарій	1) користувач вводить дійсне ім'я користувача/пошту та пароль у відповідні поля. 2) користувач натискає кнопку «Sign in»; 3) система відкриває користувачу сторінку його профілю.
Розширення	1) користувач вводить некоректні/неіснуючі у БД дані (ім'я користувача/email або пароль); 2) система виводить повідомлення про неправильно введені дані.

Кінець таблиці 4.2

	1) користувач не заповнює обов’язкові поля; 2) система виводить повідомлення про необхідність заповнення таких полів.
	1) користувач натискає на кнопку «Forgot password»; 2) система відкриває користувачу форму введення електронної пошти для надсилання інструкції відновлення паролю; 3) користувач переходить за посиланням у отриманому повідомленні; 4) система відкриває користувачу форму зміни паролю; 5) користувач вводить новий пароль та підтверджує його; 6) система зберігає новий пароль.
	1) користувач натискає кнопку «Sign out»; 2) система виконує деавторизацію користувача; 3) система відкриває користувачу сторінку «Home».
Статус	Виконано

Таблиця 4.3 – Сценарій тестування «Створення подорожі»

ID	3
Мета	Створити подорож
Передумови	1) користувач авторизований у системі; 2) користувач відкрив сторінку створення подорожі.
Успішний сценарій	1) користувач вводить дані: місце призначення, дати початку та закінчення подорожі у відповідні поля, встановлює видимість (private/public); 2) користувач натискає кнопку «Plan a trip». 3) система відкриває користувачу сторінку редагування подорожі.
Розширення	1) користувач не заповнює обов’язкові поля; 2) система виводить повідомлення про необхідність заповнення таких полів.
	1) користувач вводить вводить пошту або username користувача, якого хоче додати, у формі додання друзів; 2) система надсилає доданому користувачу сповіщення.

Кінець таблиці 4.3

	1) користувач хоче додати в подорож незареєстрованого користувача; 2) система виводить повідомлення, що такого користувача не існує.
Статус	Виконано

Таблиця 4.4 – Сценарій тестування «Редагування подорожі»

ID	4
Мета	Редагувати подорож
Передумови	1) користувач авторизований у системі; 2) користувач вже має створену подорож.
Успішний сценарій	1) користувач обирає подорож для редагування; 2) користувач завантажує фото, додає опис, заповнює дати місцями у формі «Add place», додає витрати у формі «Add expense»; 3) користувач натискає кнопку «Save»; 4) система зберігає внесені користувачем зміни.
Розширення	1) користувач не заповнює обов’язкові поля; 2) система виводить повідомлення про необхідність їх заповнення.
	1) користувач додає у подорож іншого користувача для спільного редагування; 2) система надсилає вказаному користувачу сповіщення про спільну подорож.
Статус	Виконано

Таблиця 4.5 – Сценарій тестування «Редагування профілю користувача»

ID	5
Мета	Редагувати профіль
Передумови	Користувач авторизований у системі.
Успішний сценарій	1) користувач переходить на сторінку редагування профілю; 2) користувач завантажує фото, додає опис (bio), змінює username та ім’я; 3) користувач натискає кнопку «Save»; 4) система зберігає внесені користувачем зміни.

Кінець таблиці 4.5

Розширення	1) користувач натискає кнопку «Delete my account»; 2) система виводить модальне вікно про підтвердження видалення облікового запису; 3) користувач підтверджує видалення; 4) система видаляє обліковий запис користувача та завантажує сторінку «Home».
	1) користувач натискає на кнопку «Change password»; 2) система відкриває користувачу форму зміни паролю; 3) користувач вводить новий пароль та підтверджує його; 4) система зберігає новий пароль.
Статус	Виконано

Таблиця 4.6 – Сценарій тестування «Пошук подорожей»

ID	6
Мета	Знайти опубліковані користувачами подорожі.
Передумови	Користувач авторизований у системі.
Успішний сценарій	1) користувач вводить у хедері або на сторінці Trips у формі пошуку ключові слова; 2) користувач натискає кнопку «Search»; 3) система відображає знайдені подорожі; 4) користувач відкриває потрібну подорож.
Розширення	1) система не знаходить подорожі за вказаними даними та виводить відповідне повідомлення.
	1) користувач обирає подорож для перегляду; 2) користувач пише коментар у формі додання коментарів.
Статус	Виконано

Отже, розглянуто 6 тестових сценаріїв для вебзастосунку планування подорожей. Протестовано такі функції системи, як:

– реєстрація;
– авторизація;
– створення подорожі;

– редагування подорожі;
– редагування профілю користувача;
– пошук подорожей.

Всі тестові сценарії разом з розширеннями успішно виконано.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи бакалавра наведено аспекти програмної реалізації вебзастосунку. Наведено частини розробленого програмного коду для деяких функцій системи та сторінок, зокрема головної сторінки вебзастосунку. Наведено частину коду, що забезпечує адаптивність сторінок вебзастосунку. Також наведено список деяких контролерів та моделей, що відповідають за функції системи.

Розроблено керівництво користувача: наведено мокапи сторінок, до яких має доступ звичайний користувач, описано деякий функціонал та сторінки, по яким може перейти користувач.

Показано результати виконаної роботи з кодування програмного забезпечення: наведено частини коду контролерів, моделей Laravel та розмітку головної сторінки.

Проведено тестування розробленого ПЗ. Для цього, розроблено тестові сценарії, що описують як успішний сценарій, так і можливі розширення.

При виконанні тестових сценаріїв для розробленого ПЗ виявлено деякі недоліки та помилки у проєкті, після чого їх виправлено. Описані сценарії тестування успішно виконано.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра розроблено вебзастосунок планування подорожей для автоматизації процесу планування та організації подорожей.

Для досягнення поставленої мети вирішено наступні завдання:

– проаналізовано аналогічні застосунки планування подорожей;

– визначено функціонал застосунку, що розробляється;

– розроблено специфікацію вимог до програмного забезпечення;

– розроблено UML-діаграми, що моделюють роботу програмного забезпечення;

– розроблено дизайн застосунку;

– виконано кодування програмного забезпечення;

– проведено тестування програмного забезпечення.

Проведено аналіз аналогічних застосунків: виділено їх основні функції, переваги та недоліки. На основі проведеного аналізу сформовано вимоги до програмного забезпечення вебзастосунку планування подорожей. Визначено функціонал та користувачів системи, розроблено специфікацію вимог.

Для моделювання програмного забезпечення, побудовано UML-діаграми, які демонструють роботу вебзастосунку. Описано варіанти використання системи. Розроблено дизайн (мокапи) вебзастосунку.

Обґрунтовано вибір технологій шляхом аналізу розроблених вимог до програмного забезпечення та сучасних досліджень технологій, вимог ринку. Виконано кодування вебзастосунку планування подорожей та проведено його тестування. Розроблено керівництво користувача.

Результатом проведеної роботи є вебзастосунок планування подорожей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Funliday makes trip planning easier. Funliday. URL: https://www.funliday.com/en (Last accessed: 27.03.2024).

2) About Polarsteps. Polarsteps. URL: https://www.polarsteps.com/about (Last accessed: 27.03.2024).

3) About Grouppi. Grouppi. URL: grouppii.com (Last accessed: 29.03.2024).

4) Eriksson H.-E., Penker M., Lyons B. UML 2 Toolkit. John Wiley & Sons, 2003. 515 P. URL: https://nuleren.be/edocumenten/uml-2-toolkit.pdf (Last accessed: 02.05.2024).

5) Agner L. T. W., Lethbridge T. C. A Survey of Tool Use in Modeling Education. *2017 ACM/IEEE 20th International Conference on Model Driven Engineeri ng Languages and Systems (MODELS)*. P. 303– 311. URL: https://ieeexplore.ieee.org/d ocument/8101276 (Last accessed: 10.05.2024).

6) Staiano F. Designing and Prototyping Interfaces with Figma. Packt Publishing Ltd, 2022. 382 P.

7) Tatroe K., MacIntyre P. Programming PHP: Creating dynamic web pages. O'Reilly Media, 2020. 493 P.

8) Olanrewaju R. F., Islam T., Ali N. An empirical study of the evolution of PHP MVC framework. *Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering* (2015). P. 399–410. URL: https://doi.org/10.1007/978-3-319-07674-4_40 (Last accessed: 12.05.2024).

9) Laaziri M., Benmoussa K., Khoulji S. A Comparative study of PHP frame works performance. *Procedia Manufacturing*. 2019. Vol. 32. P. 864– 871. DOI: 10.101 6/j.promfg.2019.02.295.

10) Laravel, Symfony, Yii, CodeIgniter, CakePHP. Google Trends. URL:https://trends.google.com/trends/explore?q=Laravel,Symfony,Yii,CodeIgniter,Cak ePHP&hl=en (Last accessed: 12.05.2024).

11) Li Y., Manoharan S. A performance comparison of SQL and NoSQL datab ases. *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Pr*

ocessing (PACRIM) (2013). P. 15– 19. URL: https://doi.org/10.1109/PACRIM.2013.6625441 (Last accessed: 12.05.2024).

12) Mihai G. Comparison between relational and NoSQL databases. Econ. Appl. Inform. 2020. P. 38–42. URL: doi.org/10.35219/eai15840409134 (Last accessed: 12.05.2024).

13) MySQL, Microsoft SQL Server, PostgreSQL, SQLite, MariaDB. Google Trends. URL: trends.google.com/trends/explore?q=%2Fm%2F04y3k,%2Fm%2F09gc20r,%2Fm%2F0120vr,%2Fm%2F05ynw&hl=en (Last accessed:15.05.2024).

14) Tongkaw S., Tongkaw A. A comparison of database performance of MariaDB and MySQL. *2016 IEEE Conference on Open Systems (ICOS)*(2016). P. 117–119.URL: doi.org/10.1109/ICOS.2016.7881999 (Last accessed: 15.05.2024).

15) Wodyk R., Skublewska-Paszkowska M. Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework. *Journal of Computer Sciences Institute*. Vol. 17, 2020. P.358–364. DOI:10.35784/jcsi.2279.

16) Juba S., Vannahme A., Volkov A. Learning PostgreSQL. Packt Publishing Ltd, 2015. 425 P.

17) PhpStorm Features. PhpStorm. URL: https://www.jetbrains.com/phpstorm/features/ (Last accessed: 12.05.2024).

18) Verma A. MVC Architecture: A comparative study between Ruby on Rails and Laravel. *Indian Journal of Computer Science and Engineering (IJCSE)*. Vol. 5, Issue 5. P. 196–198. URL: https://ijcse.com/docs/INDJCSE14-05-05-053.pdf (Last accessed: 30.05.2024).

19) Khan S., Khanam A. T. Study on MVC Framework for Web Development in PHP. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol*. 2023. P. 414– 419. DOI: 10.32628/CSEIT2390450.

20) Eloquent: Relationships. Laravel. URL: https://laravel.com/docs/11.x/eloquent-relationships (Last accessed: 19.05.2024).

21) Umar M. A. A Study of Software Testing: Categories, Levels, Techniques and Types. 2020. DOI:10.36227/techrxiv.12578714.v1.s.