

Travels and Tales

A PROJECT REPORT

for

Mini Project (KCA353)

Session (2023-24)

Submitted by

Hritik Srivastava

University Roll No 2200290140074

Himanshu Raghav

University Roll No 2200290140073

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr Vipin Kumar
(Associate Professor)**



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

FEBRUARY 2024

CERTIFICATE

Certified that **Hritik Srivastava (2200290140074)**, **Himanshu Raghav (2200290140073)** have carried out the project work having “**Travels and Tales**” (**Mini Project KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Hritik Srivastava (2200290140074)

Himanshu Raghav (2200290140073)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The Travels and Tales website, built on the React framework, is a dynamic platform designed to immerse users in a world of exploration and storytelling. This abstract encapsulates the essence of the website

Travels and Tales is a captivating online destination that seamlessly combines the art of travel with the power of storytelling. Leveraging the versatility and efficiency of the React framework, our website offers users an immersive experience like no other. From inspiring destinations to personalized travel plans, users can embark on unforgettable journeys tailored to their preferences.

Through intuitive interfaces and seamless navigation, travelers can effortlessly browse and book flights, hotels, and activities, all in one place.

Real traveler reviews provide invaluable insights, while our dedicated customer support team ensures peace of mind throughout the adventure. With a focus on user engagement and interactivity,

Travels and Tales invites users to share their own stories, connect with like-minded explorers, and create lasting memories.

Join us on a journey of discovery and inspiration, where every click is a step towards a new adventure.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Hritik Srivastava

Himanshu Raghav

TABLE OF CONTENTS

Certificate	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv-v
List of Tables	vi
List of Figures	vii
1 Introduction	1-2
1.1 Background	1
1.2 Project overview	1
1.3 Objective	1
1.4 Key features	2
1.5 Scope of the project	2
2 Problem identification & feasibility study	3-4
2.1 Problem Identification	3
2.2 Feasibility Study	3
2.2.1 Technical Feasibility	3
2.2.2 Operational Feasibility	3
2.2.3 Economic Feasibility	4
3 Requirement Analysis	5-6
3.1 Introduction	5
3.2 Functional Requirements	5
3.2.1 User Module	5
3.2.2 Administrator Module	5
3.2.3 Package Module	5
3.3 Non-Functional Requirements	5
3.3.1 Performance	5
3.3.2 Security	6
3.3.3 Scalability	6
3.3.4 Usability	6
4 Project Planning and Scheduling	7
4.1 Pert Chart	7
5 Hardware and Software Specification	8-9
5.1 Hardware Specification	8
5.2 Software Specification	9
6 Choice of Tools & Technology	10-13
6.1 React	10
6.2 MongoDB	10
6.3 Data Flow Diagram	10
6.4 Context Level Diagram	11-12

7	ER-Diagram	14-16
7.1	Entity-relationship model	14
7.2	Class Diagram	15
8	Database	17-20
8.1	Admin	17
8.2	User	18
8.3	Packages	20
9	Form Design	21-23
9.1	Login	21
9.2	Signup	22
9.3	User	22
9.3.1	package create/add	23
9.3.2	search package	24
9.3.3	homepage	25
10	Coding	26-77
11	Testing	78-81
11.1	Introduction	
11.2	Types of Testing	
	Future Scope and Further Enhancement of the Project	
	Conclusion & References	

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
4.1	Pert Chart	7
6.1	0 Level DFD	12
6.2	1 Level DFD	12
6.3	2 nd Level DFD	13
7.1	Entity-relationship Model	14
8.1	Admin	18
8.2	User	19
8.3	Packages	20

CHAPTER 1

INTRODUCTION

1.1 Background

Travel and Tales serve as comprehensive platforms addressing various challenges encountered by travelers. Additionally, they simplify travel planning with tools for itinerary management and destination exploration, ensuring a smooth and organized experience. Travel websites prioritize safety and security, providing resources and assistance to travelers, along with personalized recommendations tailored to individual preferences. With customer support services and reliable reviews, these platforms aim to enhance the overall travel experience, making it more convenient, enjoyable, and worry-free for users worldwide.

1.2 Project Overview

Travels and Tales is a MERN based web application revolutionizing travel planning and management. It offers seamless user authentication, dynamic itinerary creation, real-time booking integration, interactive destination guides, social collaboration features, personalized recommendations, and responsive design for optimal accessibility. With its cutting-edge technology and user-centric approach, Travels and Tales simplifies travel experiences and fosters meaningful connections among users.

1.3 Objective

The primary objectives of a travel website include:

Efficiency: Optimizing the search, booking, and itinerary management processes for travellers.

User Convenience: Offering travellers a seamless and intuitive interface to explore destinations, book accommodations, and plan activities.

1.4 Key Features

Travels and Tales include a number of features

Booking Interface: Provide travelers with an intuitive interface to book travel services and accommodations according to their preferences, such as destination, dates, and amenities. This interface should streamline the booking process, making it easy for users to specify their travel requirements.

Create package: Admin wants to create a package for user source to destination.

1.5 Scope of the project

The scope of Travel and Tales extends to the following:

User Interface and Experience: Designing an intuitive and user-friendly interface for travellers to easily navigate, search, and book travel services such as flights, accommodations, transportation, and activities.

Booking and Reservation System: Implementing a robust booking and reservation system that allows travellers to search for available options, compare prices, and make secure bookings directly through the website.

CHAPTER 2

PROBLEM IDENTIFICATION & FEASIBILITY STUDY

2.1 Problem Identification

Identifying potential problems in a traveling website is crucial for ensuring a smooth user experience. Common issues include poor user interface and experience, limited search and filtering options, incomplete or inaccurate information, booking and payment issues, ineffective customer support, poor mobile responsiveness, limited personalization, and security concerns. Addressing these issues through regular user testing, feedback gathering, and continuous improvement efforts can enhance the overall user satisfaction and usability of the website.

2.2 Feasibility Study

Before initiating the development of the traveling website, a thorough feasibility study was conducted to evaluate the feasibility and practicality of the proposed platform. This study examined technical, operational, and economic aspects to ensure the viability of the project.

2.2.1 Technical Feasibility

Assess the technical requirements and capabilities needed to develop and maintain the website. Consider factors such as web hosting, scalability, security measures, and integration with third-party services like booking systems and payment gateways.

2.2.2 Operational Feasibility

Assess the practicality of managing and operating the website on a day-to-day basis. Consider factors such as staffing requirements, content management processes, customer support mechanisms, and strategic partnerships with travel suppliers and service providers

2.2.3 Economic Feasibility

The economic study of a traveling website involves assessing its financial viability and sustainability through various analyses. This includes evaluating costs associated with development, maintenance, and operations, as well as identifying potential revenue streams such as booking commissions, advertising, and subscription fees. Market research is conducted to understand demand, competition, and growth opportunities, while financial projections and ROI calculations provide insights into the website's potential performance. Risk assessment and sensitivity analysis help identify and mitigate potential challenges, ensuring the website's resilience in fluctuating market conditions. Overall, the economic study informs stakeholders about the feasibility and profitability of the traveling website, guiding strategic decisions for its successful implementation and operation.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Introduction

Travels and Tales is a groundbreaking web-based platform designed to revolutionize the way travelers plan, book, and experience their journeys. Rooted in the MERN (MongoDB, Express.js, React.js, Node.js) technology stack, Travels and Tales offers an innovative approach to travel management, integrating cutting-edge features and intuitive user interfaces to cater to the diverse needs of modern-day explorers.

3.2 Functional Requirements

3.2.1 User Module

Allow users to create accounts by providing necessary details such as name, email, and password. Implement authentication mechanisms to ensure secure access to user accounts.

3.2.2 Administrator Module

Creating a package for user administration from source to destination involves organizing the necessary components and functionalities to manage users' travel journeys. Here's a detailed breakdown of the process:

3.2.3 Package Module

The package module of a traveling website refers to a structured organization of components, functionalities, and dependencies related to specific features or modules within the website. Here's an outline of how you might structure the package module for a traveling website:.

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 PERFORMANCE

Response Time: Ensure that pages load quickly to provide a seamless user experience, especially during peak traffic times.

Scalability: The website should be able to handle increasing user loads without significant degradation in performance.

Reliability: Minimize downtime and ensure high availability to prevent disruptions in service.

3.3.2 SECURITY

Data Encryption: Implement secure communication protocols (e.g., HTTPS) to encrypt data transmission between the user's browser and the server.

Authentication and Authorization: Ensure secure user authentication mechanisms and role-based access control to protect user accounts and sensitive information.

Data Protection: Adhere to data protection regulations such as GDPR (General Data Protection Regulation) to safeguard user data and privacy.

3.3.3 SCALABILITY

Horizontal Scalability: Design the website architecture to scale horizontally by adding more servers or resources to handle increased traffic.

Vertical Scalability: Ensure that the website can scale vertically by optimizing code, databases, and server configurations to handle increased load on existing resources.

3.3.4 USABILITY

Intuitive Interface: The website should be easy to navigate, with clear menus, intuitive search functionalities, and user-friendly design elements.

Accessibility: Ensure that the website is accessible to users with disabilities, complying with accessibility standards such as WCAG (Web Content Accessibility Guidelines).

Multilingual Support: Provide support for multiple languages to cater to a diverse user base.

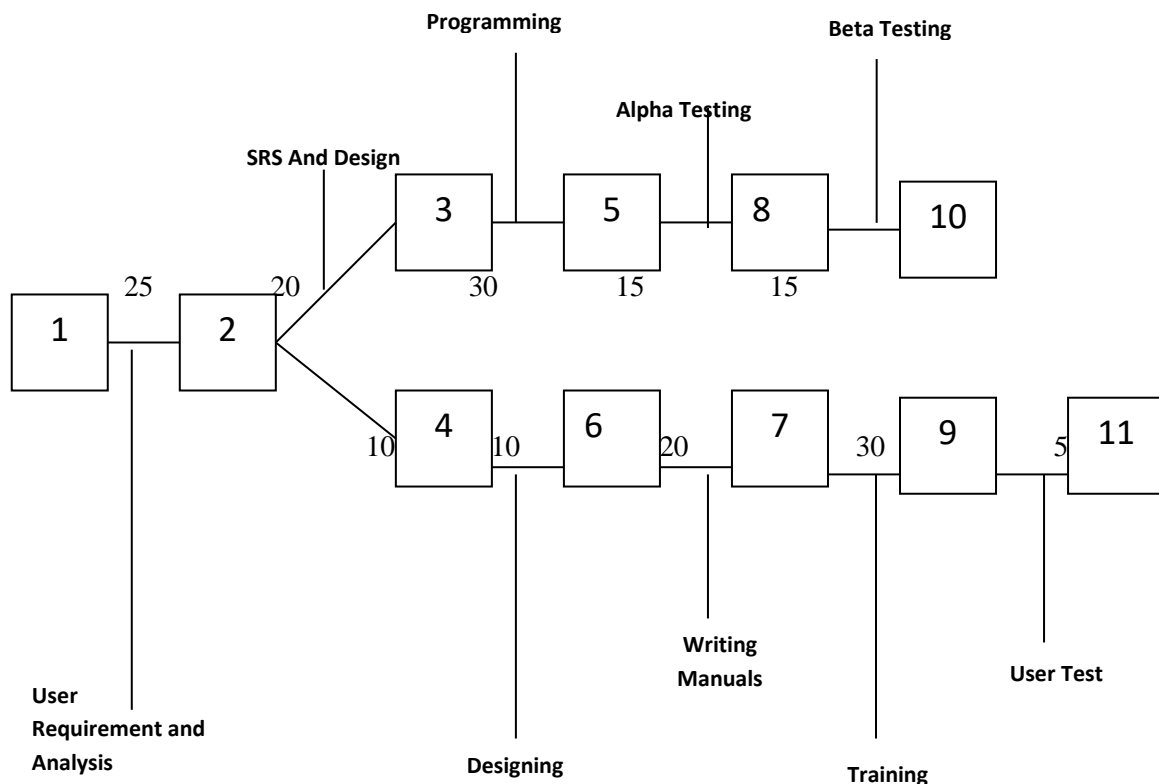
CHAPTER 4

PROJECT PLANNING AND SCHEDULING

4.1 Pert Chart:

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.



CHAPTER 5

HARDWARE & SOFTWARE SPECIFICATION

5.1 Hardware Specification

The Travels and Tales will be developed and deployed on a hardware infrastructure that ensures optimal performance and reliability. The recommended hardware specifications are as follows:

Server:

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

Database Server:

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

Network Interface: Gigabit Ethernet

Client Machines:

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 128 GB SSD or higher

Network Interface: 100 Mbps Ethernet or Wi-Fi

5.2 Software Specification

The traveling website's Travels and tales will be developed using a combination of server-side and client-side technologies within the MERN (MongoDB, Express.js, React.js, Node.js) stack. The software specifications for the traveling website encompass:

Server-Side Technologies:

Operating System: Windows Server 2016 or later

Web Server: Node.js

Database Management System: MongoDB

Server-Side Scripting Language: JavaScript (Node.js)

Client-Side Technologies:

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Client-Side Scripting: JavaScript

Development Tool

Integrated Development Environment (IDE): Visual Studio Code

Version Control:

Git: Version control for collaborative development

Security:

SSL/TLS: Ensure secure data transmission over the network

Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

CHAPTER 6

CHOICE OF TOOLS & TECHNOLOGY

6.1 React

React is a JavaScript library used for building user interfaces. React allows developers to describe the UI's appearance at any point in time, and it automatically updates and renders the right components when the data changes.

React organizes the UI into reusable components, making it easier to manage and maintain complex applications. React uses a virtual DOM to optimize and update the actual DOM efficiently, improving performance by minimizing unnecessary re-rendering.

React uses JSX, a syntax extension that looks similar to XML or HTML, to describe the structure of UI components in a more concise and readable way. React components can manage their internal state, and data can be passed to components through props, allowing for dynamic and interactive UIs

6.2 MongoDB

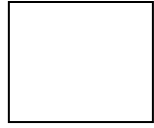
MongoDB is a NoSQL (non-relational) database management system that stores data in a flexible, JSON-like format known as BSON (Binary JSON). It is designed to handle large volumes of unstructured or semi-structured data, making it particularly suitable for applications with evolving and dynamic data requirements.

6.3 Data Flow Diagram

The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

In the DFD, four symbols are used and they are as follows.

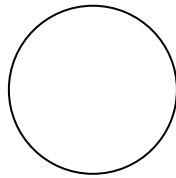
1. A square defines a source (originator) or destination of system data.



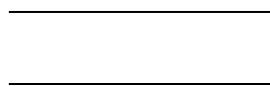
2. An arrow identifies data flow-data in motion. It is a pipeline through which information flows.



3. A circle or a “bubble” (Some people use an oval bubble) represents a process that transfers incoming data flows into outgoing data flows.

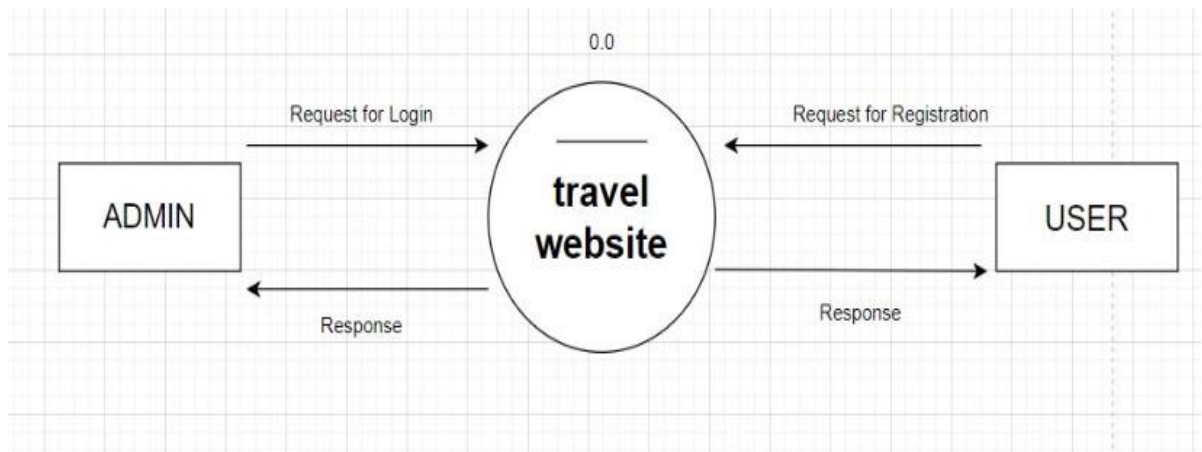


4. An open rectangle is a data store-data at rest, or a temporary repository of data.



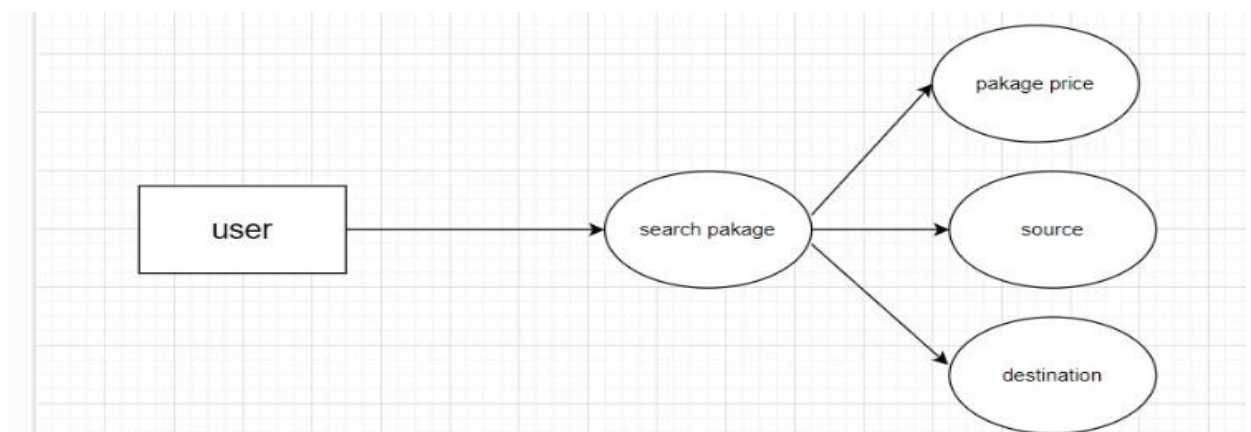
6.4 Context Level Diagram

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Travels and Tales is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays an important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.



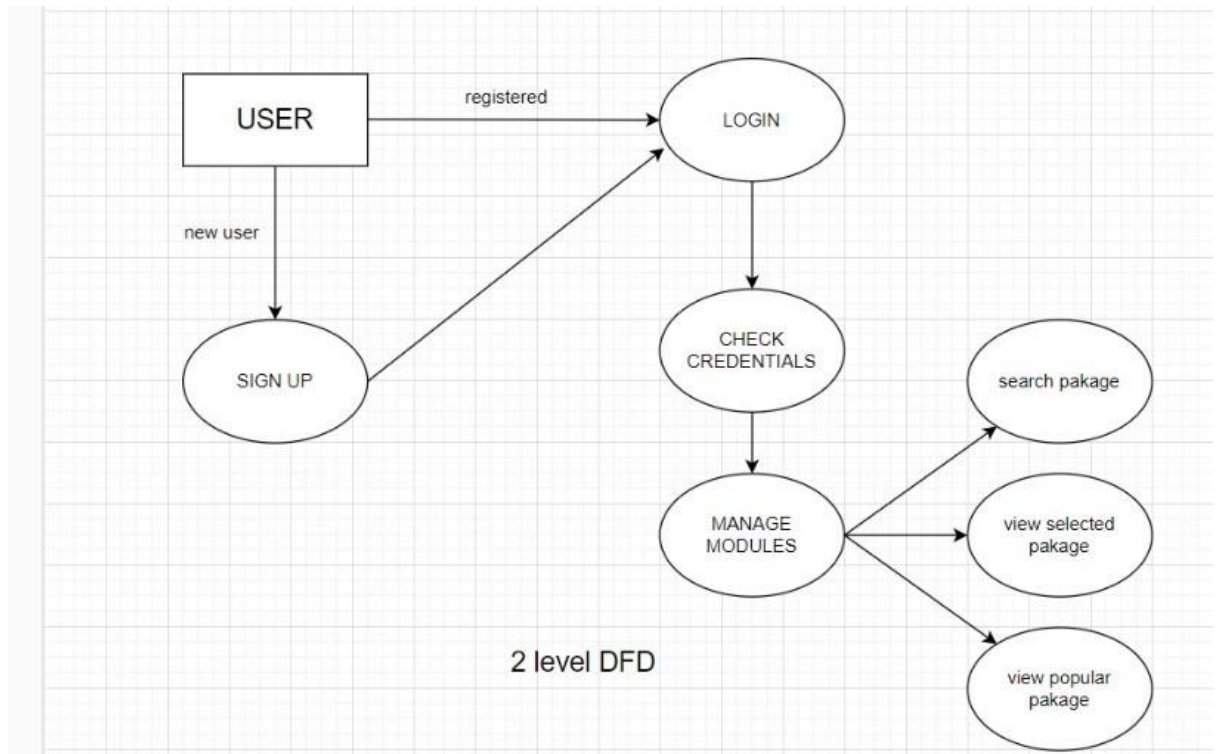
0 Level DFD

Fig 6.1



1st Level DFD

Fig 6.2



2nd Level DFD

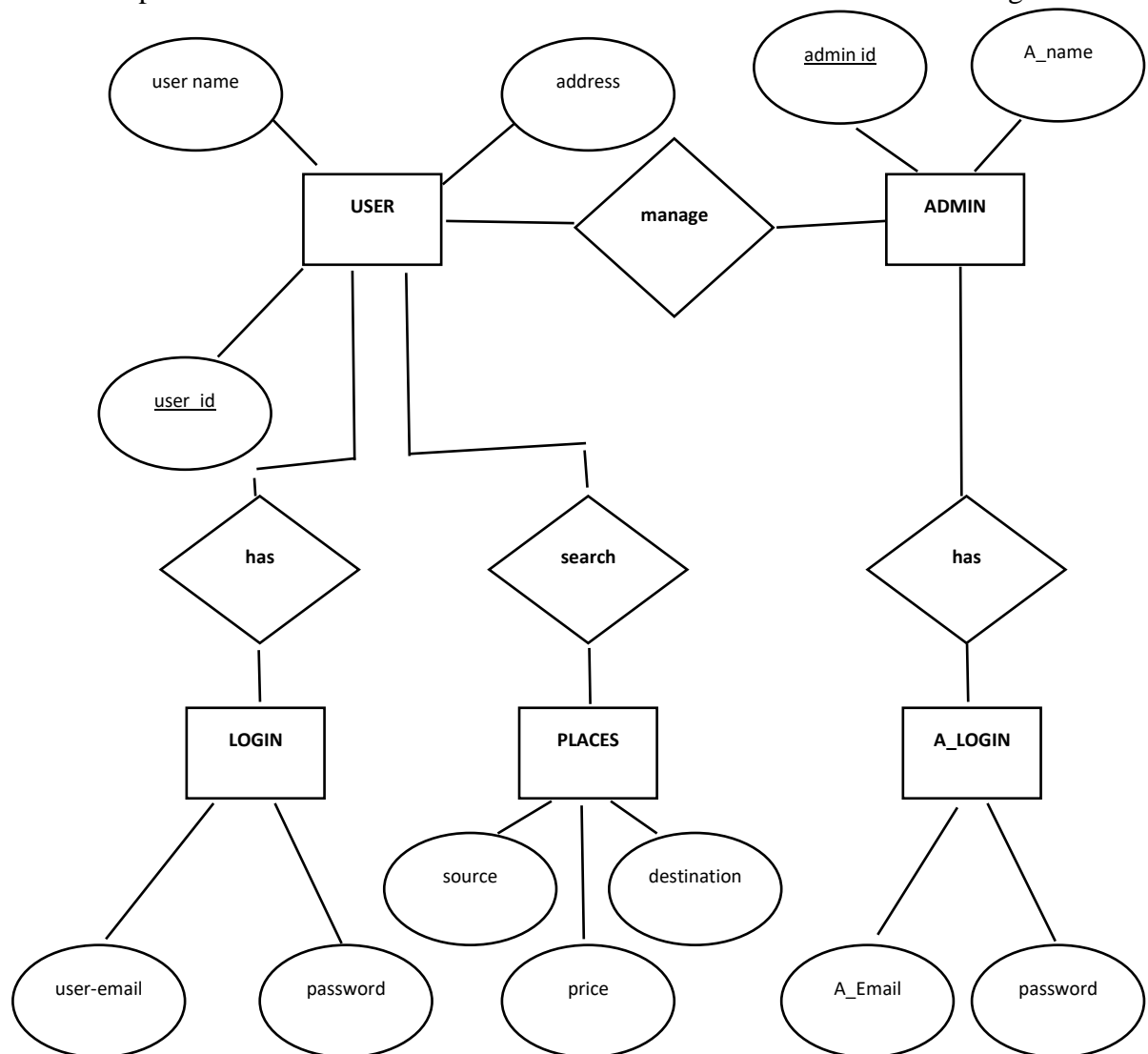
Fig 6.3

CHAPTER 7

ER-DIAGRAM

7.1 Entity-relationship model: -

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams.



7.2 Class Diagram: -

Authentication:

- Classification: Weak Class
- Description: Represents user authentication details, including username and password. This class is responsible for user login functionality.

User:

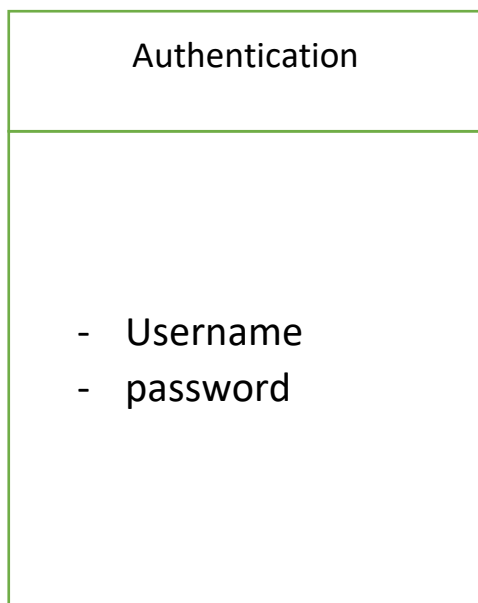
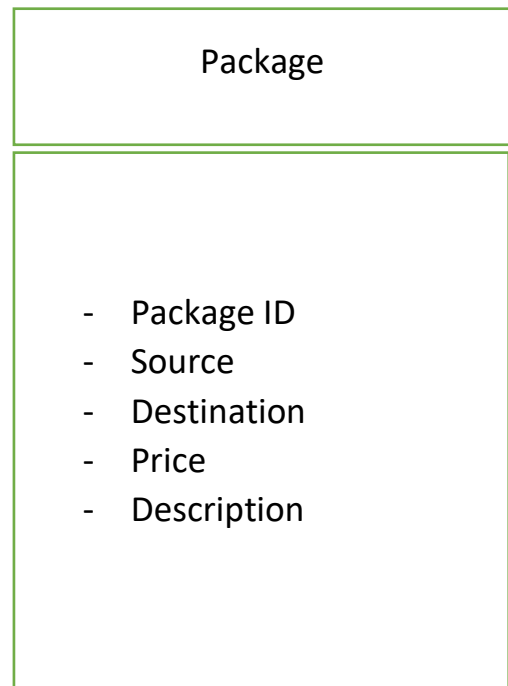
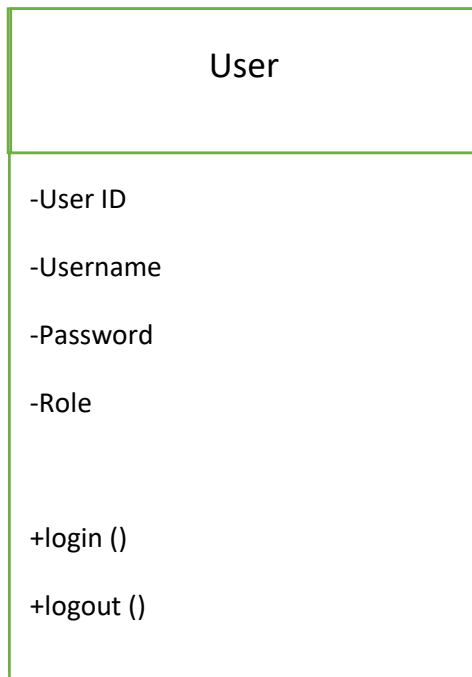
- Classification: Strong Class
- Description: Represents the users of the system, including administrators and employees.

Package:

- Classification: Strong Class
- Description: Represents the package source and destinations including details such as Package ID ,source and destination price and description.

Packages:

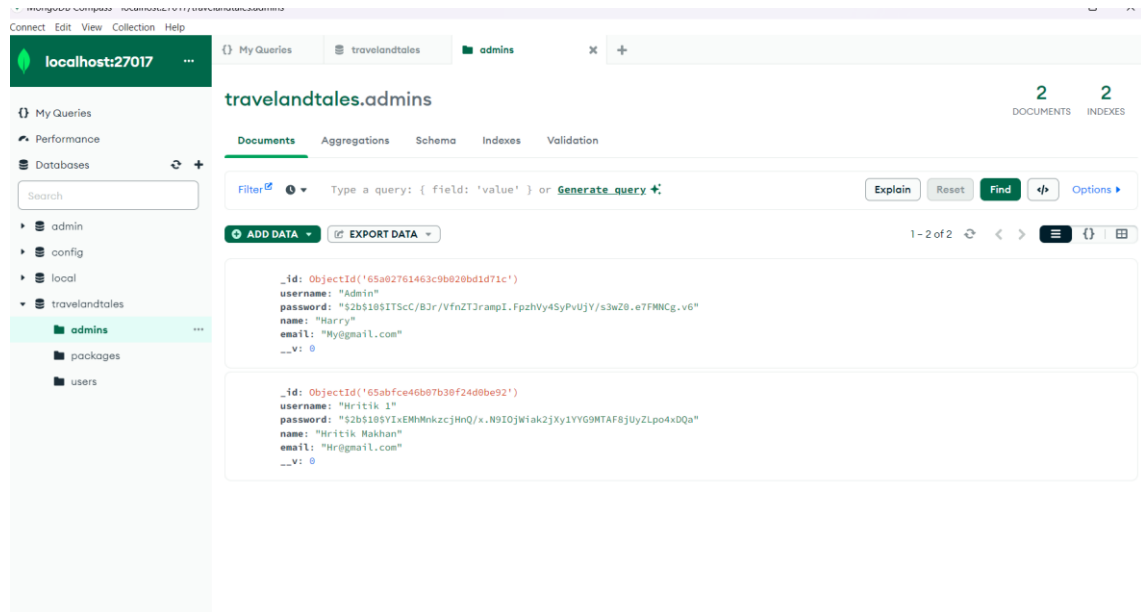
- Classification: Strong Class
- Description: Represents the packages from source to destination in the package cart, including details such as package ID, Description and price.



CHAPTER 8

DATABASE

8.1 Admin



Email:

This attribute stores the email address of an individual. It is a unique identifier and is commonly used for user authentication and communication.

Name:

The "name" attribute typically stores the full name of an individual. It might be divided into first name, middle name, and last name for more detailed records.

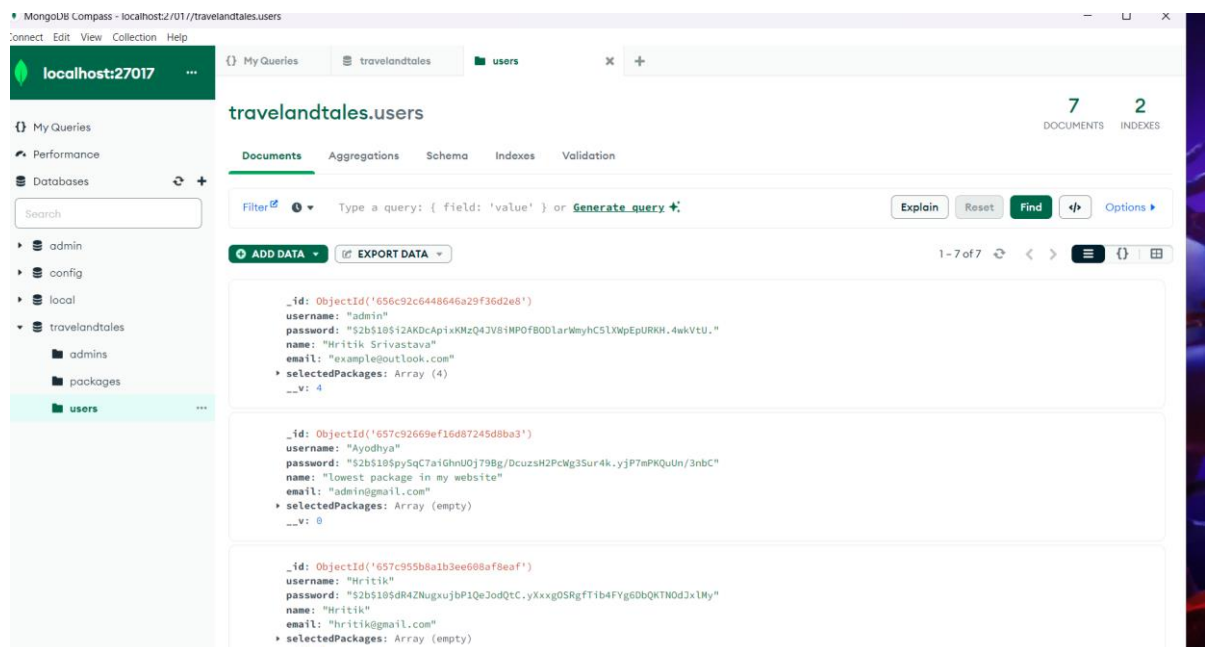
Address:

The "Address" attribute stores the physical address or location details of an individual. It could include components such as street address, city, state, and postal code..

Password:

The "password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring secure access to the system.

8.2 User



Users Credentials

Id:

The "Id" attribute is typically a unique identifier assigned to each individual in the database. It serves as a primary key, ensuring that each record can be uniquely identified and referenced.

Name:

The "Name" attribute stores the full name of an individual. It is a fundamental piece of personal information and is often used for identification and communication purposes.

Email:

The "Email" attribute stores the email address of an individual. It serves as a unique identifier for user accounts and is commonly used for communication and login credentials.

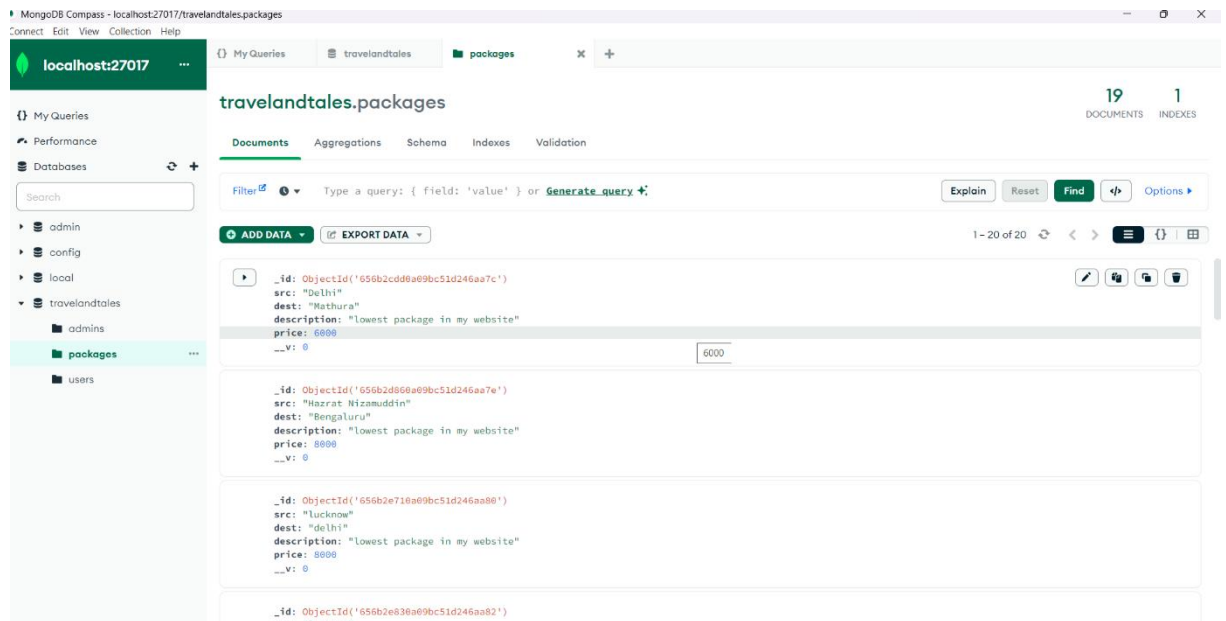
Address:

The "Address" attribute captures the physical address or location details of an individual. It might include components such as street address, city, and state, providing a comprehensive overview of an individual's residence.

Password:

The "Password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring the security of user accounts by protecting access to sensitive information.

8.3 Packages



Packages

Packages_Id:

The “Id” attribute typically serves as a unique identifier for each record in the database table. It is a primary key that ensures each entry can be uniquely identified and referenced..

Description:

The “Description” attribute provides additional details or a brief description of the product. This field allows for a more comprehensive understanding of the product’s characteristics or features.

Src(Source):

The source refers to the origin or starting point of a project, including the initial problem statement, objectives, and the resources utilized for its implementation. It encapsulates the background information, research findings, and any existing frameworks or methodologies that have contributed to the project's inception.

Desc (Destination):

The source refers to the origin or starting point of a project, including the initial problem statement, objectives, and the resources utilized for its implementation. It encapsulates the background information, research findings, and any existing frameworks or methodologies that have contributed to the project's inception.

CHAPTER 9

FORM DESIGN

9.1 Login

Login

Sign in to your account

Username

admin

Password

☐ Remember me [Forgot password?](#)

Submit

[Don't have an account yet? Signup](#)

login page

The authentication module in the Travels and Tales website is a fundamental component tasked with securely verifying user identities and controlling access to the platform. This module validates user credentials, typically comprising a username and password, to authenticate users effectively. Upon successful authentication, the module assigns appropriate roles to users, distinguishing between administrators and regular users. Session management functionalities are employed to maintain user sessions throughout their interactions on the website.

The authentication module in the Travels and Tales website is a fundamental component tasked with securely verifying user identities and controlling access to the platform. This module validates user credentials, typically comprising a username and password, to authenticate users effectively. Upon successful authentication, the module assigns appropriate roles to users, distinguishing between administrators and regular users. Session management functionalities are employed to maintain user sessions throughout their interactions on the website

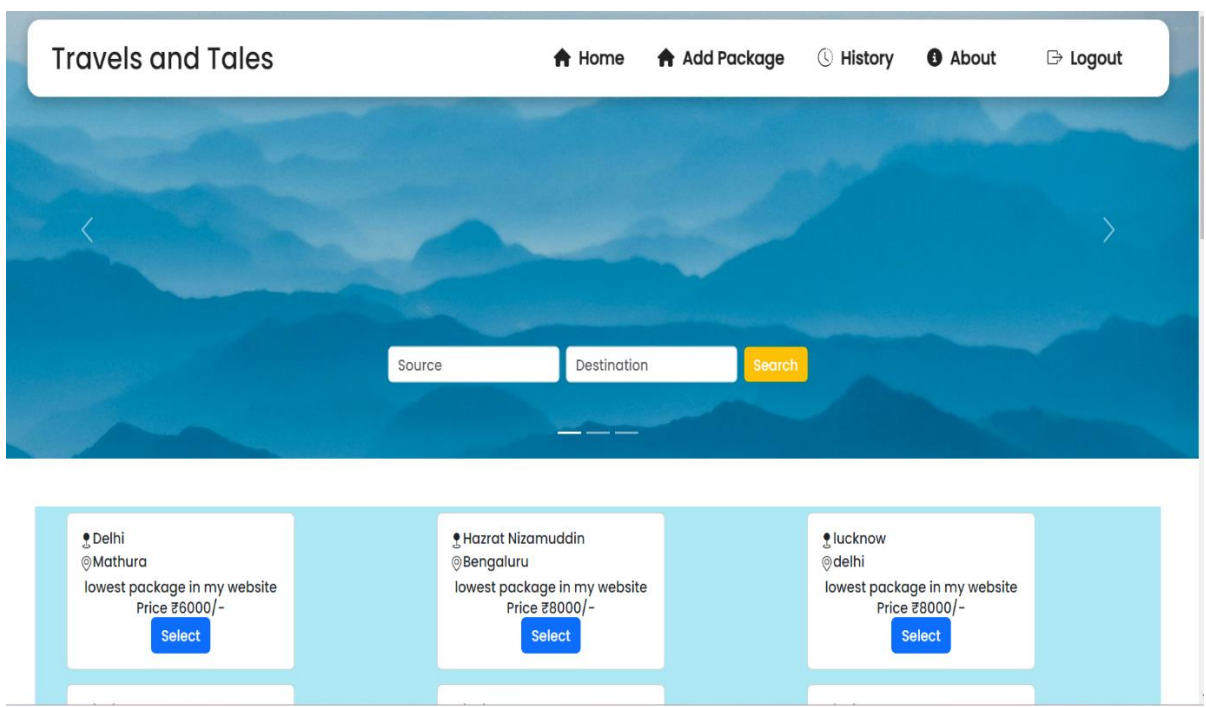
9.2 Signup

The image shows a 'Sign Up to your account' form. It has five input fields: Username (with 'admin' entered), Email (with 'name@company.com' entered), Name (with 'ABC' entered), Address (empty), and Password (masked with dots). Below the fields is a black 'Signup' button. At the bottom, there is a link: 'Already have an account? Sign In'.

Signup Page

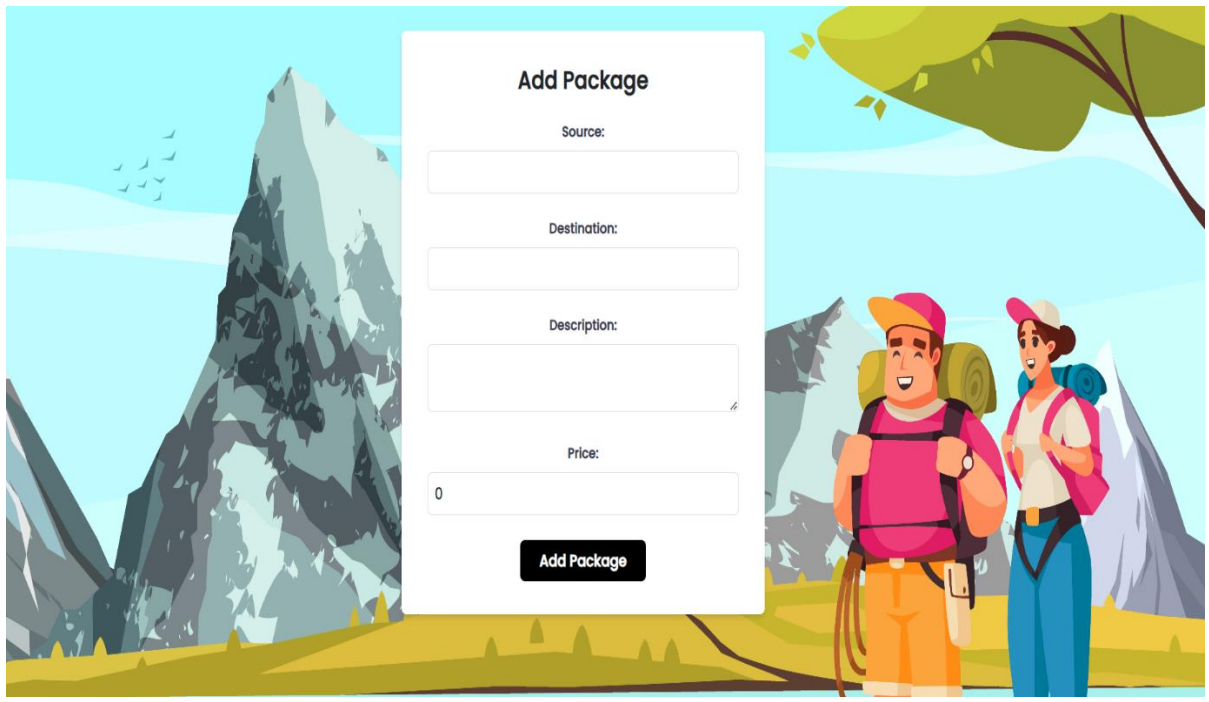
The signup page on Travels and Tales facilitates easy registration for users eager to join our community. By providing basic details like name and email, visitors can swiftly create personalized accounts. Optional fields may capture interests, enriching user profiles. Our secure process includes email verification to ensure authenticity. Join us to share stories, connect with travellers, and embark on new adventures!

9.3 Users



The user module on the Travels and Tales website offers members a dynamic platform to share and explore captivating travel stories and experiences. Through an intuitive interface, users can effortlessly create and publish their narratives, engage with fellow travellers through comments and likes, and personalize their feeds to discover inspiring content. With profile management tools at their disposal, members can easily update their information, tailor preferences, and immerse themselves in a vibrant community of storytellers and adventurers. Join us to embark on a journey of discovery and connection through the art of travel storytelling.

9.3.1 Package Create/Add



Add Package

Source:

Destination:

Description:

Price:

Add Package

Package Create

As an admin on our travel website, creating a package is a breeze. Simply log in, select "Add New Package," and fill in the essential details like the destination, duration, and highlights. Upload captivating images, set the price, and outline the itinerary with key activities. Once done, publish it for travelers to explore and book their next adventure effortlessly.

9.3.2 Search Package

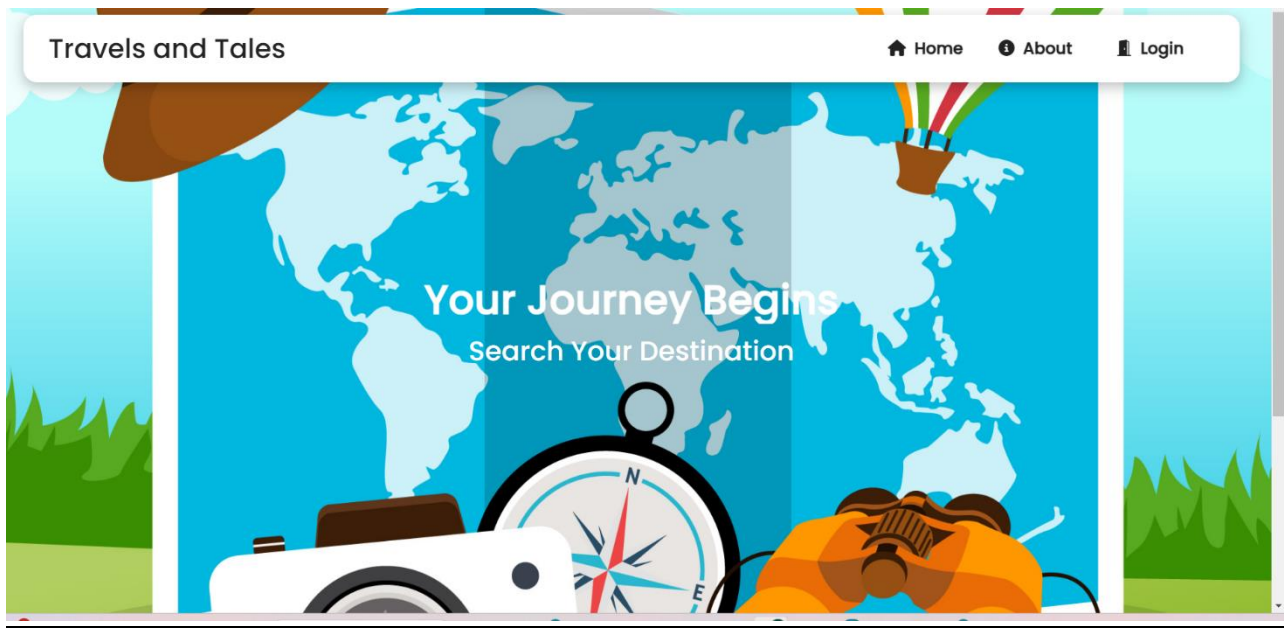
The screenshot shows the 'Travels and Tales' website interface. At the top, there is a navigation bar with links: Home, Add Package, History, About, and Logout. Below the navigation bar is a large banner image of mountains. In the center of the banner, there is a search bar with two input fields labeled 'Source' and 'Destination', and a yellow 'Search' button. Below the banner, there are three package cards. Each card displays a route, the lowest package price, and a 'Select' button.

Route	Lowest Package Price	Action
Delhi to Mathura	₹6000/-	Select
Hazrat Nizamuddin to Bengaluru	₹8000/-	Select
lucknow to delhi	₹8000/-	Select

Search Package

To search for a package on our travel website, simply use the search bar located at the top of the homepage. Enter keywords such as destination names, activity types, or specific dates to narrow down your search. You can also browse through our categories or use filters to refine your results based on criteria like destination, duration, price range, and more. Once you find a package that interests you, click on it to view more details and booking options. Happy searching and safe travels!

9.3.3 Homepage



Homepage

Welcome to our travel website, your gateway to unforgettable adventures and remarkable journeys around the globe. Whether you're dreaming of exotic destinations, seeking thrilling experiences, or craving serene getaways, you'll find inspiration and opportunities aplenty right here.

Explore our vibrant homepage, where a world of possibilities awaits. From breathtaking landscapes to cultural gems, our curated selection of destinations caters to every traveler's wanderlust. Dive into our enticing packages, each meticulously crafted to deliver unforgettable memories and immersive experiences.

Discover a treasure trove of travel resources, from expert tips and guides to insider recommendations, designed to enhance your journey and make planning a breeze. Our user-friendly interface ensures seamless navigation, allowing you to effortlessly find the perfect itinerary for your next escapade.

Join our thriving community of adventurers, storytellers, and fellow wanderers as we embark on a journey of discovery together. Share your travel tales, connect with like-minded explorers, and ignite your passion for exploration.

Start your next adventure with us today. Your journey begins here. Welcome home to the world of travel.

CHAPTER 10

CODING

APP

```
import { Route, Routes } from "react-router-dom";  
import Home from "../routes/Home";  
import About from "../routes/About";  
import Contact from "../routes/Contact";  
import LoginSignup from "../routes/Login";
```

```
import { useEffect } from "react";  
import { useAuth } from "../Context/AuthContext";  
import SignUpForm from "../routes/SignUpForm";  
import History from "../routes/History";  
import Index from "../routes/Index";  
import AddPackages from "../routes/AddPackages";
```

```
export default function App() {  
  const { login } = useAuth();  
  useEffect(() => {  
    const token = document.cookie.split("=")[1];  
    if (token) {  
      login(token);  
    }  
  }, []);  
  return (  
    <div className="App">  
      <Routes>
```

```

    <Route path="/" element={<Home />} />
    <Route path="/about" element={<About />} />
    <Route path="/contact" element={<Contact />} />
    <Route path="/login" element={<LoginSignup />} />
    <Route exact path="/createuser" element={<SignUpForm />} />
    <Route path="/index" element={<Index/>}/>

    <Route exact path="/history" element={<History/>}/>
    <Route exact path="/addPackage" element={<AddPackages/>}/>

  </Routes>
</div>
);
}

```

INDEX

```

import React,{useEffect,useState}from 'react'
import Carousel from '../components/Carousel'
import Navbar from '../components/Navbar'
import Footer from '../components/Footer'
import Carousel1 from '../components/Carousel1'
import CardList from '../components/CardList'

```

```

export default function index() {
  const [src, setSrc] = useState("");
  const [dest, setDest] = useState("");
  const [packages, setPackages] = useState([]);

```

```

useEffect(()=>{
  const token =
document.cookie.replace(/(?:(?:^|.*;)\s*)token\s*=\s*([^;]*).*$/ , '$1');

  fetch('http://localhost:5000/packages', {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': token,
    },
  })
    .then((resp) => resp.json())
    .then((data) => {

      setPackages(data);
    });
},[])

const handleSearch = async (e) => {
  e.preventDefault();

  try {
    const token =
document.cookie.replace(/(?:(?:^|.*;)\s*)token\s*=\s*([^;]*).*$/ , '$1');

    const res = await fetch(`http://localhost:5000/search-
packages?src=${src}&dest=${dest}`, {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': token,
        // Include authentication headers if needed
      },
    });
  }
};

```

```

    const data = await res.json();

    setPackages(data)

    console.log(data); // Log the response from the backend

    // Handle the received data as needed

  } catch (error) {

    console.error(error);

    // Handle error

  }

};

return (
  <div>
    <div><Navbar/></div>

    <div><div id="carouselExampleCaptions" className="carousel slide">
      <div className="carousel-indicators">
        <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" className="active" aria-current="true" aria-label="Slide 1"></button>

        <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1" aria-label="Slide 2"></button>

        <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2" aria-label="Slide 3"></button>
      </div>
      <div className="carousel-inner">
        <div className="carousel-item active">
          
        </div>
        <div className="carousel-item">
          
        </div>
      </div>
    </div>
  </div>
);

```

```

<div className="carousel-item">
  
</div>

<div className='carousel-caption' style={{ "zIndex": "10" }}>
  <form className="d-flex position-absolute start-50 bottom-10 translate-
middle" role="search">
    <input
      className="form-control me-2"
      type="search"
      placeholder="Source"
      aria-label="Source"
      value={src}
      onChange={(e) => setSrc(e.target.value)}
    />
    <input
      className="form-control me-2"
      type="search"
      placeholder="Destination"
      aria-label="Destination"
      value={dest}
      onChange={(e) => setDest(e.target.value)}
    />
    <button
      style={{ justifyContent: "center" }}
      className="btn btn-outline-success text-white bg-warning"
      type="submit"
      onClick={handleSearch}
    >
      Search
    </button>

```

```

        </form>

    </div>

</div>

    <button className="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="prev">

        <span className="carousel-control-prev-icon" aria-hidden="true"></span>

        <span className="visually-hidden">Previous</span>

    </button>

    <button className="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="next">

        <span className="carousel-control-next-icon" aria-hidden="true"></span>

        <span className="visually-hidden">Next</span>

    </button>

</div></div>

    <div className='m-5'
style={{display:"flex",justifyContent:"center",alignItems:"center",paddingTop:"
1px"}}><CardList packages={packages}/></div>

    <Footer/>

</div>

)
}

```

LOGIN

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { useAuth } from '../Context/AuthContext';
import { Link } from 'react-router-dom';

// Define the Login component
const Login = () => {
  // State variables to store username and password
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

  const { authToken, login } = useAuth();
  const navigate = useNavigate();

  // Event handler for form submission
  const handleLogin = async (e) => {
    e.preventDefault();

    // Prepare data to be sent in the request body
    const data = {
      username: username,
      password: password,
    };

    try {
      // Send a POST request to the server

      const response = await axios.post('http://localhost:5000/login', data);
```

```

// Check if response.data is defined before accessing it
if (response && response.data) {
  login(response.data.token);
  document.cookie = `token=${response.data.token}`;
  // Handle the response as needed (e.g., redirect on success)
  console.log('Login successful');
  navigate('/index');
} else {
  // Handle the case where response.data is undefined
  console.error('Login failed. Response data is undefined.');
```

```

}
} catch (error) {
  // Handle errors (e.g., display an error message)
  console.error('Login failed', error.response ? error.response.data :
error.message);
}
};

return (
  <div className='background'>
    <section className='bg-gray-50'>
      <div className='flex flex-col items-center justify-center px-6 py-8 mx-auto
md:h-screen lg:py-0'>
        <a href="#" className='flex items-center mb-6 text-2xl font-semibold text-
gray-900'>
          Login
        </a>
        <div className='w-full bg-white rounded-lg shadow dark:border md:mt-0
sm:max-w-md xl:p-0 dark:border-gray-700'>
          <div className='p-6 space-y-4 md:space-y-6 sm:p-8'>

```



```
<h1 className="text-xl font-bold leading-tight tracking-tight text-gray-900 md:text-2xl ">
```

```
  Sign in to your account
```

```
</h1>
```

```
<form onSubmit={handleLogin} className="space-y-4 md:space-y-6" action="#">
```

```
  <div>
```

```
    <label for="username" className="block mb-2 text-sm font-medium text-gray-900">Username</label>
```

```
    <input type="text" value={username} onChange={(e) => setUsername(e.target.value)} name="username" id="username" className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5" placeholder="" required="" />
```

```
  </div>
```

```
  <div>
```

```
    <label for="password" className="block mb-2 text-sm font-medium text-gray-900">Password</label>
```

```
    <input value={password} onChange={(e) => setPassword(e.target.value)} type="password" name="password" id="password" placeholder="••••••••" className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5" required="" />
```

```
  </div>
```

```
  <div>
```

```
</div>
```

```
<div className="flex items-center justify-between">
```

```
  <div className="flex items-start">
```

```
    <div className="flex items-center h-5">
```

```
      <input id="remember" aria-describedby="remember" type="checkbox" className="w-4 h-4 border border-gray-300 rounded bg-gray-50 focus:ring-3 focus:ring-primary-300" required="" />
```

```

    </div>

    <div className="ml-3 text-sm">
      <label for="remember" className="text-gray-500">Remember
me</label>
    </div>
  </div>

  <a href="#" className="text-sm font-medium text-primary-600
hover:underline">Forgot password?</a>

</div>

  <button type="submit" className="w-full text-white bg-black hover:bg-
primary-700 focus:ring-4 focus:outline-none focus:ring-primary-300 font-medium
rounded-lg text-sm px-5 py-2.5 text-center">Submit</button>

  <p className="text-sm font-light text-gray-500">
    Don't have an account yet? <a href="#" className="font-medium text-
primary-600 hover:underline"><Link className=""
to="/createuser">Signup</Link></a>
  </p>
</form>
</div>
</div>
</div>
</section>
</div>

);

};

// Export the Login component
export default Login;

```

SIGNUP

```
import React, { useState } from 'react';  
import { Link } from 'react-router-dom';  
  
export default function Signup() {  
  const [credentials, setCredentials] = useState({  
    name: '',  
    email: '',  
    password: '',  
    geolocation: '',  
  });  
  
  const handleSubmit = async (e) => {  
    e.preventDefault();  
  
    try {  
      const response = await fetch('http://localhost:5000/register', {  
        method: 'POST',  
        headers: {  
          'Content-Type': 'application/json',  
        },  
        body: JSON.stringify({  
          name: credentials.name,  
          email: credentials.email,  
          password: credentials.password,  
          Location: credentials.geolocation,  
        }),  
      });  
  
      const json = await response.json();
```

```

    console.log(json);

    if (!json.success) {
        alert('Enter Valid Credentials');
    } else {
        // Optionally, you can redirect the user to another page upon successful
registration.

        // For example, you can use the history object from the react-router-dom.
        // history.push('/some-path');
    }
} catch (error) {
    console.error(error);
}
};

const onChange = (e) => {
    setCredentials({ ...credentials, [e.target.name]: e.target.value });
};

return (
    <>
    <div className='container'>
        <form onSubmit={handleSubmit}>
            {/* ... (your existing form fields) */}
            <div className="mb-3">
                <label htmlFor="exampleInputEmail1" className="form-label">
                    Address
                </label>
                <input
                    type="text"
                    className="form-control"

```

```

        name="geolocation"
        value={credentials.geolocation}
        onChange={onChange}
    />
</div>

<button type="submit" className="btn m-3 btn-warning">
    Submit
</button>
<Link to="/login" className='m-3 btn btn-danger'>
    Already a User
</Link>
</form>
</div>
</>
);
}

```

HOME

```
import Hero from "../components/Hero";
import Navbar from "../components/Navbar";
import heroImg from "../assets/home.jpg";
import Footer from "../components/Footer";
function Home() {
  return (
    <>
      <Navbar />
      <Hero
        cName="hero"
        heroImg={heroImg}
        title="Your Journey Begins"
        text="Search Your Destination"
        buttonText="Travel Places"
        btnClass="show"
      />
      <Footer />
    </>
  );
}

export default Home;
```

HISTORY

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

export default function History() {

  const [userPackages1, setUserPackages] = useState([]);

  useEffect(() => {

    const fetchUserPackages = async () => {

      try {

        const token =
document.cookie.replace(/(?:(?:^|.*;)\s*)token\s*=\s*([^;]*).*$/ , '$1');

        const res = await fetch('http://localhost:5000/user-packages', {

          method: 'GET',

          headers: {

            'Content-Type': 'application/json',

            'Authorization': token,

          },

        });

        const data = await res.json();

        setUserPackages(data);

      } catch (error) {

        console.error(error);

        // Handle error

      }

    };

    fetchUserPackages();

  });

}
```

}, []); // Empty dependency array ensures that the effect runs only once when the component mounts

```
return (  
  <div>  
    <h1 className='m-4 ' style={{fontSize:"larger"}}>Selected Package</h1>  
    <div className="row row-cols-1 row-cols-md-3 g-4 m-3">  
  
      {userPackages1.map((userPackage, index) => (  
        <div key={index} className="col">  
          <div className="card" style={{ width:  
"18rem",backgroundColor:"#e3d5ca" }}>  
            <div className="card-body">  
  
              <h5 className=" mb-2 "><i class="bi bi-geo-fill"></i>Source:  
{userPackage.src}</h5>  
  
              <h5 className="card-subtitle mb-2 "><i class="bi bi-geo-  
alt"></i>Destination: {userPackage.dest}</h5>  
  
              {/* <p className="card-text">Description:  
{userPackage.description}</p> */}  
  
              <h6 className="card-subtitle mb-2 text-body-secondary"><i class="bi  
bi-wallet2"></i>Price: ₹{userPackage.price}</h6>  
  
              {/* Add other properties you want to display */}  
  
            </div>  
  
          </div>  
  
        </div>  
  
      )})  
    </div>  
  
  </div>  
  
);  
}
```


ADD PACKAGES

```
import React,{useState} from 'react'
```

```
import axios from 'axios';
```

```
import './styleBack.css';
```

```
export default function AddPackages() {
```

```
  const [formData, setFormData] = useState({
```

```
    src: '',
```

```
    dest: '',
```

```
    description: '',
```

```
    price: 0,
```

```
  });
```

```
  // Define authToken here or get it from your authentication context/state
```

```
  const token =
```

```
  document.cookie.replace(/(?:(?:^|.*;)\s*)token\s*=\s*([^;]*).*$/,'$1');
```

```
  const handleChange = (e) => {
```

```
    const { name, value } = e.target;
```

```
    setFormData((prevData) => ({
```

```
      ...prevData,
```

```
      [name]: value,
```

```
    }));
```

```
  };
```

```
  const handleSubmit = async (e) => {
```

```
    e.preventDefault();
```

```
    try {
```

```
      const headers = {
```

```
        'Content-Type': 'application/json',
```

```

    'Authorization': token,
  };

  await axios.post('http://localhost:5000/create-package', formData, { headers });

  setFormData({
    src: '',
    dest: '',
    description: '',
    price: 0,
  });
} catch (error) {
  console.error('Error submitting package:', error);
  // Handle error as needed
}
};

return (
  <div className='containerBack' >
    <div className="container mx-auto mt-8 p-8 bg-white rounded shadow-md max-w-md">
      {/* Limiting the width of the container to a maximum of medium (md) width */}

      <h1 className="text-2xl font-semibold mb-6">Add Package</h1>

      {/* Form to add a new package */}
      <form onSubmit={handleSubmit}>

        <div className="mb-4">
          <label htmlFor="src" className="block text-gray-700 text-sm font-bold mb-2">

```

Source:

```

</label>

<input
  type="text"
  id="src"
  name="src"
  className="w-full p-2 border rounded"
  value={formData.src}
  onChange={handleChange}
  required
/>
</div>

{/* Destination */}
<div className="mb-4">
  <label htmlFor="dest" className="block text-gray-700 text-sm font-bold
mb-2">
    Destination:
  </label>
  <input
    type="text"
    id="dest"
    name="dest"
    className="w-full p-2 border rounded"
    value={formData.dest}
    onChange={handleChange}
    required
  />
</div>

{/* Description */}
<div className="mb-4">

```

```
<label htmlFor="description" className="block text-gray-700 text-sm font-  
bold mb-2">
```

```
  Description:
```

```
</label>
```

```
<textarea
```

```
  id="description"
```

```
  name="description"
```

```
  className="w-full p-2 border rounded"
```

```
  value={formData.description}
```

```
  onChange={handleChange}
```

```
  required
```

```
></textarea>
```

```
</div>
```

```
{/* Price */}
```

```
<div className="mb-4">
```

```
  <label htmlFor="price" className="block text-gray-700 text-sm font-bold  
mb-2">
```

```
    Price:
```

```
</label>
```

```
<input
```

```
  type="number"
```

```
  id="price"
```

```
  name="price"
```

```
  className="w-full p-2 border rounded"
```

```
  value={formData.price}
```

```
  onChange={handleChange}
```

```
  required
```

```
</div>
```

```

    { /* Submit Button */}

    <button type="submit" className="bg-black hover:bg-gray-800 text-white
font-bold py-2 px-4 rounded">

        Add Package

    </button>

</form>

</div>

</div>

)
}

```

ABOUT

```

import Hero from "../components/Hero";
import Navbar from "../components/Navbar";
import Footer from "../components/Footer";
import aboutImg from "../assets/About.jpg";
import owner from "../assets/about.png";
import "../styleBack.css";

function About() {
    return (
        <>
        <div className="about">
            <Navbar />
            { /* <Hero cName="hero-mid" btnClass="hide" /> */}

        </div className="containerAbout ">

            <div className="flex items-center justify-center h-screen">
                <blockquote className="text-2xl text-yellow-400 drop-shadow-2xl italic">
                    "Embrace the journey, learn from the challenges, and celebrate the victories.
                    Life is a beautiful adventure waiting to be explored."
                </blockquote>
            </div>
        </div>
    )
}

```

```

    </blockquote>
  </div>
</div>
  { /* <Hero cName="hero" heroImg={owner} title="owner" btnClass="hide" />
  */}
  <Footer />
</div>
</>
);
}

```

```
export default About;
```

CARDLIST

```

import React, { useEffect, useState } from 'react';
import { useAuth } from '../Context/AuthContext';
import Card from './Card';

export default function CardList({packages}) {
  const { authToken } = useAuth();
  //const [packages, setPackages] = useState([]);

  const handleSelect = async (packageId) => {
    try {
      const res = await fetch('http://localhost:5000/choose-package', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': authToken,
        },
        body: JSON.stringify({ packageId }),

```

```

});

const data = await res.json();
console.log(data); // Log the response from the backend
// Handle success or error as needed
} catch (error) {
  console.error(error);
  // Handle error
}
};

useEffect(() => {

}, []);

return (
  <div className="" style={{ backgroundColor: "#ade8f4" }}>
    <div style={{ display: 'flex', flexWrap: 'wrap'
,padding:'7px',paddingLeft:'12px'}}>
      {packages.map((data, index) => (
        <div key={index} style={{ flex: '0 0 33.33%', marginBottom: '15px' }}>
          <div className="card ml-7" style={{ width: "18rem", maxWidth:
"400px" }}>
            <div className="card-body">
              <div style={{ display: "flex" }}>
                <i class="bi bi-geo-fill"></i>
                <h2 className="">{data.src}</h2>
              </div>
              <div style={{ display: "flex" }}>
                <i class="bi bi-geo-alt"></i><h2 className="">{data.dest}</h2>
              </div>
            </div>
          </div>
        )
      )}
    </div>
  </div>
);

```

```

        <p className="card-text">{data.description}</p>
        <h3>Price ₹{data.price}</h3>
        <button
            className="btn btn-primary"
            onClick={() => handleSelect(data._id)} // Pass packageId to handleSelect
function
        >
            Select
        </button>
    </div>
</div>
</div>
    )}
</div>
</div>
);

```

CAROUSEL

```
import React from 'react'
```

```

export default function Carousel() {
    return (
        <div className=' '>
            <div id="carouselExampleFade " className="carousel slide carousel-fade"
data-bs-ride="carousel" style={{ "objectFit": "contain !important" }}>
                <div className="carousel-inner " id='carousel'>
                    <div className='carousel-caption' style={{ "zIndex": "10" }}>
                        <form className="d-flex " role="search">
                            <input className="form-control me-2 start-20 bottom-20 end-20 position-
absolute " type="search" placeholder="Source" aria-label="Search"/>
                            <input className="form-control me-2 start-20 bottom-100 end-50 position-
absolute " type="search" placeholder="Destination" aria-label="Search"/>

```



```

    <button className="btn btn-outline-success text-white bg-warning bottom-0
end-0 position-absolute" type="submit">Search</button>

  </form>

</div>

<div className="carousel-item active">

</div>

<div className="carousel-item">

</div>

<div className="carousel-item">

</div>

</div>

<button className="carousel-control-prev" type="button" data-bs-
target="#carouselExampleFade" data-bs-slide="prev">

  <span className="carousel-control-prev-icon" aria-hidden="true"></span>

  <span className="visually-hidden">Previous</span>

</button>

<button className="carousel-control-next" type="button" data-bs-
target="#carouselExampleFade" data-bs-slide="next">

  <span className="carousel-control-next-icon" aria-hidden="true"></span>

  <span className="visually-hidden">Next</span>

</button>

</div>

</div>

)
}

```

Navbar

```
import React, { Component } from "react";
import "./NavbarStyles.css";
import { MenuItems } from "./MenuItems";
import { Link } from "react-router-dom";
import { Navigate } from "react-router-dom";

class Navbar extends Component {
  state = { clicked: false, hasCookie: false };

  componentDidMount() {
    // Check if the cookie exists

    const token =
document.cookie.replace(/(?:(?:^|.*;\s*)token\s*=\s*([^;]*).*$/g, '$1');
    this.setState({ hasCookie: !!token });
  }

  handleClick = () => {

    this.setState({ clicked: !this.state.clicked });
  };

  handleLogout = () => {
    // Perform logout logic here (e.g., clear cookies, update state)

    // Example: document.cookie = "token=; expires=Thu, 01 Jan 1970 00:00:00
UTC; path=/;";

    // Update state accordingly
  }
}
```

```
document.cookie = 'token=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/';
```

```
// Update state accordingly
```

```
this.setState({ hasCookie: false });
```

```
window.location.href = '/';
```

```
};
```

```
render() {
```

```
  const homeButton = this.state.hasCookie ? (
```

```
    <>
```

```
    <li>
```

```
      <Link className="nav-links" to="/index">
```

```
        <i className="fa-solid fa-house"></i>Home
```

```
      </Link>
```

```
    </li>
```

```
    <li>
```

```
      <Link className="nav-links" to="/addPackage">
```

```
        <i className="fa-solid fa-house"></i>Add Package
```

```
      </Link>
```

```
    </li>
```

```
    <li>
```

```
      <Link className="nav-links" to="/history">
```

```
        <i className="bi bi-clock-history"></i>History
```

```
      </Link>
```

```
    </li>
```

```
  </>
```

```
) : (
```

```
  <li>
```

```
    <Link className="nav-links" to="/">
```

```
      <i className="fa-solid fa-house"></i>Home
```

```

    </Link>
  </li>
);

return (
  <nav className="NavbarItems" onClick={this.handleClick}>
    <h1 className="navbar-logo">Travels and Tales</h1>
    <div className="menu-icons">
      <i className={this.state.clicked ? "fas fa-times" : "fas fa-bars"}></i>
    </div>

    <ul className={this.state.clicked ? "nav-menu active" : "nav-menu"}>
      {homeButton}
      {MenuItems.map((item, index) => (
        <li key={index}>
          <Link className={item.cName} to={item.url}>
            <i className={item.icon}></i>
            {item.title}
          </Link>
        </li>
      ))}

      { /* Conditionally render the Login and Logout buttons */ }

      <li>
        {!this.state.hasCookie ? (
          <Link className="nav-links" to="/login">
            <i className="bi bi-door-open-fill"></i>Login
          </Link>
        ) : (
          <Link to="/" className="btn ">

```

```

        <button className="nav-links" onClick={this.handleLogout}>
          <i className="bi bi-box-arrow-right"></i>Logout
        </button>
      </Link>
    )}
  </li>
</ul>
</nav>
);
}
}

```

```
export default Navbar;
```

FOOTER

```

import "./FooterStyles.css";

const Footer = () => {
  return (
    <div className="footer">
      <div className="top">
        <div>
          <h1>Travels and Tales</h1>
          <p>Search your favourite destinations</p>
        </div>
        <div>

```

```

<a href="/" className="facebook">
  <i className="fa-brands fa-facebook-square"></i>
</a>
<a href="/">
  <i className="fa-brands fa-instagram-square"></i>
</a>
<a href="/">
  <i className="fa-brands fa-github-square"></i>
</a>
</div>
</div>
<div className="bottom">
  <div>
    <h3>project</h3>
    <a href="/">Status</a>
    <a href="/">License</a>
    <a href="/">ChangeLog</a>
  </div>
  <div>
    <h3>comunity</h3>
    <a href="/">Github</a>
    <a href="/">Issues</a>
    <a href="/">Twitter</a>
  </div>
  <div>
    <h3>help</h3>
    <a href="/">Support</a>
    <a href="/">Troubleshoot</a>
    <a href="/">Contact</a>
  </div>
</div>

```

```
<div>
  <h3>others</h3>
  <a href="/">Policy</a>
  <a href="/">Service</a>
  <a href="/">License</a>
</div>
</div>
</div>
);
};

export default Footer;
```

VALIDATE

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

```
const validateSignUpForm = payload => {
```

```
  const errors = {};
```

```
  let message = "";
```

```
  let isValid = true;
```

```
  if (
```

```
    !payload ||
```

```
    typeof payload.username !== "string" ||
```

```
    payload.username.trim().length === 0
```

```
  ) {
```

```
    isValid = false;
```

```
    errors.username = "Please provide a user name.";
```

```
  }
```

```
  if (
```

```
    !payload ||
```

```
    typeof payload.email !== "string" ||
```

```
    !validator.isEmail(payload.email)
```

```
  ) {
```

```
    isValid = false;
```

```
    errors.email = "Please provide a correct email address.";
```

```
  }
```

```
  if (
```

```
    !payload ||
```



```

    typeof payload.password !== "string" ||
    payload.password.trim().length < 8
  ) {
    isValid = false;
    errors.password = "Password must have at least 8 characters.";
  }

  if (!payload || payload.pwconfirm !== payload.password) {
    isValid = false;
    errors.pwconfirm = "Password confirmation doesn't match.";
  }

  if (!isValid) {
    message = "Check the form for errors.";
  }

  return {
    success: isValid,
    message,
    errors
  };
};

const validateLoginForm = payload => {
  const errors = {};
  let message = "";
  let isValid = true;

  if (
    !payload ||

```

```

    typeof payload.username !== "string" ||
    payload.username.trim().length === 0
  ) {
    isValidForm = false;
    errors.username = "Please provide your user name.";
  }

  if (
    !payload ||
    typeof payload.password !== "string" ||
    payload.password.trim().length === 0
  ) {
    isValidForm = false;
    errors.password = "Please provide your password.";
  }

  if (!isValidForm) {
    message = "Check the form for errors.";
  }

  return {
    success: isValidForm,
    message,
    errors
  };
};

module.exports = {
  validateLoginForm: validateLoginForm,
  validateSignUpForm: validateSignUpForm
}

```

};

LOGIN .CSS

```
.container{  
    display: flex;  
    flex-direction:column;  
    margin: auto;  
    margin-top: 200px;  
    width: 600px;  
    background: #d85e5e;  
    padding-bottom: 30px;
```

```
}
```

.all

```
{  
    max-width: 100%;  
    max-height: 100%;  
    background-color: grey;  
}
```

.header {

```
    display :flex;  
    flex-direction: column;  
    align-items: center;  
    gap:9px;  
    width:100% ;  
    margin-top: 30px;
```

```
}
```

.text{

```
    color:#3c009d;
```

```

    font-size: 48px;
    font-weight:700;

}
.underline{
    width: 61px;
    height: 6px;
    background: #3c009d;
    border-radius:9px;
}
.inputs{
    margin-top:55px;
    display:flex;
    flex-direction: column;
    gap:25px;
}
.input{
    display: flex;
    align-items: center;
    margin:auto;
    width: 480px;
    height:80px;
    background: #eaeaea;
    border-radius: 6px;

}
.input img{
    margin: 0px 30px;
}
.input input{

```

```

    height: 60px;
    width: 400px;
    background: transparent;
    border: none;
    outline:none;
    color: #797979;
    font-size: 20px;

}

.forgot-Password{
    padding-left: 70px;
    margin-top: 27px;
    color: #797979;
    font-size: 18px;
}

.forgot-Password span{
    color:#4c00b4;
    cursor: pointer;
}

.submit-container{
    display: flex;
    gap:30px;
    margin: 60px auto;
}

submit{
    display:flex;
    justify-content: center;
    align-items: center;
    width: 220px;
    height:59px;

```

```
color:#fff;
background: #4c00b4;
border-radius:50px;
font-size: 20px;
font-weight: 700;
cursor:pointer;

}

.gray{
background: #eaeaea;
color:#676767;

}
```

PACKAGE.JSON

```
{  
  "name": "travels-and-theses",  
  "version": "1.0.0",  
  "description": "",  
  "keywords": [],  
  "main": "src/index.js",  
  "dependencies": {  
    "axios": "^1.6.2",  
    "jsonwebtoken": "^9.0.2",  
    "loader-utils": "3.2.1",  
    "react": "18.2.0",  
    "react-dom": "18.2.0",  
    "react-router-dom": "6.17.0",  
    "react-scripts": "^5.0.1"  
  },  
  "devDependencies": {  
    "@babel/runtime": "7.13.8",  
    "tailwindcss": "^3.3.5",  
    "typescript": "4.1.3"  
  },  
  "scripts": {  
    "start": "react-scripts start",  
    "build": "react-scripts build",  
    "test": "react-scripts test --env=jsdom",  
    "eject": "react-scripts eject"  
  },  
  "browserslist": [  
    ">0.2%",
```

```

    "not dead",
    "not ie <= 11",
    "not op_mini all"
  ]
}

```

INDEX.HTML

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="utf-8" />
```

```
<meta
```

```
  name="viewport"
```

```
  content="width=device-width, initial-scale=1, shrink-to-fit=no"
```

```
/>
```

```
<meta name="theme-color" content="#000000" />
```

```
<!--
```

```
  manifest.json provides metadata used when your web app is added to the
```

```
  homescreen on Android. See
```

```
https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/
```

```
-->
```

```
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
```

```
<link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" />
```

```
<link
```

```
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
```

```
  rel="stylesheet" integrity="sha384-
```

```
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
```

```
<script
```

```
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
```

```
  integrity="sha384-
```


C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46
cDfL" crossorigin="anonymous"></script>

<!--

Notice the use of %PUBLIC_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEVDwwwyk2MPK8
M2HN" crossorigin="anonymous">

<link

rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.2/css/all.min.css"

integrity="sha512-
z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCF
kKxLhei6S9CQXFEbbKuqLg0DA=="

crossorigin="anonymous"

referrerpolicy="no-referrer"

/>

<title>React App</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.2/font/bootstrap-icons.min.css">

</head>

<body>

<noscript>

You need to enable JavaScript to run this app.

`</noscript>`

`<div id="root"></div>`

`<!--`

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the `<body>` tag.

To begin the development, run ``npm start`` or ``yarn start``.

To create a production bundle, use ``npm run build`` or ``yarn build``.

`-->`

`<script`

`src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js`

`" integrity="sha384-`

`C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46`

`cDfL" crossorigin="anonymous"></script>`

`</body>`

`</html>`

BACKEND

SERVER

```
// server.js

const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const app = express();
const PORT = process.env.PORT || 5000;

app.use(express.json());
app.use(cors({
  origin: 'http://localhost:3000',
  methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
  credentials: true,
}));

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/travelandtales', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Define User schema
const userSchema = new mongoose.Schema({
```

```

username: {
  type: String,
  required: true,
},
password: {
  type: String,
  required: true,
},
name: {
  type: String,
  required: true,
},
email: {
  type: String,
  required: true,
  unique: true,
  validate: {
    validator: function (value) {
      // Using a simple regex to check for a valid email format
      return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(value);
    },
    message: 'Invalid email format',
  },
},
Location: {
  type: String,
},

```

```

    selectedPackages: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Package',
    }],
  });

const User = mongoose.model('User', userSchema);

// Register a new user
app.post('/register', async (req, res) => {
  try {
    const { name, username, password, email, Location } = req.body;

    // Check if the username already exists
    const existingUser = await User.findOne({ username });
    if (existingUser) {
      return res.status(400).json({ message: 'Username already exists' });
    }

    // Hash the password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Create a new user
    const newUser = new User({
      username,
      password: hashedPassword,
      email,
      name,
      Location,
    });
  }

```

```

    await newUser.save();

    res.status(201).json({ message: 'User registered successfully' });
  } catch (error) {
    console.error(error);

    // Check if the error is a validation error
    if (error.name === 'ValidationError') {
      return res.status(400).json({ message: error.message });
    }
    res.status(500).json({ message: 'Internal Server Error' });
  }
});

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

  try {
    const { username, password } = req.body;

    // Find the user in the database
    const user = await User.findOne({ username });
    if (!user) {
      return res.status(401).json({ message: 'Invalid username or password' });
    }

    // Compare the provided password with the hashed password in the database
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {

```

```

        return res.status(401).json({ message: 'Invalid username or password' });
    }

    // Generate a JWT token

    const token = jwt.sign({ userId: user._id, username: user.username }, 'your-secret-key');

    res.status(201).json({ token });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Internal Server Error' });
  }
});

const isAuthorized = (req, res, next) => {
  const token = req.header('Authorization');
  if (!token) {
    return res.status(401).json({ message: 'Unauthorized' });
  }
  jwt.verify(token, 'your-secret-key', (err, user) => {
    if (err) {
      return res.status(401).json({ message: 'Unauthorized' });
    }
    req.user = user;
    next();
  });
}

const isAdmin = (req, res, next) => {
  // Verify the JWT token
  const token = req.header('Authorization');

```

```

    if (!token) {
        return res.status(401).json({ message: 'Unauthorized' });
    }
    jwt.verify(token, 'your-secret-key', (err, user) => {
        if (err) {
            return res.status(401).json({ message: 'Unauthorized' });
        }
        // Check if the user is an admin
        user.username === 'admin' ? next() : res.status(401).json({ message:
'Unauthorized' });
    });
}

app.get('/protected', isAdmin, (req, res) => {
    // Verify the JWT token
    const token = req.header('Authorization');
    if (!token) {
        return res.status(401).json({ message: 'Unauthorized' });
    }
    jwt.verify(token, 'your-secret-key', (err, user) => {
        if (err) {
            return res.status(401).json({ message: 'Unauthorized' });
        }
        res.json({ message: 'You have access to this protected route', user });
    });
});

app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});

```



```

// Add the package schema
const packageSchema = new mongoose.Schema({
  src: {
    type: String,
    required: true,
  },
  dest: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  }
});

const Package = mongoose.model('Package', packageSchema);

// Endpoint to get a list of available packages
app.get('/packages', async (req, res) => {
  try {
    const packages = await Package.find();
    res.json(packages);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Internal Server Error' });
  }
});

```

```

    }
  });

// Endpoint to choose a package
app.post('/choose-package', isAuthenticated, async (req, res) => {
  try {
    const { packageId } = req.body;
    const userId = req.user.userId;

    // Find the user by ID
    const user = await User.findById(userId);
    if (!user) {
      return res.status(404).json({ message: 'User not found' });
    }

    // Find the package by ID
    const selectedPackage = await Package.findById(packageId);
    if (!selectedPackage) {
      return res.status(404).json({ message: 'Package not found' });
    }

    // Add the selected package to the user's model
    user.selectedPackages.push(selectedPackage);
    await user.save();

    res.json({ message: 'Package chosen successfully', selectedPackage });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Internal Server Error' });
  }
});

// Endpoint to view user's selected packages
app.get('/user-packages', isAuthenticated, async (req, res) => {

```

```

try {
  const userId = req.user.userId;
  // Find the user by ID with populated selectedPackages
  const user = await User.findById(userId).populate('selectedPackages');
  if (!user) {
    return res.status(404).json({ message: 'User not found' });
  }
  res.json(user.selectedPackages);
} catch (error) {
  console.error(error);
  res.status(500).json({ message: 'Internal Server Error' });
}
});

app.post('/create-package', isAdmin, async (req, res) => {
  try {
    const { src, dest, description, price } = req.body;
    // Create a new package
    const newPackage = new Package({
      src,
      dest,
      description,
      price,
    });
    await newPackage.save();
    res.status(201).json({ message: 'Package created successfully' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Internal Server Error' });
  }
}

```

```
});
```

```
app.get('/search-packages', isAuthenticated, async (req, res) => {  
  try {  
    let { src, dest } = req.query;  
    console.log(src, dest);  
    let packages;  
    if (src && dest) {  
      packages = await Package.find({ src, dest });  
    } else if (src) {  
      packages = await Package.find({ src });  
      console.log(packages);  
    } else if (dest) {  
      packages = await Package.find({ dest });  
    } else {  
      return res.status(400).json({ message: 'Invalid search query' });  
    }  
    res.json(packages);  
  } catch (error) {  
    console.error(error);  
    res.status(500).json({ message: 'Internal Server Error' });  
  }  
});
```

CHAPTER 11

TESTING

10.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

10.2 Types of Testing

10.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

10.2.2 Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

10.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT

- The future scope of traveling websites lies in leveraging advanced technologies such as AI, VR, and AR to offer personalized and immersive travel experiences. These websites will focus on sustainability, eco-friendly options, and authentic cultural exchanges. They will also provide AI-driven assistance, AR-enhanced navigation, and digital storytelling platforms to inspire and guide travelers in their journeys.
- Another enhancement would be allowed for Customer also.

CONCLUSION & REFERNCES

The successful design and development of our project on Web Based Travels and Tales using React mark a significant milestone in our journey. Through this endeavor, we have gained invaluable experience in website development, honing our skills and understanding of the intricacies of creating engaging online platforms. As we move forward, we anticipate further growth and learning opportunities, leveraging our newfound expertise to continue innovating and delivering exceptional experiences for our users. Our commitment to excellence and passion for storytelling will drive us to push boundaries, inspiring wanderlust and fostering connections through the captivating world of travel and tales.

References

Coding phase: -

1. React

Referenced Sites:

[www.w3school. com](http://www.w3school.com)

[www. https://react.dev/learn](https://react.dev/learn)