FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computer Science & Mathematics*

FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computer Science & Mathematics*

BSc DEGREE

IN

Computer Science (Software Engineering) Top Up

FINAL REPORT

Name: **Mohamed Fariq Abdulla Sabith**

ID Number: **E026503**

Project Title: **Leisure Diary**

Project Type: **Build**

Date: **20.04.2023**

Supervisor: Ms. Sampa Rasanie

Kingston University London

Did you discuss and agree the viability of your project idea with your supervisor?	Yes
Did you submit a draft of your proposal to your supervisor?	Yes
Did you receive feedback from your supervisor on any submitted draft?	Yes

Abstract

Leisure Diary is a solution proposed by AFADAMAD (Pvt) Ltd, a travel agent that aims to assist travelers with their inbound and outbound travel needs to ensure their satisfaction by providing a hassle-free, all-in-one trip organizer. The solution offers a wide range of features that reduce the workload of the organizer and provide the required resources at their fingertips. It provides a convenient platform for all the requirements of a traveler and offers an opportunity for related stakeholders to grow their businesses. This report contains all the details about the technologies, methodologies, work plan, constraints, and ways to overcome them.

Contents

List of Figures

List of Tables

Glossary of Terms

HTML: Hyper Text Markup Language

IDE: Integrated Development Environment

CSS: Cascading Style Sheet

RAM: Random Access Memory

JS: JavaScript

API: Application Programming Interface

Introduction & Literature Review

1.1 Introduction

AFADAMD (Pvt) Ltd is a premier travel agency that specializes in catering to the travel needs of both inbound and outbound travelers. As a leading trip advisor, the company is committed to ensuring that all travelers have a hassle-free and satisfying travel experience. With a wide network of hotels, resorts, and other leisure activities, the company provides an extensive range of related services within its chain.

Tourism has grown significantly over the years, impacting various economic aspects of countries, businesses, and individuals. To maintain this growth and improve the industry, countries and related businesses must work together to mitigate problems faced by travelers and other entities. This project aims to achieve these objectives through effective communication.

The project's primary goal is to provide an all-in-one solution for planning and organizing trip-related aspects with ease, streamlining the entire process. Moreover, the project serves as a communication platform for all stakeholders to enhance communication efficiency, minimizing delays and errors.

In conclusion, AFADAMD (Pvt) Ltd is dedicated to providing an exceptional travel experience for all its clients while working to enhance the tourism industry's growth through effective communication and collaboration among stakeholders.

1.2 Background and Motivation

In today's modern world, individuals and families often find themselves facing the daily stresses of life. As a result, traveling has become one of the most popular ways to unwind and spend quality time together. However, planning a trip can be a daunting task, given the multitude of associated entities. Furthermore, businesses play a crucial role in the tourism industry and are considered stakeholders in trips.

With the majority of internet users accessing the web through mobile devices, reaching travelers through mobile applications has become the most effective method. Proper communication is essential to resolve any potential issues faced by foreign travelers [1]. Despite the plethora of travel applications available in the market, there is currently no all-in-one solution.

Separate applications for each entity do not effectively connect them together, and travelers and organizers seek convenience in a single application. Ultimately, the success of an application lies in customer satisfaction, which is the ultimate goal. The organization of a trip involves multiple tasks that can make it challenging for organizers to engage in the arrangement of the trip while others are enjoying it. The tourism service chain is continually improving, and the involvement of each sector is essential to its overall success.

The primary focus of mobile application development or any other type of application development is to improve the user experience (UX), as it is crucial to the value of the application. Users always expect a more convenient solution, and thus, solution providers must evaluate user requirements as a critical initial task. A user-friendly solution will attract new users to the application.

1.3 Problem in brief

Planning a trip can be a daunting process, particularly when it comes to accommodating the interests and preferences of multiple individuals. The process typically involves a myriad of tasks, including choosing a destination, arranging transportation, finding suitable accommodation, planning meals, and organizing daily activities throughout the trip. For many

people, the purpose of a trip is to unwind and relax from the daily grind, making the planning process an unwelcome chore.

One major challenge that arises during trip planning is the burden that falls on those responsible for organizing the trip. While others get to enjoy the trip, the planners often remain preoccupied with making arrangements and ensuring that everyone's expectations are met. This problem is particularly acute when planning trips involving large groups, such as corporate or school trips, where coordinating transportation, accommodation, and other logistics can be a formidable task.

Currently, there are numerous applications on the market that provide booking services for hotels, transport, and food. However, there is no all-in-one solution that enables users to plan their entire trip and share the plan with service providers without encountering any additional hassles during the trip.

1.4 Aim & Objectives

1.4.1 Aim

The aim of this project is to develop an all-in-one mobile application to plan, organize and review a hassle-free trip to travelers' solution with all the related stakeholder involvement.

1.4.2 Objectives

- To identify and provide all in one solution to the travelers to address all their needs through one application.
- To suggest recommendations for the travelers to get decisions on planning a trip.
- To design an application which can find fulfill all the aspects of needs of all the stakeholders.
- To build a communication and interaction method between travelers, service providers and the organizers.

1.5. Scope

The scope of the project that refers to the specific set of outcomes or results that need to be delivered based on the project requirements. It outlines what is included and excluded from the project and governs what can be added or removed during the implementation phase. In this project the scope implements a mobile application along with a web application to serve the travelers a hassle-free travel experience where all their needs to be found in one place. This it to make the travelling experience more efficient and time saving. Also, which will help the businesses to reach their clients easily. Throughout a SWOT analysis we have identified the identify the project's strengths, weaknesses, opportunities, and threats.

1.1.Project Scope Statement

A project scope statement is a document that defines the objectives, goals, deliverables, and boundaries of a project. It outlines what the project will accomplish, what is included, and what is not included. The scope statement also includes a description of the project's stakeholders, assumptions, constraints, and risks. It serves as a reference for the project team and stakeholders to ensure that the project stays on track and meets its objectives [1].

Project Scope Statement

Project Name	Leisure Diary
Project owner	Sabith Fariq
Date	
Product Scope Description	Scope: • Requirement analyzing and engineering. • Identifying the flow and mapping the Flow charts, Data Flow Diagrams, ER Diagrams and Prototyping

	• Designing and developing the Leisure Diary Mobile app and Web Application. • Testing the implementation • User Manual • Documenting
Project Deliverables	• Project proposal • Project Gantt Chart and Milestone Schedule • Design document • Develop Mobile app • Develop Web App • Test Document • User manual • Final Report
Constrains	• Time • Resource
Acceptance Criteria	The project scope is agreed and accepted by the management.

Table 1- Project scope statement

1.6 Deliverables

The deliverables describe the primary outcome of the project as the result of the scope. Hence, we have different roadmaps and tracking tools to reach the success without any constrains. In the form of charts, the tools will be laid as the path to reach project objectives as mentioned above.

The project delivers followings,

1. Project Gantt Chart and Milestone Schedule

2. Design document
3. A Mobile app for the travelers
4. A Web App for the service providers
5. Test Document
6. User manual
7. Final Report

1.5.1. Milestone Schedule

Number	Task	Deliverables	Time period
1	Requirement gathering and analysis	• Functional and non-Functional requirements • Gantt chart • Milestone Schedule • Project proposal	December-January 2023
2	System Design	• Prototyping • UI/UX Designing • Requirement Engineering • Design Diagrams	January – March 2023
3	System Implementation	• Mobile Application • Web Application	March – April 2023

4	Testing	• Testing reports	March – April 2023
5	System deployment and submission	• Final report • Developed product	March – April 2023

Table 2 Milestone Schedule

1.5.2 Gantt Chart

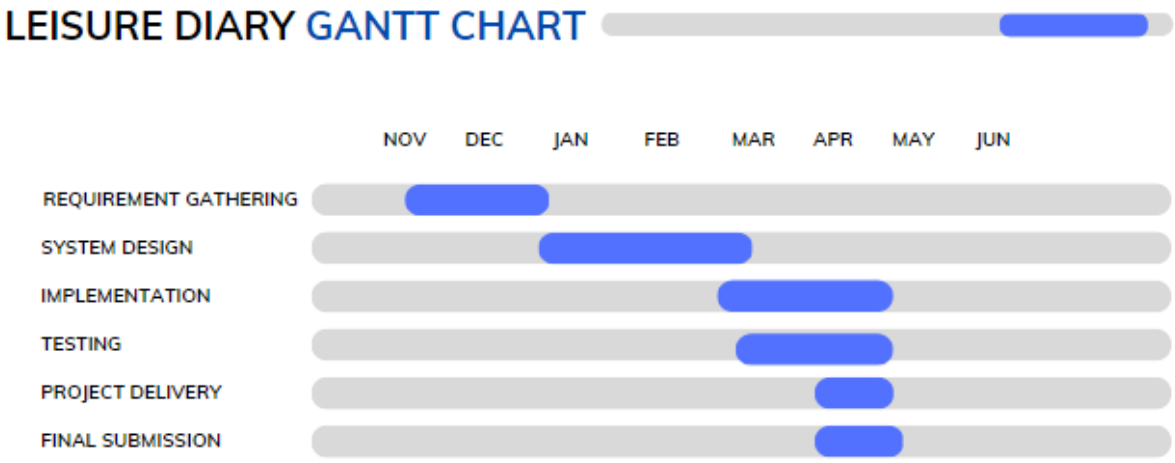


Figure 1 Gantt Chart

1.7 Technologies & Resources

1.7.1 Technologies

1.7.2 Resource requirements

Software requirement	• Visual Studio code editor • Windows 7 or above • MongoDB/Firebase • Android studio
Hardware Requirements	• Personal Computer • 8GB RAM or higher • 500GB HDD • 1GB Video Graphics card or Higher
Client requirement	• Mobile phone with internet connection • A PC with internet connection
Technological requirements	• HTML • CSS • JavaScript • Flutter • MongoDB / Firebase • EJS

1.7.3 Libraries and APIs used

The Leisure Diary application has been made using many APIs and libraries to make the development process faster and smoother. Following are the APIs and Libraries used to build the Leisure Diary web application and Mobile application.

Google Map API.

Google Cloud Console is a web-based tool that lets the developer to manage the resources and services on the Google Cloud Platform. It is a central location where you can access and control your resources, such as virtual machines, databases, and storage buckets.
The Google Maps API is used to access Google Maps in the applications. This means it provides the ability to create and customized maps, add markers, and display directions. It's a very powerful tool and widely used in many different industries.

A separate project has been created in Google Cloud Console for Leisure Diary system implementation.

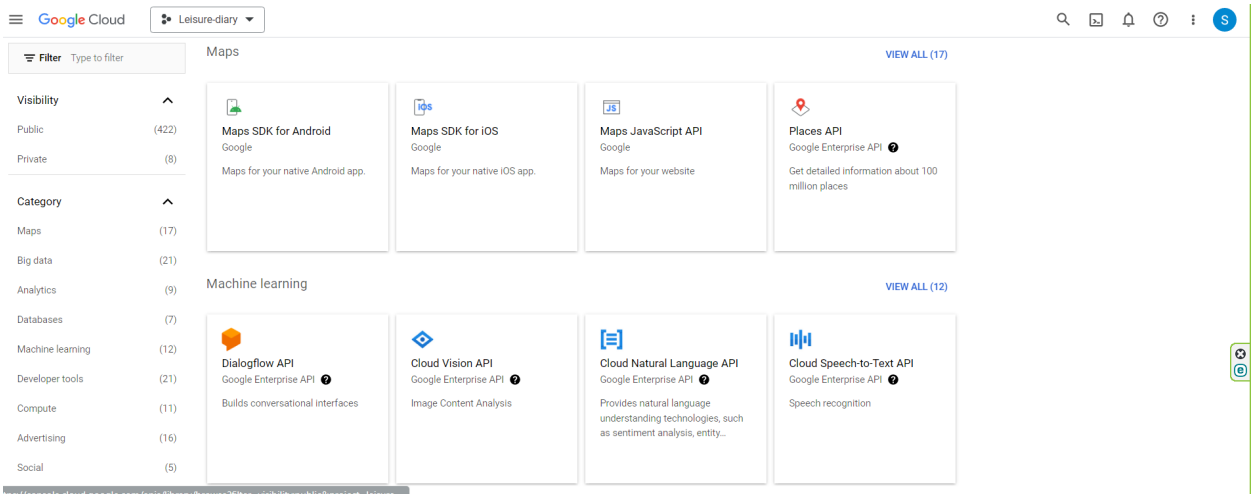


Figure 2- Google cloud console integration

The one of the library Maps JavaScript API helps the user to search and find the exact location when creating new services. Also, it will assist other component to render the linked location API whenever its required.

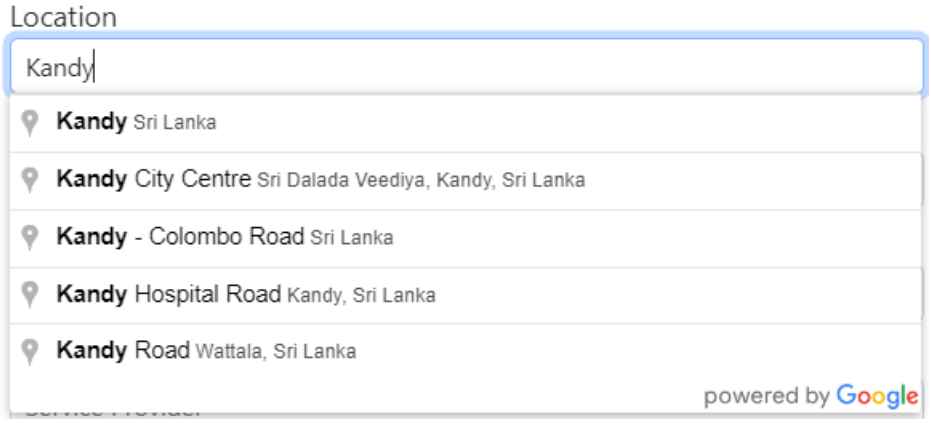


Figure 3- Location API

Express.js

Express.js is a JavaScript popular framework which is used to build web applications and related APIs. Express.js can be used to develop route to the pages and endpoints. The framework also helps to handle and interact HTTP requests.

mongooseValidationErrorHandler

mongooseValidationErrorHandler is a popular library which helps as a middleware in handling validation errors when interacting with mongoDB. mongooseValidationErrorHandler also helps to understand the error easily.

Multer

Multer is also a middleware which helps to handle file uploading. Node.js uses the Multer library to simplify the file handling and limiting the file sizes.

Fs

The Leisure Diary system has a component to read, write and modify the files. To achieve the task node.js uses its inbuilt library “Fs”

Dotenv

Dotenv files are used to store sensitive and confidential data such as Database connections, API Keys and Port details. Main concept of the Dotenv files is, If the developer builds an application and had to share it with other stakeholders or third-party people. Dotenv file which has the confidential data can be removed. The recipient will have to create their own credential and access the application.

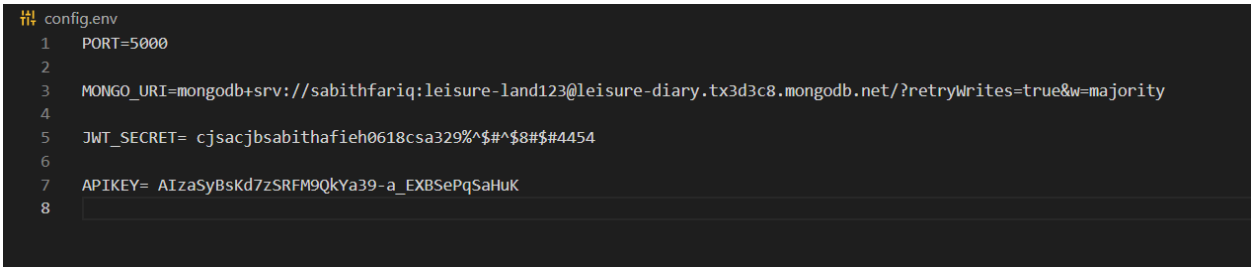


Figure 4- Dotevn configuration

```
// Assigning a PORT
dotenv.config({path:'config.env'});

const JWT_SECRET= process.env.JWT_SECRET;

const PORT=process.env.PORT || 3000;
```

Figure 5 Dotenv Usage in the file

bcrypt

The Node.js library bcrypt serves to hash passwords. By using a salt and a one-way hash function to hash the passwords, it offers a secure means to store them, making it impossible for

hackers to decipher the original password. In the Leisure Diary application, there are two types of users, can register them self in the system. When they enter the password, it will **bcrypt** and saved in the database. Whereas no one can access or read the password.

```
_id: ObjectId('6425056603a72a72585e6637')
username: "esoft"
email: "esoft@gmail.com"
password: "$2a$10$HKCkYvs.Ym0g0BvXm.TJg.EhCYB6xl712DLJ61wAjDIHyikFEbKle"
__v: 0
```

Figure 6- Service provider registration

Travelers Signing up through the mobile app

```
_id: ObjectId('6439ad3d14620a690b81acb3')
name: "Sabith2"
email: "sabith@frontierlk.com"
password: "$2b$10$mFl9CMnY0S2mBJCdThpGZ.SPO31xt8Xe9Sn8NiNDp6BGN0.98byQS"
__v: 0
```

Figure 7- Traveler registration

JWT

JWT is used to generating and validating secure tokens in web applications, or JSON Web Tokens. Users are authenticated using JWTs to grant access to restricted resources.

Morgan

Morgan is a middleware for that handles and logs HTTP requests and responses. It quickly records the details about each request, such as the method used, the URL, the response status code, and more.

```
POST http://192.168.8.188/api/Reservations 302 314.456 ms - 45
GET http://connectivitycheck.gstatic.com/generate_204 404 0.597 ms - 151
GET http://www.google.com/gen_204 404 0.384 ms - 146
GET http://connectivitycheck.gstatic.com/generate_204 404 0.548 ms - 151
GET http://play.googleapis.com/generate_204 404 0.401 ms - 151
GET http://192.168.8.188/api/getAccomodations 200 931.065 ms - 161843
reservation body {
```

Figure 8- Morgan Model

Session

Node.js's Session middleware helps manage user sessions. Users can maintain their authentication over numerous requests due to a method for storing and retrieving session data across requests.

BodyParser

BodyParser is a middleware which is used to parse HTTP request bodies. It offers a mechanism to extract JSON, form data, and other forms of data from HTTP request bodies. [2]

Libraries used for the Mobile application

cupertino_icons: ^1.0.2

Cupertion icons library is used to add icons in the UI of the application. The library has large number of icons which ease the UI development process.

flutter_svg: ^1.1.6

A picture cache with an ui: Picture class is implemented in the flutter svg package. This is the Skia graphics engine's SkPicture wrapper, which stores particular SVG rendering instructions in binary mode [3].

flutter_swiper: ^1.1.6

With the use of the touch slider plugin in wiper flutter, a user can move any view, such as an image slider. We may utilize a variety of layouts using the swiper library. This library works with both iOS and Android devices [4].

shared_preferences: ^2.1.0

Let's say users want to store a small item which will be referred to at a later point when the user runs the application. Such activities are performed by shared preference library [5]

jwt_decoder: ^2.0.1

The library which helps to decode Json web tokens which is used to login authentication in Leisure Diary application.

google_fonts: ^4.0.3

A library used to fetch google fonts from internet.

pdf: ^3.10.1

To create and access PDF files within the flutter application

path_provider: ^2.0.5

path_provider library is used to provide paths for specific actions within the application

dio: ^4.0.0

Dio package is used to is a powerful library which used when working with Http request.

printing: ^5.10.3

Printing library is the tool assist the developer to work print documents

permission_handler: ^10.2.0

Permission handler helps with overriding the device permission on such required actions.

1.8 Literature Review

Trip planning is an essential part of tourism and travel. It entails planning, selecting, and analyzing key modules of a journey, including lodging, activities, transportation, and finances. Travelers now have access to a variety of tools that can help them plan their vacations thanks to the development of technology and the ease with which information is available online.

Wang and Chen stated [6] in their study that majority of tourists prefer to use online platforms for travel planning and organizing via websites, social media contents and travel agencies. Also, their study indicates that travelers who have used online platforms to plan and organize their trips are more satisfied. People engage in a variety of activities to have fun and refresh their minds from monotonous tasks at work or in daily life, including fishing, watching TV or movies, hanging out with friends, and many others. Touring is one of those activities. according to the Indonesian lexicon. Tourism is a leisure travel-related activity.

According to Soekadijo [7] Tourism refers to both the temporary movement of individuals to locations other than their usual areas of residence and employment, as well as the activities they engage in while there. There are a few components that tourists can appreciate to make their trip memorable and interesting.

A study conducted by Lee et al. (2008) indicates the factors which effects the decision-making process of travelers through his study. It also shows how attractive is the destination, convenience and cost are the major factors considered by the travelers when making decisions. Further, the study clearly indicates that travelers more often use online reviews and recommendations posted by other travelers as a key assistant to make decisions [8].

The study which was carried by Deegan et al, which evaluated the planning and organizing element on overall travel experience says the quality of travel planning was positively related to travel satisfaction and loyalty. Also, the study indicated that travel agencies and companies should focus more on creating more personalized packages and travel planning services to enhance and satisfy the travelers [9].

As we aware the tourism is a one of the fastest growing industries in the world. The function of travel planning in encouraging sustainable travel behavior has drawn more attention as the importance of sustainable tourism has grown. The desire of tourists to engage in sustainable travel behavior during the trip preparation phase was examined in a study by Vu and Tasci (2018). Tourism attitudes toward sustainable travel, perceptions of the place, and perceived behavioral control were discovered to be important determinants of travelers' intentions to practice sustainable travel [10].

T. D. Hinch and J. E. S. Higham [11] carried in their study and analyzed that the rise in sport tourist activities caused by the availability of different sports. In certain ways, sport tourism contributes to all aspects of life. The subject of research is different sports that are used by tourists.

Leiper [12] has stated there are three main activity occurring elements available,

1. Travelers

   Travelers are the main role play of the activities of tourism. In such activities they gain memories and experiences lasts their whole period of life.

2. Geographic components

   Above travelers' movement are takes in to account as Geographical components.

   Geographic component has three sub components,

   - Region of Origin – The place of origin of the travelers. Where they do all their daily lifestyle activities.
   - Regional Transit- Not every passenger was required to make a stop there. Yet, all travelers undoubtedly pass through that region, therefore the object destinations will play a crucial part.
   - Destination Region-Creates demand for travel to a certain tourist site and acts as a booster for the entire tourism system.

3. The tourism industry

   The industry which provides the travel related services and other activities.

The definition of tourism can be drawn from these explanations. It is a happy pastime geared for amusement and leisure pursuits for travelers. Up to the trip's conclusion, there is no melancholy. Being joyful all the time is one of the things tourists do. Because of tourists who appreciate tours, they can choose the type of tourism they prefer in advance. After work or other activities, the majority of people utilize travel as a mood enhancer and media to recharge their brains. Tourists can feel better than previously by participating in the activities they receive as part of their tour.

Tourists have a wide range of tourism options. People can choose and explore before enrolling in tourism-related activities. Also, they can be more particular about the type of tourism they prefer. According to Spillane [13] the types are determined by the selected destination. Types of the tourism as follows,

- **Enjoyment trip**

  Pleasure tourism is the kind of tourism practiced by individuals who travel away from home for a holiday in order to take in some fresh air, quench their curiosity, dry their eyes, see something new, take in the beauty of nature, learn about the locals, or simply to find calm in the village.

- **Recreation**

  Tourism recreation refers to travel that is done by individuals who use their vacation time to rest, restore their sense of freshness, or who wish to relieve fatigue.

- **Cultural**

  The kind of this tourism is distinguished by a reason, such as the desire to study at research and teaching facilities, to discover the traditions and way of life of people in other nations, or to see historical sites or artifacts from earlier civilizations.

- **Sport**

  Sport events are the main emphasis of sport tourism. Sport tours or sport tourism refers to everything that is connected to sports and includes travel. There are two distinct categories for sport tourism:

- Major sporting events Large-scale sporting events like the Olympics, the ski world championships, the World Cup, and others garner attention from both fans and spectators.
- The practitioners' use of sporting tourism For those who like to do sports alone, such as mountain climbing, horseback riding, hunting, fishing, and other activities, sport tourism is a good option.

- **Business**

  Theorists contend that this was a business trip taken by a travel expert or that it was necessary given the nature of the suspect's position or employment, which limited his or her options for travel dates and locations.

- **Convention**

  A tourism convention or meeting is a gathering that draws hundreds or even thousands of attendees, who often spend a few days in the host city or nation. [13]

While Yoeti (1995) [14] defined tourism as a conscious activity that rotates among the people in a country that includes the standing of another region for a while in pursuit of the satisfaction of various and different from what happened in their home country, this is not the case now. Thus, a journey or activity that is centered on providing pleasure and relaxation is called tourism. If such phrases are put together, the result will be a trip that focuses on horror activities as amusement and leisure pursuits.

According to specialists, there are many different types of tourism, including commercial tourism, sport tourism, and nature tourism. A type of tourism that belongs in pilgrimage or ziarah is the horror tour. Due of horror tourism, visitors are invited to a location that many people regard as sacred. Pendit (1994) [15]defined pilgrimage tourism as a journey to holy locations, the tombs of great or exalted leaders, or to hills or mountains that are thought to be sacred graveyards of human figures or leaders as full-fledged magical legends.

Tour packages are a type of commercially produced media that tour operators offer to travelers. Mansur (2009) [16]states that a tour package is a tourists' itinerary of activities that has been put together and is maintained at a set cost, and that includes travel, lodging, sights and attractions, as well as other ancillary services that are offered under the terms of the tour package. According to Kotler and Armstrong (1989) [17], a tour package is something that is provided to the customer or market share to appease a whim and desire that includes tangible goods, services, human resources used by the firm, and ground-breaking or innovative ideas.

Mobile Application for travel

Buhalis (2008) has reviewed in his article that there are variety of mobile applications available for the travelers also the author has mentioned there are plenty of features and functionalities on travel applications which bring the convivence for the travelers [18]. In recent years it has proved that travel apps have become increasingly popular among travelers. Studies and researches have explored the factors that influence the use and getting adopted to travel apps.

Wang and Chen (2007) stated travelers are more likely to adopt and use travel apps if they perceive them as useful for their travel needs. This includes features such as booking accommodations and transportation, finding local attractions and activities, and accessing travel guides and maps [6].

A study conducted by Yu, Lio, and Yao (2018) indicates that travelers are more likely to use the travel mobile apps when travelers find the convenience to locate their navigations through it [19].

Moreover the study explored by Xiang, Du, and Ma (2017) stated that adoption of the travel apps are influenced by society. Such as reviews and recommendations influence the decision making of travelers [20].

Security and privacy concerns could be a potential barrier to the adoption of travel apps. People concern more about their privacy and security of data which could reduce consumers to relucted on adopting to travel apps. Kim and Lee (2018) highlighted that developers have to focus more on security to avoid aforementioned types of security and privacy issues [21].

Overall, these factors can have a significant impact on the adoption and use of travel apps by consumers. App developers and marketers should consider these factors when designing and promoting travel apps to ensure that they meet the needs and expectations of users.
Ali, F., Ryu, K., & Hussain, K has indicated in his study that Influence of online reviews on consumers' hotel booking decisions. International Journal of Hospitality Management, 53, 74-85. , [22]how online reviews affect consumers' decisions to book hotels. The researchers used structural equation modeling to examine the data from a survey of 386 participants (SEM). The findings imply that consumer decisions to book hotels are significantly influenced by online reviews. Customers are more likely to reserve rooms at hotels that have received good reviews and higher ratings. The study also discovered that consumers' trust in online reviews is significantly influenced by their utility and credibility.

2. Analysis

Analysis is a crucial stage in software development that is typically the first step in the software development life cycle. During this phase, the development team collaborates closely with stakeholders to identify and document their requirements. This includes understanding the business objectives, user needs, and technical constraints. Additionally, the team may conduct market research and competitor analysis to better understand the target audience and industry landscape. The analysis phase is critical in ensuring that the software project is feasible, meets the needs of its stakeholders, and identifies potential risks and challenges. It also enables the team to define the project's scope, objectives, and deliverables, which serves as a basis for project planning and estimation. Furthermore, the analysis phase provides a foundation for designing the software architecture and creating a software requirements specification (SRS) that outlines the software's functional and non-functional requirements, performance, security, and usability considerations. Accurately documenting requirements is essential to ensuring that the software meets the stakeholders' needs, is delivered on time, and stays within budget. Overall,

the analysis stage is a critical component of software development that sets the foundation for the project's success [23].

2.1 Use case diagram

A use case diagram is a graphical representation commonly used in software development that displays how a system interacts with its users or other systems. Its purpose is to present the system's functional requirements in a simplified way. It provides a broad perspective of the system's functionality and presents the various user types or actors involved in the system. Use case diagrams help to document and convey requirements, and can serve as a blueprint for building software features [24].

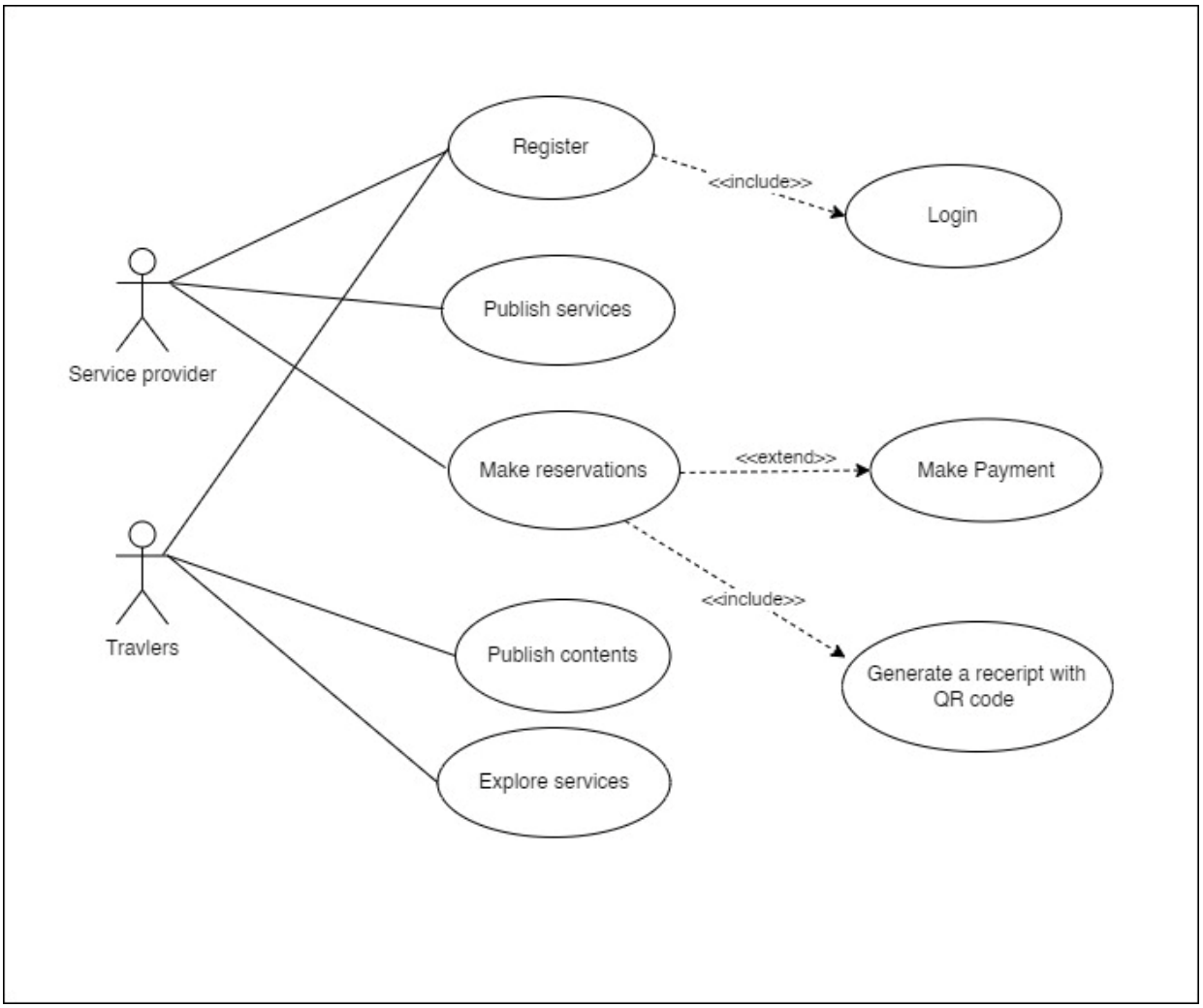


Figure 9Use case diagram

2.2 Clear problem definition

Planning a trip can be a complex process depends on the type people involved and their interests. The process involves numerous tasks, such as searching and selecting a destination, arranging transportation, finding suitable accommodations, organizing food arrangements, and planning day-to-day activities throughout the trip. Many individuals are hesitant to get involved in the

planning process because the ultimate goal is to satisfy all co-travelers and their expectations. Those who are often stressed or busy may view a trip as an opportunity to relax and unwind.

2.3 User stories

User stories are a technique used in agile software development to capture and communicate requirements from the perspective of the end-user. Each user story describes a feature or functionality that the user wants to achieve and follows a simple template: "As a [user], I want [feature], so that [benefit]. [25]"

Service provider

- As a service provider, I want to add my services to the system, so that the travelers can explore them and reserve them and that will help to grow my business.
- As a service provider, I can add, edit or remove the published services.
- As a service provider, I want to generate a report of the reservations.

Traveler

- As a traveler, I want to explore the services as per my requirements and find the most suitable one.
- As a traveler, I want to reserve the required services to fulfill my travel needs.

2.4 SWOT Analysis

Strength	Weakness
• Brand recognition establishment for the businesses. • Diverse range of travel offerings and packages • Strong partnerships with hotels, airlines, and other travel-related companies • Convince of travelers on finding related services. • Save time and misleading by the fraud guiders.	• Limited geographic reach or scope of services • Limited differentiation from competitors • Competitor pricing
Opportunities	**Threat**
• Expanding the business to reach more audience from new geographical locations • New and innovative ways to develop and reach the travelers. • Offering personalized and customized travel services to clients	• Competition from online travel booking platforms and price comparison websites • Fluctuating currency exchange rates • Changes in government regulations or travel restrictions • Natural disasters, health crises, or other unforeseen events that could impact travel demand

• Developing partnership with co-service providers and expand the business and brand recognition	• Increasing costs of travel-related services, such as fuel prices and hotel rates.

Table 3-SWOT Analysis

2.5 Requirement Engineering

Requirement engineering is a process of eliciting, analyzing, specifying, validating, and managing the requirements of a software system. It involves understanding the needs of stakeholders, including end-users, customers, and business owners, and translating those needs into software requirements that can guide the development process.

The goal of requirement engineering is to ensure that the software system being developed meets the needs of its intended users and stakeholders. It involves a range of activities, including gathering and analyzing user needs, creating use cases, and defining functional and non-functional requirements [26].

2.5.1 Functional requirements

Id	Description	Priority
FR1	**Title:** Service provider/ Travelers Registration Requirements Any anonymous user should see a “Register Account” button to register a user account.	High

	When the user clicks the “Register Account” button, the application navigates to the Register Account page. The “Register Account” page displays a registration form with the required details: Email address should be validated against proper email structure and existing emails. If the email doesn’t have the format user@domain.com, a validation message should be displayed which says, “Improper email address”, and if the email already exists in the system the error message should be “User with the given email already exists, please login.” The “Submit” button must be disabled until all mandatory fields are filled and validated. The email ID should be validated to avoid duplicating. Travelers can create one account with one email address. And Travelers should create their accounts through the mobile app.	
FR2	**Title:** User registration When the user clicks “Submit” on the registration page, the system should create a new user account and the service providers and travelers can login to the system. Service providers can log in to the web portal and publish their services.	High

	Travelers can login to the mobile app and explore their needs and services published by the service providers.	
FR3	**Title:** User Login When an anonymous user visits the website, they can click on the login button to access the login page. On the login page, they will see a form with fields for entering their username and password. They can also see a login button and a cancel button. If the user clicks the cancel button, they will be taken back to the home page. If the user enters an incorrect username or password and tries to log in, they will see an error message that says "Invalid username or password". If they enter the correct login credentials and click the login button, they will be successfully logged into the system and redirected back to the home page.	High
FR4	**Title:** Common Site Layout There are different layouts for the service provider which is a web application and a mobile application for the travelers. Web application: • Login Screen • Profile • Services grid • Create Service form • Reservation list Mobile Application:	High

	• Login screen • Profile • Feed • Services feed • Reservation options	
FR5	**Title:** Reservation Traveler should be able to access all the services published by the service provider. Traveler should be able to view each service in a detailed list. If the traveler wishes to reserve the service, should be able to click on "Book Now" button and reserve the service. Once the traveler clicked the "Book Now" a dialog message should be popup and request the user to download the receipt. On successful reservation the reservation details should pass to node.js web application and create a table with the details	High

Table 4 Functional Requirements

2.5.2 Nonfunctional requirements

Performance Requirements

Id	Description	Priority
P.NFR1	All images stored on the server file system must be encoded with WEBP file format and they must stay within 20KB and 100KB file size for optimal transfer.	HIGH
P.NFR2	All CSS and JavaScript assets must be minified and bundled for optimal size.	HIGH

Table 5 Performance requirements

Privacy and Security Requirements

Id	Description	Priority
S.NFR1	All usernames and other sensitive user information must be encrypted.	HIGH

Table 6 Privacy and security requirements

Software Quality Requirements

Id	Description	Priority
Q.NFR 1	Source code should have a unit test coverage of above 90%	MEDIUM
Q.NFR 2	It is advised that developers must implement features with testing in mind, therefore they may follow a suitable test-driven development methodology.	MEDIUM

Table 7 Software Quality Requirements

2.5.3 Feasibility Study

A feasibility study is the phase of looking into an idea, project, or business's viability or sustainability of the project. The research looks at whether there are sufficient and what is feasible resources to involve in to the project and whether the idea has the potential to make profit. Also, it will show caste the rewards for taking the chance to invest in the concept [27].

We have to look in how feasible below sectors are,

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility
- Schedule Feasibility
- Legal and Ethical Feasibility

Technical Feasibility

The Leisure Diary project which contains and web application and a mobile application. The feasibility study evaluates the if the system will show is performance and functions as mentioned. Technical feasibility also includes evaluating the availability of skilled personnel, necessary equipment, materials, and infrastructure to carry out the project. And the ability to build the said products. The leisure diary web application will be built with the association of the technologies mentioned in the resource requirements. The Technical Feasibility evaluates how the technologies are used in collaboration with the required skills. Most of the technologies are free for to use, therefore it will save huge sum of the cost for the company when building the project. As an important factor that the technologies currently have used to build the Leisure Diary web application and the mobile application has the potential of enhancing the further and add new features. Finally, Technical feasibility assessment is an important element of a feasibility study, as it helps to ensure that the project can be executed efficiently and effectively to achieve the desired outcomes as mentioned above.

Operational Feasibility

Operational feasibility phase is used to evaluate the ability of the Leisure Diary to be implemented and delivered successfully in an operational perspective. In this phase Leisure diary were evaluated if the web and mobile applications are workable within the specified audience and achieve its said functions, goals and values. The operational evaluation also takes into

account the influence of the project on the organization's current procedures and workflows. These elements include the availability of staff, the skills, and training needed to conduct the project successfully. The project's compliance with the organization's culture, regulations, and procedures as well as the availability and accessibility of resources like technology, infrastructure, and money are further considerations. Operational feasibility is important for establishing if a project or business endeavor can be carried out and sustained over the long term, and it influences stakeholders' decision-making greatly. Hence, analytics confirms that the Leisure diary can sustain on above requirements, can be conclude that the Leisure Diary project is operationally feasible.

Economic Feasibility

Planning is necessary for an organization to accomplish one of its primary goals, which is to expand and become profitable. The economic feasibility analysis is one of the most crucial elements in this process of investment selection and prioritization. In addition to establishing if a particular approach is profitable, this analysis seeks to determine whether it is economically viable or not. Since resources are limited and there are other financial priorities in the competition for investments with higher returns, it will be necessary to conduct a careful analysis before adding an initiative to the portfolio, even if it is creative, innovative, and can produce excellent results for some stakeholders. The economic feasibility analysis is a study that will compare the returns to be obtained with the investment required to point out its feasibility, being, therefore, an extremely important process, which brings business intelligence, avoids losses, and leads to more assertive decisions [28]. Service providers and customers can access the Leisure Diary from anywhere. They can signup and publish their services with no cost. From the companies end most of the technologies are free for to use, therefore it will save huge sum of the cost for the company when building the project. Only cost the company would be bearing the cost of hosting, developer salaries and the service payment for google cloud console.

Technical Requirements

Table 8 Technical Requirements

Technical Requirements		
Unit	Qty	Price
Personal Computer	1	80,000
Mobile Phone	1	65,000
Human Resource Requirements		
Business Analysist	1	50,000
Web Developer	1	80,000
Mobile App developer	1	80,000

Leisure Diary is planning to charge on a commission basis from the service provider and travelers as service charge from each service they render from the system. Hence, the business will grow as sooner the Leisure Diary get promoted.

Schedule Feasibility

A project can be implemented within a reasonable time frame if the schedule is feasible. If a project is not finished within its allotted time range, it may occasionally not be successful. Here, we can estimate how long it will take to accomplish each project activity. The Leisure Diary project can now be finished before it becomes superfluous or obsolete thanks to the schedule feasibility study. The Leisure Diary project's timetable viability was evaluated taking into account how time and money interact. The Leisure Diary project is expected to take 4 months to fulfill its many duties. Thus, the project schedule was determined to be feasible by taking into account the project activities and time need. The author has created a Gantt chart that details the project's many tasks and the time needed to finish each one.

Legal/Ethical Feasibility

Legal and ethical feasibility in software engineering refers to a software system's capacity to abide by relevant laws, rules, and moral guidelines. This includes making sure the system complies with all security, privacy, and intellectual property rules. To make sure that the system does not hurt its users or other stakeholders, ethical factors including fairness, openness, and accountability must also be taken into account. Due to the potential for negative legal and reputational repercussions for the firm, legal and ethical feasibility is essential in software engineering. In order to guarantee that their software systems are conceived, created, and maintained in compliance with applicable laws and ethical standards, software developers and engineers must be aware of legal and ethical factors [29]. By having confirmed that, the Leisure Diary website would not have to face any legal issues.

While studying the legal and ethical requirements, the privacy, nepotism, and accountability were considered as well. Leisure Diary utilizes freely available development tools, and provide the system as an open-source system. Only the service commission will be charged from customers and service providers. Software libraries that are utilized in this system are free open-source libraries. Hence, the author concluded that the Leisure Diary project is legally and ethically feasible.

2.6 Resource requirements

Software requirement	• Visual Studio code editor • Windows 7 or above • MongoDB/Firebase • Android studio
Hardware Requirements	• Personal Computer • 8GB RAM or higher • 500GB HDD

	• 1GB Video Graphics card or Higher
Client requirement	• Mobile phone with internet connection • A PC with internet connection
Technological requirements	• HTML • CSS • JavaScript • Flutter • MongoDB / Firebase • EJS

Table 9 Resource requirements

2.7 Outcome of the analysis

The requirement elicitation is the primary objective achieved by the analysis process. User stories and use case diagram assisted to get a clear picture of the actual scenario and to identify the flow of the system. Clear problem identification phase helped to elaborate the actual problem and align it with the requirement identified in the early phrases. Along with that we have conducted a SWOT analysis which is the best way to identify Strengths, Weakness, Opportunities, and Threats in the related field and forecast the future. As a summary of the requirement elicitation, we have finalized the functional requirements and non- functional requirements to implement the system.

3. Design

3.1 Design Techniques

Design is the highly creative and important stage in software development process. Design process will be held iteratively through entire development process in different number of ways. A good system design will arrange the program modules so that they are simple to create and modify. Developers can manage the size and complexity of programs with the aid of structured design strategies. The developers are given guidance by analysts on how to write code and how to put parts of code together to create programs. The process of developing, creating, testing, and maintaining software is known as software engineering. The creation of software that is dependable, effective, and simple to maintain is the aim of software engineering. Making choices about a software system's architecture, components, modules, interfaces, and data is a crucial aspect of system design in software engineering.

3.1.1 Flow Chart

A flowchart is a visual representation of a process or algorithm using various symbols and shapes to show the sequence of steps and decisions involved. It is a useful tool for understanding and documenting complex processes or systems. Each symbol in a flowchart represents a specific action or decision point in the process, and arrows connect the symbols to show the flow of the process. Flowcharts are used in a variety of fields, including software development, engineering, and business. They help to simplify complex information and make it easier to understand [30].

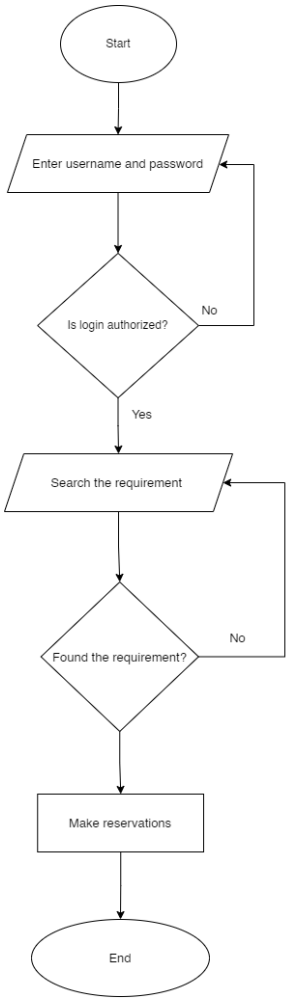


Figure 11 Travelers flow chart

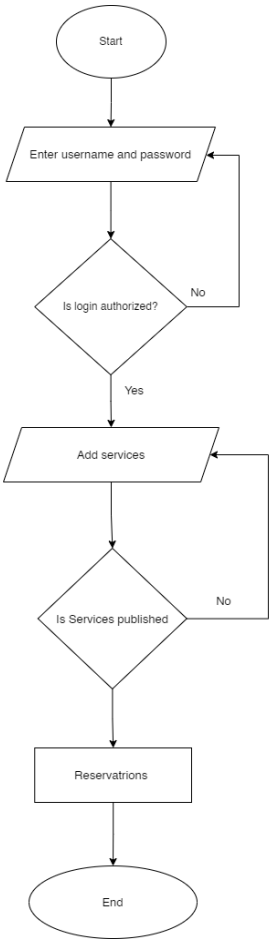


Figure 10 Service provider Flow chart

3.1.2 Use Case Diagram

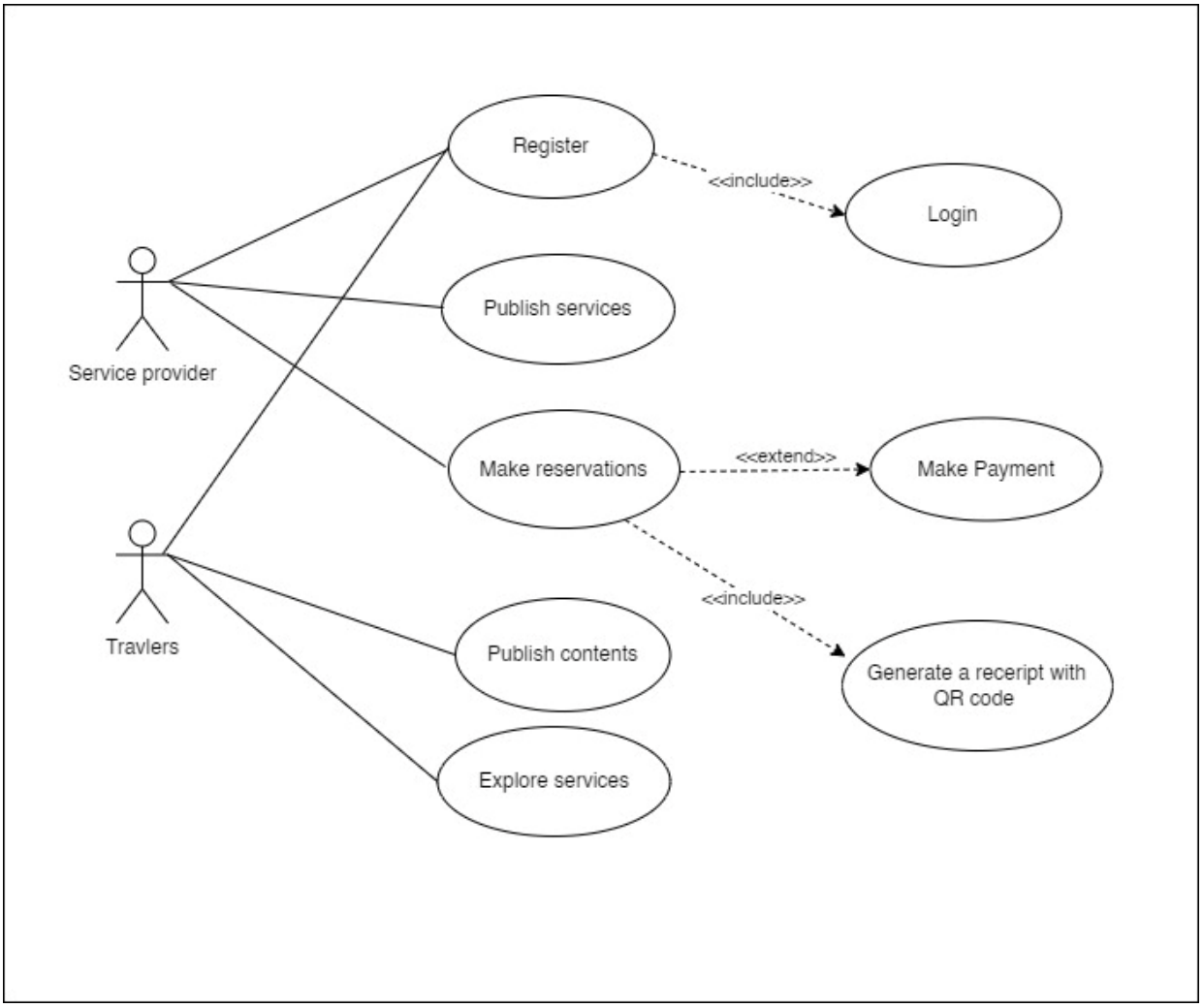


Figure 12- User case diagram

3.1.3 ER Diagram

An Entity Relationship Diagram (ERD) is a type of flowchart that illustrates the relationships between entities such as objects, people, or concepts in a system. ERDs are often used to design and debug relational databases in various fields such as software engineering, business information systems, education, and research. These diagrams use a specific set of symbols, including diamonds, rectangles, ovals, and connecting lines, to represent the interconnectedness

of entities, relationships, and their attributes. ERDs follow a grammatical structure where entities are represented as nouns and relationships as verbs. The main purpose of an ERD is to show the infrastructure of the entity framework, as illustrated in the example of the leisure diary project.

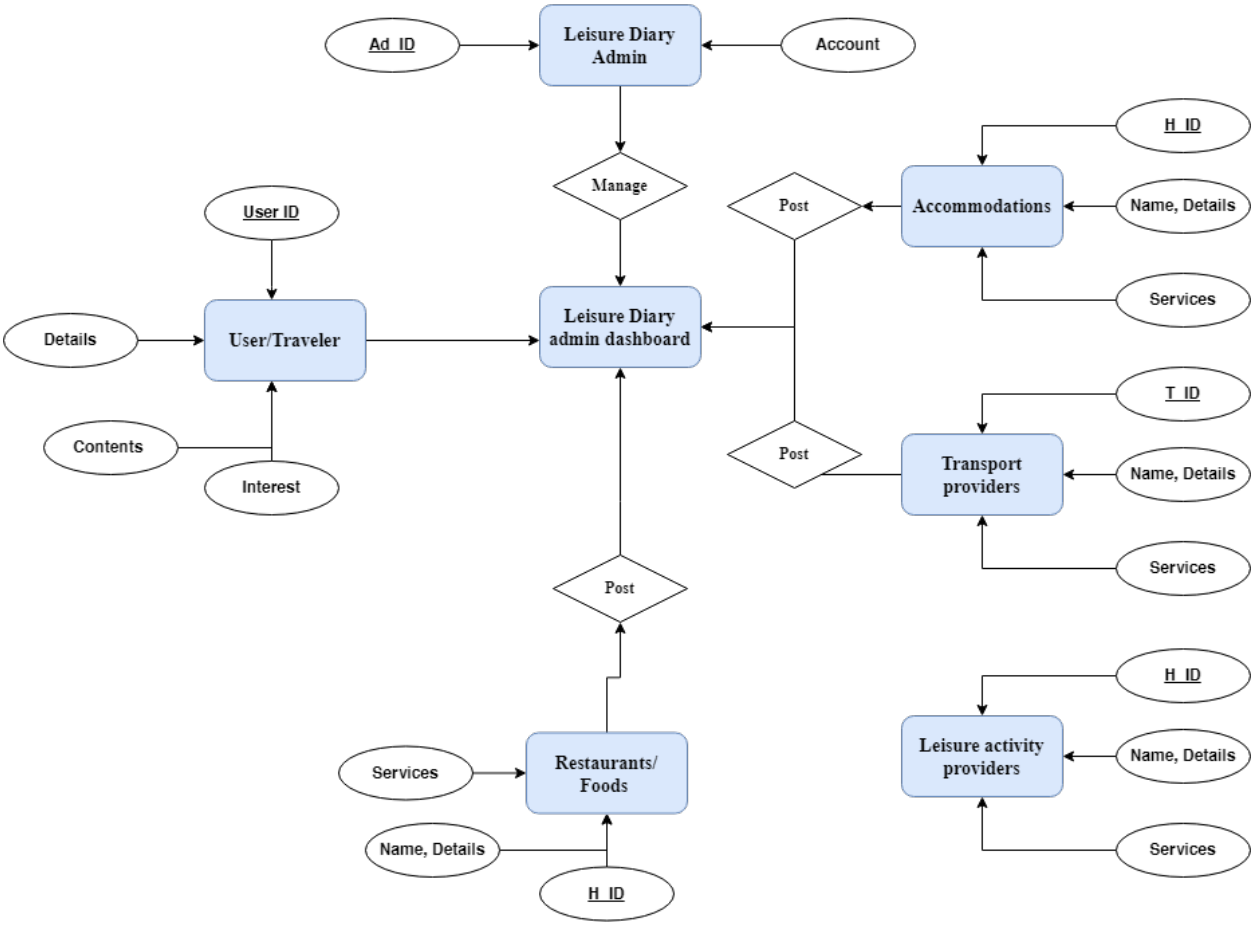


Figure 13 ER Diagram

3.1.4 High Level System Component Diagram

Below is the high-level structure of the application. The backend consists of a REST API coupled with two databases. A HTTP port and a TCP port is exposed for external communication interfaces. WebSocket communication shares the same HTTP port.

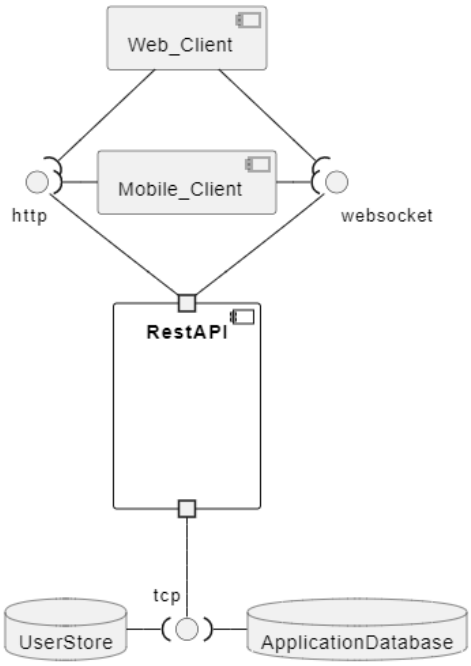


Figure 14 System component diagram

3.1.5 Data Flow Diagram

Data flow across a system is shown visually in a DFD (Data Flow Diagram). It is frequently used for information system analysis, design, and documentation. Below is the illustration of Leisure Diary's Level 0 Data Flow Diagram which describes the scenario including web and mobile applications:

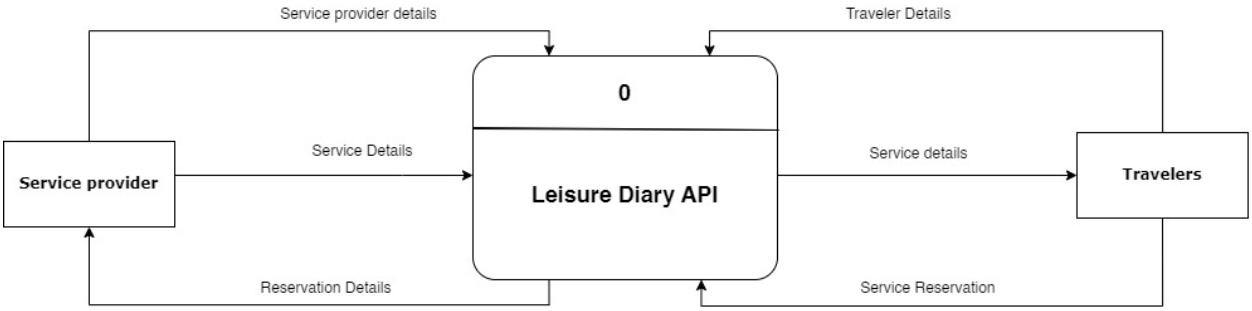


Figure 15- Data Flow Diagram Level 0

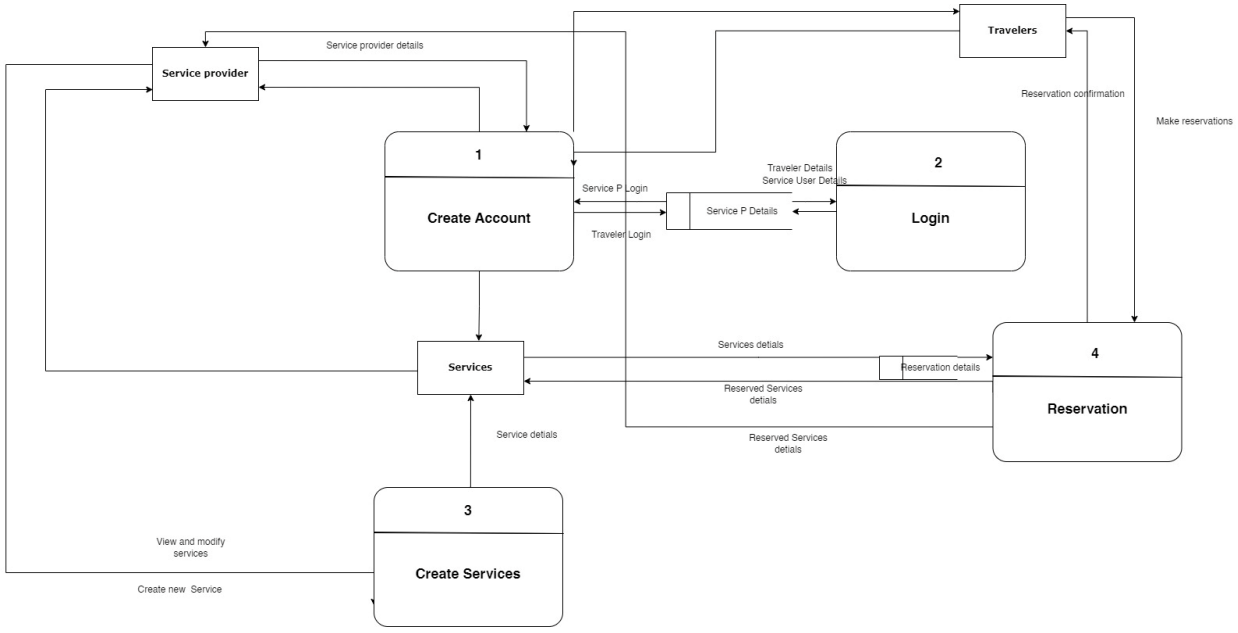


Figure 16 - DFD level 1

3.1.6 Class Diagram

A class diagram in the UML (Unified Modeling Language) is a form of static structural diagram used in software engineering that depicts the classes, properties, operations (or methods), and interactions between objects to represent the structure of a system. The class diagram created to create the Leisure Diary Web & Mobile application is shown below.

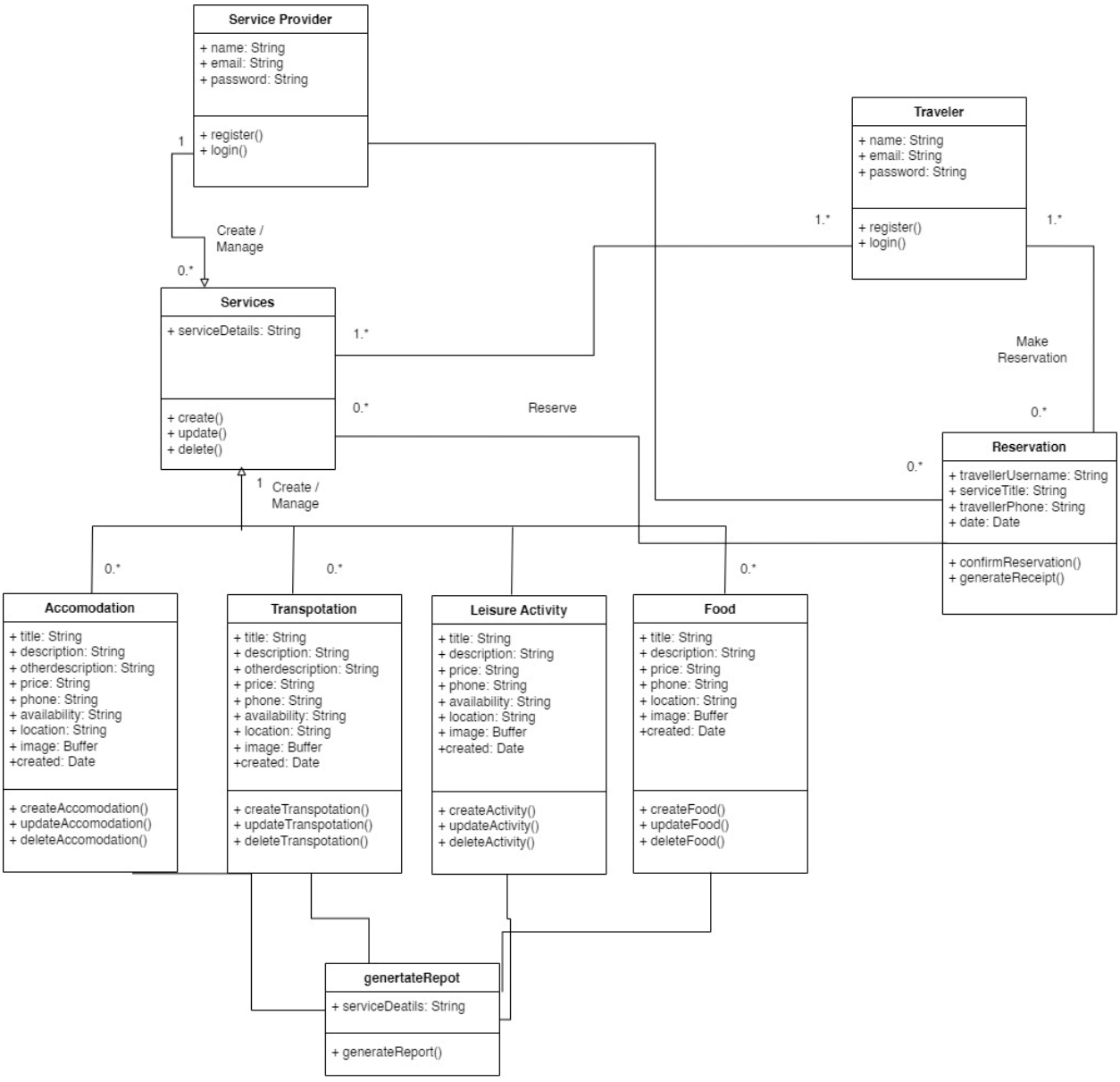


Figure 17- Class Diagram

3.1.7 Activity Diagram

A sort of Unified Modeling Language (UML) flowchart called a "activity diagram" depicts the progression of one activity across a system or process. It is referred to as a "behavior diagram" because it outlines what ought to occur in the modeled system and is used to depict the various

dynamic characteristics of a system. Activity diagrams can be used to visualize even extremely complicated systems. As a result, activity diagrams are frequently employed in organizational business process modeling or to outline the processes of a use case diagram. They display each step in an activity along with their relative timing. They can also display the data flow between several activities [31].

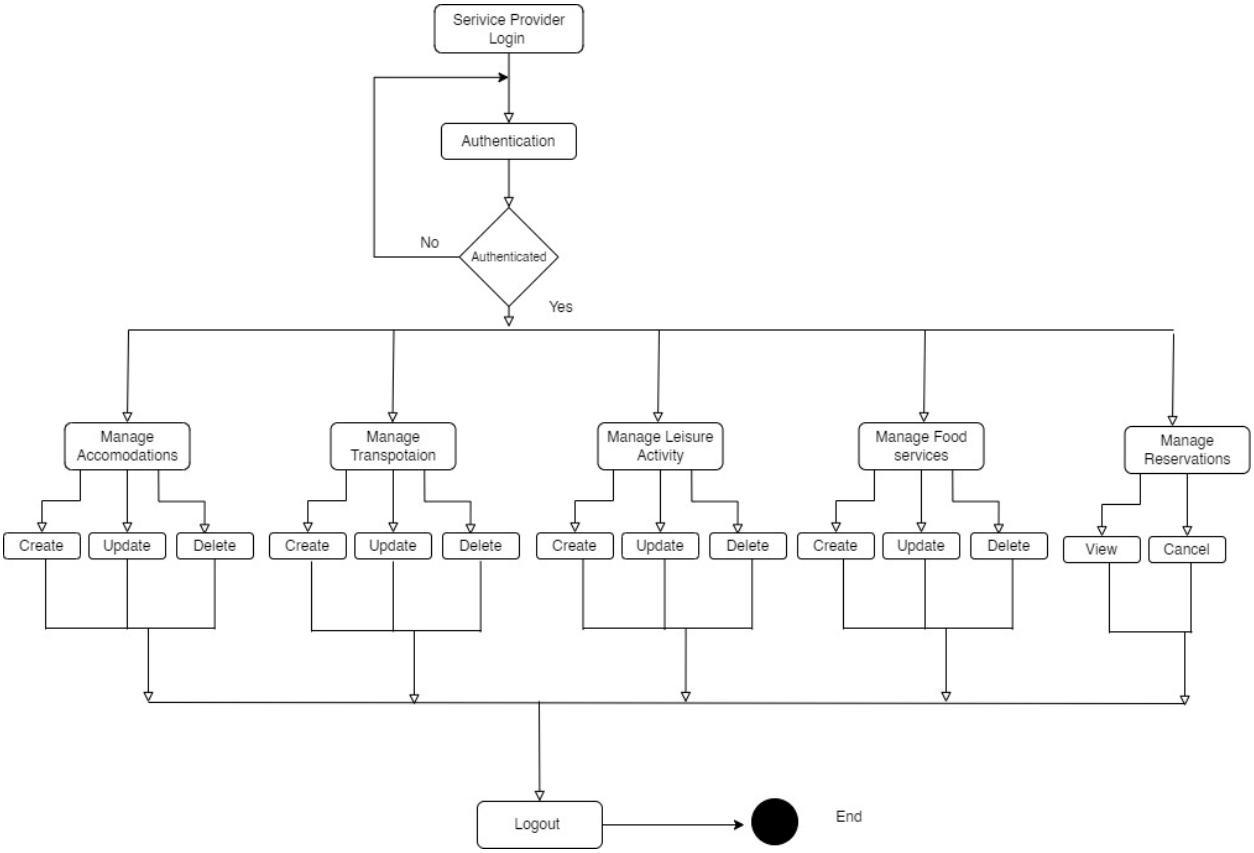


Figure 18- Service provider Activity Diagram

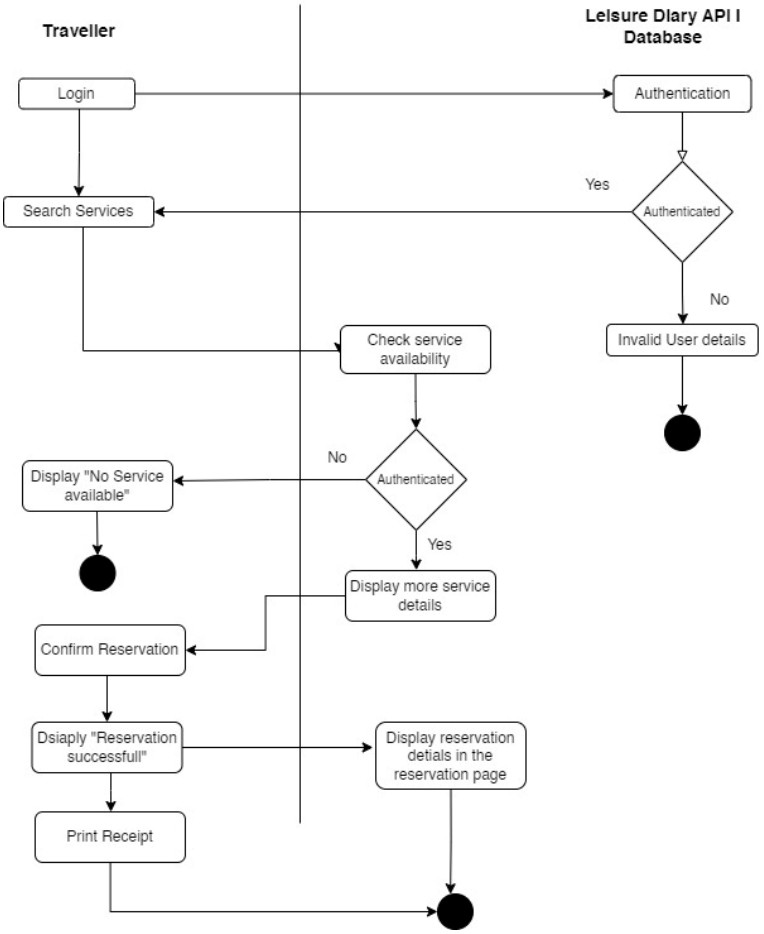


Figure 19 Traveler Activity Diagram

3.1.8 Sequence Diagram

A sequence diagram is a diagram created using the Unified Modeling Language (UML) that shows the flow of messages sent and received by objects during an interaction. A group of objects that are represented by lifelines and the messages they exchange over the course of an interaction make up a sequence diagram [32].

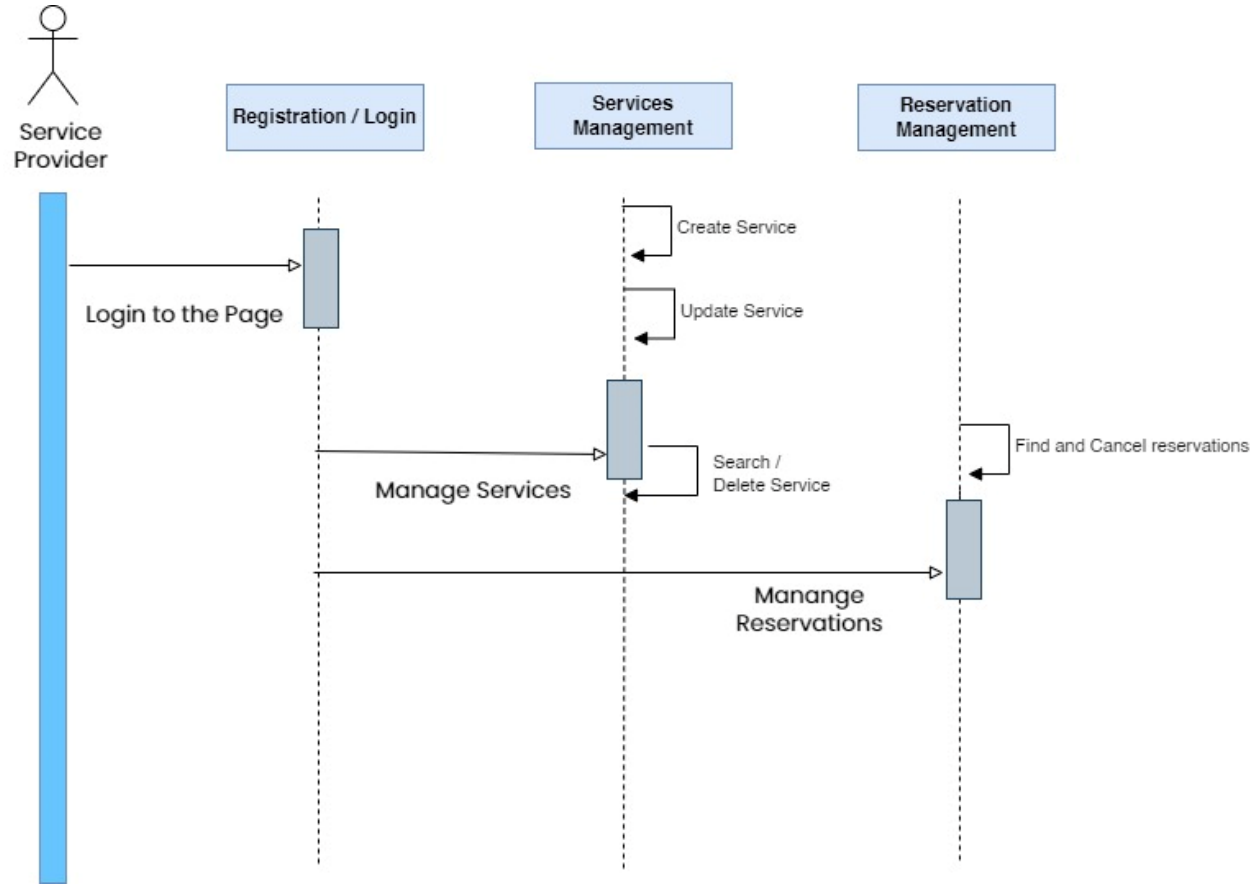


Figure 20 Sequence Diagram service provider

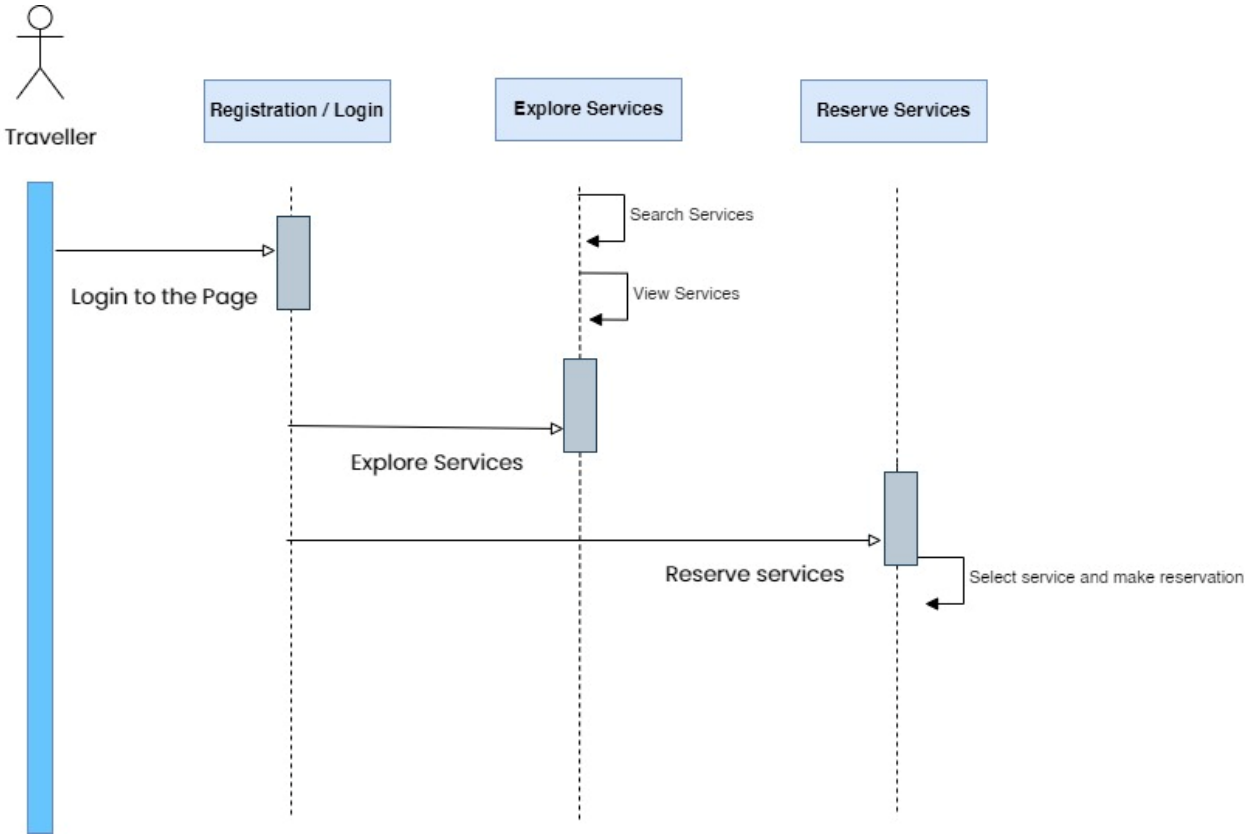


Figure 21- Sequence diagram - Traveller

3.2 System Overview

Leisure Diary is the proposed solution for the identified significant problems and requirements addressed by the travelers and other service providers. The solution is a web based mobile application connected to an admin panel through a rest API technology. Application will have many entities such as Location accommodators (Hotels and resorts), transport providers, foods, other restaurants and other leisure activities. The solution is like a story creator which attracts the user's interest with going forward with the application.

A user can register to the application by creating an account. The story begins the user can start to create the travel package. User can base locations, type of foods/restaurants or leisure activities as the initial point of selection. The application shows recommendations based on your interest. With

adding the no. of people to the system it filters the available options to proceed. After creating the whole story of the trip, you can share it among your co-travelers and other service providers on their approval of the reservations.

Data Models and Data Structure

A data model is a conceptual representation of the data in a system, and it defines the structure, relationships, and rules that govern how data is stored and manipulated. In your scenario, the data model would define the structure of the data that is created and managed by your web application. This could include information such as user accounts, service descriptions, and pricing information.

A data structure, on the other hand, refers to the organization of data within a system. It determines how data is stored and accessed, and it can have a significant impact on the efficiency and performance of your application. In your scenario, the data structure would determine how the data created by your web application is organized and accessed by your mobile application via the API.

To facilitate the exchange of data between your web application and mobile application, it is important to establish a clear and consistent data model and data structure. This will ensure that both applications understand the data being exchanged and can manipulate it effectively. Additionally, it is crucial to implement appropriate security measures to protect the integrity and confidentiality of the data being exchanged. By doing so, you can create a robust and reliable system that meets the needs of your users while maintaining the security and privacy of their data.

3.2.1 Wireframes

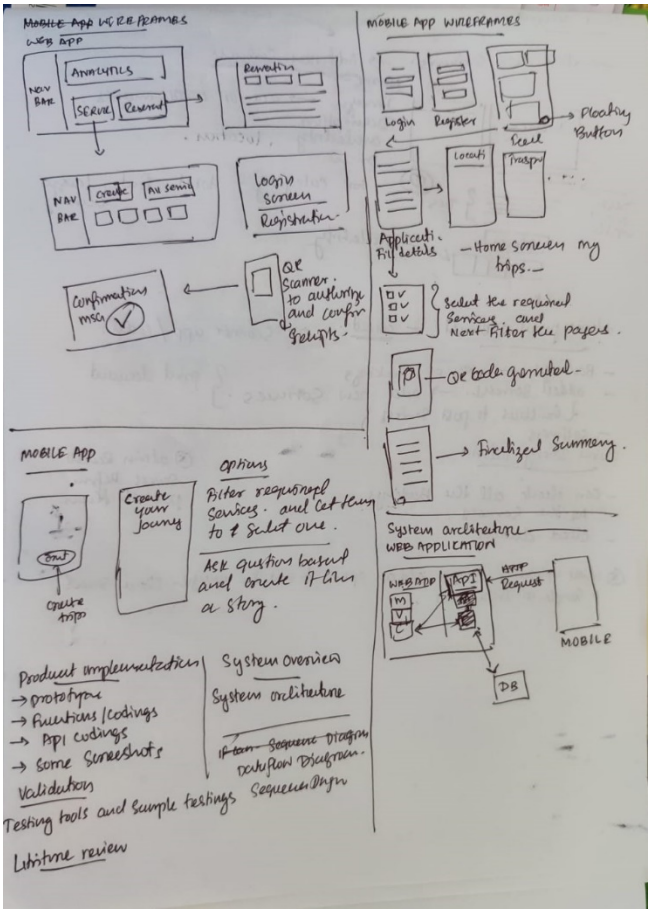


Figure 22 Wireframe Sketch

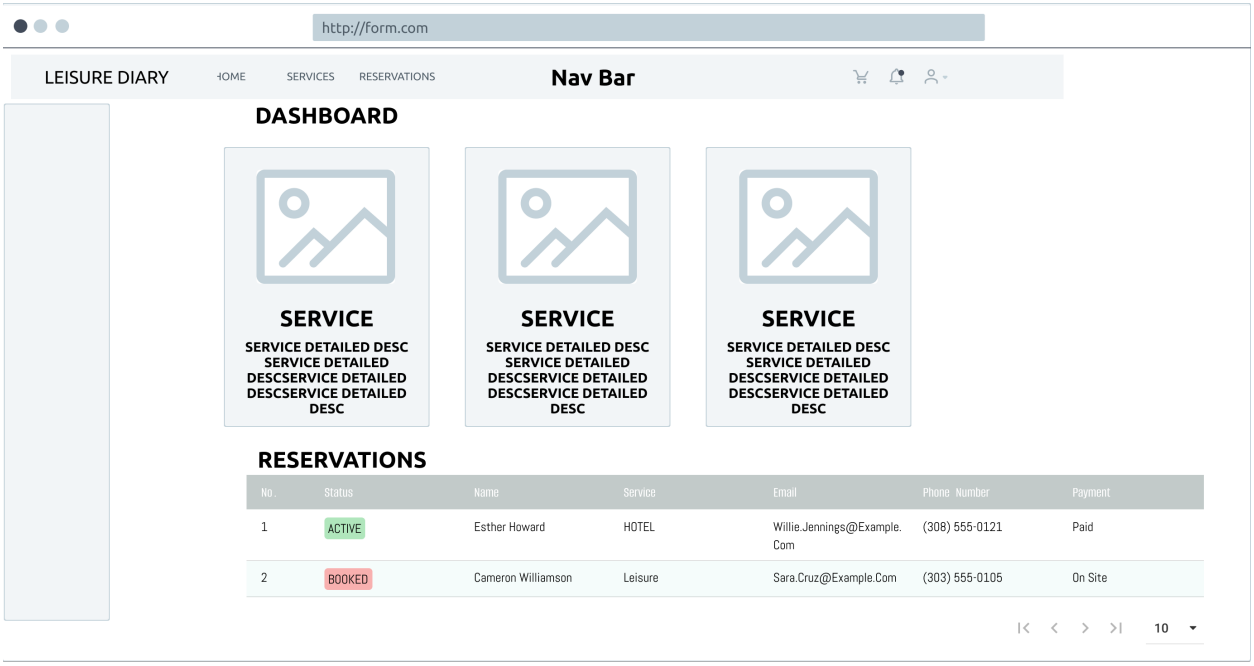


Figure 23 Wireframe

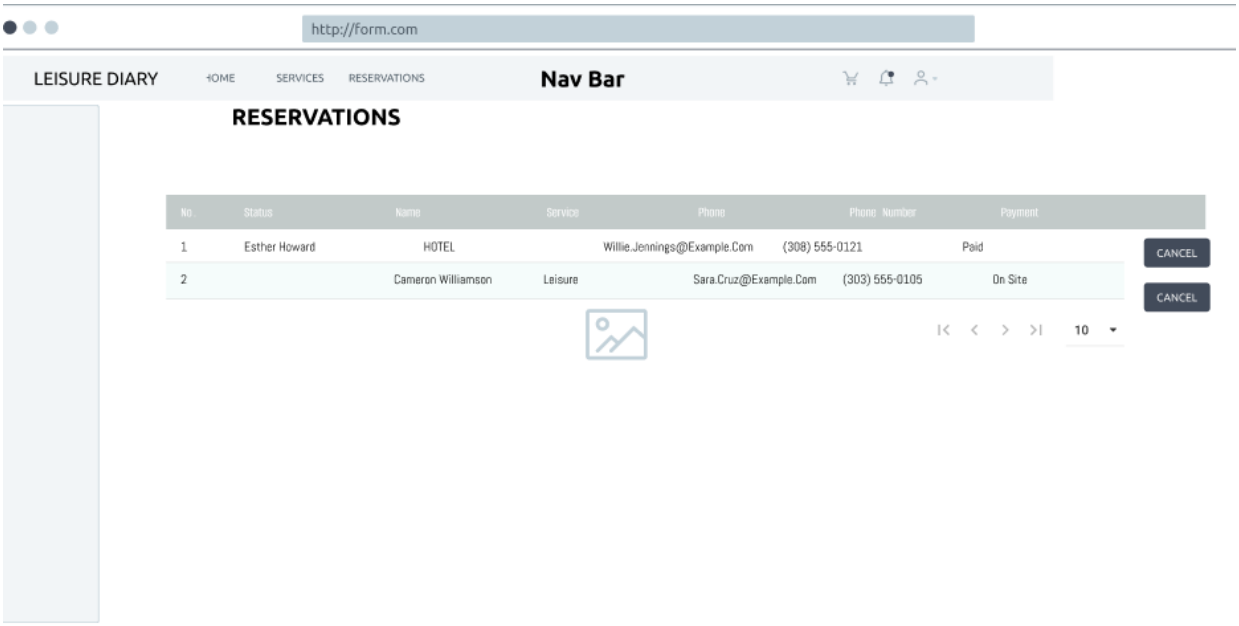


Figure 24 Wireframe

Figure 25 Wireframe

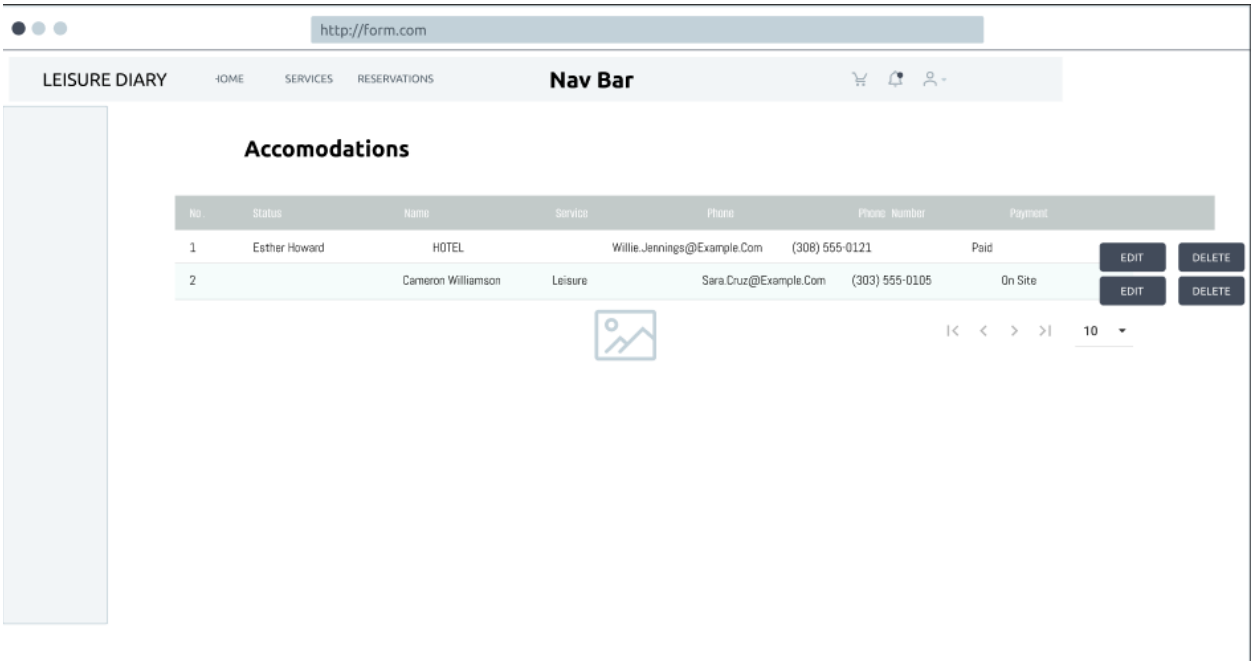
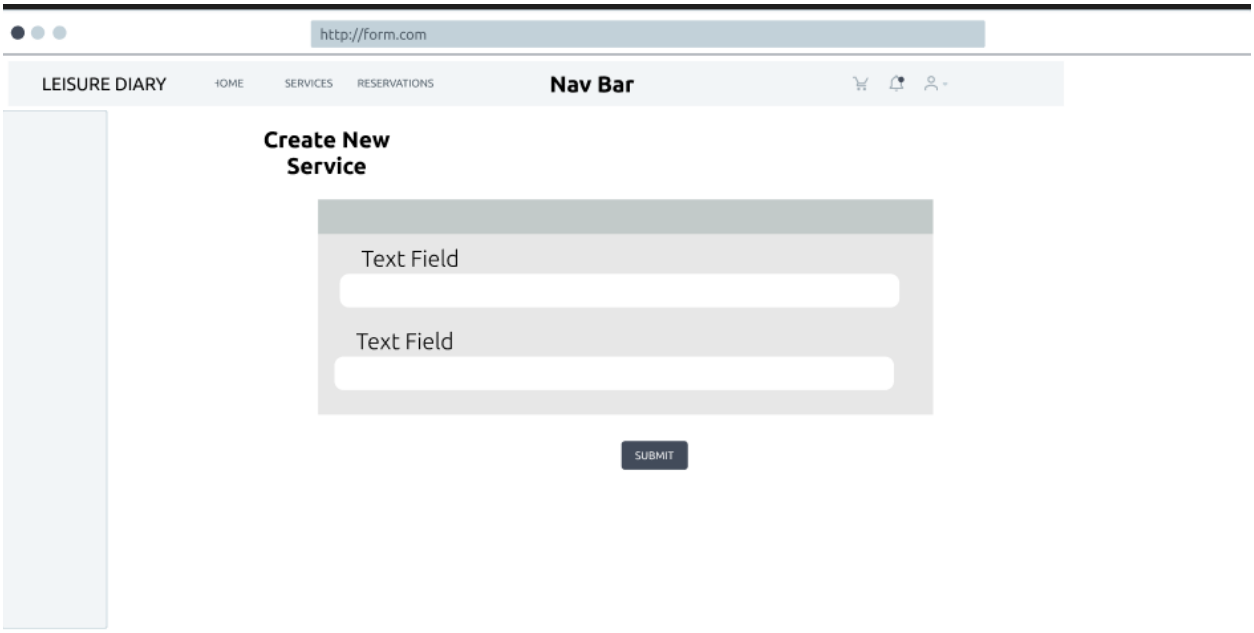


Figure 26 Wireframe form



Mobile App wireframes

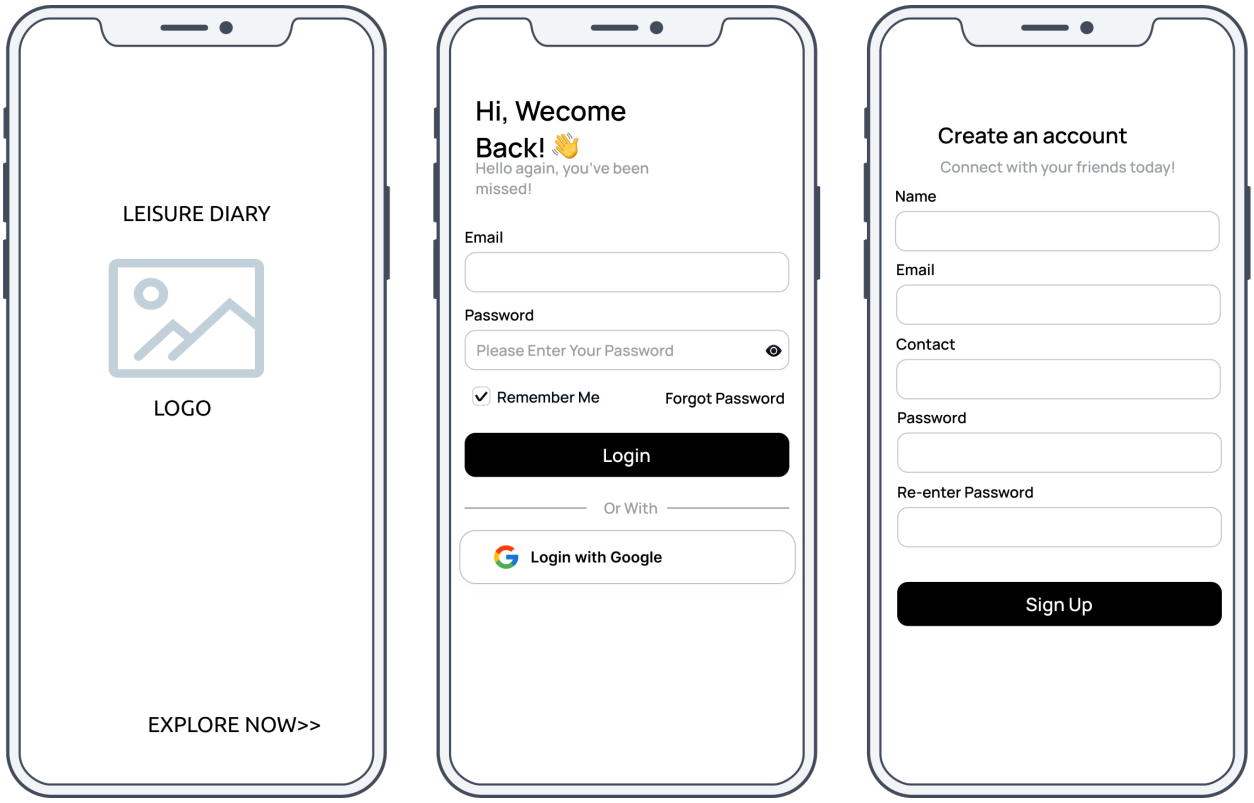
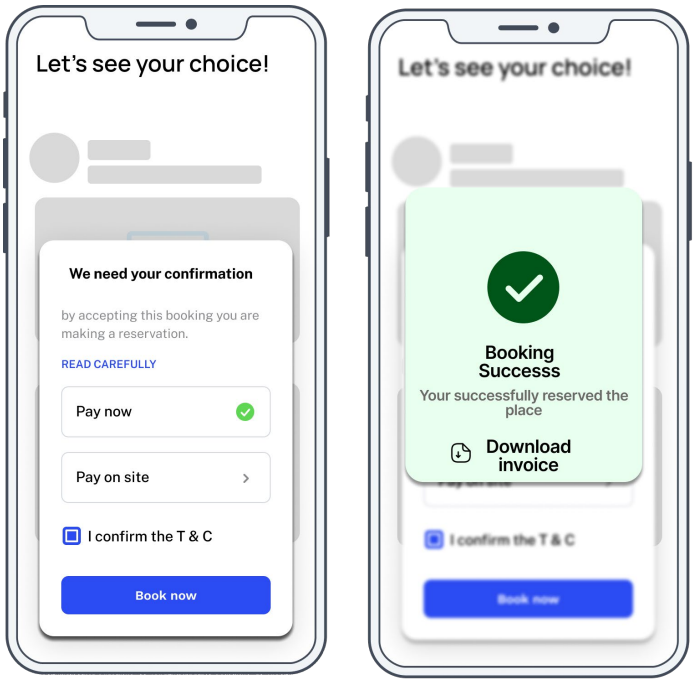


Figure 27 Wireframe


Create a trip +
Create your Journey
Let's see your choice!
Book now



3.2.2 System Architecture

The design of a complicated system that consists of hardware, software, network infrastructure, and other components is referred to as system architecture. It aids in ensuring that all parts perform properly by defining a system's structure, behavior, and functioning.

The system architecture is crucial because it gives stakeholders a high-level picture of the system and makes it possible for them to comprehend the connections and interactions among various components. Making wise decisions about trade-offs between cost, performance, and usefulness is made easier by early identification of potential problems and hazards [33].

As it indicates in the below architecture the web application creates the data and store it to an API which will be saved in the DB. The REST API will be used to communicate with the mobile application. API can directly access the model and the services of the web application and save the changes to cloud database.

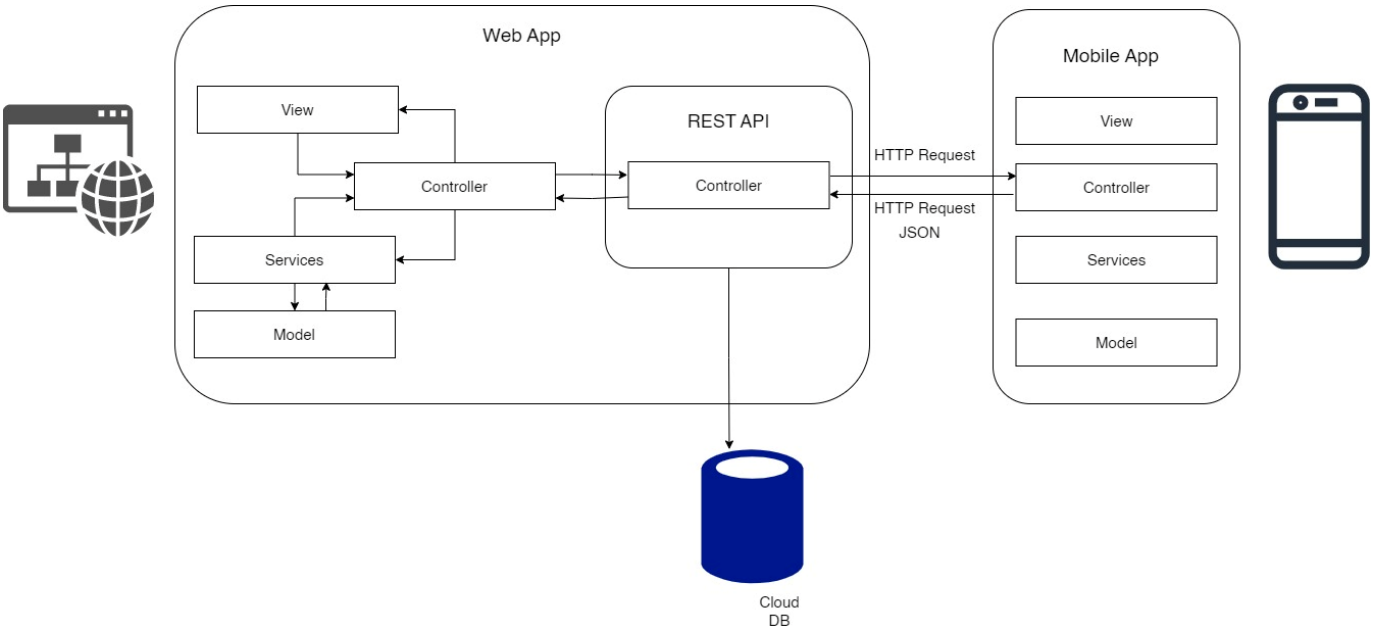


Figure 28 System Architecture

The system architecture of the Leisure Diary for web application and mobile application show cast the front-end interfaces are developed using the languages; HTML, CSS, and JavaScript and flutter for the mobile application. The users of the Leisure Diary system are able to manipulate data using the front-end interfaces of the web & mobile application.

MVCS architecture has been used to implement the leisure land web application. The model-view-controller-service (MVCS) design adds a service layer to the conventional MVC paradigm. This architecture encourages a modular method of web development, in which various components are divided according to their function. The Model is used to retrieve, store, and change data and is in charge of the application's data and business logic. The View, which can be any kind of user interface, such as a webpage, a mobile application, or a desktop program, is in

charge of showing the user the data. The Controller manages user input, converts it into actions, and manages the application's flow [34].

3.2.3 Actual Design

Actual Design contains two main applications as mentioned above. One is the web application which is used by the service provider to sign up and manage services.

Leisure Diary website

Leisure Diary has a web site where the user can search and land to the site. Site has Navigation panel with the links Home, About, Services, Blog and Contact form, where the visitors can quickly navigate to the particular page. Also, the user can create new account using Sign Up button and returning user can log in to the page using sign in button which has been displayed on the Nav bar.

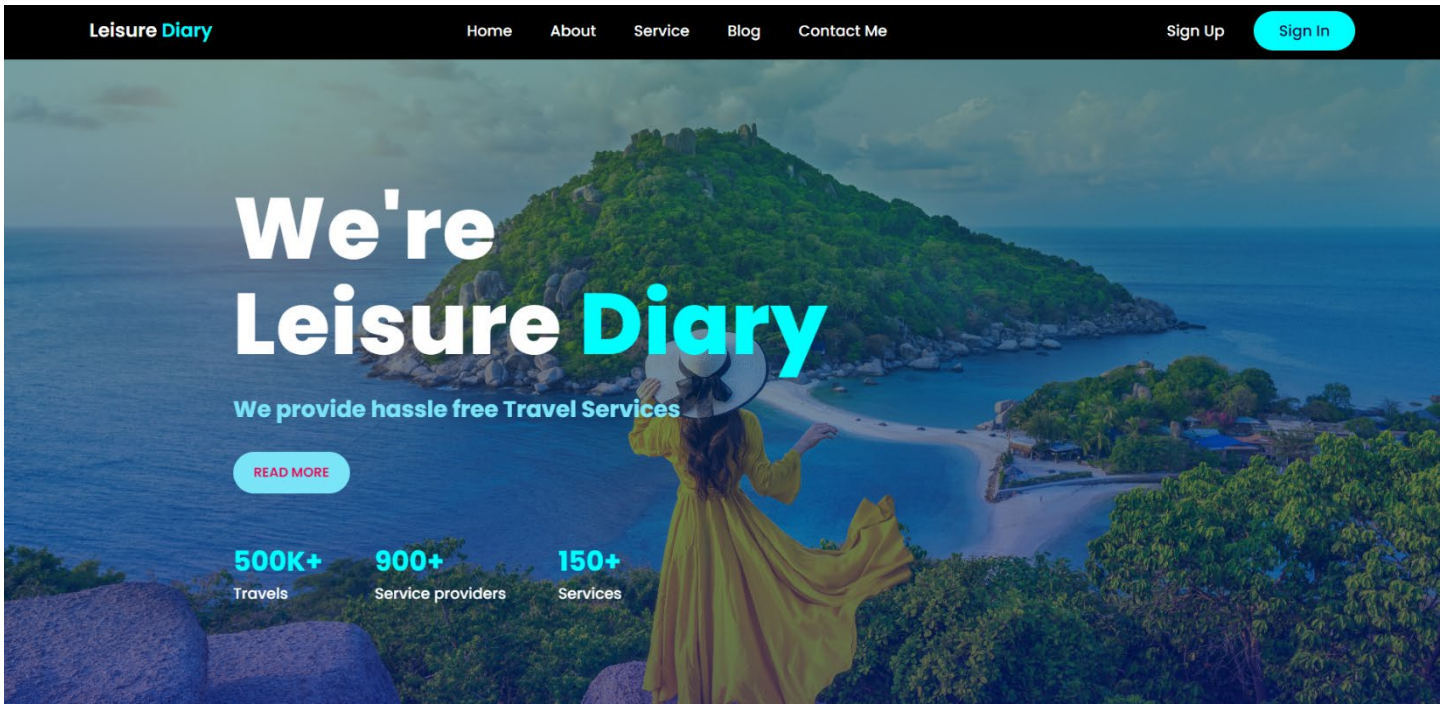


Figure 29 Leisure Diary website

About Leisure Diary

Traveller & Service provider

We provide hassle free travel experience for the **Travelers** as well for the travel related **Service providers**

MORE DETAILS

Figure 30 Website about page

Travel Related Services

Services Provided

Priority is Customer Satisfaction

Transportation

We porvide all island Transportation services for our Travellers

Accommodation

We porvide all island Accommodation services for our Travellers

Leisure Activities

We porvide all island Leisure Activities services for our Travellers

Food

We porvide all island Food services for our Travellers

Figure 31Service details page

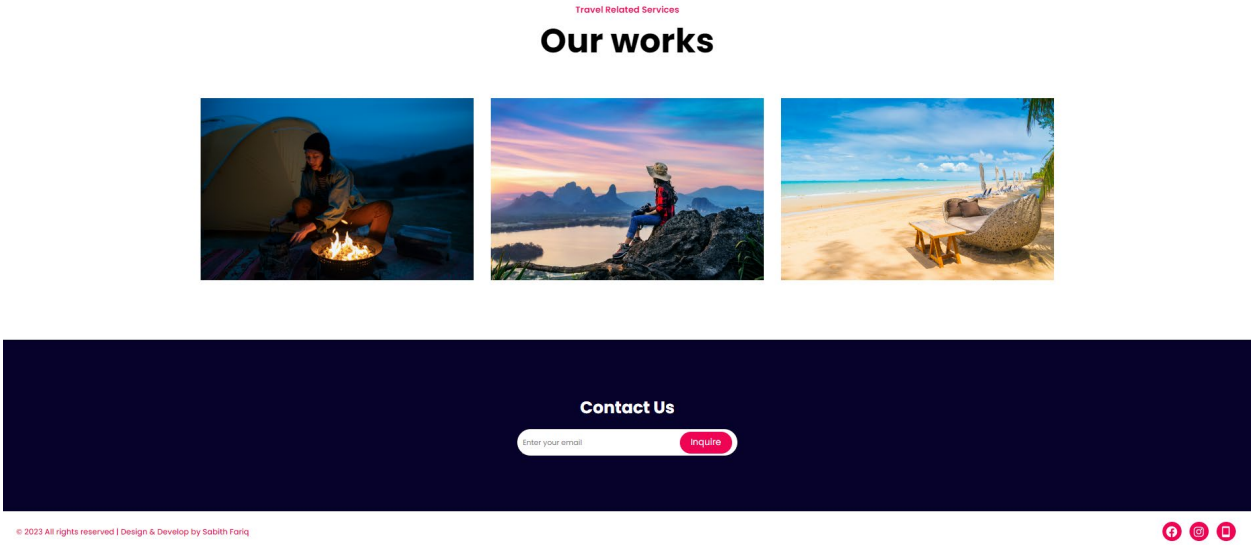


Figure 32 Our Works

Login Screen

A user will redirect to this login screen when they click on the sign in button in the website. User can use their username and password to authenticate their self to log in to the dashboard.

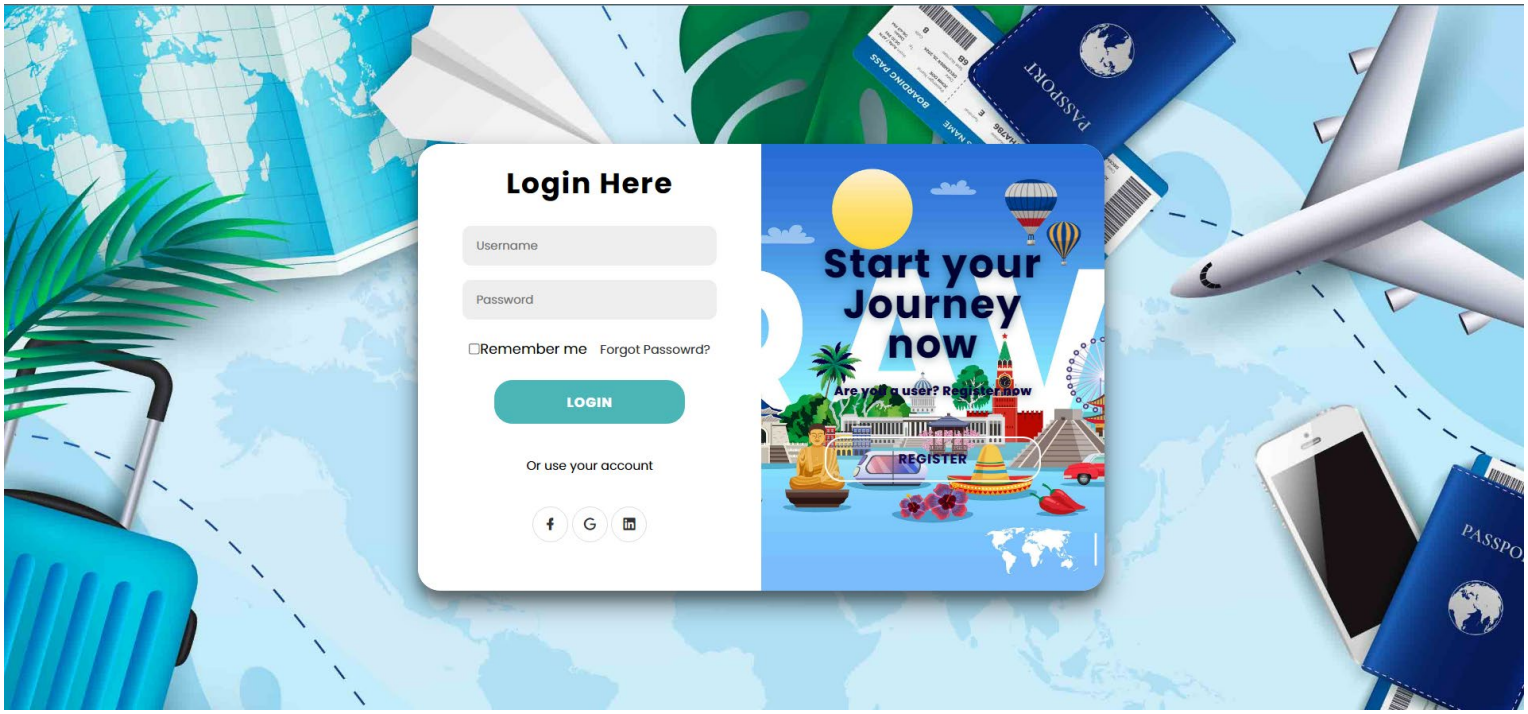


Figure 33 Login Screen

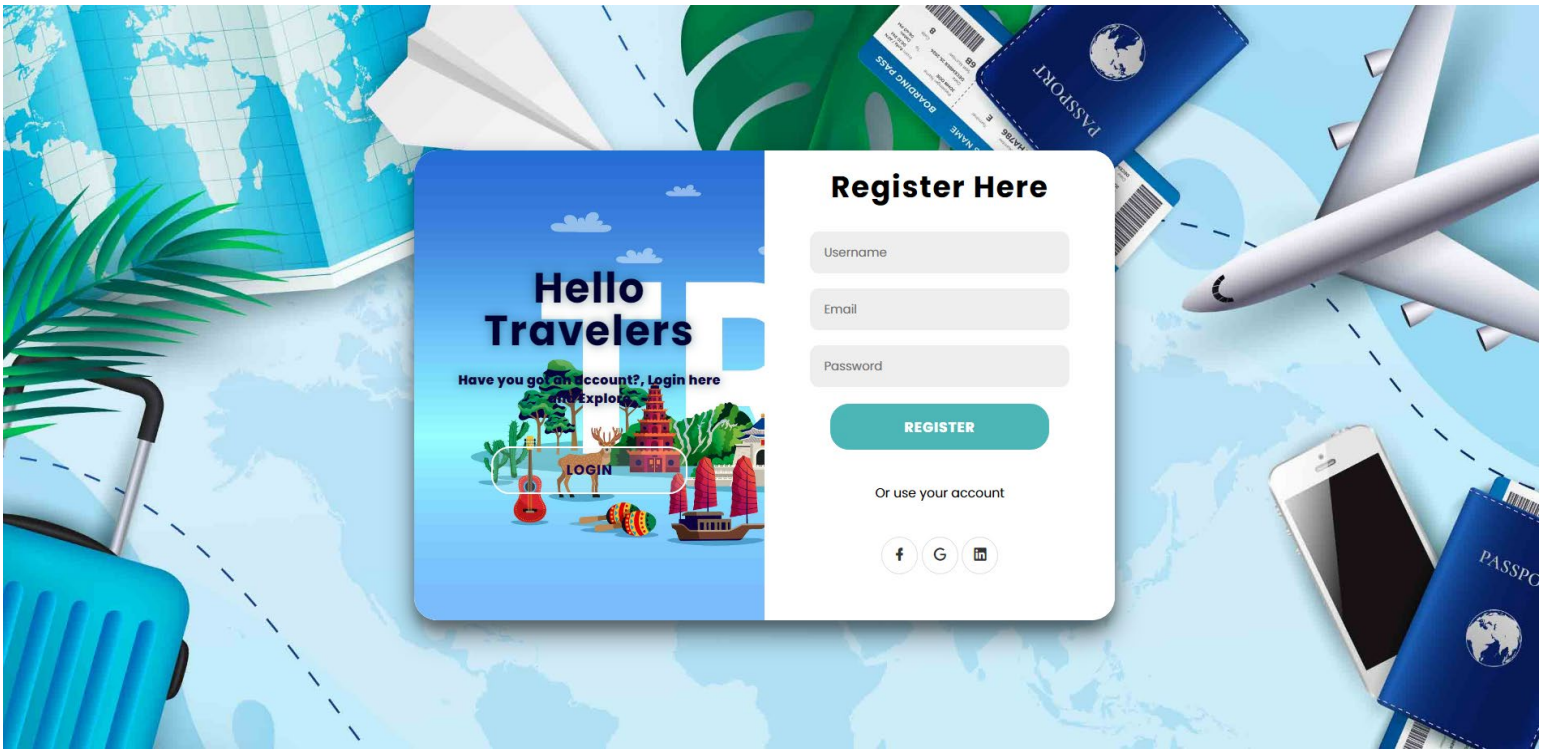


Figure 34 Registration page

User can provide their desired username, email and password to create new account. Password will be bcrypted and saved in the database. User can use the username and password to login to the system Dashboard.

Dashboard

User will be navigated to the dashboard once they authenticated from the login screen. Dash board has the available Services as cards. And the Links to each services.

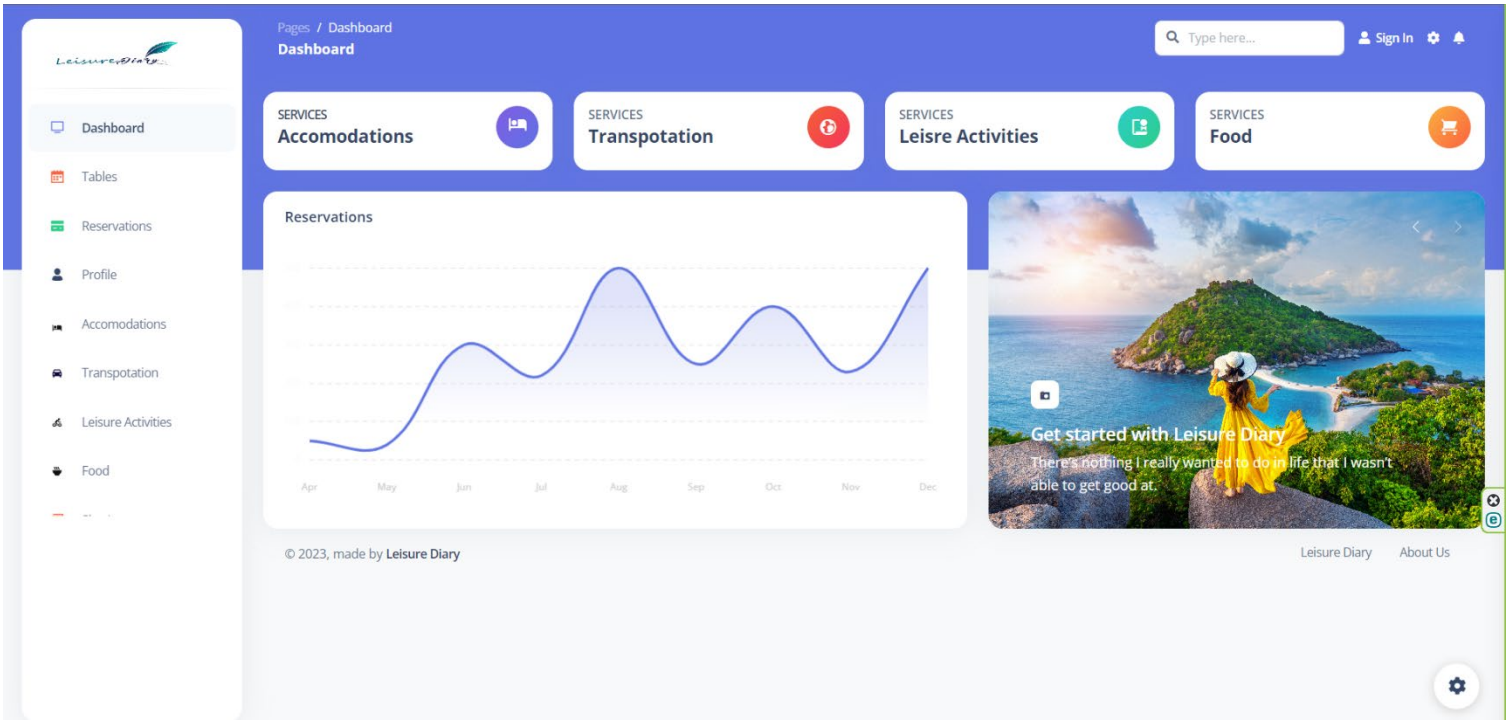


Figure 35- Leisure diary dashboard

Accommodations Dashboard

Accommodation dashboard will be empty when the new user got registered. It shows no services found as in the below snippet.

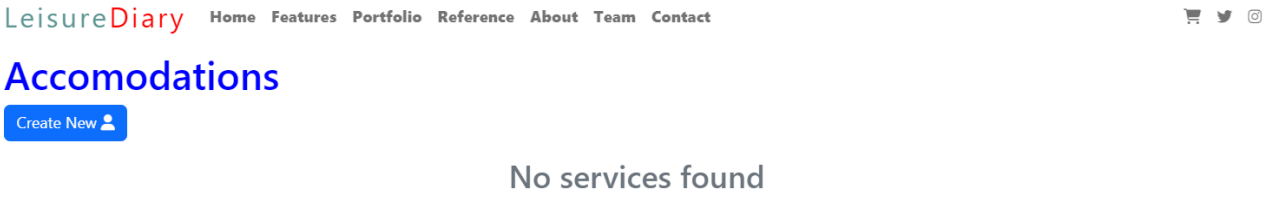


Figure 36- Accommodation Dashboard

Create new accommodation

Once we click the “Create new service” button a form will be displayed and all the details must be entered. Fields are validated against being null.

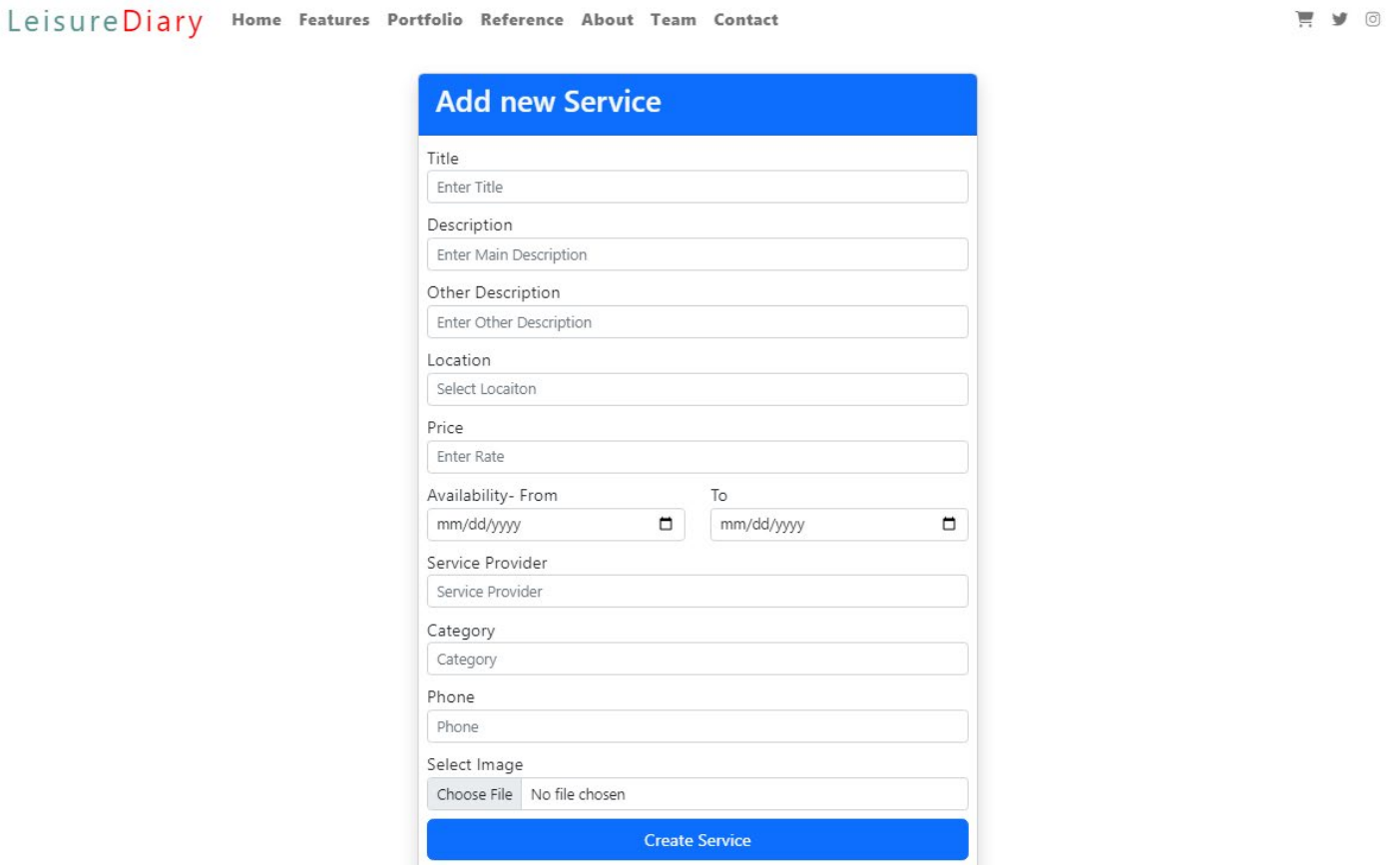


Figure 37 Service creating form

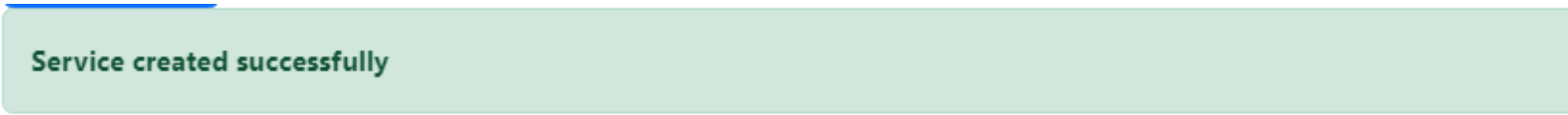
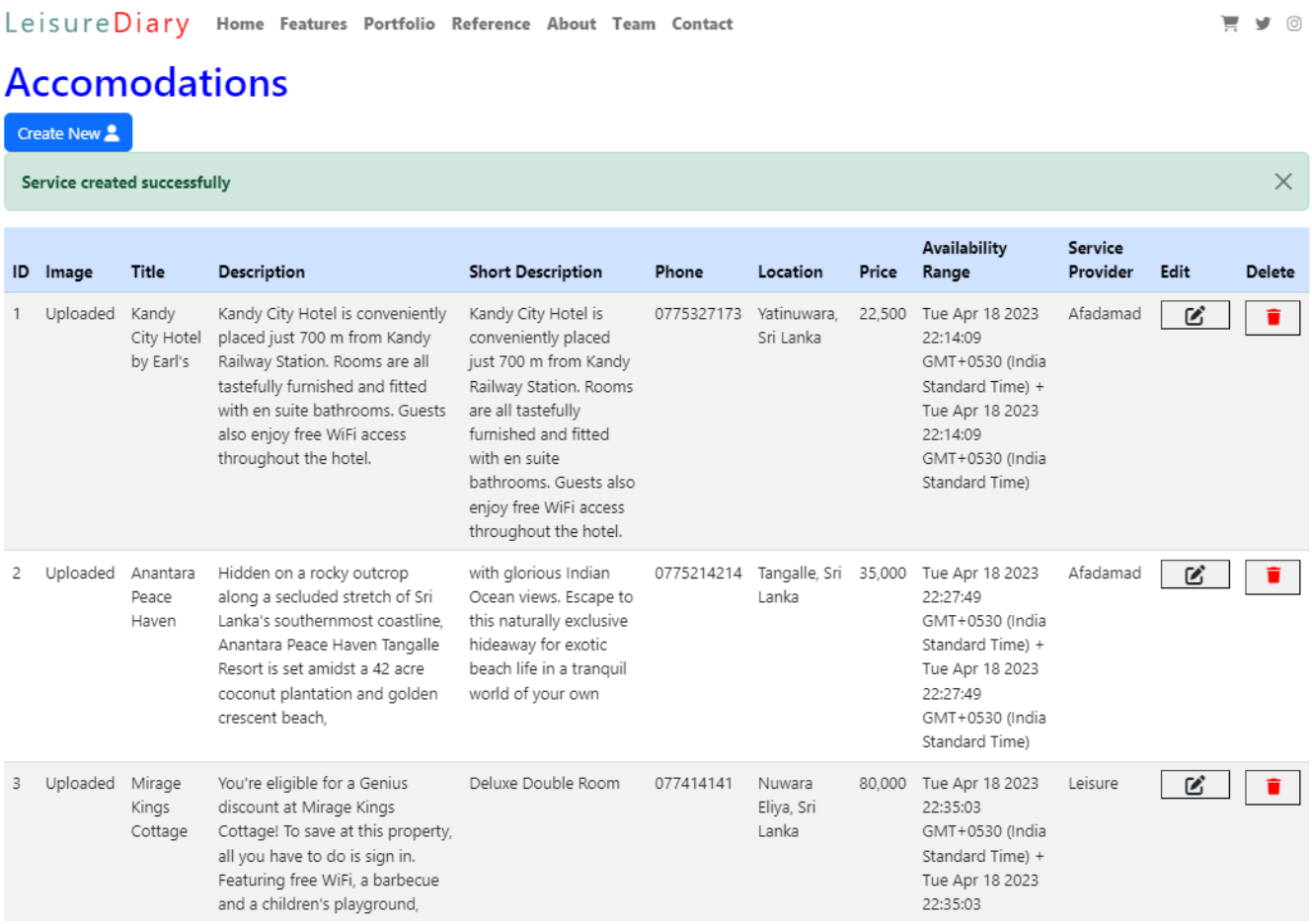


Figure 38- Success message

Accommodation dashboard is where the service provider can see all the created accommodations and their details. The dashboard has the details such the ID, Image, Title, Description, Short Description, Phone, Location, Availability and the service. Also, there is a column to edit or delete the created service.



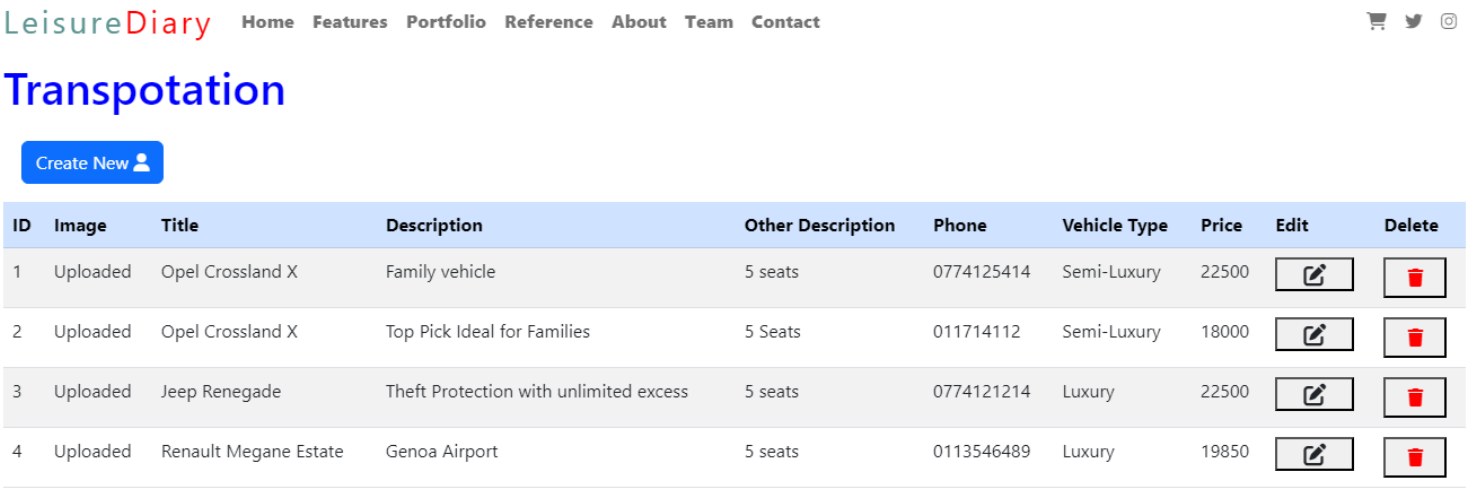
ID	Image	Title	Description	Short Description	Phone	Location	Price	Availability Range	Service Provider	Edit	Delete
1	Uploaded	Kandy City Hotel by Earl's	Kandy City Hotel is conveniently placed just 700 m from Kandy Railway Station. Rooms are all tastefully furnished and fitted with en suite bathrooms. Guests also enjoy free WiFi access throughout the hotel.	Kandy City Hotel is conveniently placed just 700 m from Kandy Railway Station. Rooms are all tastefully furnished and fitted with en suite bathrooms. Guests also enjoy free WiFi access throughout the hotel.	0775327173	Yatinuwara, Sri Lanka	22,500	Tue Apr 18 2023 22:14:09 GMT+0530 (India Standard Time) + Tue Apr 18 2023 22:14:09 GMT+0530 (India Standard Time)	Afadamad		
2	Uploaded	Anantara Peace Haven	Hidden on a rocky outcrop along a secluded stretch of Sri Lanka's southernmost coastline, Anantara Peace Haven Tangalle Resort is set amidst a 42 acre coconut plantation and golden crescent beach,	with glorious Indian Ocean views. Escape to this naturally exclusive hideaway for exotic beach life in a tranquil world of your own	0775214214	Tangalle, Sri Lanka	35,000	Tue Apr 18 2023 22:27:49 GMT+0530 (India Standard Time) + Tue Apr 18 2023 22:27:49 GMT+0530 (India Standard Time)	Afadamad		
3	Uploaded	Mirage Kings Cottage	You're eligible for a Genius discount at Mirage Kings Cottage! To save at this property, all you have to do is sign in. Featuring free WiFi, a barbecue and a children's playground,	Deluxe Double Room	077414141	Nuwara Eliya, Sri Lanka	80,000	Tue Apr 18 2023 22:35:03 GMT+0530 (India Standard Time) + Tue Apr 18 2023 22:35:03	Leisure		

Figure 39- Accommodations Dashboard

Accommodation dashboard is where the service provider can see all the created accommodations and their details. The dashboard has the details such the ID, Image, Title, Description, Short

Description, Phone, Location, Availability and the service. Also, there is a column to edit or delete the created service.

Transportation- Dashboard



ID	Image	Title	Description	Other Description	Phone	Vehicle Type	Price	Edit	Delete
1	Uploaded	Opel Crossland X	Family vehicle	5 seats	0774125414	Semi-Luxury	22500		
2	Uploaded	Opel Crossland X	Top Pick Ideal for Families	5 Seats	011714112	Semi-Luxury	18000		
3	Uploaded	Jeep Renegade	Theft Protection with unlimited excess	5 seats	0774121214	Luxury	22500		
4	Uploaded	Renault Megane Estate	Genoa Airport	5 seats	0113546489	Luxury	19850		

Figure 40- Transportation Dashboard

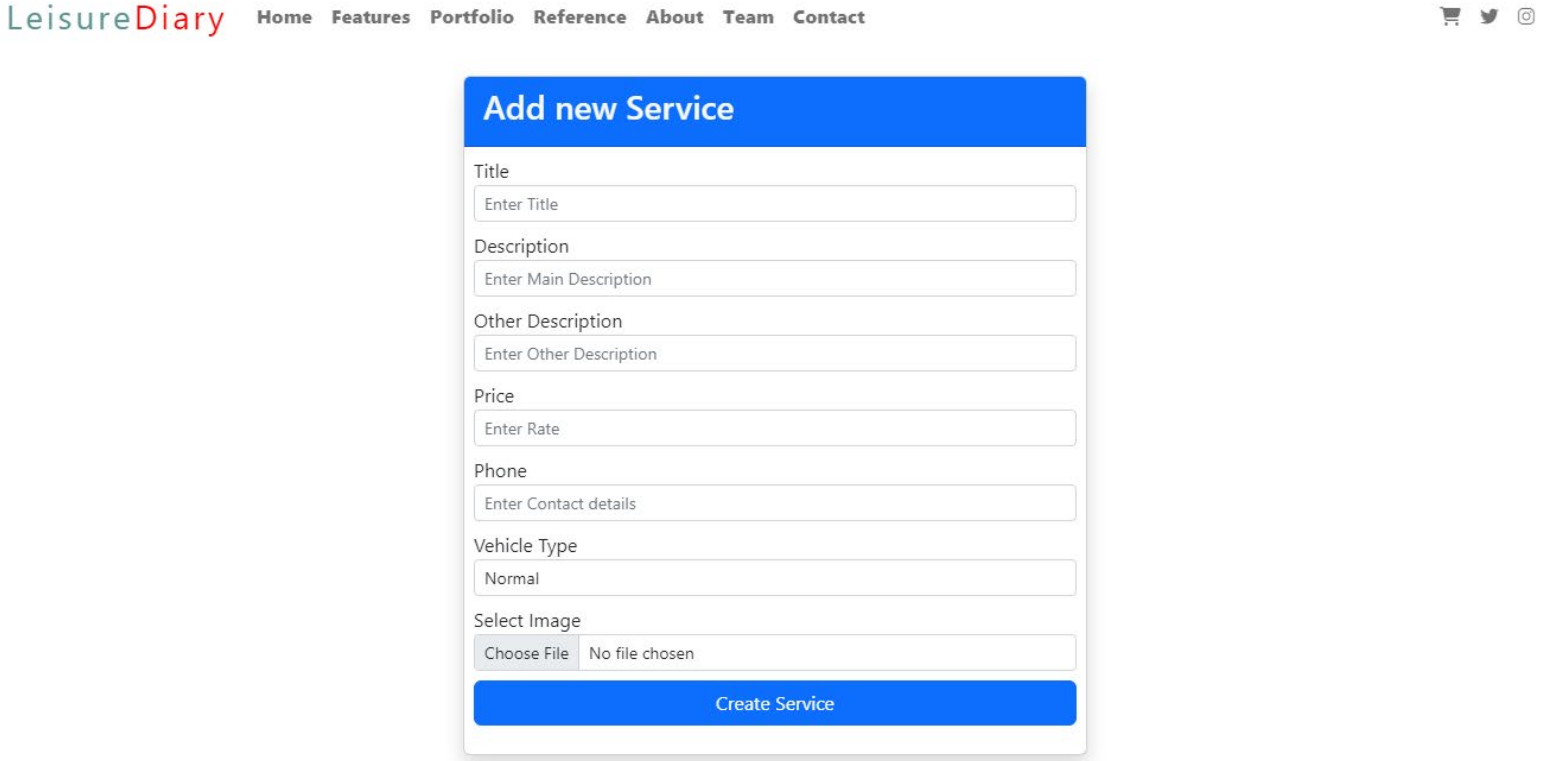


Figure 41Create new transportation

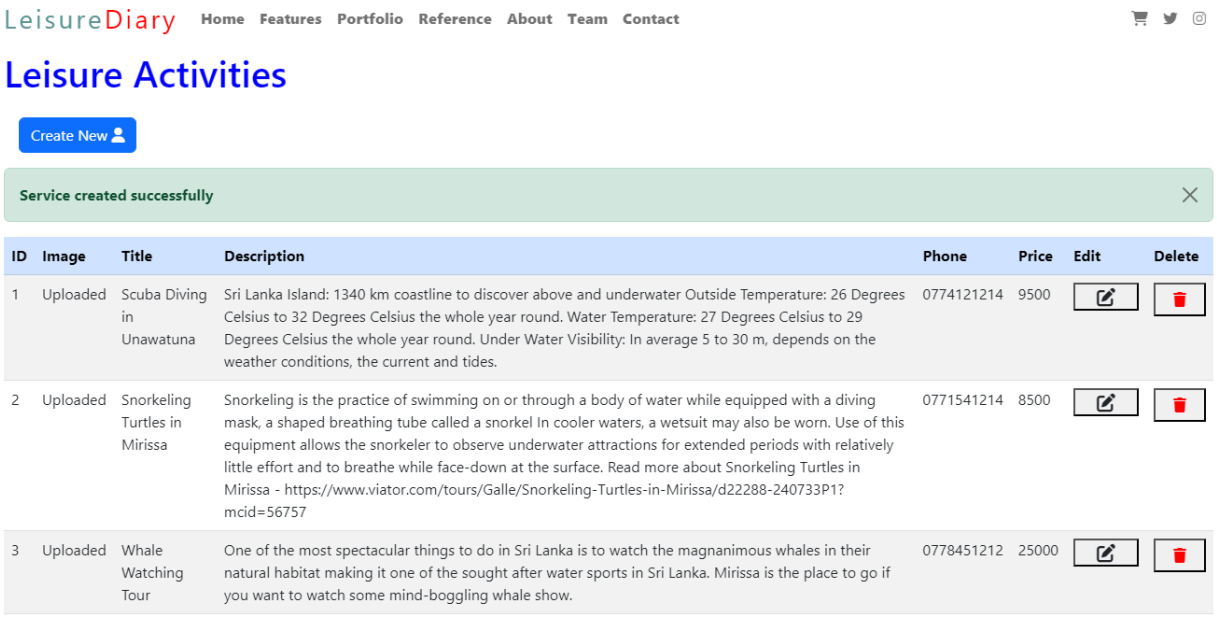


Figure 42 Leisure activity dashboard

Food Dashboard

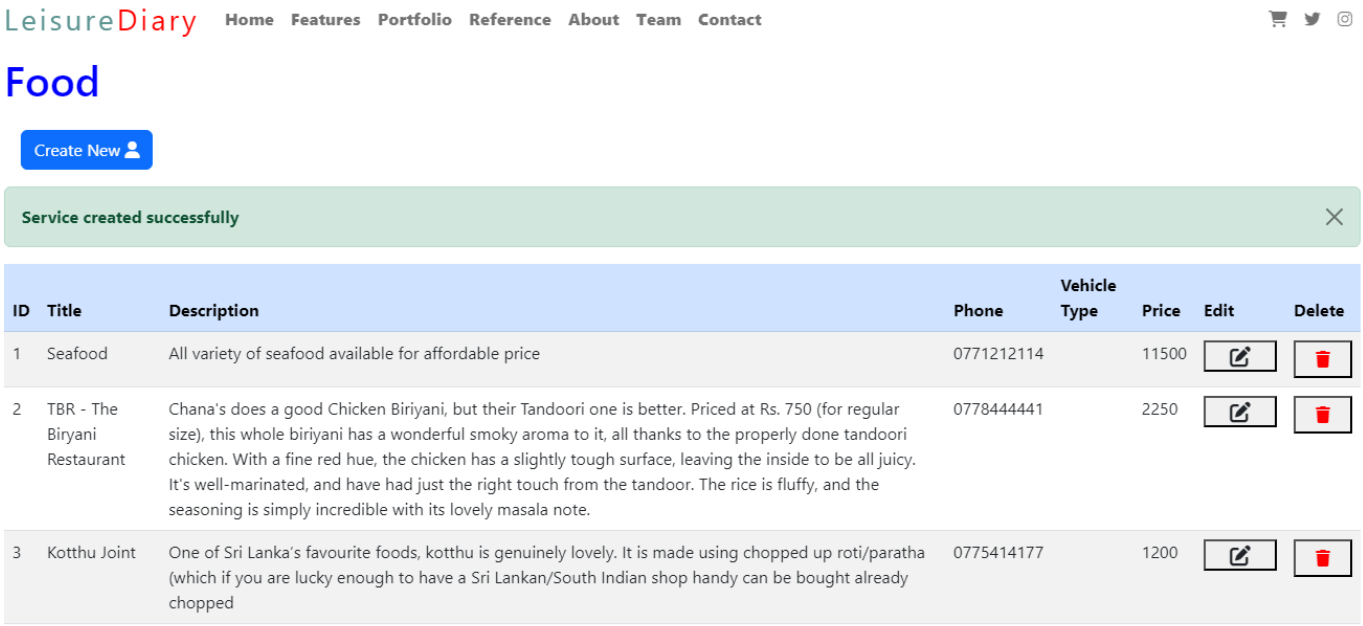


Figure 43 Food Dashboard

Reservation

Once the services are created from the backend users are able to find the created services from their mobile application. They can explore the services and make a reservation by clicking the book now button. In the reservation table which is in the backend a service provider can check and

Dashboard

Figure 44 Reservation Dashboard

Leisure Diary

Mobile application

Welcome Page

Leisure Diary has a welcome page with two buttons in displayed in the below image. After careful analyzation User interface and user experience the buttons and navigation are placed in favour of user convenience.

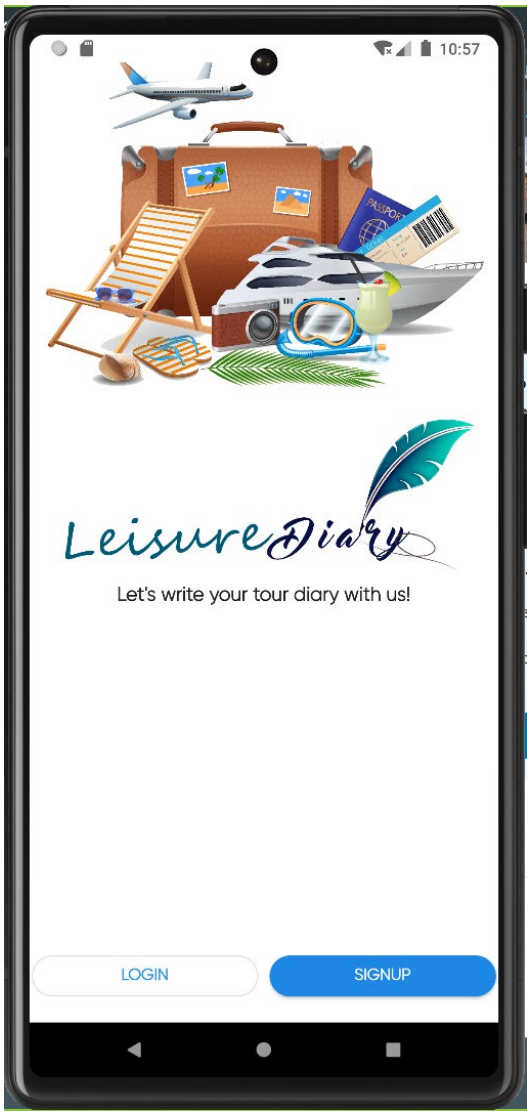


Figure 45- Mobile app welcome page

Registration Page

A new user can click on Signup button and navigate to the registration page. For a quick registration process a user only need to provide their username, email and desired password. Other additional details can be provided in the profile page.

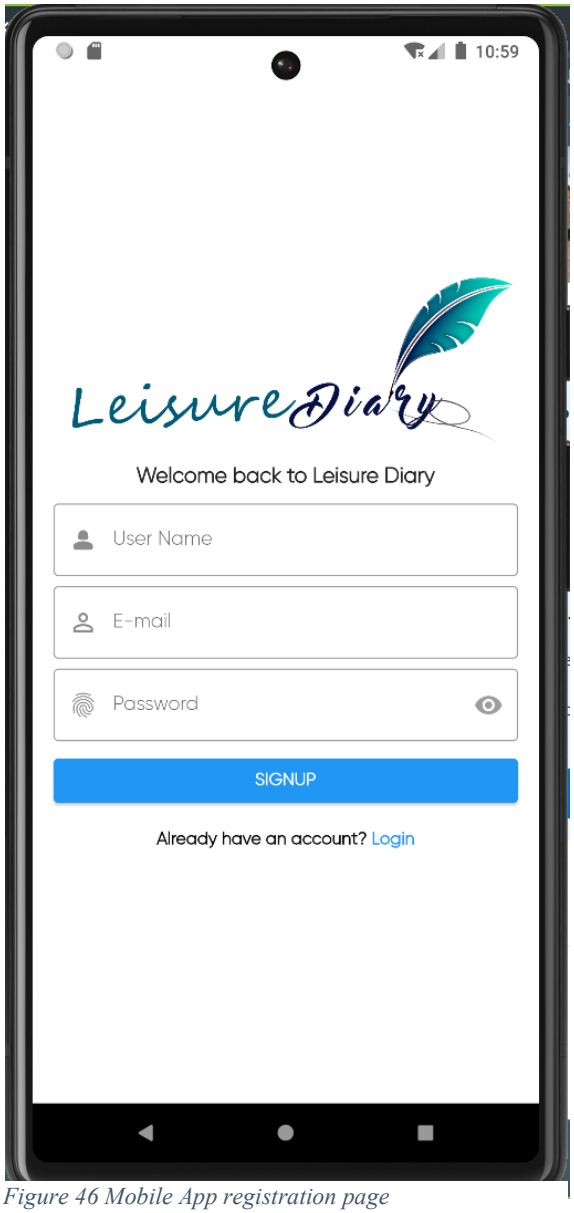


Figure 46 Mobile App registration page

Login Page

Once the user created a new account, they can use the email and password to enter into the system.

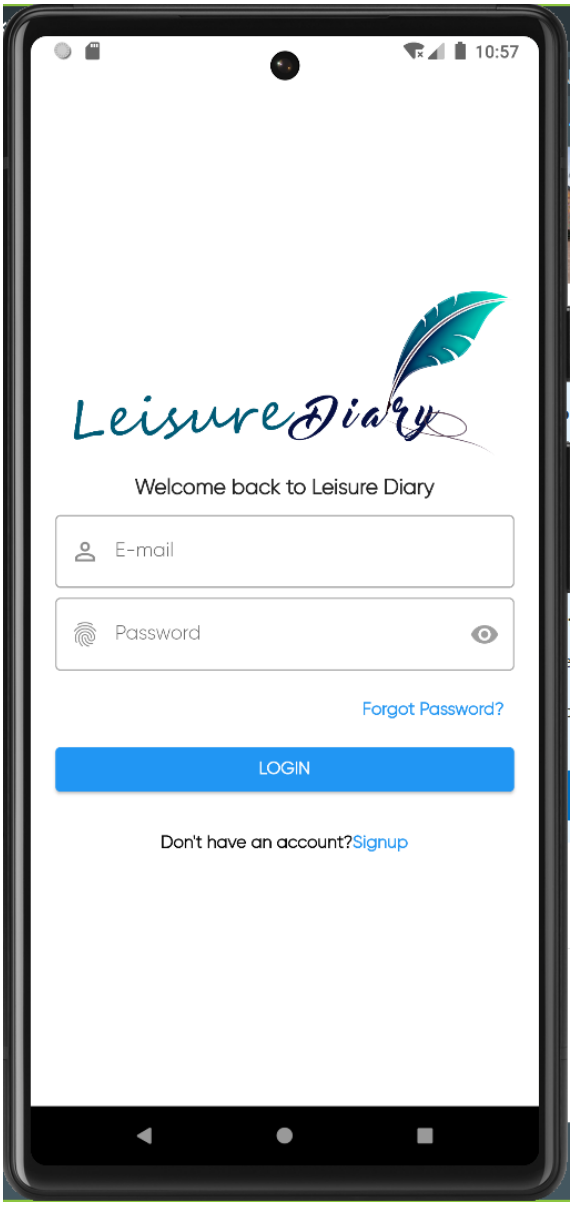


Figure 47 Mobile app login page

Mobile App Home Page

The leisure diary mobile application home page is divided into four main sections. Namely, The App bar with search bar and button to navigate to the drawer. Secondly, the categories section. There are four main categories as mentioned above Accommodations, Transport, Leisure Activities and Foods. Third is the feed where the user can find common services as in card view which can be slide horizontally. User can vertically slide and find other services.

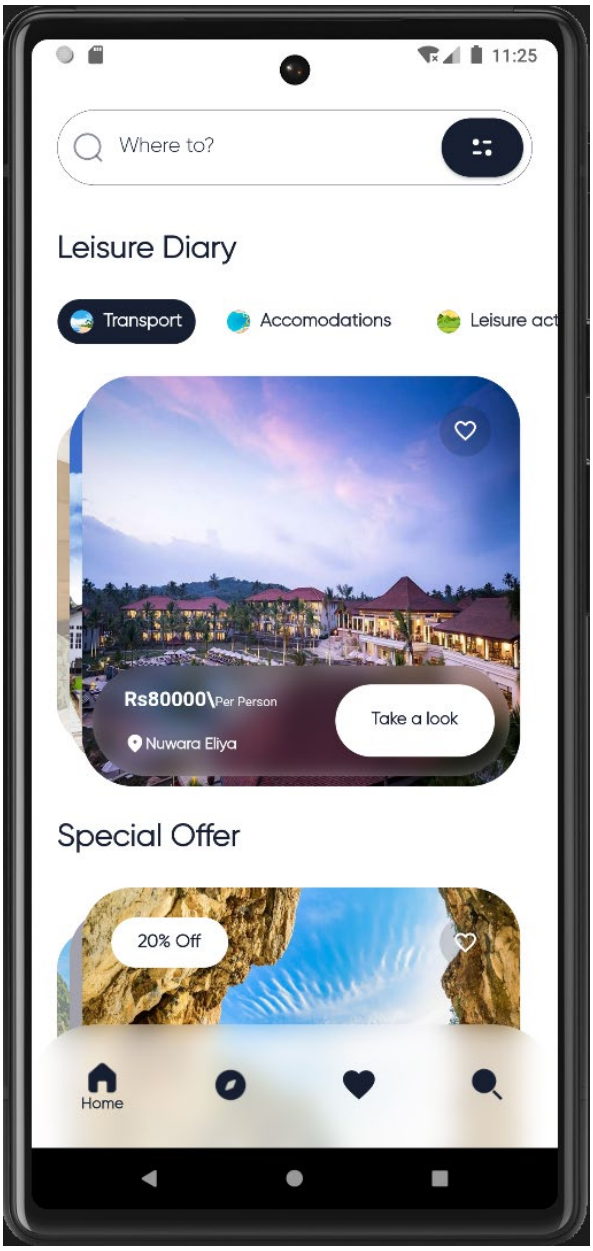


Figure 48 Mobile app home page

Drawer

Once the drawer button in the search bar clicked the drawer will be slide in. Drawer has three main buttons which are Settings, Home and Logout buttons.

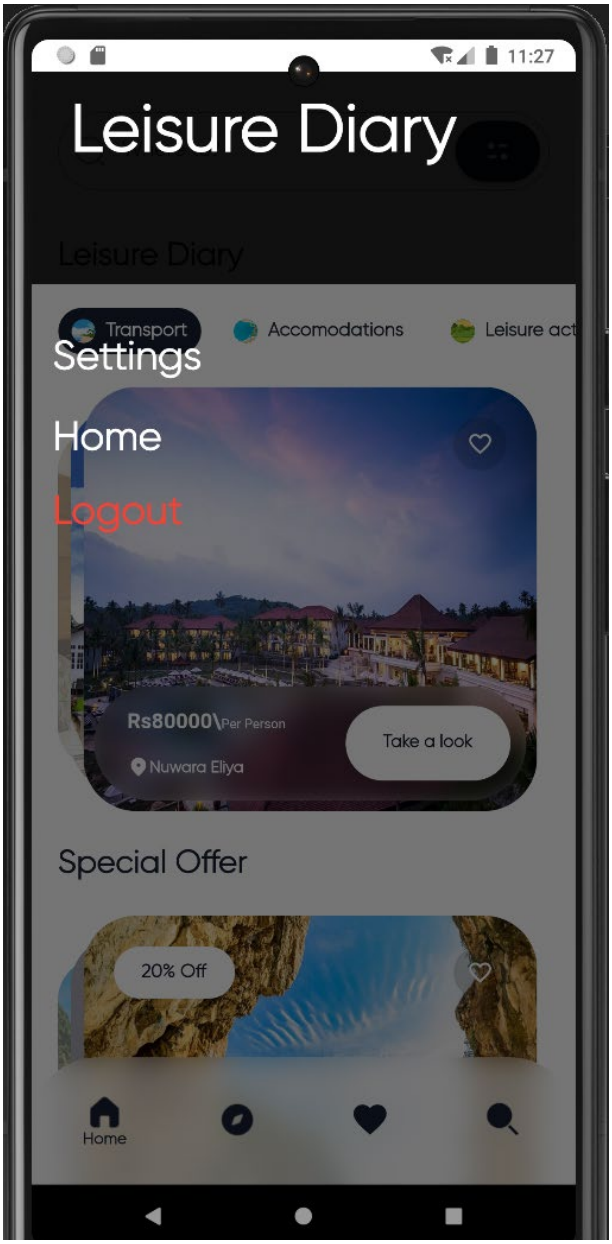


Figure 49 Mobile app drawer page

Accommodation service dashboard

The services created by the service provide in the backend web application will be rendered here under specific categories. A list view builder method has been used to render the data from API and build cards. Each card has a button as “Take a Look”. Once the button is pressed user will be navigated to a detailed view page.

Details page

From the services page, once the user selects a specific card it will be navigated to the page screen. The screen passes the relevant details in to its detailed page. The detail page has the image, title, description, location, other description and amount. If the user is satisfied with the particular service, they can click on the book now button and reserve the service. Once the user clicks on the book now button a dialog box with the confirmation and user can download the receipt through the “Download button”.

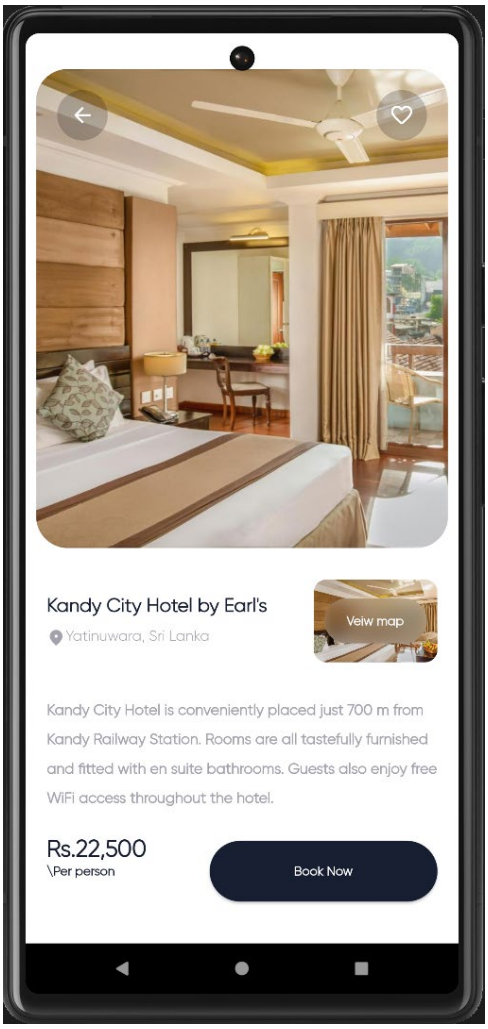


Figure 50- Service page

Figure 51 Mobile app details page

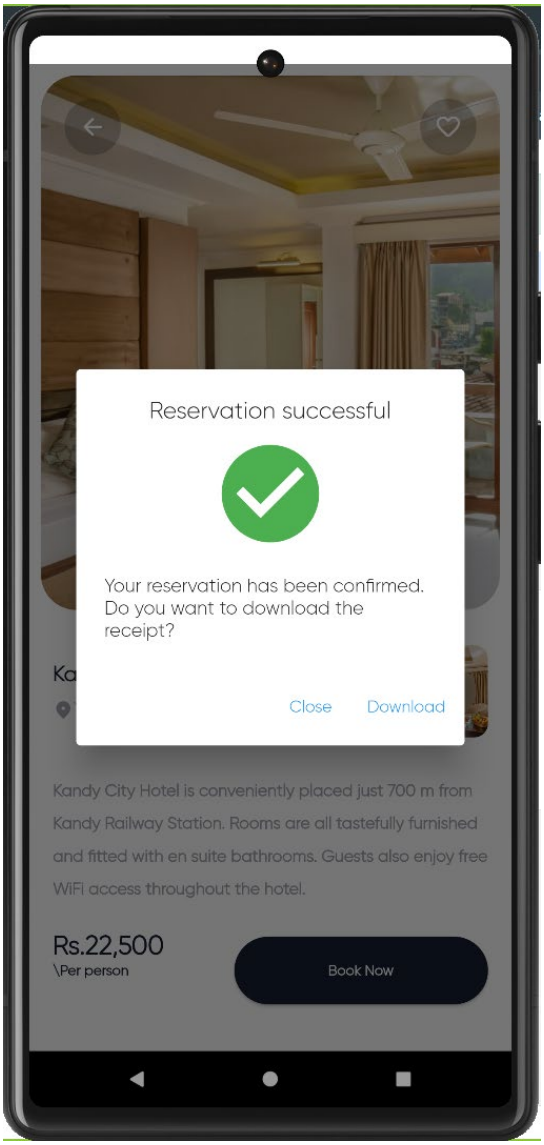


Figure 52- Mobile app reservation success message

Reservation table after its confirmation by the traveler

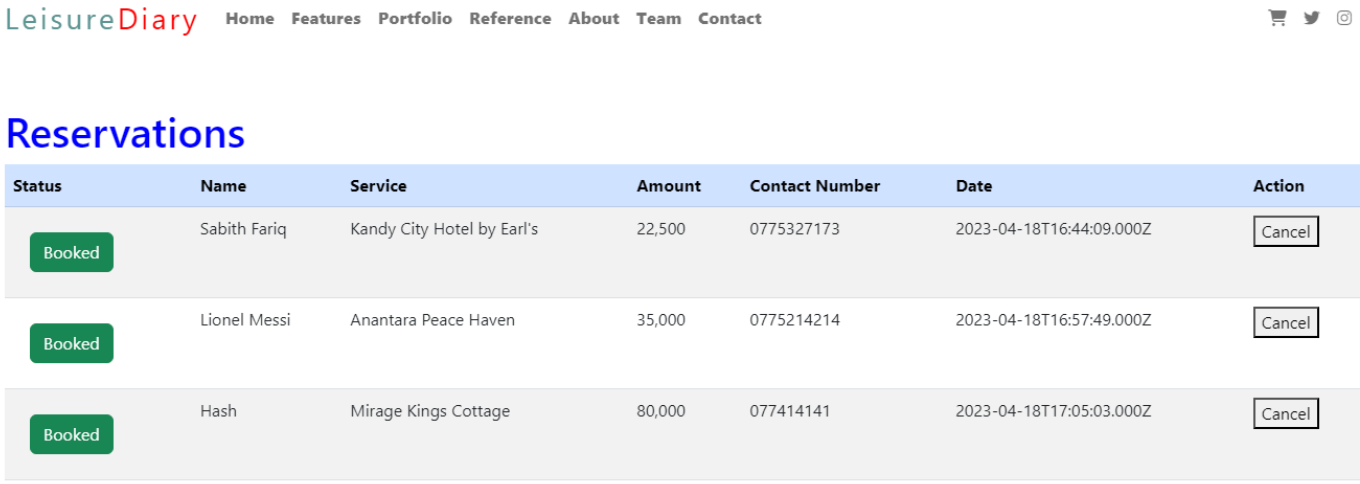


Figure 53- Reservation confirmation table

3.2.4 Coding Snippets

Web Application

Node.js Server configuration

```
const express = require('express');
const path =require('path');
const morgan = require('morgan');
const dotenv = require('dotenv');
const connectDB = require('./server/database/connection');
const serProvModel = require('./server/model/serviceProvider')
const bcrypt =require('bcryptjs');
const { response } = require('express');
const jwt = require('jsonwebtoken')
const session = require('express-session');
const travellerModel = require('./server/model/traveller');


const app=express();



// app.use(bodyParser.json());
app.use(express.json());
app.use(express.urlencoded({ extended: true}));
app.disable('etag');

app.use(
    session({
        secret: "My Secret Key",
        saveUninitialized: true,
        resave: false,
    })
);

app.use((req, res, next)=> {
    res.locals.message = req.session.message;
    delete req.session.message;
    next();
})

// Assigning a PORT
dotenv.config({path:'config.env'});

const JWT_SECRET= process.env.JWT_SECRET;

const PORT=process.env.PORT || 3000;

// Morgan Module--log request
app.use(morgan('dev'));
```

Figure 54 Server Configuration

```
// Database coonection
connectDB();

// app.use(express.static("uploads"));

app.set('view engine', 'ejs');



// load assets
app.use('/css', express.static(path.resolve(__dirname, "assets/css")));
app.use('/img', express.static(path.resolve(__dirname, "assets/img")));
app.use('/js', express.static(path.resolve(__dirname, "assets/js")));
app.use('/', express.static(path.join(__dirname, 'views')));

app.use('', require('./server/routers/router'));

app.get('/dashboard', (req, res)=>{
    res.render('dashboard')
     })

app.listen(PORT, ()=> {console.log('Server is running on', PORT)})
```

Figure 55 Server configuration 2

Database Configuration

Leisure Diary system has been used MongoDb as its database. A separate cluster has been created to store data generating from the web application.

```
const mongoose = require('mongoose');

const connectDB = async() => {
    try {
        const con = await mongoose.connect(process.env.MONGO_URI, {

        })

        console.log('Database Connected:', con.connection.host);


    } catch (error) {
        console.log(error);
        process.exit(1);

    }
}

module.exports=connectDB
```

Figure 56 DB connection

DotEnv file configuration

Figure 57 DotEnv file

PORT=5000

MONGO_URI=mongodb+srv://sabithfariq:leisure-land123@leisure-diary.tx3d3c8.mongodb.net/?retryWrites=true&w=majority

JWT_SECRET= cjsacjbsabithafieh0618csa329%^$#^$8#$#4454

APIKEY= AIzaSyBsKd7zSRFM9QkYa39-a_EXBSePqSaHuK8

User registration

```
// USER REGISTRATION
app.post('/api/register', async (req, res)=>{
    // get the inputs
    const {username, email, password: plainTextPassword}= req.body

    // validation error- USERNAME
    if(!username || typeof username != 'string'){
        return res.json({status: 'error', error: 'Invalid username'})
    }
    // validation error- EMAIL
    if(!email || typeof email != 'string'){
        return res.json({status: 'error', error: 'Invalid email'})
    }
    // validation error- PASSWORD
    if(!plainTextPassword || typeof plainTextPassword != 'string'){
        return res.json({status: 'error', error: 'Invalid password'})
    }

    if(plainTextPassword.length <5){
        return res.json({status: 'error', error: 'Password should be at least
more than 6 Characters'})
    }

    const password = await bcrypt.hash(plainTextPassword, 10)

    try {
        const response = await serProvModel.create({
            username, email, password
        })
        console.log('Service provider created successfully',response)
    } catch (error) {
        // 11000 duplicate key error
        if(error.code == 11000){
            return res.json({status: 'error', error: 'User Name / Email already
in use'})
        }
        throw error

    }

    res.json({status: 'ok'})
})
```

Figure 58 Service provider login authentication

JWT tokens have been used to save the session in as a token. JWT is used to generating and validating secure tokens in web applications, or JSON Web Tokens. Users are authenticated using JWTs to grant access to restricted resources.

```
app.post('/api/login', async (req, res)=> {

    // Validating username and password
    const {username, password}= req.body
    console.log('Data input by the user is',req.body)
    const servPro = await serProvModel.findOne({username}).lean()

    // check if the record is exist
    if(!servPro){
        return res.json({status:'error', error: 'User not available'})
    }

    if (await bcrypt.compare(password, servPro.password)){

        // if the username password combination is successful
        const token = jwt.sign({
            id: servPro._id,
            username: servPro.username
        }, JWT_SECRET)
        return res.json({status: 'ok', data: token})
        // res.redirect('dashboard');

        //
    }

    res.json({status: 'error', error: 'Invalid username/password'});
})
```

Figure 59- User Login

Service provider Model

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;
// const bcrypt = require('bcryptjs');


const serviceProvideSchema = new Schema ({
    username: {
        type: String,
        unique: true,
        required: true,

    },
    email: {
        type: String,
        unique: true,
        required: true
    },

    password:{
        type: String,
        required: true

    },


}, {collection: 'serviceProvider'}
// {timestamps: true}
);


const ServiceProvider = mongoose.model('serviceProvideSchema',
serviceProvideSchema);
module.exports = ServiceProvider;
```

Figure 60 Service provider model

Traveler Login and registration

API end point for the travelers to sign and sign up using the mobile application. Leisure diary mobile application has no separate database. It will post all the data the API hosted via Node.js backend. API will do all the communication between the mobile application and the database.

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;
const bcrypt = require('bcrypt');

const travellerModel = new Schema ({
    name: {
        type: String,
        require: true,
        unique: true

    },
    email: {
        type: String,
        lowercase: true,
        require: true,
        unique: true

    },
    password: {
        type: String,
        require: true
    }
})

travellerModel.pre('save', async function(){
    try {
        var traveller = this;
        const salt = await(bcrypt.genSalt(10));
        const hashpass = await bcrypt.hash(traveller.password, salt);

        traveller.password= hashpass;
    } catch (error) {
        throw error;
    }
},{collection: 'Travellers'}
)

travellerModel.methods.comparePassword = async function(userPassword){
    try {

        console.log('no password', this.password)
        const isMatch = await bcrypt.compare(userPassword,this.password);
        return isMatch;
    } catch (error) {
        throw error
```

Figure 61 Traveler model

```
const TravellerMod = require('../model/traveller')
const jwt = require('jsonwebtoken');

class TravellerService{
    static async registerUser(name, email, password){
        try {
            const createUser = new TravellerMod({name,email,password});
            return await createUser.save();
        } catch (error) {
            throw error;

        }
    }

    static async checkUser(email){
        try {
            return await TravellerMod.findOne({email});
        } catch (error) {
            throw error

        }
    }

    static async generateToken(tokenData, JWTSecret_Key, JWT_EXPIRE){
        return jwt.sign(tokenData, JWTSecret_Key, {expiresIn:JWT_EXPIRE});

    }
}

module.exports = TravellerService;
```

Figure 62 Traveler model

Service Models- Accommodation Model

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;


const AccomodationModel = new Schema({

    serviceProvider: {
        type: String,
        required: true,
    },

    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
        required: true,
    },
    otherdesc: {
        type: String,

    },
    location: {
        type: String,
        required: true,
    },
    image: {
        data: Buffer,
        contentType:String,

    },
    price: {
        type: String,
        required: true,
    },
    category: {
        type: String,
        required: true,
    },
    phone: {
        type: String,
        required: true,
    },
```

```
startDate: {type: Date},

    endDate: {type: Date},


    created:{
        type: Date,
        required: true,
        default: Date.now
    }



},{collection: 'Accomodation'}, {timestamps: true}
);

// Accomodation is the model here
module.exports =mongoose.model('Accomodation', AccomodationModel);
```

Figure 63 Accommodation model

Transportation service model

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;


const transportation = new Schema({


    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
        required: true,
    },
    otherdesc: {
        type: String,

    },
    image: {
        data: Buffer,
        contentType:String,

    },
    price: {
        type: String,
        required: true,
    },
    vehicletype: {
        type: String,
        required: true,
    },
    phone: {
        type: String,
        required: true,
    },

    created:{
        type: Date,
        required: true,
        default: Date.now
    }
```

```
},{collection: 'Transpotation'}, {timestamps: true}
);

// Accomodation is the model here
module.exports =mongoose.model('Transpotation', transportation);
```

Figure 64 Transportation model

Leisure Activity model

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;


const leisureActivity = new Schema({
    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
        required: true,
    },
    image: {
        data: Buffer,
        contentType:String,

    },
    price: {
        type: String,
        required: true,
    },
    location: {
        type: String,
        required: true,
    },

    phone: {
        type: String,
        required: true,
    },




    created:{
        type: Date,
        required: true,
        default: Date.now
    }



},{collection: 'Leisure Activity'}, {timestamps: true}
);
```

Food Model

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;


const food = new Schema({


    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
        required: true,
    },
    image: {
        data: Buffer,
        contentType:String,

    },
    price: {
        type: String,
        required: true,
    },
    location: {
        type: String,
        required: true,
    },

    phone: {
        type: String,
        required: true,
    },


    created:{
        type: Date,
        required: true,
        default: Date.now
    }



},{collection: 'Food'}, {timestamps: true}
);
```

Figure 65 Food Model

Routing libraries

```
const express = require('express');
const route = express.Router();
const services = require('../services/render');
const AuthController = require('../controller/authController');
const Accomodation = require('../model/accomodation');
const Reservations = require('../model/reservation');
const Transpotation = require('../model/transport');
const LeisureActivity = require('../model/leisure');
const FoodModel = require('../model/food');
const multer = require('multer');
const mongooseValidationErrorHandler = require('mongoose-validation-error-
message-handler');
const accomodation = require('../model/accomodation');
const { off } = require('../model/serviceProvider');
const moment = require('moment/moment');
const fs = require('fs');
const TravellerController = require('../controller/travellerController');
const leisure = require('../model/leisure');
const food = require('../model/food');
```

Figure 66 Routing libraries

Service routing codlings

The coding part of the services is divided in to a few categories and hosted in to separate endpoints. End points have been created to Get, Create, Update, Delete and Transfer in to API. According to the specific endpoints client-side pages will be rendered.

Routing codes to get data from the Database

```
route.get('/dashboard/accomodation', async (req, res)=>{

    if(req.query.id){
        const id = req.query.id;
        await Accomodation.findById(id).then(data=>
            {
                if(!data){
                    res.status(404).send({message: "Service not Found"})
                }else{
                    res.send(data)
                }

            })
            .catch(err =>{
                res.status(500).send({message: "Error retreving service by ID"})
            })
    }else{
        Accomodation.find().then(accomodation=>{

            res.render('accomodation', {title: 'Accomodations- Leisure Diary',
accomodation: accomodation});
            // console.log('Accomodation body',accomodation);

        })
        .catch(err=>{
            res.status(500).send({message:err.message || "Error on getting all
the data"});
        })
    }

 });
```

Figure 67 Accommodation Get method

Transportation GET method

```
route.get('/dashboard/transpotation', async (req, res)=>{

    if(req.query.id){
        const id = req.query.id;
        await Transpotation.findById(id).then(data=>
            {
                if(!data){
                    res.status(404).send({message: "Service not Found"})
                }else{
                    res.send(data)
                }

            })
            .catch(err =>{
                res.status(500).send({message: "Error retreving service by ID"})
            })
    }else{
        Transpotation.find().then(transpotaion=>{

            res.render('transpotaion', {title: 'Transpotation- Leisure Diary',
transpotaion: transpotaion});
            // console.log('Accomodation body',accomodation);

        })
        .catch(err=>{
            res.status(500).send({message:err.message || "Error on getting all
the data"});
        })
    }

 });
```

Figure 68 Transpotation get method

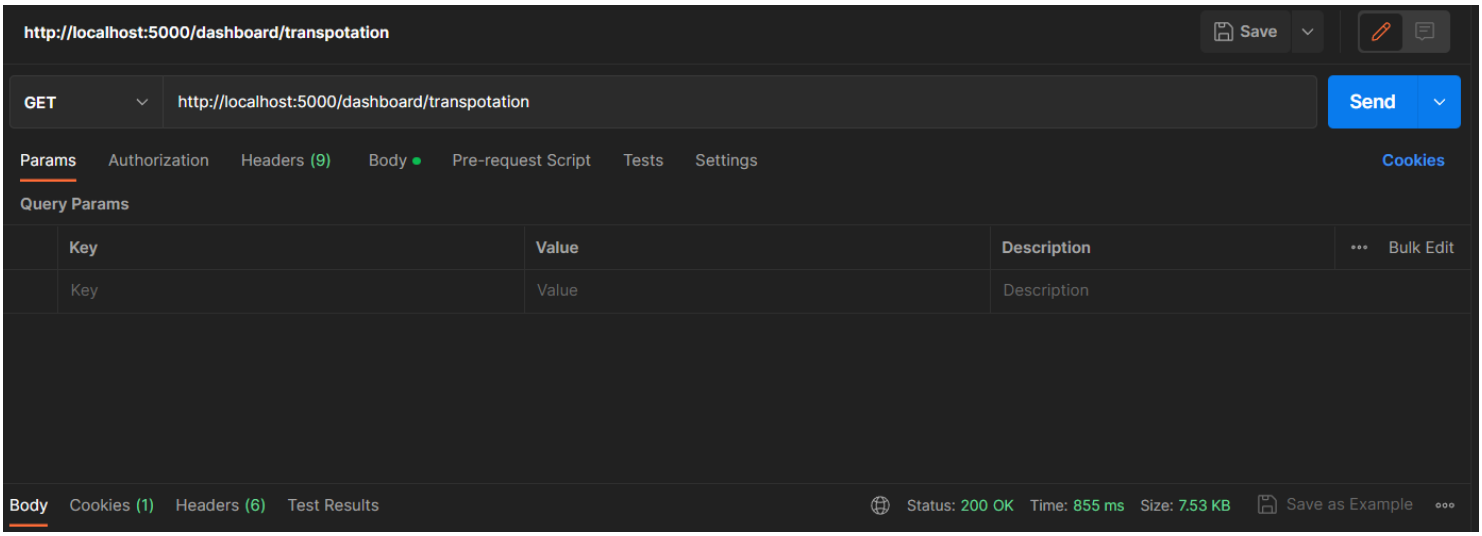


Figure 69 GET API Transportation

```
route.get('/dashboard/leisureactivity', async (req, res)=>{

    if(req.query.id){
        const id = req.query.id;
        await LeisureActivity.findById(id).then(data=>
            {
                if(!data){
                    res.status(404).send({message: "Service not Found"})
                }else{
                    res.send(data)
                }

            })
            .catch(err =>{
                res.status(500).send({message: "Error retreving service by ID"})
            })
    }else{
        LeisureActivity.find().then(leisure=>{

            res.render('leisureactivity', {title: 'Leisure Activities- Leisure
Diary', leisure: leisure});
            // console.log('Accomodation body',accomodation);

        })
        .catch(err=>{
            res.status(500).send({message:err.message || "Error on getting all
the data"});
        })
    }

 });
```

Figure 70 Leisure activity GET method

```
route.get('/dashboard/food', async (req, res)=>{

    if(req.query.id){
        const id = req.query.id;
        await FoodModel.findById(id).then(data=>
            {
                if(!data){
                    res.status(404).send({message: "Service not Found"})
                }else{
                    res.send(data)
                }

            })
            .catch(err =>{
                res.status(500).send({message: "Error retreving service by ID"})
            })
    }else{
        FoodModel.find().then(food=>{

            res.render('food', {title: 'Accomodations- Food', food: food});
            // console.log('Accomodation body',accomodation);

        })
        .catch(err=>{
            res.status(500).send({message:err.message || "Error on getting all
the data"});
        })
    }

 });
```

Figure 71- Food Get method

Service POST routes

```
// Accomdation inserting Saving
route.post('/add-accomodation', upload.single('image'), async (req, res) =>{
    const accomodation = new Accomodation({
    serviceProvider:req.body.serviceProvider,
    title: req.body.title,
    description: req.body.description,
    otherdesc: req.body.otherdesc,
    location: req.body.location,
    image: {
        data: req.file.buffer,
        contentType: req.file.mimetype
    },
    price: req.body.price,
    category: req.body.category,
    phone: req.body.phone,
    startDate: Date(req.body.startDate),
    endDate: Date(req.body.endDate),

    });
    await accomodation.save(accomodation)

    .then(data=> {
        req.session.message= {
            type: 'Success',
            message: 'Service created successfully'
        };

        res.redirect('/dashboard/accomodation');


    })
    .catch(err =>{
        const error = mongooseValidationErrorHandler(err);
        console.log(error);
        res.status(500).send({
            message:err.message || "Error in creating"
        })
    })

})
```

Figure 72 Posting routes

```
route.post('/add-transpotaion', upload.single('image'), async (req, res) =>{
    const transpotaion = new Transpotation({
    title: req.body.title,
    description: req.body.description,
    otherdesc:req.body.otherdesc,
    image: {
        data: req.file.buffer,
        contentType: req.file.mimetype
    },
    price: req.body.price,
    vehicletype: req.body.vehicletype,
    phone: req.body.phone,

    });
    await transpotaion.save(transpotaion)

    .then(data=> {
        req.session.message= {
            type: 'Success',
            message: 'Service created successfully'
        };

        res.redirect('/dashboard/transport');


    })
    .catch(err =>{
        const error = mongooseValidationErrorHandler(err);
        console.log(error);
        res.status(500).send({
            message:err.message || "Error in creating"
        })
    })

})

// Creating Leisure activity services
route.post('/add-leisureactivity', upload.single('image'), async (req, res) =>{
    const leisure = new LeisureActivity({
    title: req.body.title,
    description: req.body.description,
    image: {
        data: req.file.buffer,
        contentType: req.file.mimetype
    },
```

```
location: req.body.location,
    price: req.body.price,
    phone: req.body.phone,


    });
    await leisure.save(leisure)

    .then(data=> {
        req.session.message= {
            type: 'Success',
            message: 'Service created successfully'
        };

        res.redirect('/dashboard/leisureactivity');


    })
    .catch(err =>{
        const error = mongooseValidationErrorHandler(err);
        console.log(error);
        res.status(500).send({
            message:err.message || "Error in creating"
        })
    })

})

// Creating Leisure activity services
route.post('/add-food', upload.single('image'), async (req, res) =>{
    const food = new FoodModel({
    title: req.body.title,
    description: req.body.description,
    image: {
        data: req.file.buffer,
        contentType: req.file.mimetype
    },
    price: req.body.price,
    phone: req.body.phone,
    location: req.body.location,

    });
    await food.save(food)

    .then(data=> {
```

```
req.session.message= {
            type: 'Success',
            message: 'Service created successfully'
        };

        res.redirect('/dashboard/food');


    })
    .catch(err =>{
        const error = mongooseValidationErrorHandler(err);
        console.log(error);
        res.status(500).send({
            message:err.message || "Error in creating"
        })
    })

})
```

Figure 73 Post routing

Front end page rendering on routing

```
//  add new accomodation
route.get('/create-accomodation', (req, res) =>{
    res.render('create-accomodation', {title: 'Create New Accomodation'} )
})
route.get('/create-transport', (req, res) =>{
    res.render('create-transport', {title: 'Create New Transportation'} )
})
route.get('/create-leisureactivity', (req, res) =>{
    res.render('create-leisureactivity', {title: 'Create New Leisure Activity'} )
})
route.get('/create-food', (req, res) =>{
    res.render('create-food', {title: 'Create New Food'} )
})
```

Figure 74 Front-end page rendering

Update routing

```
route.post('/update-accomodation/:id', async (req, res)=>{

    let id=req.params.id;
    await Accomodation.findByIdAndUpdate(id, req.body)
    .then(accomodation=>{
        if(!accomodation){
            res.status(404).send({message: 'Service Not Found'})
        }else{

            // res.send(accomodation)
            res.render('update-accomodation', {title: 'Create New Accomodation',
accomodation:accomodation} )
        }
    })
   })
```

Figure 75 Update routing

Delete routing

```
//Delete Routes
route.post('/delete-accomodation/:id', (req, res)=>{
    const id = req.params.id;
    Accomodation.findByIdAndDelete(id).then(accomodation=>{
        res.redirect('/dashboard/accomodation');
    })

})
route.post('/delete-transpotaion/:id', (req, res)=>{
    const id = req.params.id;
    Transpotation.findByIdAndDelete(id).then(transpotaion=>{
        res.redirect('/dashboard/transpotation');
    })

})
route.post('/delete-food/:id', (req, res)=>{
    const id = req.params.id;
    FoodModel.findByIdAndDelete(id).then(food=>{
        res.redirect('/dashboard/food');
    })

})
route.post('/delete-leisure/:id', (req, res)=>{
    const id = req.params.id;
    LeisureActivity.findByIdAndDelete(id).then(leisure=>{
        res.redirect('/dashboard/leisureactivity');
    })
```

Figure 76 Delete routing

Reservation coding

Reservation POST method

```
route.post('/api/Reservations', async (req,res) =>{
    const reservation = new Reservations(req.body
    );
  await reservation.save(reservation)
  .then(data=>{
    res.redirect('/dashboard/accomodation')
  })
  console.log('reservation body', reservation);
})
```

Figure 77 Reservation POST method

Reservation GET method

```
route.get('/reservation-accomodation', (req, res)=>{
    Reservations.find().then(reservation=>{

        res.render('reservationsPage', {title: 'Reservations- Leisure Diary',
reservation: reservation});
        // console.log('Accomodation body',accomodation);

    })

})
```

Figure 78 Reservation GET method

```
route.post('/delete-reservation/:id', (req, res)=>{
    const id = req.params.id;
    Reservations.findByIdAndDelete(id).then(reservation=>{
        res.redirect('/reservation-accomodation');
    })

})
```

Figure 79 Reservation delete method

Kingston University London

Leader in Professional ICT Education

Coding snippets

Mobile application

Main function

Main function directly navigate the mobile application welcome screen.

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:travel_app_ui/constants.dart';
import 'package:travel_app_ui/screens/home/homepage.dart';
import 'package:travel_app_ui/screens/login-register/login-reg-main-Screen.dart';

void main() {
  SystemChrome.setSystemUIOverlayStyle(
    SystemUiOverlayStyle(
      statusBarIconBrightness: Brightness.dark,
      statusBarColor: backgroundColor,
      systemNavigationBarColor: Colors.black,
    ),
  );

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'LEISURE DIARY',
      theme: ThemeData(
        fontFamily: 'Gilroy',
        scaffoldBackgroundColor: backgroundColor,
        primarySwatch: Colors.blue,
      ),
      home: const LoginScreenMain(),
    );
  }
}
```

Figure 80- Mobile app main function

Models

Accommodation

```dart
import 'package:flutter/material.dart';

class Accomodation {
  final String? serviceProvider;
  final String? title;
  final String? description;
  final String? otherdesc;
  final String? location;
  final String? image;
  final String? price;
  final String? category;
  final String? phone;
  final String? startDate;
  final String? endDate;

  Accomodation(
      {this.serviceProvider,
      this.title,
      this.description,
      this.otherdesc,
      this.location,
      this.image,
      this.price,
      this.category,
      this.phone,
      this.startDate,
      this.endDate});
}
```

Figure 81 Accommodation model

Reservation Model

```
class Reservation {
  final String? username;
  final String? title;
  final String? date;
  final String? price;
  final String? phone;

  Reservation({this.username, this.title, this.date, this.price, this.phone});

  Map<String, dynamic> toJson() => {
        'username': username,
        'title': title,
        'date': date,
        'price': price,
        'phone': phone,
      };
}
```

Figure 82 Reservation model

Home page UI

```
class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      extendBodyBehindAppBar: true,
      extendBody: true,
      drawer: DrawerWid(),
      bottomNavigationBar: const BottomGlassBox(
        child: BottomBar(),
      ),
      body: Column(
        children: [
          const SizedBox(height: 35),
          const Searchform(),
          Expanded(
            child: SingleChildScrollView(
              physics: const BouncingScrollPhysics(),
              child: Column(
                // mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const SizedBox(height: 15),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: defaultPadding),
                    child: Text(
                      'Leisure Diary',
                      style: TextStyle(
                        color: primaryColor,
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                  const SizedBox(height: 25),
                  const Categories(),
                  const SizedBox(height: 25),
                  const SwipeCard(),
                  const SizedBox(height: 25),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: defaultPadding),
                    child: Text(
                      'Special Offer',
                      style: TextStyle(
```

```
  color: primaryColor,
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                  const SizedBox(height: 25),
                  const SpecialCard(),
                  const SizedBox(height: 100),
                  const SwipeCard(),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Figure 83 Home screen UI

Login Function

```
class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  TextEditingController emailController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  bool _isNotValidate = false;

  late SharedPreferences prefs;

  @override
  void initState() {
    super.initState();
    initSharedPref();
  }

  void initSharedPref() async {
    prefs = await SharedPreferences.getInstance();
  }

  void loginUser() async {
    if (emailController.text.isNotEmpty && passwordController.text.isNotEmpty) {
      // mapping the body

      var reqBody = {
        "email": emailController.text,
        "password": passwordController.text
      };

      var response = await http.post(Uri.parse(login),
          headers: {"Content-Type": "application/json"},
          body: jsonEncode(reqBody));

      var jsonResponse = jsonDecode(response.body);

      if (jsonResponse['status']) {
        var myToken = jsonResponse['token'];
        prefs.setString('token', 'myToken');

        Navigator.push(
```

```
ontext, MaterialPageRoute(builder: (context) => HomePage()));
      } else {
        print('Something went wrong');
      }
    } else {
      setState(() {
        bool _isNotValidate = true;
      });
    }
  }
```

Figure 84 Login Function

Signup Function

```
class Signup extends StatefulWidget {
  const Signup({super.key});

  @override
  State<Signup> createState() => _SignupState();
}

class _SignupState extends State<Signup> {
  TextEditingController emailController = TextEditingController();
  TextEditingController nameController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  bool _isNotValidate = false;

  // reg user function
  void registerUser() async {
    if (emailController.text.isNotEmpty &&
        passwordController.text.isNotEmpty &&
        nameController.text.isNotEmpty) {
      // mapping the body

      var regBody = {
        "name": nameController.text,
        "email": emailController.text,
        "password": passwordController.text
      };
      var response = await http.post(Uri.parse(registration),
          headers: {"Content-Type": "application/json"},
          body: jsonEncode(regBody));

      var jsonResponse = jsonDecode(response.body);

      print(jsonResponse['status']);
      if (jsonResponse['status']) {
        Navigator.push(
            context, MaterialPageRoute(builder: (context) => LoginScreen()));
      } else {
        print('Something went wrong');
      }
    } else {
      setState(() {
        bool _isNotValidate = true;
      });
    }
  }
```

Figure 85 Signup function

Configuration codes

```
final url = 'http://192.168.8.188:5000/';
final registration = url + 'traveller-registration';

final login= url + 'traveller-login';
```

Figure 86 Configuration

GET service end point

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

import '../models/AccomModel.dart';

class Api {
  static getServices() async {
    List<Accomodation> accomodations = [];

    var url = Uri.parse('http://192.168.8.188/api/getAccomodations');

    try {
      final res = await http.get(url);

      if (res.statusCode == 200) {
        var data = jsonDecode(res.body);
        // print('res.body ${data}');

        data['accomodations'].forEach((value) => {
              accomodations.add(Accomodation(
                  serviceProvider: value['serviceProvider'].toString(),
                  title: value['title'].toString(),
                  description: value['description'].toString(),
                  otherdesc: value['otherdesc'].toString(),
                  location: value['location'].toString(),
                  image: value['image'].toString(),
                  price: value['price'].toString(),
                  category: value['category'].toString(),
                  phone: value['phone'].toString(),
                  startDate: value['startDate'].toString(),
                  endDate: value['endDate'].toString()))
            });
        return accomodations;
      } else {
        return [];
      }
    } catch (e) {
      print(e.toString());
      print('Errorrrrrrrrrrrrrrr');
    }
  }
}
```

Figure 87 Service Data modeling

Reservation POST function

Below is the function created to send the reservation details to the back-end web application.

Once the user confirmed a booking the details will be redirect the web application.

```
import 'package:http/http.dart' as http;
import 'dart:convert';

import '../../models/reservation.dart';

Future<void> sendReservation(Reservation reservation) async {
  final String apiUrl = 'http://192.168.8.188/api/Reservations';

  final response = await http.post(
    Uri.parse(apiUrl),
    headers: {
      'Content-Type': 'application/json',
    },
    body: json.encode(reservation.toJson()),
  );

  if (response.statusCode == 200) {
    print('Reservation sent to backend');
  } else {
    // Error sending reservation to backend
    print('Error sending reservation to backend');
  }
}
```

Figure 88 Reservation POST function

Details page with data passing

```
class DetailsPage extends StatefulWidget {
  final Places places;
  final Accomodation accomodation;
  late Reservation reservation;

 etailsPage({
    Key? key,
    required this.accomodation,
    required this.places,
  }) : super(key: key);

  @override
  State<DetailsPage> createState() => _DetailsPageState();
}

class _DetailsPageState extends State<TestDetailsPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Stack(
          alignment: Alignment.bottomCenter,
          children: [
            SingleChildScrollView(
              physics: const BouncingScrollPhysics(),
              child: Column(
                children: [
                  Padding(
                    padding: EdgeInsets.all(defaultPadding / 2),
                    child: Container(
                      padding: const EdgeInsets.all(20),
                      height: MediaQuery.of(context).size.height / 1.9,
                      alignment: Alignment.topCenter,
                      decoration: BoxDecoration(
                        image: DecorationImage(
                          fit: BoxFit.cover,
                          image: AssetImage(widget.places.image),
                        ),
                        borderRadius: BorderRadius.circular(30),
                      ),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Container(
                            decoration: BoxDecoration(
```

```
borderRadius: BorderRadius.circular(50),
                            color: Colors.black.withOpacity(0.2),
                          ),
                          child: IconButton(
                            onPressed: () {
                              Navigator.pop(context);
                            },
                            icon: const Icon(
                              Icons.arrow_back_rounded,
                              color: Colors.white,
                            ),
                          ),
                        ),
                        Container(
                          decoration: BoxDecoration(
                            borderRadius: BorderRadius.circular(50),
                            color: Colors.black.withOpacity(0.2),
                          ),
                          child: IconButton(
                            onPressed: () {},
                            icon: const Icon(
                              Icons.favorite_border_rounded,
                              color: Colors.white,
                            ),
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
                Padding(
                  padding: EdgeInsets.all(defaultPadding),
                  child: Column(
                    children: [
                      Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                widget.accomodation.title.toString(),
                                style: TextStyle(
                                  fontWeight: FontWeight.bold,
                                  fontSize: 18,
```

```
color: primaryColor,
                                  ),
                                ),
                                const SizedBox(height: 10),
                                Row(
                                  crossAxisAlignment: CrossAxisAlignment.start,
                                  children: [
                                    Icon(
                                      Icons.location_on_rounded,
                                      color: greyColor,
                                      size: 18,
                                    ),
                                    Text(
                                      widget.accomodation.location.toString(),
                                      style: TextStyle(color: greyColor),
                                    ),
                                  ],
                                ),
                              ],
                            ),
```

```
MaterialButton(
                    onPressed: () async {
                      final reservation = Reservation(
                        username: widget.file.username.toString(),,
                        phone: widget.accomodation.phone.toString(),
                        title: widget.accomodation.title.toString(),
                        price: widget.accomodation.price.toString(),
                        date: widget.accomodation.startDate.toString(),
                      );

                      await sendReservation(reservation);
                      // ScaffoldMessenger.of(context).showSnackBar(
                      //   SnackBar(content: Text('Reservation successful')),
                      // );

                      _showReservationSuccessDialog(reservation);
                    },
```

Figure 89 Data passing on Click button

```
void _showReservationSuccessDialog(Reservation reservation) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Column(children: const [
            Text("Reservation successful"),
            SizedBox(
              width: 8.0,
              height: 8.0,
            ),
            Icon(
              Icons.check_circle,
              color: Colors.green,
              size: 100.0,
            ),
          ]),
          content: Text(
              "Your reservation has been confirmed. Do you want to download the
receipt?"),
          actions: <Widget>[
            TextButton(
              child: Text("Close"),
              onPressed: () {
                Navigator.of(context).pop();
                Navigator.of(context)
                    .push(MaterialPageRoute(builder: (context) => HomePage()));
              },
            ),
            TextButton(
              child: Text("Download"),
              onPressed: () {
                generateReceipt(reservation);

                Navigator.of(context).pop();
              },
            ),
          ],
        );
      },
    );
  }
```

Figure 90Success Dialog box

Receipt generating function

```
Future<void> generateReceipt(Reservation reservation) async {
    final pdf = pw.Document();

    pdf.addPage(pw.Page(
      build: (pw.Context context) {
        return pw.Center(
            child: pw.Column(
                mainAxisAlignment: pw.MainAxisAlignment.center,
                children: [
              pw.Text('Reservation Details'),
              pw.SizedBox(height: 20),
              pw.Text('Name: ${reservation.username}'),
              pw.SizedBox(height: 10),
              pw.Text('Title: ${reservation.title}'),
              pw.SizedBox(height: 10),
              pw.Text('Date: ${reservation.date}'),
              pw.SizedBox(height: 10),
              pw.Text('Price: ${reservation.price}'),
              pw.SizedBox(height: 20),
              pw.Text('Phone: ${reservation.phone}'),
              pw.SizedBox(height: 20),
              pw.Container(width: 200, height: 200)
            ]));
      },
    ));

    Future<void> openFile(File file) async {
      try {
        final bytes = await file.readAsBytes();
        await Printing.sharePdf(
            bytes: await pdf.save(), filename: 'invoice.pdf');
      } catch (e) {
        print(e.toString());
      }
    }
```

Figure 91 Receipt generating function

Other code snippets are attached in to the **Annexure C**

3.2.5 Leisure Diary Database

Leisure diary database has been created using MongoDB which is a popular NoSQL database management system. MongoDB uses a document-oriented data model. It saves data in Binary JSON format.

Collections

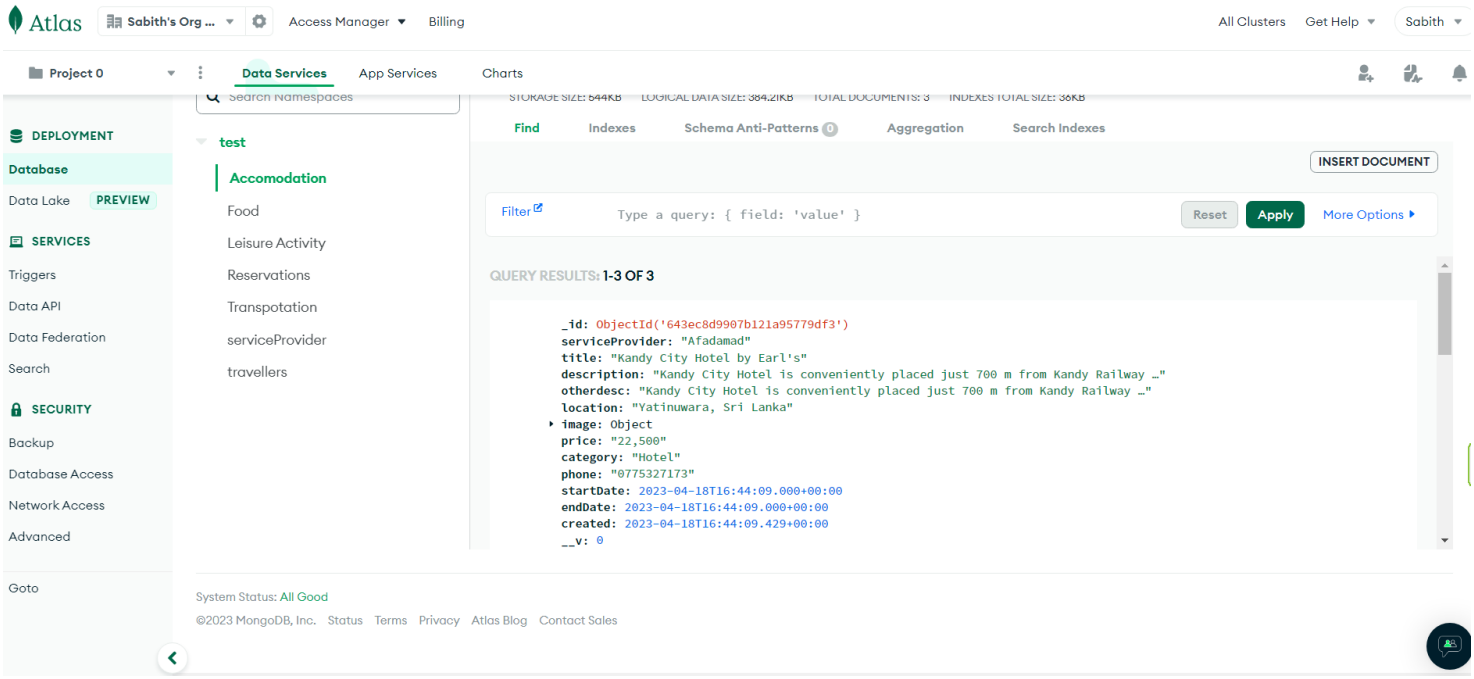


Figure 92- Database Dashboard

Figure 94 Accommodation Collection

Figure 93 Food Collection

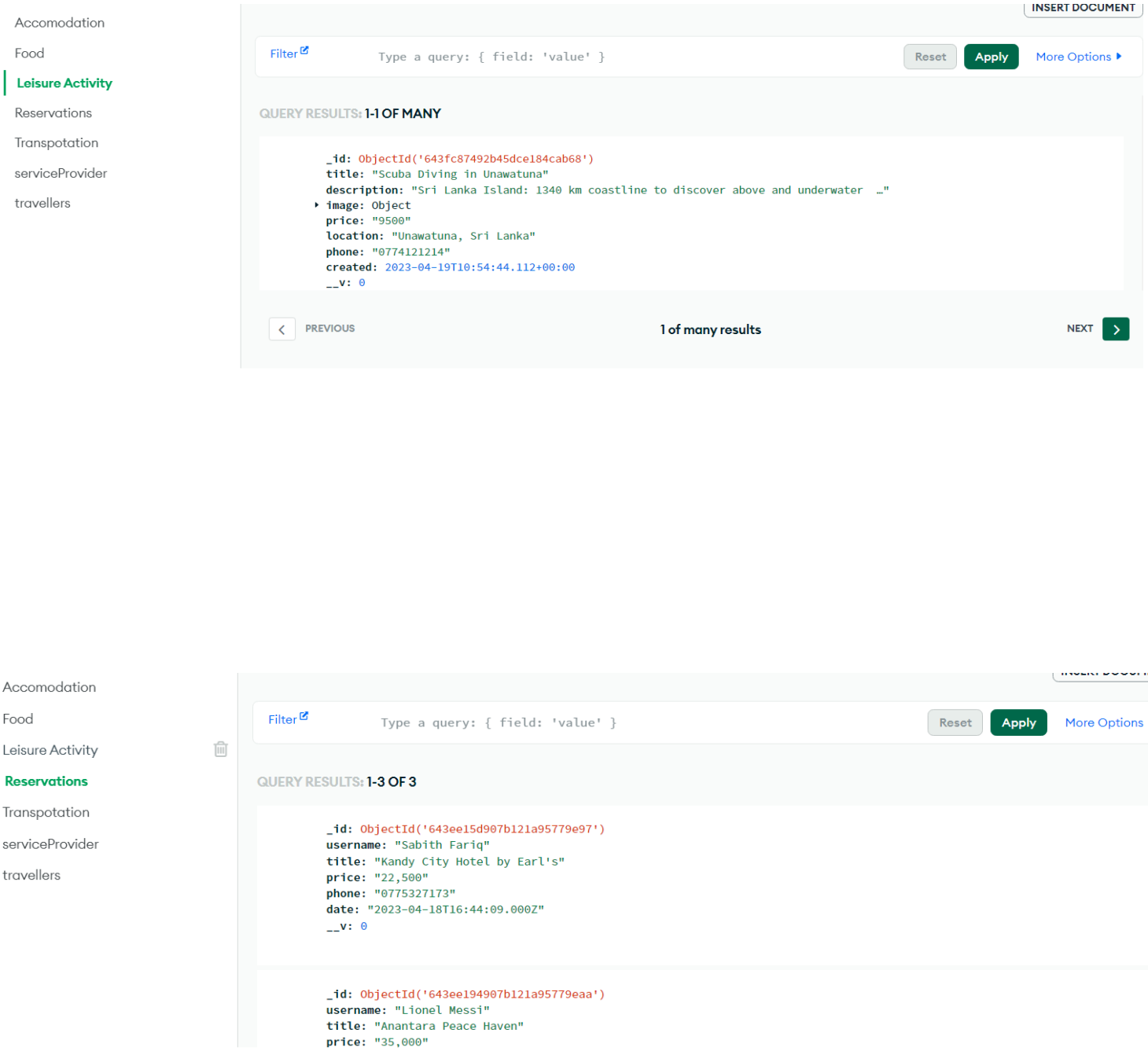


Figure 96 Reservation Details

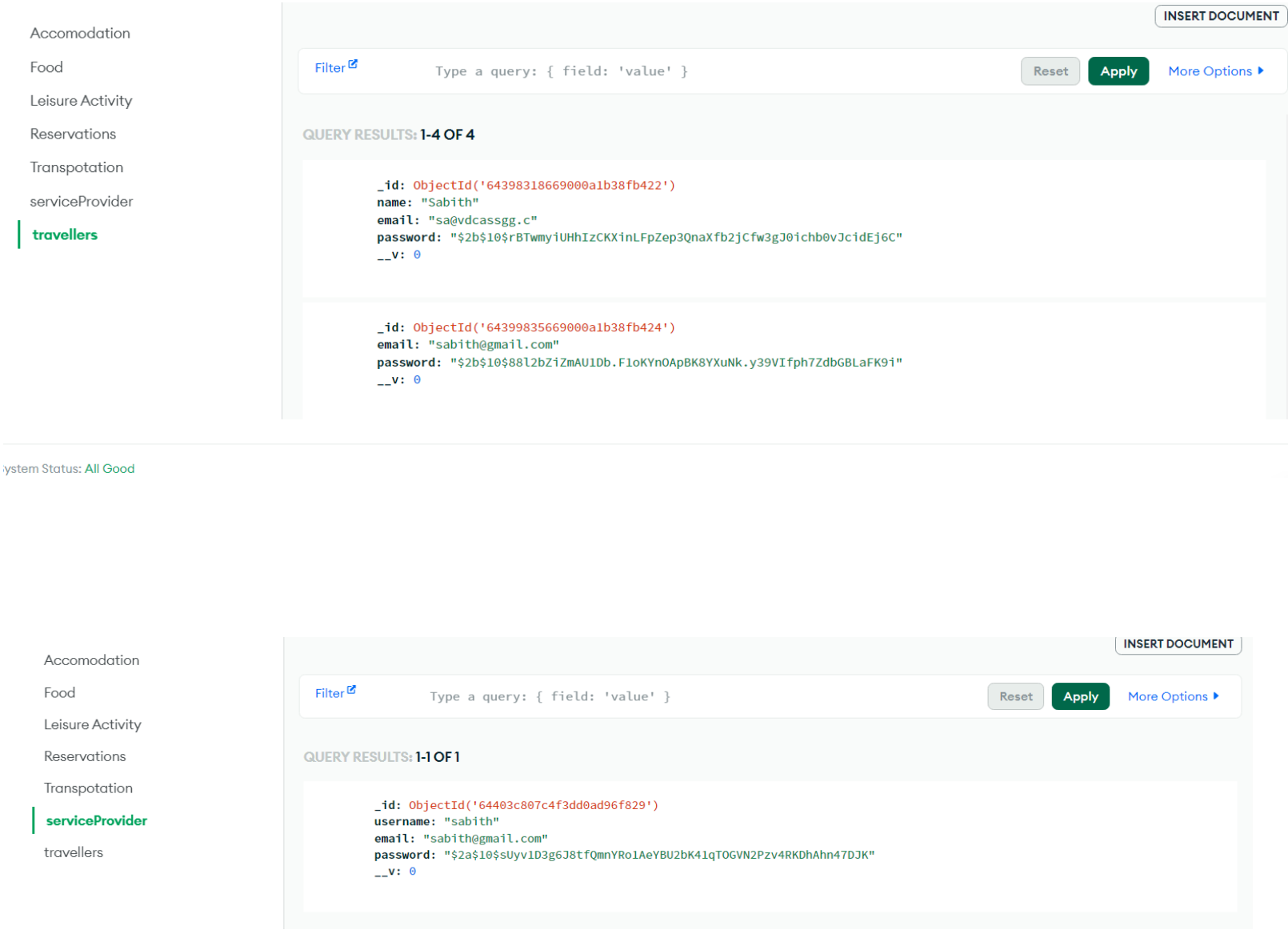


Figure 97 Traveler collection

Figure 98Service provider details

4. Product Implementation

As we identified through requirement elicitation, a web app and a mobile app are to be created and connected through a REST API. The decided technologies that are to be used when implementing the web application are ExpressJs which a frame work of Node.Js to build the back end and HTML, CSS and JavaScript have been used to frontend of the application. To save more time when building the frontend, a templating language EJS (Embedded JavaScript) has been used. It will make the code look simply, reduce the developing time, speedy execution, easy debugging and active development. Also, to avoid the code being more sophisticated to understand we have refactored the code into separate folders as per the MVC Architecture. By using expressJs the REST API endpoints have been created to communicate with the mobile application and being saved in the mongoDb database which is a cloud Database. Also, the system will be utilizing the object-oriented programming (OOP) concept in order to organize and manipulate the objects in the web application and mobile application.

5. Validation

In the validation phase the system is being tested and document the process to meet the effectiveness of the application.

5.1 Test Plan

The strategy, goals, timetable, estimation, deliverables, and resources needed to carry out testing on a software product are all described in detail in a test plan. The test plan aids in estimating the amount of work required to verify the application's quality. The test manager carefully monitors and controls every aspect of the test plan to ensure that software testing activities are carried out according to a defined methodology [35].

Process of writing test plan

- Analyze the product
- Design the Test Strategy
- Define the Test Objectives
- Define Test Criteria
- Resource Planning
- Plan Test Environment
- Schedule & Estimation
- Determine Test Deliverables

Leisure Diary Test Plan

Table 10 Test Plan

Test Plan ID	001
Objectives	Leisure Diary test plan is to elaborate how the application is to be tested in order to deliver an effective product. Testing the Web and mobile applications by checking whether the systems allows the user to log in without any problems if the correct Username and Password are entered, inserting data through the interfaces, updating existing data through the interfaces, deleting data in

	the database through the interfaces and the API connectivity.
Products to be tested	• Leisure diary Web application • Leisure Diary Mobile application
Testing features	Web application Login and Registration functions. Mobile application Login and Registration functions. Web application CRUD operations Mobile application API fetching
Approach	Unit testing
Testing Environment	VS Code
Testing tasks	Planning, Implementing, designing, Executing
Deliverables	Unit testing codes, Results

5.2 Testing

Unit Testing

Test case #: 1

Module: Web application user registration

Component: User registration

Scenarios: User should be able to register in to a new account using their username, email and password

Preconditions:

- Server up and running.

- Test Db connection

Expected Results:

1. Should return error with status 400 if username is not provided'
2. Should return error with status 400 if email is not provided
3. Should return error with status 400 if password is not provided
4. Should return error with status 400 if password is less than 6 characters
5. Should create a new service provider if inputs are valid
6. Should return error with status 400 if username or email already in use

```
const request = require('supertest');
const bcrypt = require('bcrypt');
const app = require('../server');
const serProvModel = require('../server/model/serviceProvider');
const mongoose = require('mongoose');
const http = require('http');




describe('POST /api/register', () => {
  let server;

  beforeEach(() => {
    server = http.createServer(app);
    server = app.listen(3000);

    mongoose.connect('mongodb+srv://sabithfariq:leisurediary@leisurediarytest.r04
yhsx.mongodb.net/test', { useNewUrlParser: true });
  });

  afterEach(async () => {
    await server.close();
    await serProvModel.deleteMany();
  });

  it('should return error with status 400 if username is not provided', async ()
=> {
    const response = await request(server)
      .post('/api/register')
      .send({
        email: 'test@example.com',
        password: 'password123',
      });

    expect(response.status).toBe(400);
    expect(response.body).toHaveProperty('status', 'error');
    expect(response.body).toHaveProperty('error', 'Invalid username');
  });
it('should return error with status 400 if email is not provided', async () => {
    const response = await request(server)
      .post('/api/register')
      .send({
        username: 'testuser',
        password: 'password123',
```

```
  });

    expect(response.status).toBe(400);
    expect(response.body).toHaveProperty('status', 'error');
    expect(response.body).toHaveProperty('error', 'Invalid email');
  });

 it('should return error with status 400 if password is not provided', async ()
=> {
    const response = await request(server)
      .post('/api/register')
      .send({
        username: 'testuser',
        email: 'test@example.com',
      });

    expect(response.status).toBe(400);
    expect(response.body).toHaveProperty('status', 'error');
    expect(response.body).toHaveProperty('error', 'Invalid password');
  });

  it('should return error with status 400 if password is less than 6 characters',
async () => {
    const response = await request(server)
      .post('/api/register')
      .send({
        username: 'testuser',
        email: 'test@example.com',
        password: '123',
      });

    expect(response.status).toBe(400);
    expect(response.body).toHaveProperty('status', 'error');
    expect(response.body).toHaveProperty(
      'error',
      'Password should be at least more than 6 Characters'
    );
  });
```

```
it('should create a new service provider if inputs are valid', async () => {
    const response = await request(server)
      .post('/api/register')
      .send({
        username: 'testuser',
        email: 'test@example.com',
        password: 'password123',
      });

    expect(response.status).toBe(200);
    expect(response.body).toHaveProperty('status', 'ok');

    const serviceProvider = await serProvModel.findOne({ username: 'testuser' });
    expect(serviceProvider).toBeTruthy();

    const passwordMatch = await bcrypt.compare(
      'password123',
      serviceProvider.password
    );
    expect(passwordMatch).toBe(true);
  });

  it('should return error with status 400 if username or email already in use',
async () => {
    const existingServiceProvider = await serProvModel.create({
      username: 'testuser',
      email: 'test@example.com',
      password: 'password123',
    });

    const response = await request(server)
      .post('/api/register')
      .send({
        username: 'testuser',
        email: 'test@example.com',
        password: 'password456',
      });

    expect(response.status).toBe(400);
    expect(response.body).toHaveProperty('status', 'ok');
    expect(response.body).not.toHaveProperty('error');
    expect(typeof response.body).toBe('object');
    expect(response.headers['content-type']).toMatch(/json/);
});
});
```

Figure 99 Unit Testing 1

Result

Table 11 Test Result 1

Test Case	Result
Should return error with status 400 if username is not provided'	PASS
Should return error with status 400 if password is not provided	PASS
Should return error with status 400 if password is less than 6 characters	PASS
Should create a new service provider if inputs are valid	PASS
Should return error with status 400 if username or email already in use	PASS

Test case #: 2

Module: Web application- Login

Component: User Login

Scenarios: User should be able to login using their registered credentials

Preconditions:

1. Server up and running.
2. Test Db connection
3. Registered user

Expected Results:

1. Should return error with status 400 if username is not provided'
2. Should return error with status 400 if email is not provided
3. Should return error with status 400 if password is not provided
4. Should return error with status 400 if password is less than 6 characters

5. Should create a new service provider if inputs are valid
6. Should return error with status 400 if username or email already in use

```
const request = require('supertest');
const app = require('../server');

describe('POST /api/login', () => {
  test('valid login credentials should return 200 OK', async () => {
    const response = await request(app)
      .post('/api/login')
      .send({
        email: 'testuser@example.com',
        password: 'password123'
      });
    expect(response.statusCode).toBe(200);
    expect(response.body).toHaveProperty('status', 'ok');
  });

  test('invalid login credentials should return 401 Unauthorized', async () => {
    const response = await request(app)
      .post('/api/login')
      .send({
        email: 'testuser@example.com',
        password: 'invalidpassword'
      });
    expect(response.statusCode).toBe(401);
    expect(response.body).toHaveProperty('status', 'error');
  });
});
```

Figure 100 Test 2

Table 12 Test Result 2

Test Case	Result
Valid login credentials should return 200 OK	PASS

Invalid login credentials should return 401 Unauthorized	**PASS**

API Testing

Test case #: 3

Module: Web application- Adding new service

Component: Creating new service

Scenarios: User can create a new service

Expected Results:

1. Should create a new accommodation
2. Should handle errors when creating a new accommodation

```
describe('POST /add-accomodation', () => {
  it('should create a new accomodation', async () => {
    const accomodationData = {
      serviceProvider: 'John Doe',
      title: 'My Accomodation',
      description: 'A lovely place to stay',
      otherdesc: 'Some other details',
      location: 'Somewhere',
      price: 100,
      category: 'Hotel',
      phone: '123-456-7890',
      startDate: '2022-01-01',
      endDate: '2022-01-05',
      image: {
        buffer: Buffer.from('dummy image data'),
        mimetype: 'image/png',
      },
    };
    const response = await request(app)
      .post('/add-accomodation')
      .field(accomodationData)
      .attach('image', accomodationData.image.buffer, { mimetype:
accomodationData.image.mimetype });
    expect(response.status).toBe(302);
    expect(response.header.location).toBe('/dashboard/accomodation');
    const savedAccomodation = await Accomodation.findOne({ title:
accomodationData.title });
    expect(savedAccomodation).toBeTruthy();
    expect(savedAccomodation.serviceProvider).toBe(accomodationData.serviceProvid
er);
    expect(savedAccomodation.description).toBe(accomodationData.description);
    expect(savedAccomodation.location).toBe(accomodationData.location);
    expect(savedAccomodation.price).toBe(accomodationData.price);
    expect(savedAccomodation.category).toBe(accomodationData.category);
    expect(savedAccomodation.phone).toBe(accomodationData.phone);
    expect(savedAccomodation.startDate).toEqual(new
Date(accomodationData.startDate));
    expect(savedAccomodation.endDate).toEqual(new
Date(accomodationData.endDate));
    expect(savedAccomodation.image.contentType).toBe(accomodationData.image.mimet
ype);
  });
```

Table 13 Test 3

Test Case	Result
Should create a new accommodation	PASS
Should handle errors when creating a new accommodation	PASS

Test case #: 4

Module: Web application- Delete Service

Component: Service

Scenarios: User should be able to delete selected service

Preconditions: Created service.

Expected Results: Should delete the accommodation with the specified ID and redirect

```
const request = require('supertest');
const app = require('../server');
const Accomodation = require('../models/Accomodation');

describe('DELETE /delete-accomodation/:id', () => {
  it('should delete the accomodation with the specified ID and redirect to /dash-
board/accomodation', async () => {
    // create a new accomodation
    const newAccomodation = new Accomodation({
      serviceProvider: 'test',
      title: 'test',
      description: 'test',
      otherdesc: 'test',
      location: 'test',
      image: {
        data: 'test',
        contentType: 'test'
      },
      price: 'test',
      category: 'test',
      phone: 'test',
      startDate: 'test',
      endDate: 'test'
    });
    const savedAccomodation = await newAccomodation.save();

    // make a request to delete the accomodation
    const response = await request(app)
      .post(`/delete-accomodation/${savedAccomodation._id}`)
      .send();

    // assert that the response status is a redirect
    expect(response.status).toBe(302);

    // assert that the response redirected to /dashboard/accomodation
    expect(response.header.location).toBe('/dashboard/accomodation');

    // assert that the accomodation has been deleted from the database
    const deletedAccomodation = await Accomodation.findById(savedAccomoda-
tion._id);
    expect(deletedAccomodation).toBeNull();
  });
});
```

Figure 101 Test 4

Test Case	Result
Should delete the accommodation with the specified ID and redirect	PASS

Test case #: 5

Module: Reservation Model- New reservation

Component: Creating new reservation

Scenarios: User can create a new reservation

Expected Results:

1. Reservations created via mobile application should be posted on the backend

```
const request = require('supertest');
const app = require('../server);
const Reservations = require('../models/Reservations');

describe('POST /api/Reservations', () => {
  it('should create a new reservation and redirect to /dashboard/accomodation',
async () => {
    const newReservation = {
      name: 'John Doe',
      email: 'johndoe@example.com',
      accomodation: 'Luxury Villa',
      checkInDate: '2023-05-01',
      checkOutDate: '2023-05-07',
      guests: 4,
      message: 'I would like to request a room with a view.'
    };

    const response = await request(app)
      .post('/api/Reservations')
      .send(newReservation);

    expect(response.status).toBe(302); // Check if the response is a redirect
    expect(response.header.location).toBe('/dashboard/accomodation'); // Check if
the redirect location is correct

    const reservation = await Reservations.findOne({ email: newReservation.email
});
    expect(reservation).toMatchObject(newReservation); // Check if the created
reservation matches the request body
  });
});

describe('GET /reservation-accomodation', () => {
  it('should render the reservationsPage view and return all reservations', async
() => {
    const response = await request(app).get('/reservation-accomodation');
    expect(response.status).toBe(200); // Check if the response is OK

    const reservations = await Reservations.find();
    expect(reservations.length).toBeGreaterThan(0); // Check if there is at least
one reservation
    expect(response.text).toContain(reservations[0].name); // Check if the re-
sponse body contains the name of the first reservation
  });
});
```

Figure 102 Test 5

Test Case	Result
Reservation created successfully	PASS

Table 14 Test 5

API Testing

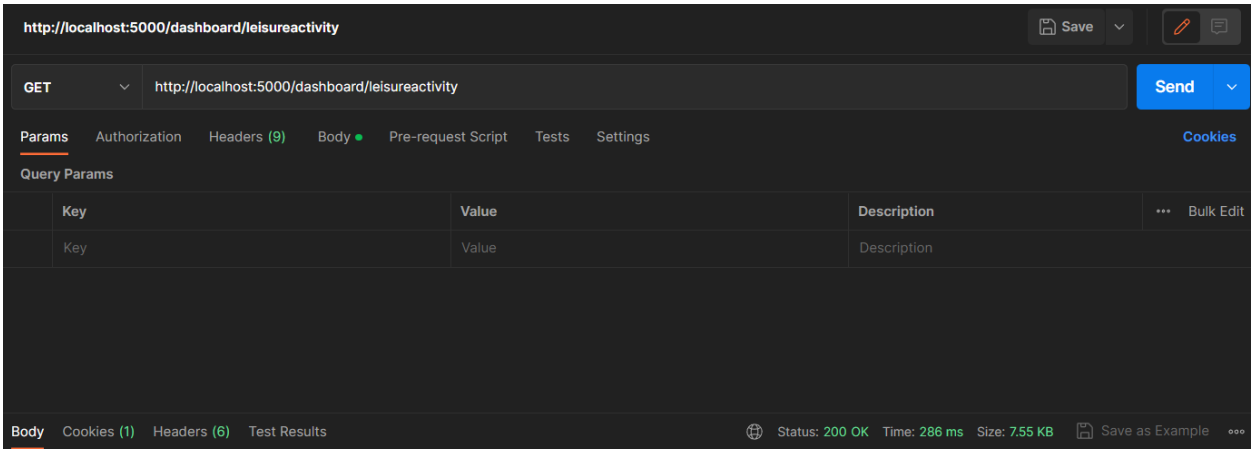


Figure 103GET API Leisure Activity

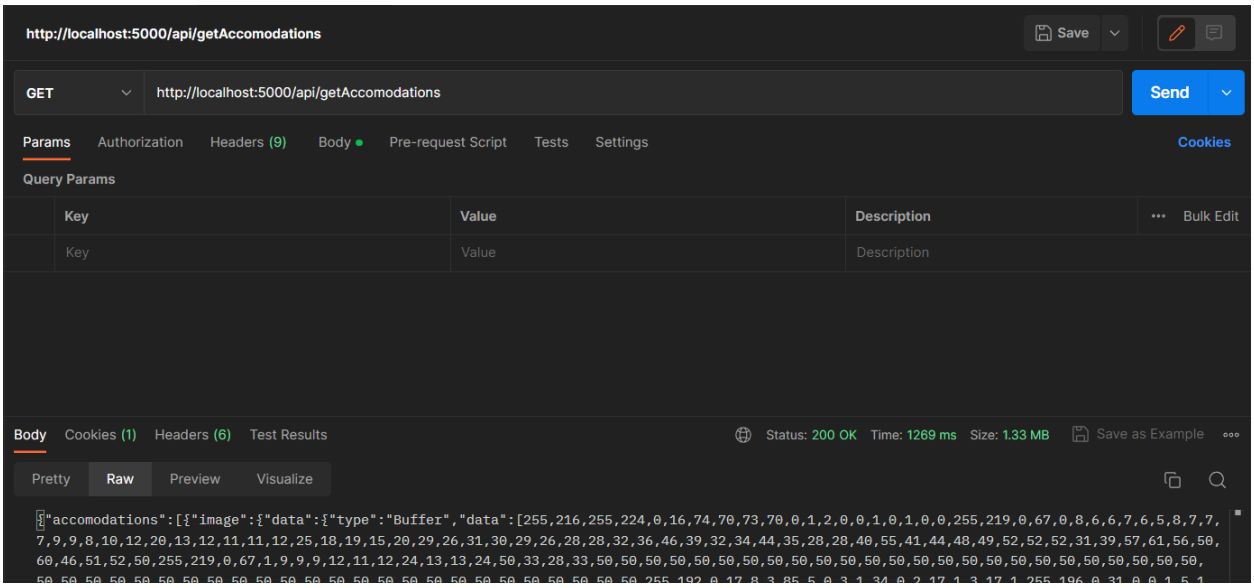


Figure 104 Get API

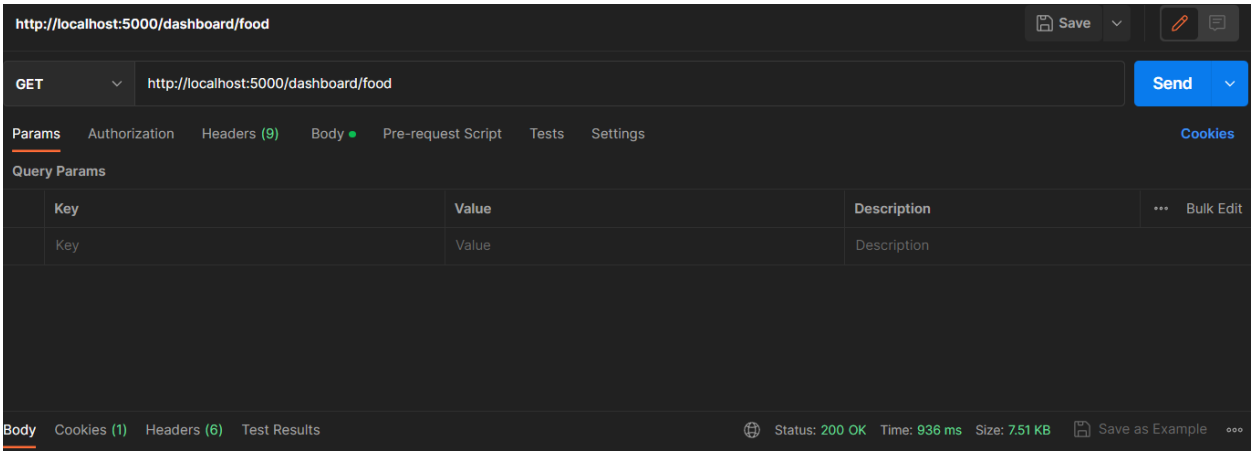


Figure 105 Food Get API

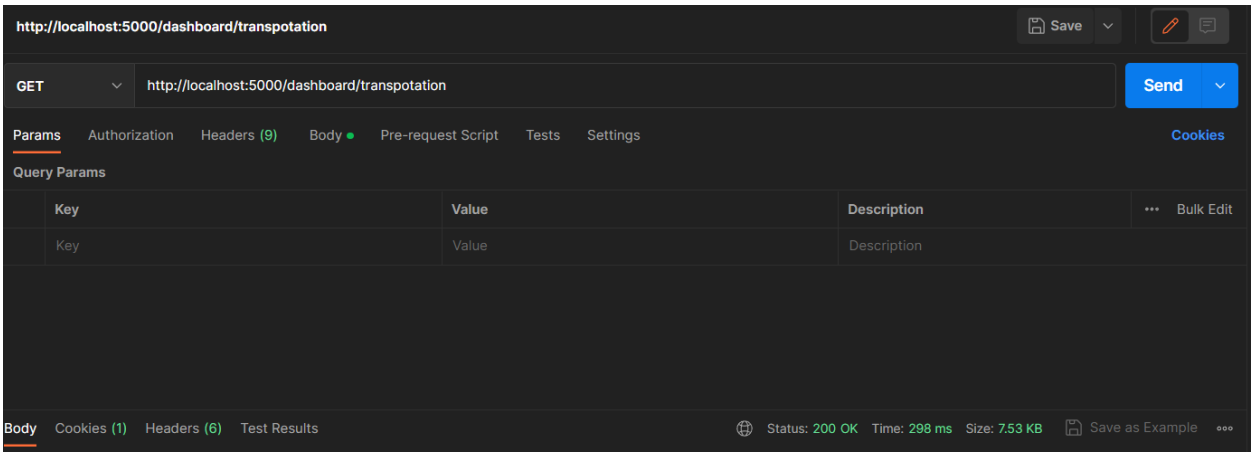


Figure 106Transportation GET API

Mobile application

Test case #: 6

Module: Mobile application- Login

Component: User Login

Scenarios: User should be able to login using their registered credentials

Preconditions:

1. Server up and running.
2. Test Db connection
3. Registered user

Expected Results:

1. Should display the Login Screen
2. Should display an error message if email and password are not entered
3. Should navigate to homepage when login is successful

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:mockito/mockito.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:travel_app_ui/screens/login/login.dart';
import 'package:travel_app_ui/utils/config.dart';
import 'package:http/http.dart' as http;

class MockSharedPreferences extends Mock implements SharedPreferences {}

class MockHttpClient extends Mock implements http.Client {}

void main() {
  group('LoginScreen', () {
    late LoginScreen loginScreen;
    late MockSharedPreferences mockSharedPreferences;
    late MockHttpClient mockHttpClient;

    setUp(() {
      mockSharedPreferences = MockSharedPreferences();
      mockHttpClient = MockHttpClient();

      loginScreen = LoginScreen();
    });

    testWidgets('should display the Login Screen', (WidgetTester tester) async {
      await tester.pumpWidget(loginScreen);
      expect(find.text('Login'), findsOneWidget);
    });

    testWidgets('should display an error message if email and password are not
entered', (WidgetTester tester) async {
      await tester.pumpWidget(loginScreen);

      // Tap the login button
      await tester.tap(find.text('Login').last);
      await tester.pump();

      expect(find.text('Please enter your email and password'), findsOneWidget);
    });

    testWidgets('should navigate to HomePage when login is successful',
(WidgetTester tester) async {

      when(mockHttpClient.post(Uri.parse(login),
              headers: {"Content-Type": "application/json"},
```

```
            body: anyNamed('body')))
          .thenAnswer((_) async => http.Response('{"status": true, "token":
"123"}', 200));


      when(mockSharedPreferences.setString('token', '123')).thenAnswer((_) async
=> true);


      loginScreen.prefs = mockSharedPreferences;
      loginScreen.client = mockHttpClient;

      await tester.pumpWidget(loginScreen);

      // Enter email and password
      await tester.enterText(find.byType(TextFormField).first,
'test@example.com');
      await tester.enterText(find.byType(TextFormField).last, 'password');

      // Tap the login button
      await tester.tap(find.text('Login').last);
      await tester.pumpAndSettle();

      // Verify that we navigate to HomePage
      expect(find.byType(HomePage), findsOneWidget);
    });
  });
}
```

Figure 107 Test Case 6

Test Case	Result
Should display the Login Screen	PASS
Should display an error message if email and password are not entered	FAIL
Should navigate to homepage when login is successful	PASS

Test case #: 7

Module: Mobile application- API Testing

Component: API Model

Scenarios: Mobile application should directly interact with the Hosted API

Preconditions:

1. Mobile App connected to web application API.
2. API end points

Expected Results:

1. API returns Data

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:flutter_test/flutter_test.dart';

import '../models/AccomModel.dart';
import '../services/Api.dart';

void main() {
  test('Test API', () async {
    List<Accomodation> accomodations = await Api.getServices();
    expect(accomodations.length, greaterThan(0));
  });
}

class Api {
  static Future<List<Accomodation>> getServices() async {
    List<Accomodation> accomodations = [];

    var url = Uri.parse('http://192.168.8.188/api/getAccomodations');

    try {
      final res = await http.get(url);

      if (res.statusCode == 200) {
        var data = jsonDecode(res.body);
        // print('res.body ${data}');

        data['accomodations'].forEach((value) => {
              accomodations.add(Accomodation(
                  serviceProvider: value['serviceProvider'].toString(),
                  title: value['title'].toString(),
                  description: value['description'].toString(),
                  otherdesc: value['otherdesc'].toString(),
                  location: value['location'].toString(),
                  image: value['image'].toString(),
                  price: value['price'].toString(),
                  category: value['category'].toString(),
                  phone: value['phone'].toString(),
                  startDate: value['startDate'].toString(),
                  endDate: value['endDate'].toString()))
            });
        return accomodations;
      } else {
```

```
        return [];
      }
    } catch (e) {
      print(e.toString());
      print('Errorrrrrrrrrrrrrrr');
      return [];
    }
  }
}
```

Figure 108 Test case 7

Test Case	Result
API returns Data	PASS

Blackbox Testing

Testing has been also by using black box testing to check other important functions.

Test case #: 8

Module: Web Application- Service creation

Component: Services Model

Scenarios: Service provider can create a service and it will be updated to API and the Database meantime

Preconditions:

1. Service up and running.
2. API end points

Expected Results:

- Service successfully added to the API and Database

Creating new service

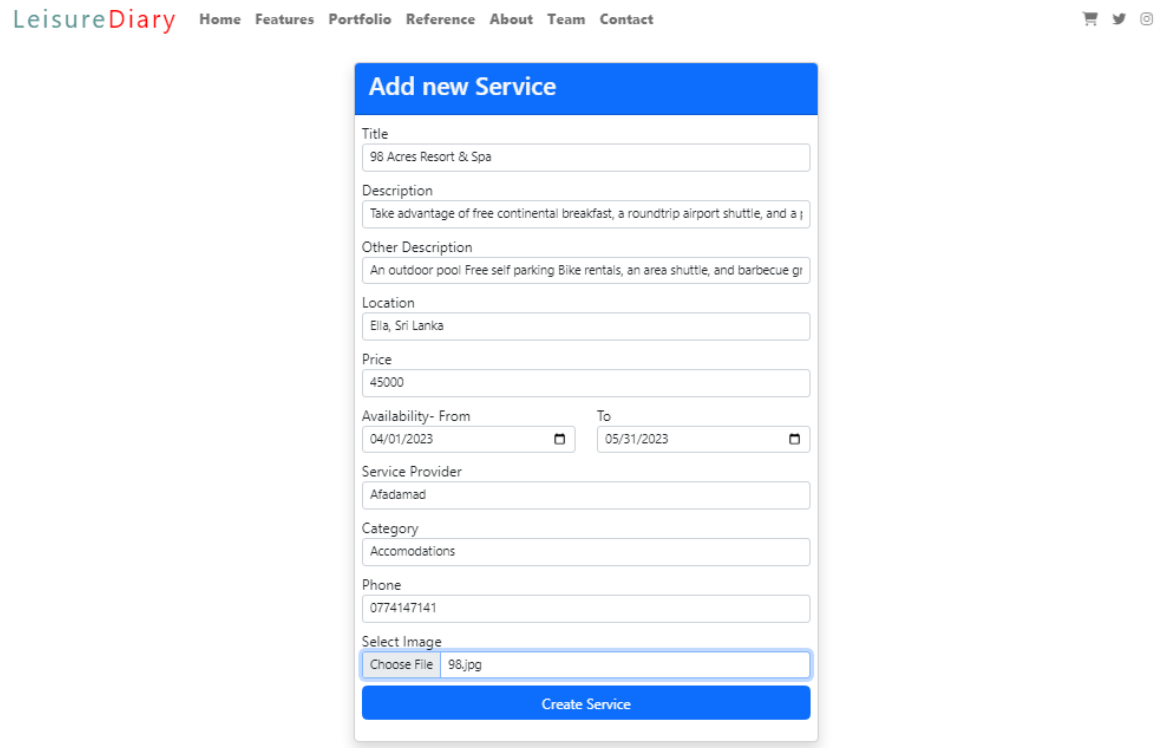


Figure 109 Testing case 8

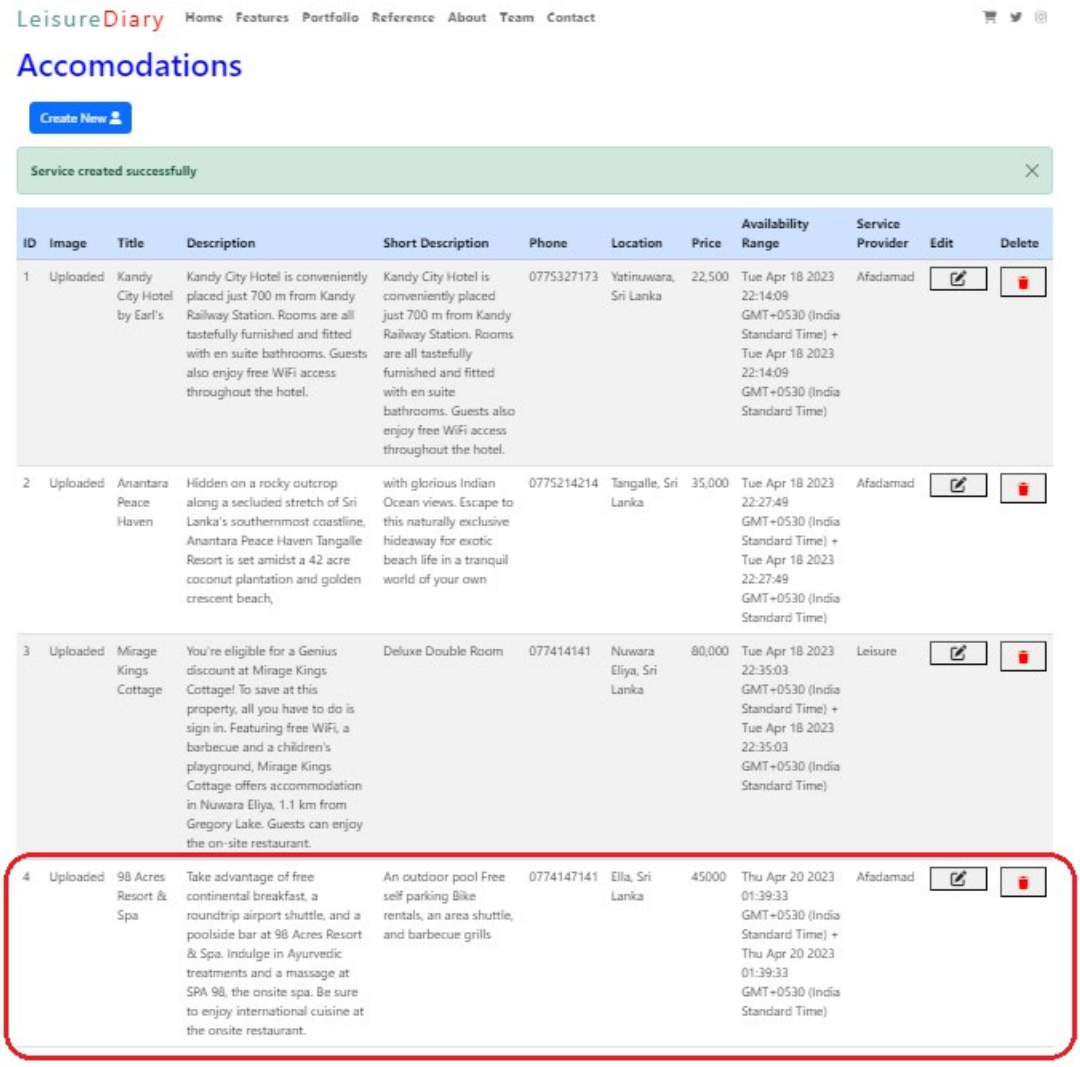


Figure 111 Created service

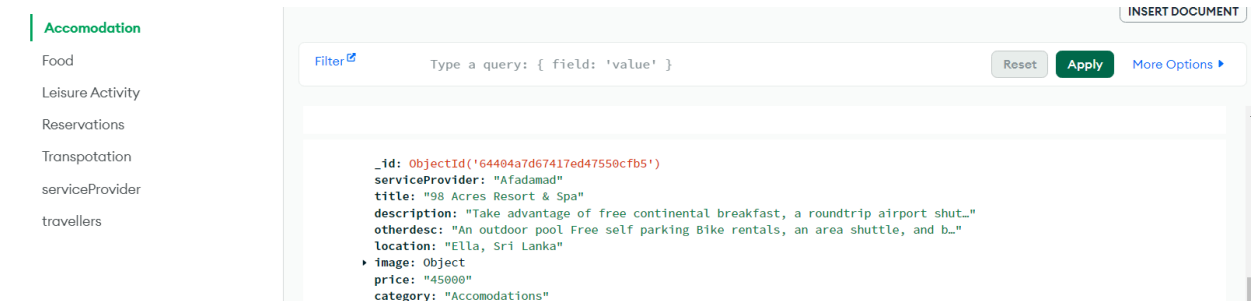


Figure 110 Database collection

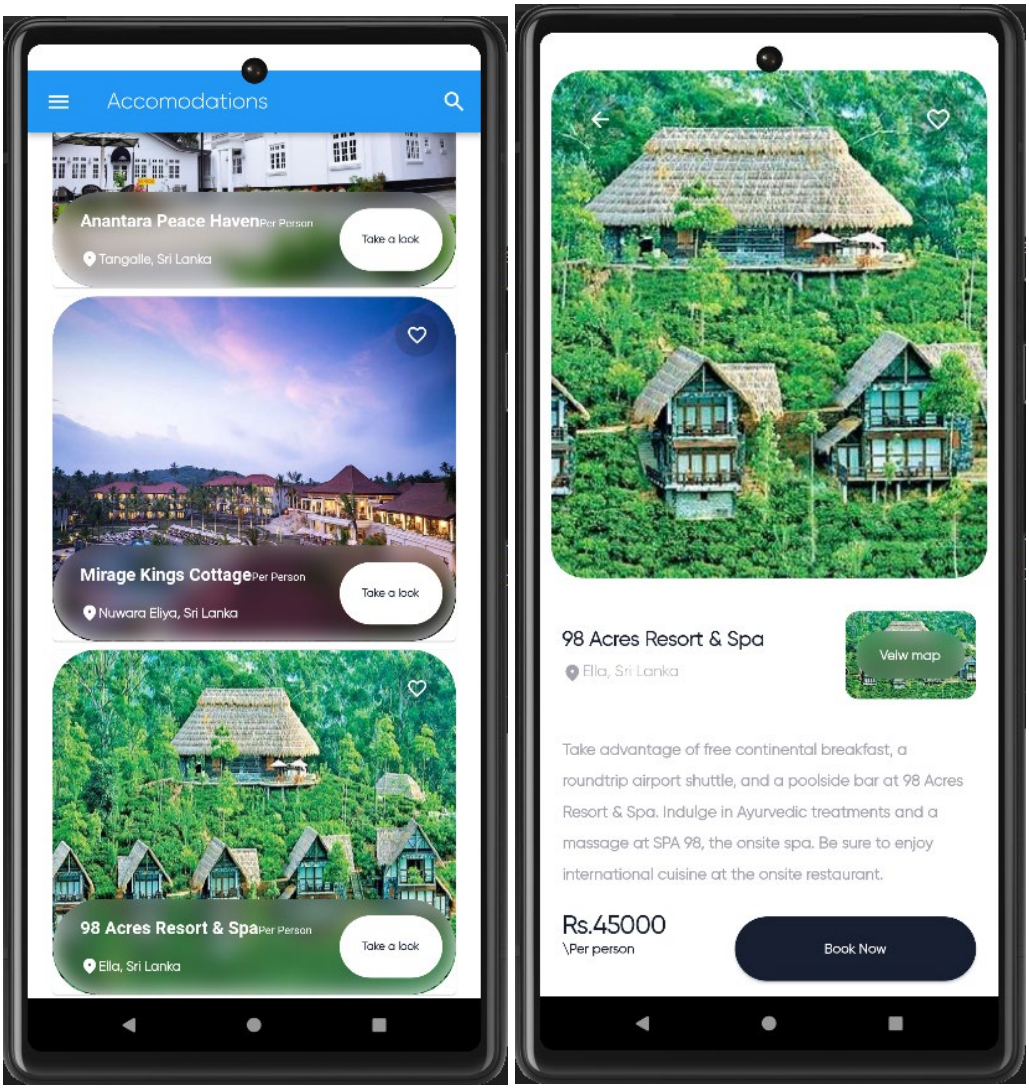


Figure 112 Test case 8

Reserving the location.

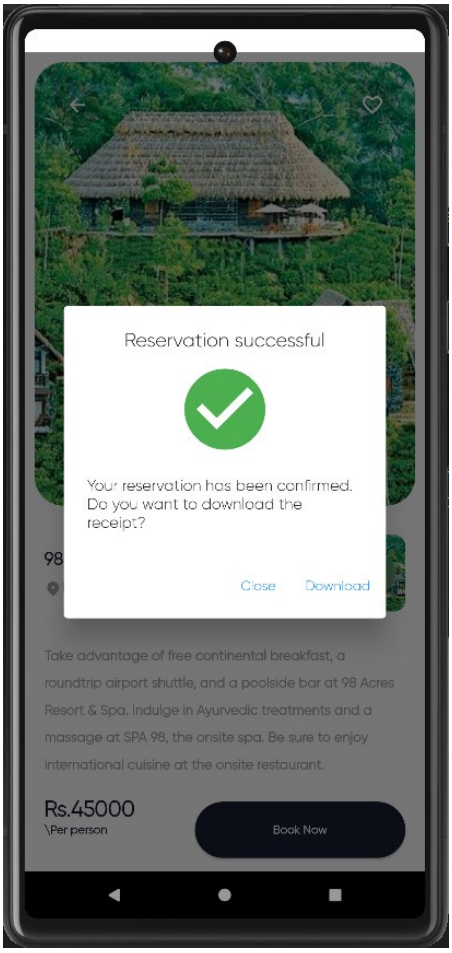


Figure 113 Test case 8

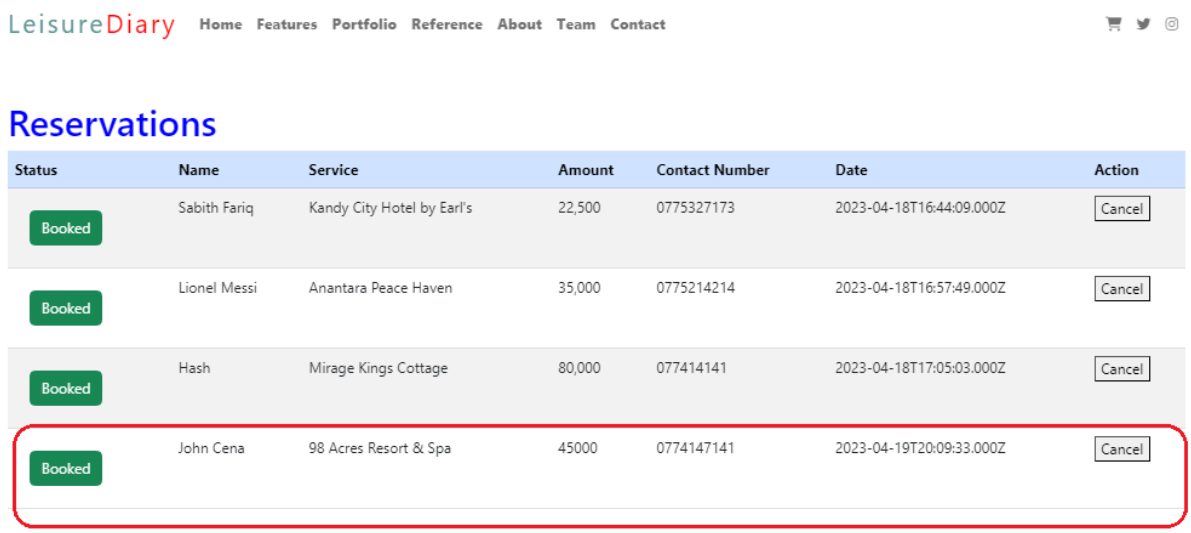


Figure 114 Test case 8

Based on the test results, we can conclude that the functions have passed 95% of the test cases. While there is always room for improvement, this is a strong indication that the functions are working as expected and can be considered reliable for their intended use. It is important to continue testing and refining the code to ensure that it meets all requirements and provides a seamless experience for users.

6. Critical Review & Conclusion

6.1 Closing executive summary

In conclusion, the development of our application and the web is in the process of development. Through the system development, it has always made sure to follow the best practices to make the application user friendly. The application includes range of features as elaborated in the functional requirement phase. The application is being tested manually and automatically using

various tools to make sure the users will be getting the error free best version of the application. Such testing brings us the confident in launching the application to the market.
Furthermore, application is being implemented under continues monitoring to provide a best version that will satisfy either party. Also, system will be undergone to get user feedbacks and make improvements where ever it is necessary. As well as we will also explore the opportunities to improve the applications features and benefits. In Addition, followings have been identified as potential evolutions to improve the application,

- Create trips and connect their friends
- Interacting option with their Travelers and service providers.
- AI to suggest options
- Language translator.
- Current update about the locations (Weather)
- Accounting system to track the spending and provide reports to others for transparent purposes.

6.2 Conclusion

Traveling is a globally fast-growing industry. Which benefits the travelers by fulfilling their needs and wants on travel planning and organizing such as finding required needs and navigating them to convenient path. Moreover, the related service providing businesses also benefits by interconnecting the travelers to reach their services. People always vote for the convenience. Majority of tourists prefer to use online platforms for travel planning and organizing via websites, social media contents and travel agencies. Also, their study indicates that travelers who have used online platforms to plan and organize their trips are more satisfied. Service providers can sustain their business if the travelers ended their journey with satisfaction.

References

[1] M. K. Pratt, "Tech Target," [Online]. Available: https://www.techtarget.com/searchcio/definition/project-scope.

[2] Retrieved, 17 April 2023. [Online].

[3] M. Hajian, "logrocket," 2021. [Online]. Available: https://blog.logrocket.com/implement-svg-flutter-apps/.

[4] N. Srivastava, "flutterdevs," 2021. [Online]. Available: https://medium.flutterdevs.com/swiper-in-flutter-56ee23b54391.

[5] A. Rawat, "flutterdevs," 2019. [Online]. Available: https://medium.flutterdevs.com/using-sharedpreferences-in-flutter-251755f07127.

[6] D. &. C. L. Wang, "Online travel review and hotel choice," *Annals of Tourism Research,* pp. 42, 214-216., 2013.

[7] Soekadijo, 2003.

[8] C. K. L. Y. K. &. L. B. Lee, " Understanding the factors influencing tourists' decision-making process. Journal of Travel Research,," Vols. 61-77, p. 57(1), 2018.

[9] C. M. K. &. W. M. Deegan, "The impact of travel planning on travel experience," *Journal of Travel Research,* Vols. 745-757, p. 56(6), 2017.

[10] H. T. &. T. A. D. Vu, " Factors influencing tourists' intention to engage in sustainable travel behavior," *An integrated model. Journal of Sustainable Tourism,* Vols. 1352-1372, p. 26(8), 2018.

[11] T. D. H. a. J. E. S. Higham, 2001.

[12] Leiper, http://eprints.polsri.ac.id/2940/3/FILE%203.pdf, 1998.

[13] Spillane, 1987.

[14] Yoeti, 1995.

[15] Pendit, 1994.

[16] Mansur, 2009.

[17] K. a. Armstrong.

[18] D. &. L. R. Buhalis, "Progress in information technology and tourism management," *20 years on and 10 years after the Internet—The state of eTourism research. Tourism Management,,* Vols. 609-623, p. 29(4), 2008.

[19] X. L. L. &. Y. L. Yu, "A study of Chinese outbound tourists," *Journal of Hospitality and Tourism Technology,* vol. 9(1), pp. 68-85, 2018.

[20] Z. D. Q. &. M. Y. Xiang, "Travelers' e-word-of-mouth communication on social networking sites," *An empirical study of Chinese tourists. Journal of Travel Research,* vol. 56(6), pp. 754-767, 2017.

[21] M. &. L. H. Kim, "Determinants of users' continued use of travel apps," *The moderating effect of privacy concern. Journal of Travel Research,* vol. 57(8), pp. 1066-1080, 2018.

[22] F. R. K. &. H. K. Ali, "Influence of online reviews on consumers' hotel booking decisions," *International Journal of Hospitality Management,* pp. 53, 74-85., 2016.

[23] P. R. S., "Software engineering: a practitioner's approach. McGraw Hill Education.," 2014. [Online].

[24] S. W. Ambler, ". The elements of UML 2.0 style. Cambridge University Press.," 2002.

[25] R. Jeffries, "User stories applied. Cigital Incorporated.," 2001. [Online].

[26] G. &. S. I. Kotonya, "Requirements engineering: Processes and techniques. John Wiley & Sons.," 1998. [Online].

[27] J. Bridges, "projectmanager," 2021. [Online]. Available: https://www.projectmanager.com/training/how-to-conduct-a-feasibility-study.

[28] blogprosperi, "prosperiglobal.com/," 16, 2021. [Online]. Available: https://blog.prosperiglobal.com/economic-feasibility-study/.

[29] C. J. M. &. M. D. Ghezzi, *Fundamentals of software engineering. Prentice Hall,* 2003.

[30] Lucidchart, "lucidchart," 2022. [Online]. Available: https://www.lucidchart.com/pages/what-is-a-flowchart.

[31] mindmanager, "mindmanager," no. https://www.mindmanager.com/en/features/activity-diagram/.

[32] P. aktualizácia:, "ibm.com," 2021-03-05. [Online]. Available: https://www.ibm.com/docs/sk/rsm/7.5.0?topic=uml-sequence-diagrams.

[33] B. Hyman, "System Architecture," 2012.

[34] R. Bhadauria, "codeproject," [Online]. Available: https://www.codeproject.com/Articles/1057645/Model-View-Controller-Service-MVCS-Architecture.

[35] T. Hamilton, "guru99," 11 February 2023. [Online]. Available: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html.

[36] N. F. P. L. S. K. a. T. S. N. Tavichaiyuth, Covid-19 Travel Planner Mobile Application Design with Lean Product Process Framework, 2022.

[37] Flutter, "flutter.dev," [Online]. Available: https://flutter.dev/.

Annexure A

Deployment plan

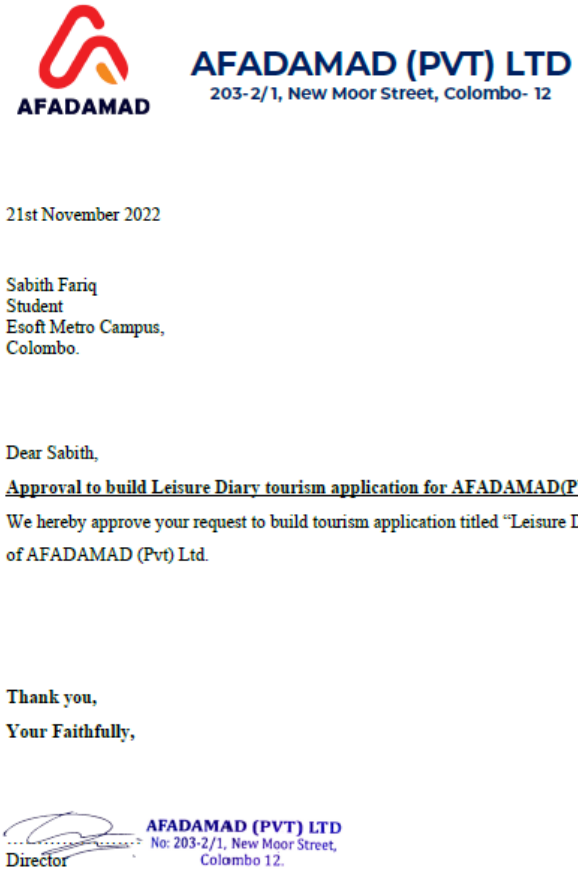
- Deployment requirements –
- Hardware requirements: Cloud server to run the web application and database servers, mobile application deployment environment
- Software requirements: Web server (Node.js), database server (MongoDb), mobile application (Flutter)
- Dependencies: Node.js, Flutter & API Gateway
- Environment preparation:
- Set up dedicated servers for backend web application and database servers.
- Install and configure necessary software on each server.
- Configure firewalls and security settings to restrict access to the servers
- Deploy mobile application and connect it with the API gateway.
- Configure the web application and Mobile application:
- Configure the web application to connect to the database server using a database connection string
- Configure the mobile application to connect to the API Gateway endpoint for retrieving data
- Configure security settings for the web application, including SSL/TLS certificates and authentication
- Build and package the application:
- Compile the web application code into the domain and host the web application.
- Compile the mobile application code as an APK file
- Create a deployment script to automate the deployment process

- Deploy the application:

- Copy the web application package to the web server and extract it
- Copy the mobile application package to the mobile application deployment environment and install it on a mobile device or emulator
- Run the deployment script to configure the application and start it
- Test the applications:
- Perform functional testing of the web application and mobile application to ensure that they are working as expected
- Perform load testing to ensure that the application can handle a high volume of requests
- Perform security testing to identify any vulnerabilities in the application
- Monitor and maintain the applications:
- Set up monitoring tools to track application performance and detect any issues
- Perform regular maintenance tasks, such as database backups and security patches
- Address any issues that arise and continue to improve the application over time

Annexure B

Approval Letter from the company.



AFADAMAD (PVT) LTD
203-2/1, New Moor Street, Colombo- 12

21st November 2022

Sabith Fariq
Student
Esoft Metro Campus,
Colombo.

Dear Sabith,

Approval to build Leisure Diary tourism application for AFADAMAD(PVT) LTD

We hereby approve your request to build tourism application titled "Leisure Diary" on behalf of AFADAMAD (Pvt) Ltd.

Thank you,

Your Faithfully,

Director

AFADAMAD (PVT) LTD
No: 203-2/1, New Moor Street,
Colombo 12.

Annexure C

Other coding snippets.

```
const TravellerService = require('../services/travellerService');


exports.register= async (req, res, next)=>{
try {
    // getting input
    const {name, email, password} = req.body;

    const isSuccess = await TravellerService.registerUser(name, email, password);

    res.json({
        status: true,
        success:"User registered successfully"
    });
} catch (error) {

}
}


exports.login= async (req, res, next)=>{
try {
    // getting input
    const {email, password} = req.body;


    let traveller = await TravellerService.checkUser(email);

    console.log('traveler', traveller);

    if(!traveller){
        throw new Error('User doesn\'t exist');
    }
        const isMatch = await traveller.comparePassword(password);

        if (isMatch=== false) {
            throw new Error('Invalid Password');
        }
```

```
let tokenData = {_id:traveller._id, email:traveller.email};
        console.log('Token data', tokenData);

        const token= await TravellerService.generateToken(tokenData, "secretKey",
'1h');
        console.log('Token', token);
        res.status(200).json({status:true, success:'sendData', token:token});


} catch (error) {
    console.log(error, 'errorrrrrr');
    next (error);

}
}
```

Front End Coding

EJS Header

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><%= title %></title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/5.2.3/js/bootstrap.min.js" integrity="sha512-
1/RvZTcCDEUjY/CypiMz+iqqtaoQfAITmNSJY17Myp4Ms5mdxPS5UV7iOfdZoxcGhzFbOm6sntTKJppjv
uhg4g==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ"
crossorigin="anonymous">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" integrity="sha512-
iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46ZmdtAc9eNBvH0H/
ZpiBw==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <link rel="stylesheet" href="css/mdb.min.css" />
    <link rel="stylesheet" href="css/dashboard.css">
    <link href="https://cdn.datatables.net/v/bs5/dt-1.13.4/datatables.min.css"
rel="stylesheet"/>
    <link href="https://cdn.datatables.net/v/bs5/dt-1.13.4/datatables.min.css"
rel="stylesheet"/>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js"
integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ih
JhDpQ==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<!-- Google Maps JavaScript library -->
    <script
src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=places&key=AIzaSyB
sKd7zSRFM9QkYa39-a_EXBSePqSaHuK8"></script>

    <script>
      var searchInput = 'search_input';
```

```
<section>
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <% if (message){ %>
                    <div class="alert alert-success alert-dismissible fade show
alert-<%= message.type %>" role="alert">
                    <button class="btn-close" type="button" data-bs-
dismiss="alert" aria-label="close"></button>

                <strong><%= message.message %></strong>
                </div> <% } %>



                <div class="table-responsive table">

                    <% if (accomodation != '') { %>
                        <table class="table table-striped">
                            <thead>
                                <tr class="table-primary">
                                    <th>ID</th>
                                    <th>Image</th>
                                    <th>Title</th>
                                    <th>Description</th>
                                    <th>Short Description</th>
                                    <th>Phone</th>
                                    <th>Location</th>
                                    <th>Price</th>
                                    <th>Availability Range</th>
                                    <th>Service Provider</th>
                                    <th>Edit</th>
                                    <th>Delete</th>
                                </tr>
                            </thead>
                            <tbody>
                                <!-- for(var i = 0; i<users.length; i++) -->
                               <%  for(var i = 0; i<accomodation.length; i++){ %>
                                <tr>
                                    <td><%= i+1 %></td>
                                    <!-- '/uploads/' + accomodation[i].image %> -
->


                                    <td>
```

```
<td><%= accomodation[i].title %></td>
                                    <td><%= accomodation[i].description %></td>
                                    <td><%= accomodation[i].otherdesc %></td>
                                    <td><%= accomodation[i].phone %></td>
                                    <td><%= accomodation[i].location %></td>
                                    <td><%= accomodation[i].price %></td>
                                    <td><%= accomodation[i].startDate %> + <%=
accomodation[i].endDate %></td>
                                    <td><%= accomodation[i].serviceProvider
%></td>
                                    <td>
                                        <form method="POST" action="/update-
accomodation/<%= accomodation[i]._id%>?_method=UPDATE">
                                            <button type="submit"><i class="fas
fa-edit fa-lg mx-1 btn"></i></button>

                                        </form>
                                    </td>

                                    <td>
                                    <form method="POST" action="/delete-
accomodation/<%= accomodation[i]._id %>?_method=DELETE">
                                            <button type="submit"> <i class="fa-
sharp fa-solid fa-trash btn" style="color:red"></i></button>

                                        </form>

                                    </td>
                                </tr>
                               <% } %>
```

```
 <div class="card-body  p-2">
                    <!-- route name -->
                    <form action="/add-accomodation" method="post" id="add-form"
enctype="multipart/form-data">

                        <div class="mb-2">
                            <label for="title">Title</label>
                            <input type="text" name='title' class="form-control
form-control-sm" placeholder="Enter Title" required/>
                        </div>
                        <div class="mb-2">
                            <label for="description">Description</label>
                            <input type="text" name='description' class="form-
control form-control-sm" placeholder="Enter Main Description" required/>
                        </div>
                        <div class="mb-2">
                            <label for="otherdesc">Other Description</label>
                            <input type="text" name='otherdesc' class="form-
control form-control-sm" placeholder="Enter Other Description" required/>
                        </div>
                        <div class="mb-2">
                            <label for="location">Location</label>
                            <input type="text" name='location' class="form-
control form-control-sm" id="search_input" placeholder="Select Locaiton"
required/>
                        </div>

                        <div class="mb-2">
                            <label for="price">Price</label>
                            <input type="text" name='price' class="form-control
form-control-sm" placeholder="Enter Rate" required/>
                        </div>
                        <div class="row" id="availabilty">
                            <div class="mb-2 col">
                                <label for="availability">Availability-
From</label>
                                <input type="date" name='startDate'
id="startDate" class="form-control form-control-sm" placeholder="Select date"
required/>
                            </div>
                            <div class="mb-2 col">
                                <label for="availability">To</label>
                                <input type="date" name='endDate' id="endDate"
class="form-control form-control-sm" placeholder="Select date" required/>
                            </div>
```

```
</div>

                        <div class="mb-2">
                            <label for="serviceProvider">Service Provider</label>
                            <input type="text" name='serviceProvider'
class="form-control form-control-sm" placeholder="Service Provider" required/>
                        </div>
                        <div class="mb-2">
                            <label for="category">Category</label>
                            <select id="category" name="category" class="form-
control form-control-sm" placeholder="Category" required>
                                    <option
value="Accomodations">Accomodations</option>
                                    <option
value="Transpotation">Transpotation</option>
                                    <option value="leisureactivities">Leisure
Activities</option>
                                    <option value="food">Food</option>
                            </select>
                            <!-- <input type="text" name='category' class="form-
control form-control-sm" placeholder="Category" required/> -->
                        </div>
                        <div class="mb-2">
                            <label for="phone">Phone</label>
                            <input type="text" name='phone' class="form-control
form-control-sm" placeholder="Phone" required/>
                        </div>
                        <div class="mb-2">
                            <label for="image" class="form-lable">Select
Image</label>
                            <input type="file" name='image' id="image"
class="form-control form-control-sm" required/>
                        </div>
                        <div class="mb-3 d-grid">
                            <input type="Submit" name="submit" value="Create
Service" class="btn btn-primary btn-md">
                        </div>

                    </form>
```

Appendix D

Monitoring the Progress of Final Year Projects Final Project Log Sheet

Kingston University BSc. (Hons)

Project Title: Leisure Diary

Student Name: Mohamed Fariq Abdulla Sabith

Name of the Supervisor: Ms. Sampa Rasanie

#	Feedback criteria	Suggestion	Supervisor	Date
1	Project topic approval	Supervisory requested to improve the scope of the project	Prof. Ruvan	Fri, Oct 28, 2022
2		Supervisory approved the improved topic and requested to visit him for further improvement.	Prof. Ruvan	Sat, Oct 29, 2022
3		Visited the supervisor to improve the scope of the project.	Prof. Ruvan	Sun, Nov 06, 2022

4		Improved scope is approved by the Supervisor	Prof. Ruvan	Sun, Nov 13, 2022
5		Feedback from the supervisor for the documentation.	Prof. Ruvan	Sun, Nov 13, 2022
6	Interim Report	Sent the draft to the supervisor get obtain the feedback.	Ms. Sampa	Sat, Mar 18, 2023
7	Final Report	Sent a draft to obtain feedback	Ms. Sampa	Tue, Apr 18, 2023
8		Final report sent to obtain the approval	Ms. Sampa	Thu, Apr 20, 2023

Prof. Ruvan Abeysekera <ruvan@esoft.lk> Oct 28, 2022, 8:20 AM

to me

Dear Sabith,

Document is up to the standard. Need to improve the scope since this is a very common topic.

Kind regards,

Prof. Ruvan Abeysekara,
PhD(Doc.Eng.), MSc(CS), BSc,Dip.(Tec.Sc.), DFA, MBCS, MCS, MIEEE, MIEEECS, MIET, MIDES
Professor of Computer Science
CISCO Certified Trainer
Microsoft Certified Peer Coach

Prof. Ruvan Abeysekera <ruvan@esoft.lk> Oct 29, 2022, 4:26 PM

to me

Dear Sabith,

You can proceed with this and meet me when you come to the ESOFT.

Kind regards,

Prof. Ruvan Abeysekara,
PhD(Doc.Eng.), MSc(CS), BSc,Dip.(Tec.Sc.), DFA, MBCS, MCS, MIEEE, MIEEECS, MIET, MIDES
Professor of Computer Science
CISCO Certified Trainer
Microsoft Certified Peer Coach

Individual project

Sabith Fariq <sabithfariq@gmail.com>
to sampa

Thu, Mar 16, 9:25 PM

Dear Ms. Sampa,

I hope you are well.

I'm Sabith Fariq, SE Kingston Topup student. I have been assigned under your supervision for my individual project due to the unavailability Prof. Ruwan who was my supervisor.

To brief you about the project, I have got the approval to create a system consisting of a mobile app and a web app connected through API and cloud DB. Project tropic is Leisure Diary which is a travel-related application that is also known as an all-in-one hassle-free trip planner. There are service providers such as Hotel facilities, transport, food, and leisure activity provider who will add their services through the web portal provided. There is a mobile application that is used by travelers to make their trip plans and find all the services in one place and make their reservations.

I have attached my proposal for your perusal. I will visit you to describe further about the project. I have been working on my interim report and will send you soon for your review.

--

Sabith Fariq <sabithfariq@gmail.com>
to sampa

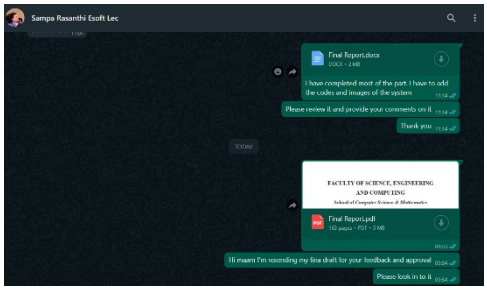
Sat, Mar 18, 1:40 PM

Dear Ms. Sampa,

I have attached my interim report. Please review and provide your feedback.

Thank you

One attachment • Scanned by Gmail



Github Link: https://github.com/SabithF/projectesoft/tree/main/Leisure%20Land