Data Structures Lab

[Project Report]

Title: TOURISTA -Tourist Travel Management System

Submitted To:

Ma'am Aisha Sattar

Submitted By:

Team Lead: Muhammad Ali Hassan (242852)
Malaika Salam (241963)
Wania Adnan (242847)
Ayesha Saleh (242893)

DEPARTMENT OF CREATIVE TECHNOLOGIES
AIR UNIVERSITY ISLAMABAD.

Contents

1. Introduction

Title: TOURISTA: A Unified Tourist Travel Management System for North Pakistan

Content: The tourism industry is one of the fastest-growing sectors globally, yet the process of planning a trip remains fragmented. **TOURISTA** is a comprehensive software assistant designed to streamline the travel experience. It serves as a centralized hub for tourists, allowing them to browse destinations and select travel packages all within a single interface. By integrating various travel necessities into one system, TOURISTA eliminates the need for users to navigate multiple websites to plan a single trip.

2. Problem Statement

Currently, when a tourist wants to plan a vacation, they face a difficult and disjointed process. They must visit one website to book flights, another to find hotels, and yet another to look for local tour packages or guides. This fragmentation leads to:

- **Time Consumption:** Excessive time spent comparing prices and schedules across different platforms.
- **Information Overload:** Managing multiple confirmations and itineraries creates confusion.
- **Lack of Centralization:** There is no single "source of truth" for the traveler's entire journey.

3. Objective

The primary objectives of the TOURISTA project are:

- To develop a unified platform where users can access all travel-related services (transport, lodging, and packages).
- To provide a user-friendly interface that simplifies the booking process for non-technical users.
- To create a system that manages data efficiently, ensuring accurate booking records and itinerary generation.
- To reduce the stress of travel planning by offering pre-curated vacation packages.

4. Problem Solution

The **TOURISTA** system solves the fragmentation problem by acting as a centralized aggregator. Instead of visiting separate service providers, the user interacts with our system, which houses a database of hotels, destinations, and packages.

- **Centralized Data:** The system combines hotel availability and tour packages with one view.
- **Streamlined Logic:** Users can add multiple elements (e.g., a hotel stays + a sightseeing package) to a single "trip" or cart.
- **Efficiency:** The system automates the calculation of costs and itinerary scheduling, providing an instant summary to the user.

5. Methodology

5.1. Implementation of Data Structures

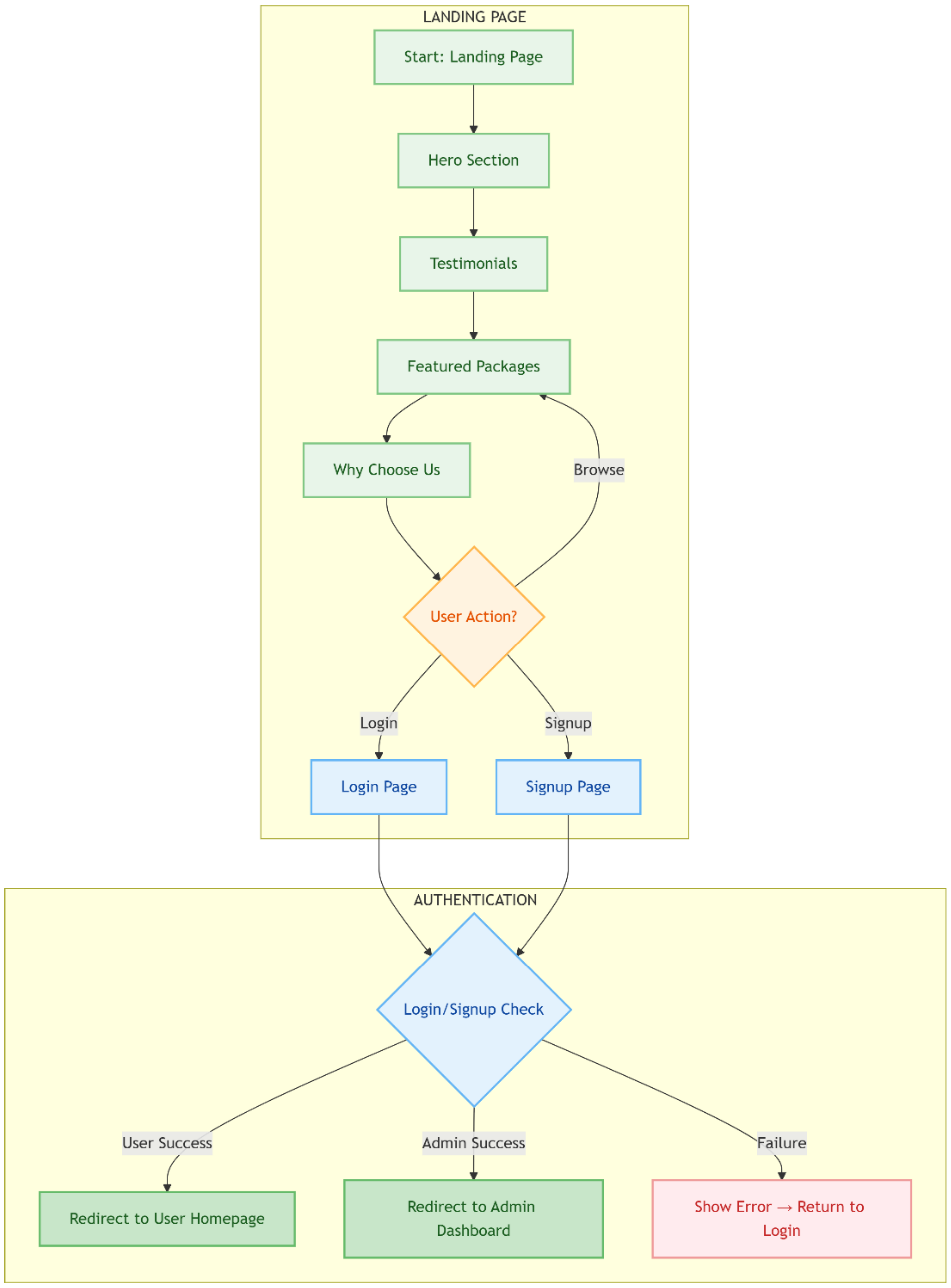
The technical architecture of TOURISTA utilizes specific data structures to solve complex travel management problems, ensuring O(1) or O(log n) efficiency for most core operations.

- Doubly Linked Lists
  - **Where it's used:** packages.h, cities.h, users.h.
  - **Why it's used:**
    - **Packages & Users:** The nodes contain *next and *prev pointers. This allows the admin to traverse the list of packages or users in both directions (forward and backward).
    - **Efficiency:** It enables efficient deletion. When a node needs to be removed (e.g., deleting a package), the system can reconnect the previous and next nodes immediately (O(1)) without having to traverse the entire list again to find the previous node·
- LIFO Stack (Last-In-First-Out)
  - **Where it's used:** announcements.h.
  - **Why it's used:**
    - **Broadcasts:** The announcement_manager adds new announcements to the head of the list.
    - **Priority:** This ensures that the most recent update (e.g., a "Weather Alert" or "New Discount") is the first thing popped/retrieved and displayed at the top of the user dashboard.
- Queue (FIFO - First-In-First-Out)
  - **Where it's used:** bookings.h (specifically the get_bookings_queue function).
  - **Why it's used:**
    - **Booking Processing:** The code retrieves bookings in a linear queue format (get_bookings_queue).
    - **Fairness:** This ensures that bookings are processed or viewed in the order they were made—the first user to book a trip is the first one to be reviewed or confirmed by the admin (First-In-First-Out).
- Nested (Hierarchical) Linked Lists
  - **Where it's used:** cities.h.
  - **Why it's used:**
    - **City Details:** Each "City Node" acts as a parent that holds pointers (spots_head and dining_head) to separate child linked lists.
    - **Memory Management:** This allows every city to have a different number of tourist spots and restaurants without reserving a fixed amount of memory (like a 2D array would), keeping the system lightweight.
- Binary Search Tree (BST)
    - **Where it's used:** Destination Search Logic (mentioned in documentation methodology).
  - **Why it's used:**

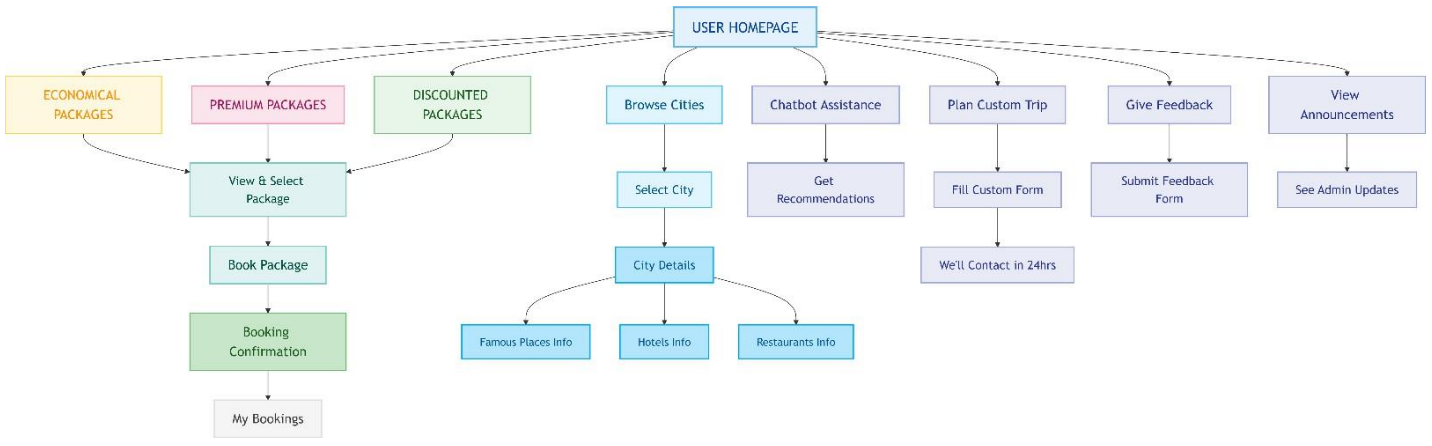
- **Searching:** City names are stored alphabetically in a tree structure.
- **Speed:** This reduces the search time from O(n) (checking every city one by one) to O(log n) (cutting the search area in half with every step), which is critical for the "Search for a city..." bar on the homepage.

6. Flowchart

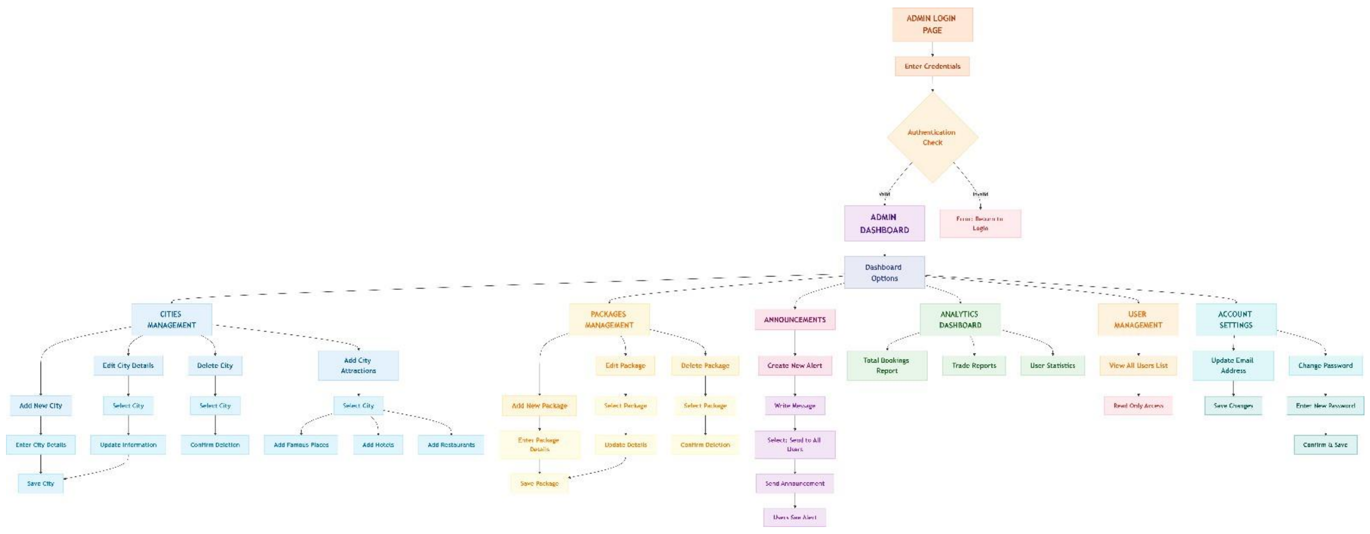
Authentication flowchart:



USER FLOWCHART:

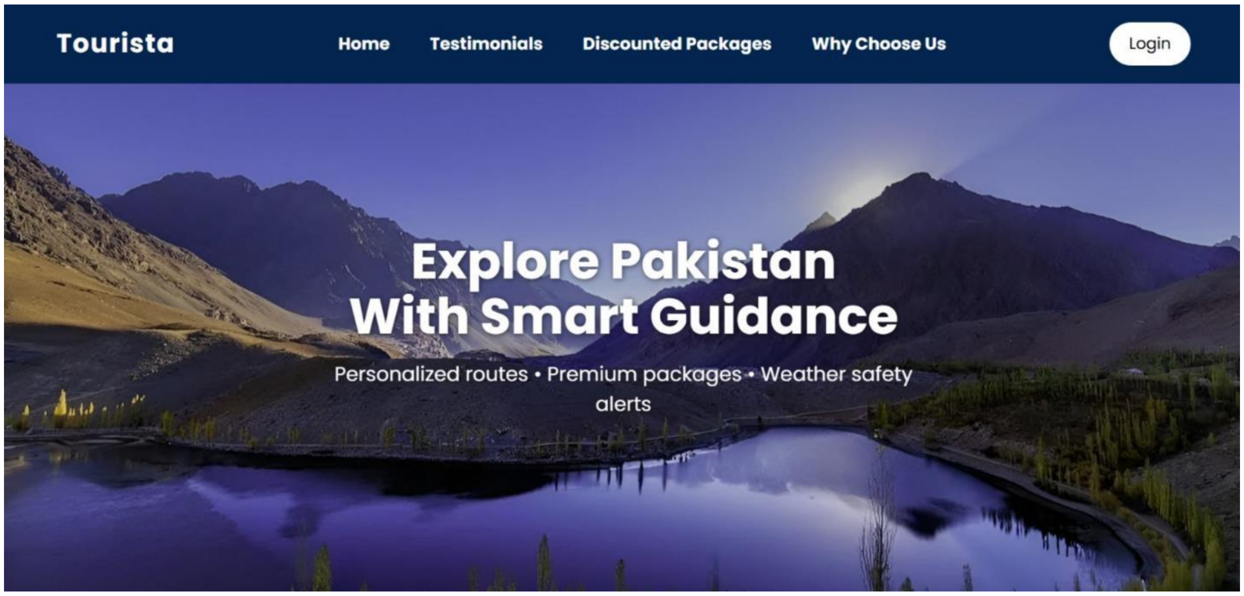


ADMIN FLOWCHART:



7. Visual Outputs

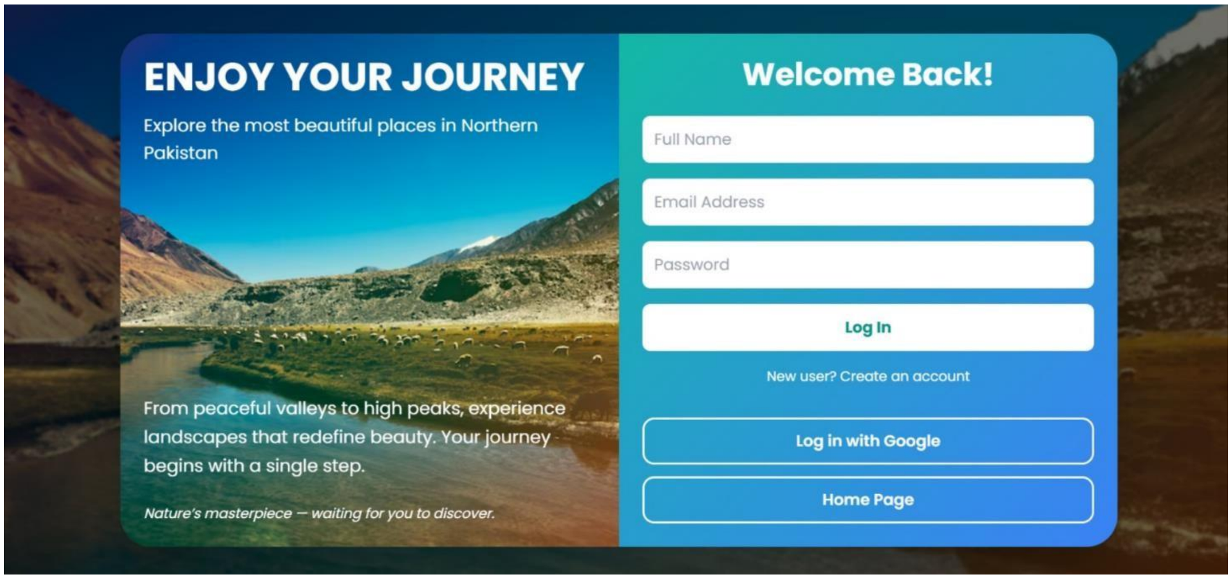
Landing Page:



Explore Pakistan With Smart Guidance

Personalized routes • Premium packages • Weather safety alerts

Login Page:



ENJOY YOUR JOURNEY

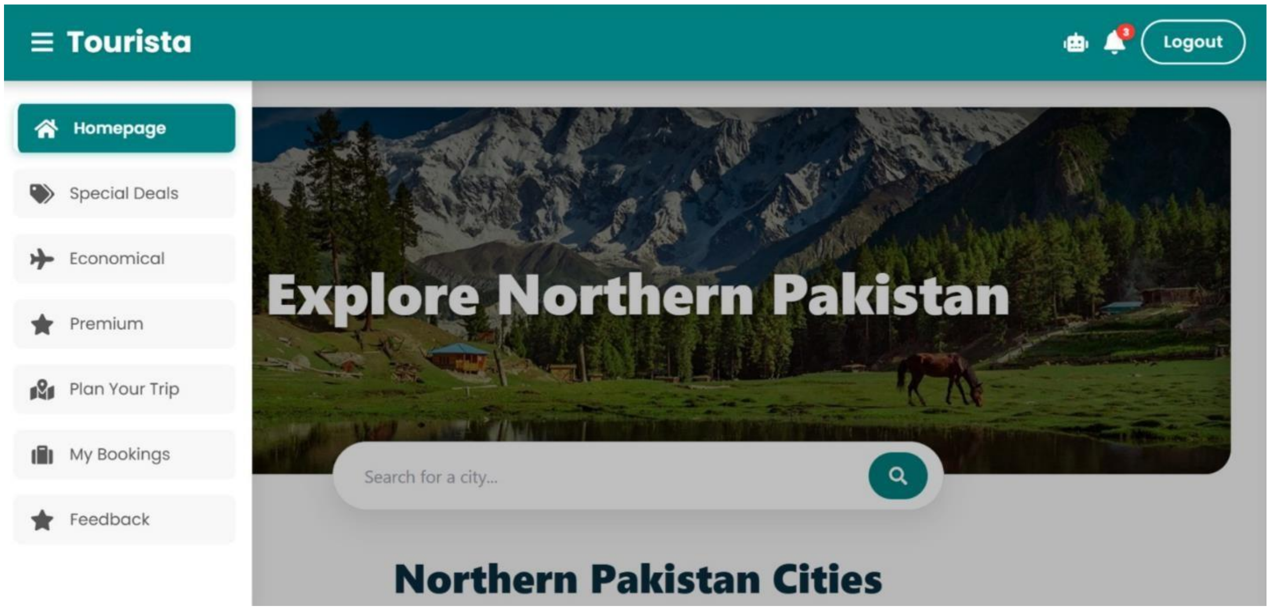
Explore the most beautiful places in Northern Pakistan

From peaceful valleys to high peaks, experience landscapes that redefine beauty. Your journey begins with a single step.

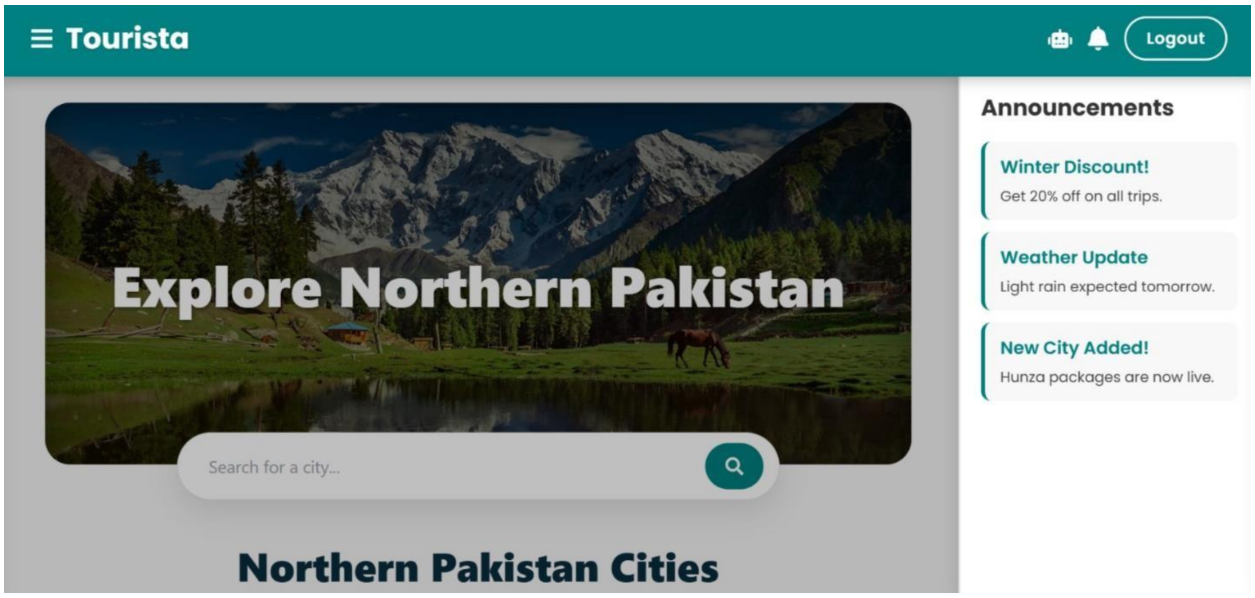
Nature's masterpiece — waiting for you to discover.

Welcome Back!

User Interfaces:



Announcements:



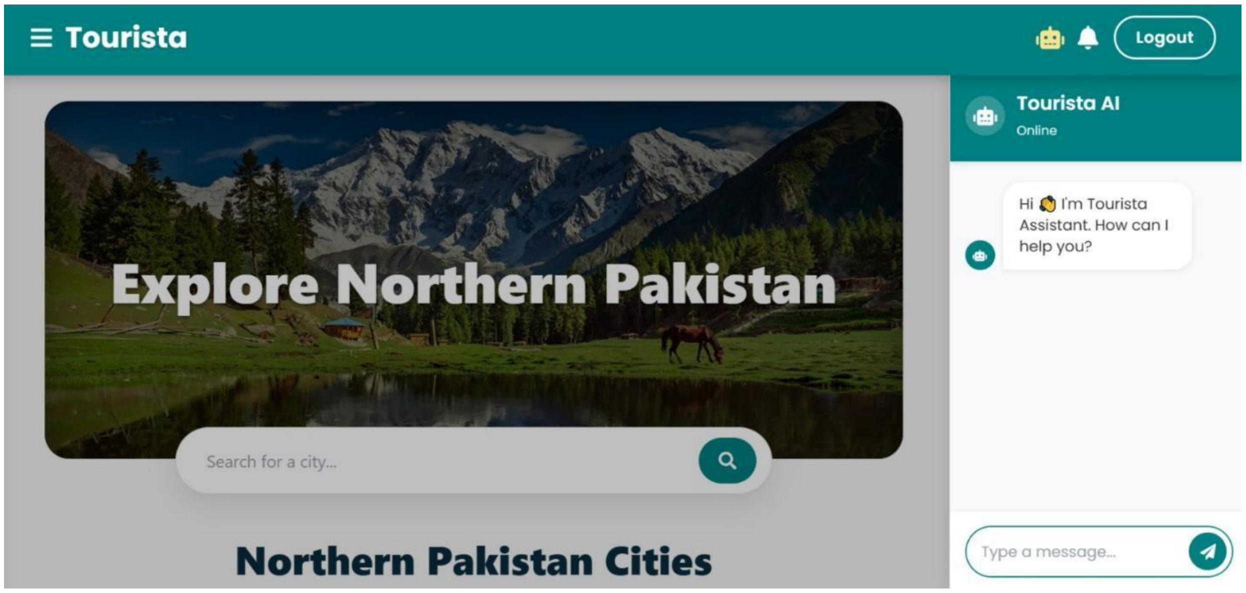
Announcements

Winter Discount!
Get 20% off on all trips.

Weather Update
Light rain expected tomorrow.

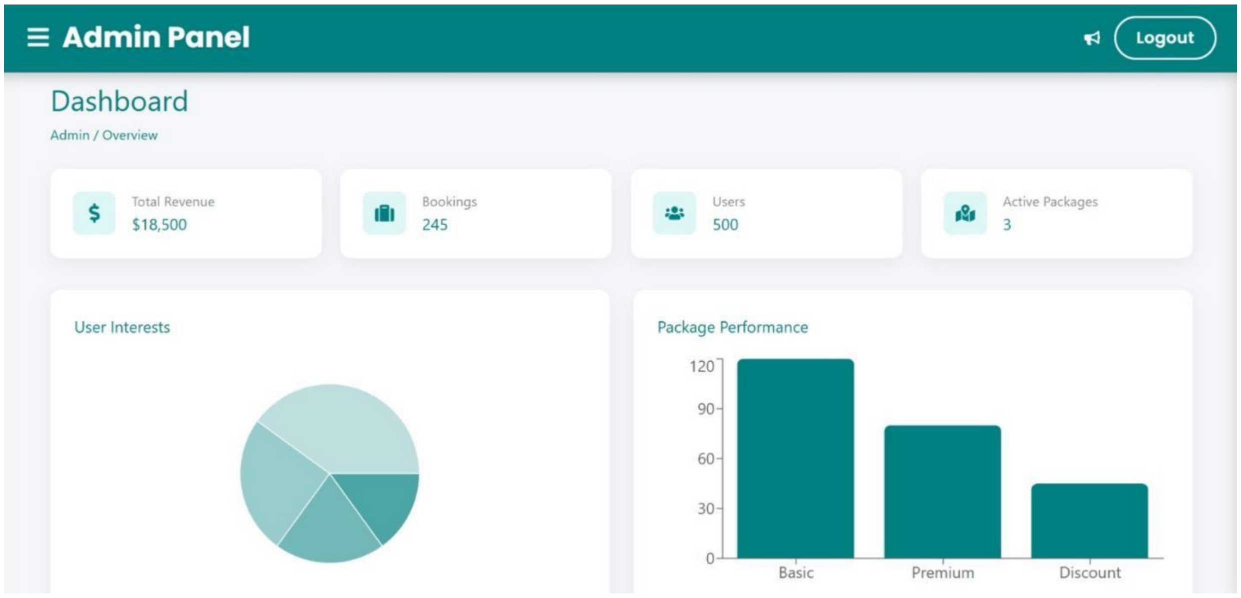
New City Added!
Hunza packages are now live.

Chatbot:



Hi 👋 I'm Tourista Assistant. How can I help you?

Admin Interfaces:

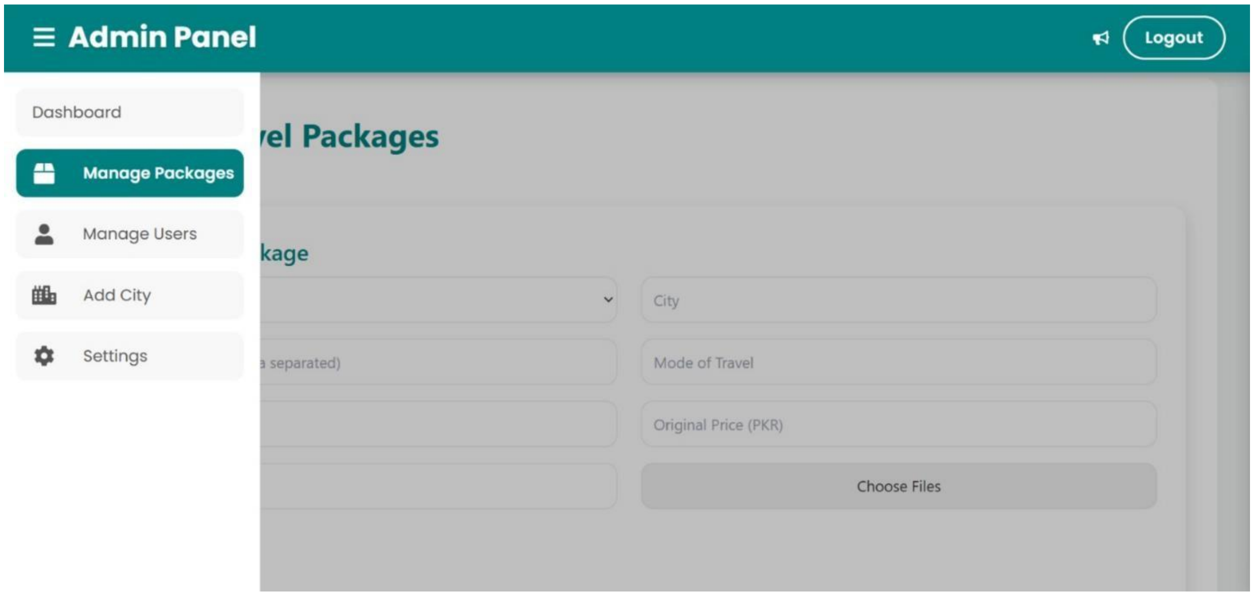


Total Revenue $18,500

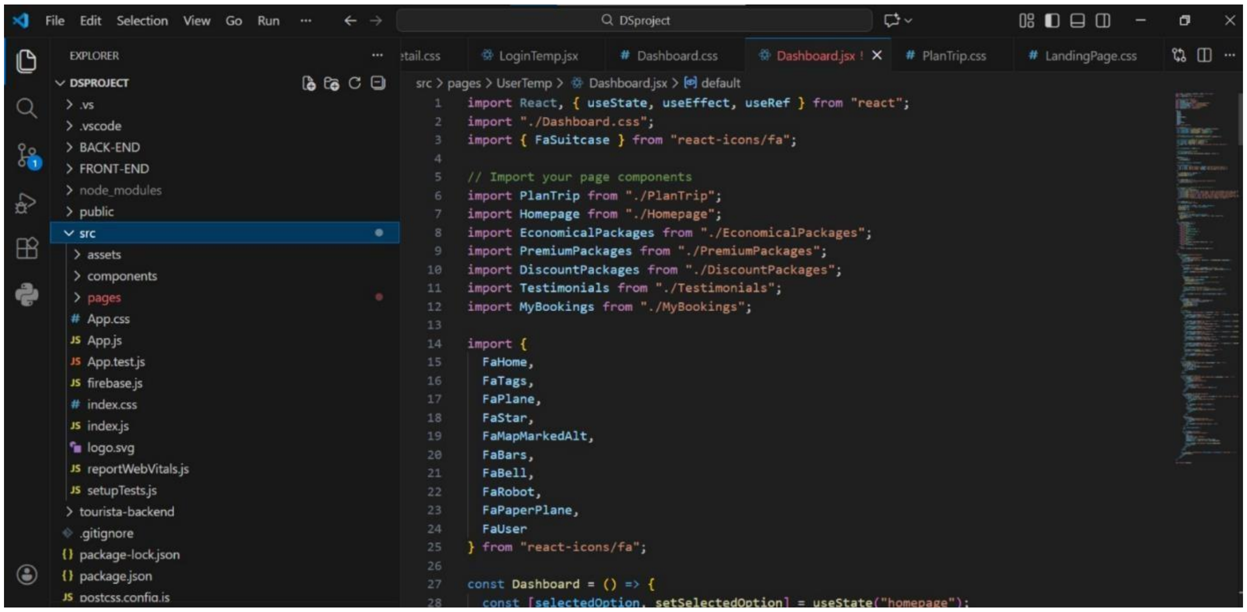
Bookings 245

Users 500

Active Packages 3

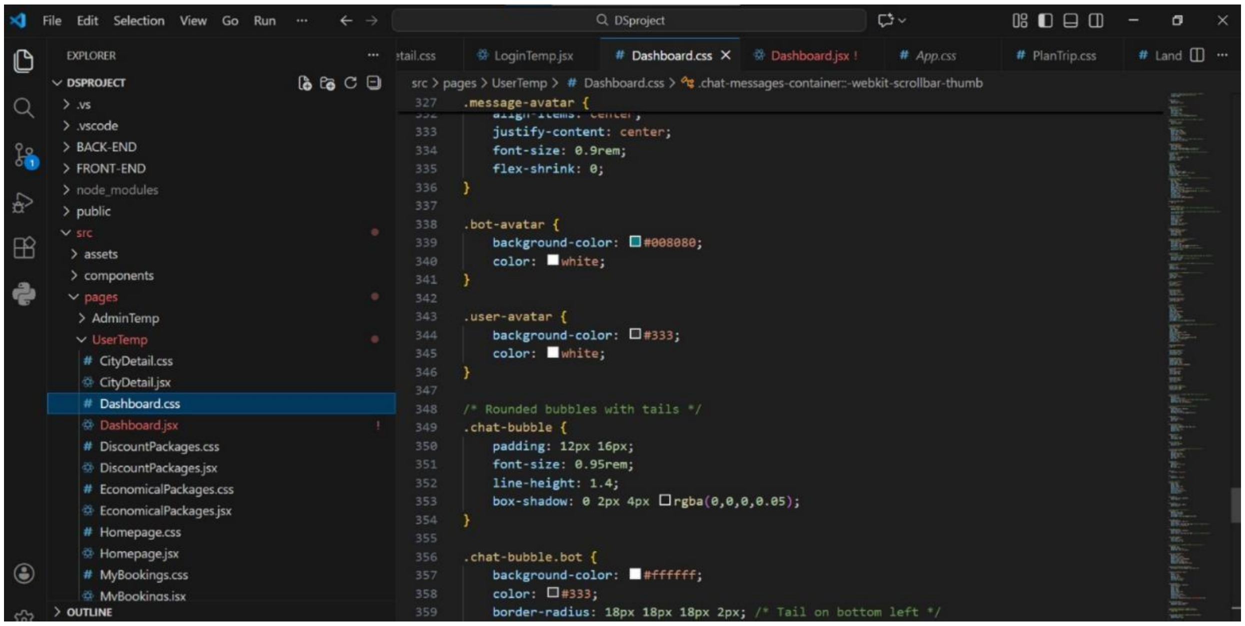


Jsx Code:



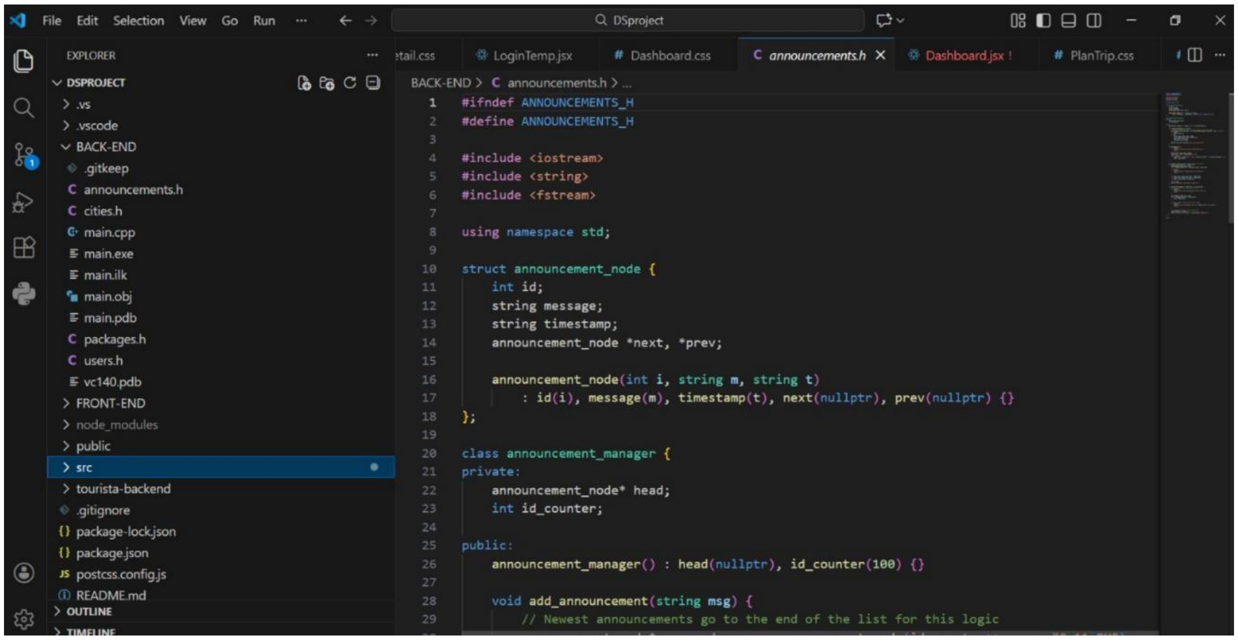
```
import React, { useState, useEffect, useRef } from "react";
import "./Dashboard.css";
import { FaSuitcase } from "react-icons/fa";

// Import your page components
import PlanTrip from "./PlanTrip";
import Homepage from "./Homepage";
import EconomicalPackages from "./EconomicalPackages";
import PremiumPackages from "./PremiumPackages";
import DiscountPackages from "./DiscountPackages";
import Testimonials from "./Testimonials";
import MyBookings from "./MyBookings";

import {
  FaHome,
  FaTags,
  FaPlane,
  FaStar,
  FaMapMarkedAlt,
  FaBars,
  FaBell,
  FaRobot,
  FaPaperPlane,
  FaUser
} from "react-icons/fa";

const Dashboard = () => {
  const [selectedOption, setSelectedOption] = useState("homepage");
```

Css Code:



```css
.message-avatar {
    align-items: center;
    justify-content: center;
    font-size: 0.9rem;
    flex-shrink: 0;
}

.bot-avatar {
    background-color: #008080;
    color: white;
}

.user-avatar {
    background-color: #333;
    color: white;
}

/* Rounded bubbles with tails */
.chat-bubble {
    padding: 12px 16px;
    font-size: 0.95rem;
    line-height: 1.4;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
}

.chat-bubble.bot {
    background-color: #ffffff;
    color: #333;
    border-radius: 18px 18px 18px 2px; /* Tail on bottom left */
```

Header File Code (Backend):



```cpp
#ifndef ANNOUNCEMENTS_H
#define ANNOUNCEMENTS_H

#include <iostream>
#include <string>
#include <fstream>

using namespace std;

struct announcement_node {
    int id;
    string message;
    string timestamp;
    announcement_node *next, *prev;

    announcement_node(int i, string m, string t)
        : id(i), message(m), timestamp(t), next(nullptr), prev(nullptr) {}
};

class announcement_manager {
private:
    announcement_node* head;
    int id_counter;

public:
    announcement_manager() : head(nullptr), id_counter(100) {}

    void add_announcement(string msg) {
        // Newest announcements go to the end of the list for this logic
```

8. Future Work

The current version of TOURISTA provides a solid foundation for travel management. Future iterations of the project will focus on the following enhancements:

- **Payment Gateway Integration:** Implementing secure API connections (e.g., Stripe or PayPal) to allow real-time online transactions and instant receipt generation.
- **Multi-Language & Currency Support:** Expanding the system to support multiple languages and automatic currency conversion for international travelers.
- **Mobile Application:** Porting the desktop system to a dedicated Android/iOS app for on-the-go access and notifications.

9. Conclusion

• TOURISTA successfully addresses the issue of decentralized travel planning by integrating package selection into a single, user-friendly interface. By centralizing these features, the system effectively reduces the time and effort required for tourists to plan their trips, ensuring a smooth and efficient experience from planning to booking.

• The project also demonstrates the practical application of Data Structures and Algorithms (DSA) concepts. Core functionalities are supported using Linked Lists, including Singly Linked Lists and Weighted Linked Lists, which help in managing dynamic travel packages, itineraries, and booking data efficiently.

• Additionally, the system is designed around two main actors: User and Admin. The User can explore travel packages, book hotels, and manage itineraries, while the Admin is responsible for managing system data. This role-based design ensures better control, security, and scalability of the system.

Overall, TOURISTA provides a robust and scalable platform that simplifies the complexities of travel management while reinforcing key programming and DSA concepts through real-world implementation.