



## **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
1. Introduction .....	3
2. Problem Statement .....	3
3. Objective .....	3
4. Problem Solution .....	3
5. Methodology .....	4
6. Flowchart .....	5
7. Visual Outputs .....	7
8. Future Work .....	12
9. Conclusion .....	12

## 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 stay + 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

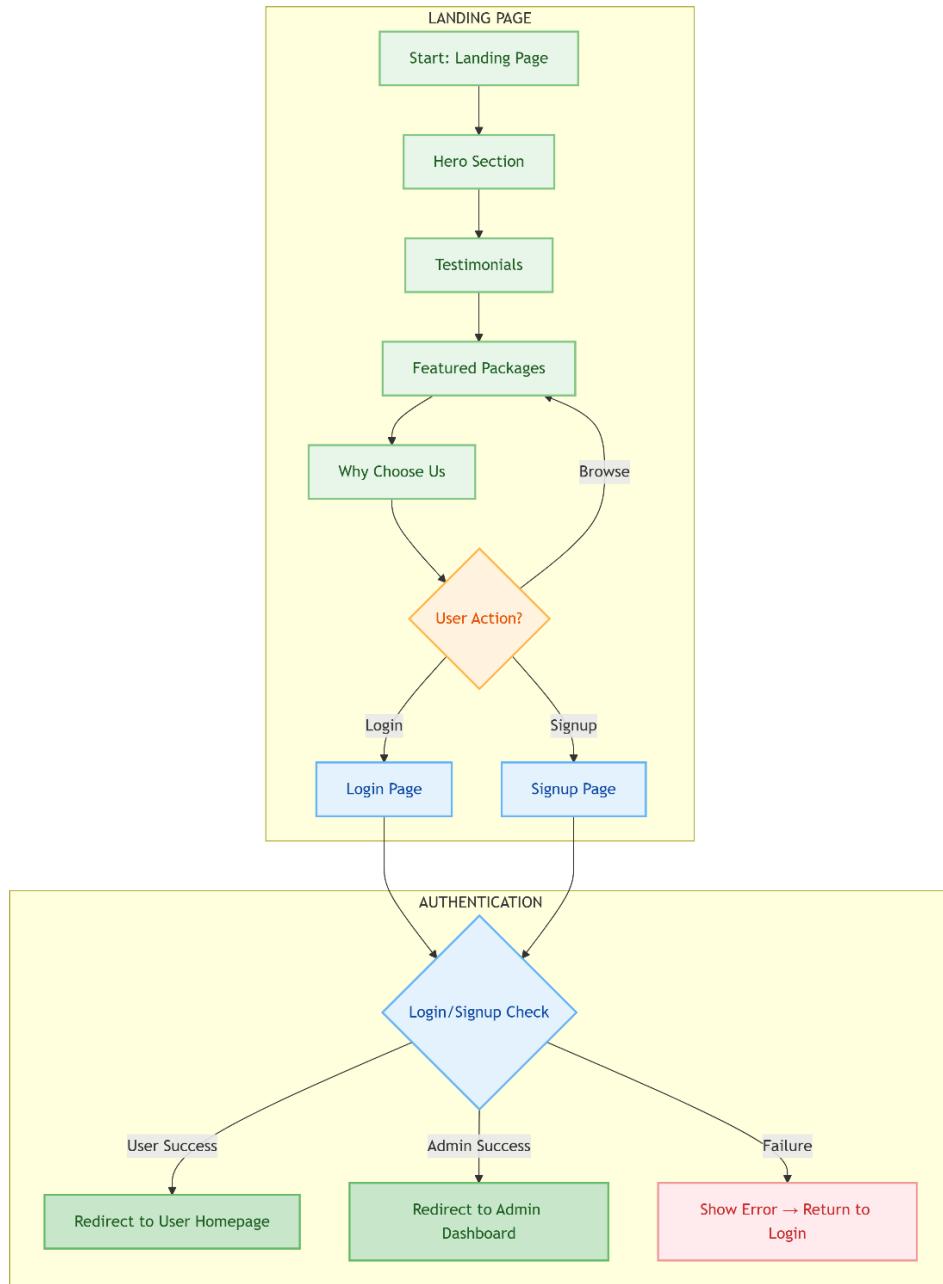
The technical architecture of TOURISTA utilizes specific data structures to solve complex travel management problems, ensuring O(1) or O(log n) efficiency for most core operations.

- 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:**

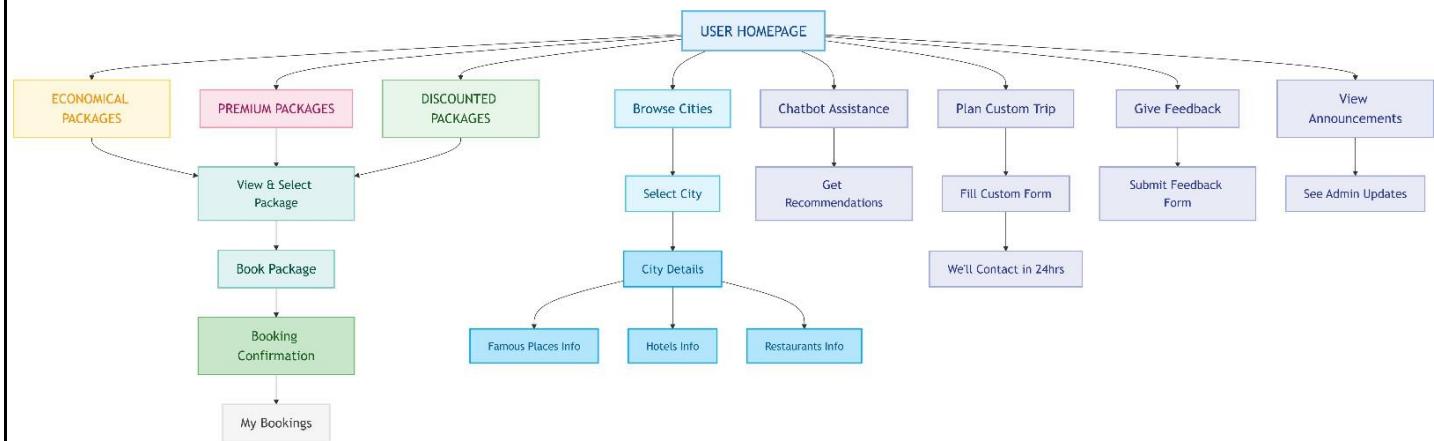
- **Searching:** City names are stored alphabetically in a tree structure.
- **Speed:** This reduces the search time from  $O(n)$  (checking every city one by one) to  $O(\log n)$  (cutting the search area in half with every step), which is critical for the "Search for a city..." bar on the homepage.

## 6. Flowchart

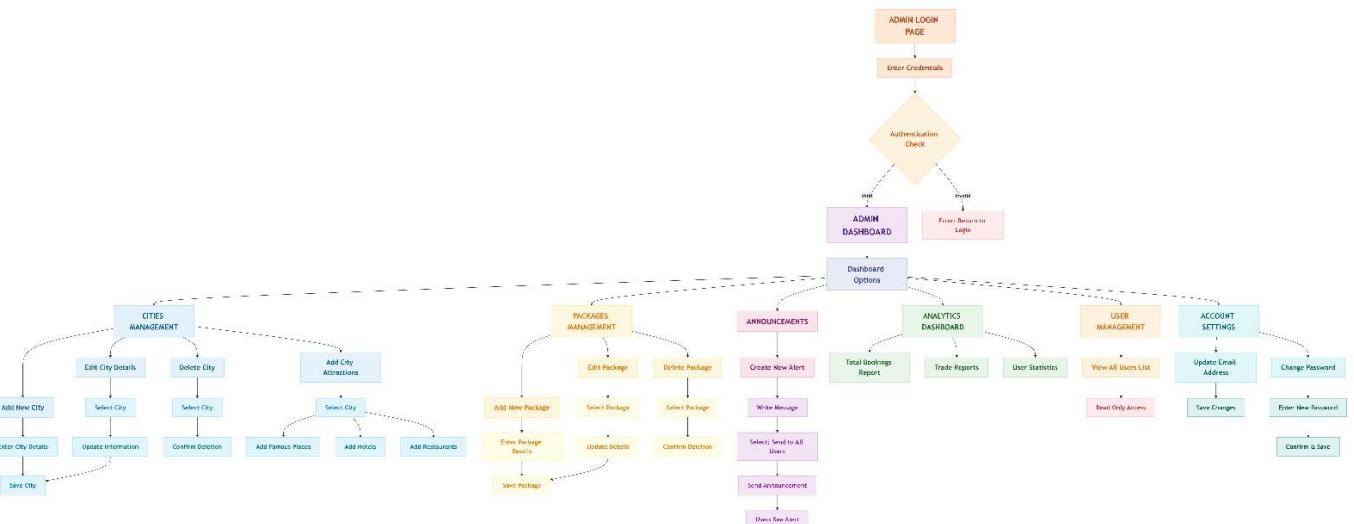
### Authentication flowchart:



## USER FLOWCHART:



## ADMIN FLOWCHART:

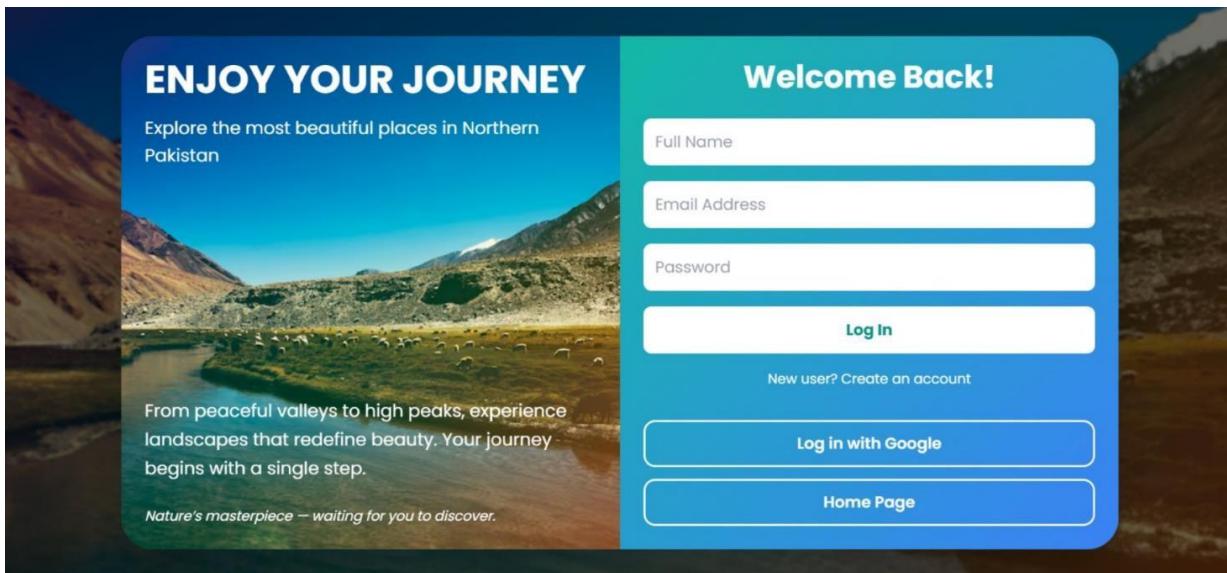


## 7. Visual Outputs

Landing Page:



Login Page:



## User Interfaces:

The screenshot shows the mobile application interface for 'Tourista'. At the top, there is a teal header bar with the 'Tourista' logo and three vertical dots. On the right side of the header are icons for notifications (with a red dot) and logout. Below the header is a navigation menu with the following items: 'Homepage' (selected), 'Special Deals', 'Economical', 'Premium', 'Plan Your Trip', 'My Bookings', and 'Feedback'. The main content area features a large image of a mountain landscape with a horse grazing. Overlaid on the image is the text 'Explore Northern Pakistan'. Below the image is a search bar with the placeholder 'Search for a city...' and a magnifying glass icon. At the bottom of the screen, there is a dark grey footer bar with the text 'Northern Pakistan Cities'.

## Announcements:

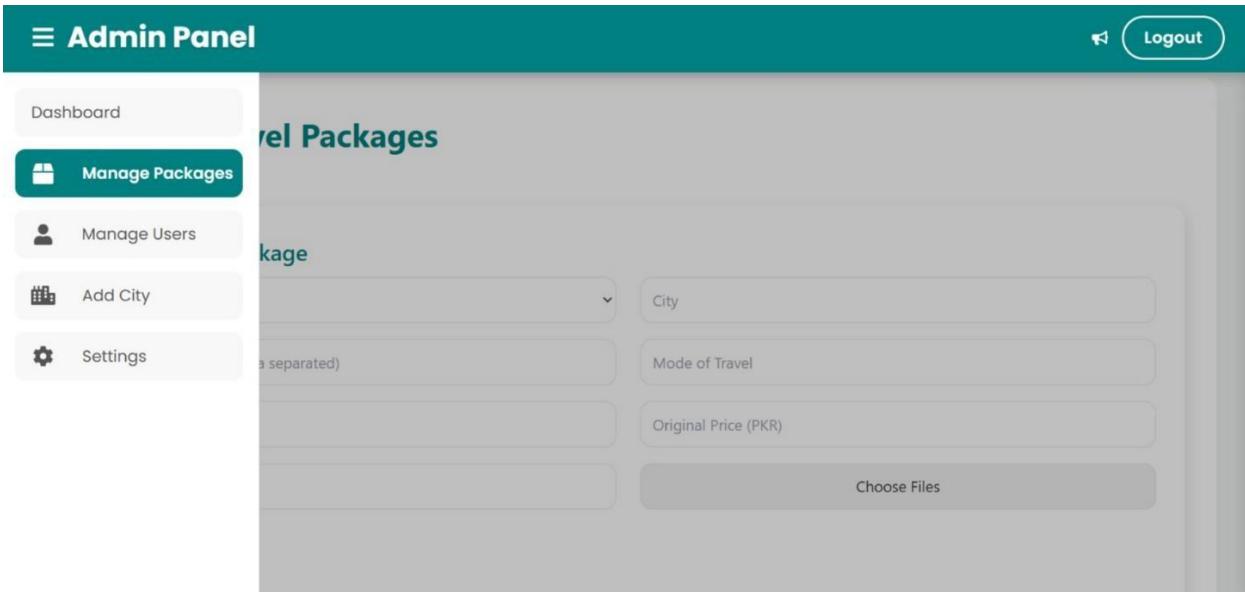
The screenshot shows the mobile application interface for 'Tourista' with an 'Announcements' sidebar. The header and navigation menu are identical to the previous screenshot. The main content area features the same mountain landscape image and 'Explore Northern Pakistan' text. A search bar is also present. On the right side, there is a white sidebar titled 'Announcements' containing three sections: 'Winter Discount!', 'Weather Update', and 'New City Added!'. Each section has a brief description and a small blue arrow icon pointing to the right.

## Chatbot:

The screenshot shows the Tourista AI chatbot interface. At the top, there's a teal header bar with the 'Tourista' logo and a search bar containing the placeholder 'Search for a city...'. Below the header is a large image of a mountainous landscape with a horse grazing. A prominent white banner across the image reads 'Explore Northern Pakistan'. To the right of the banner is a sidebar for 'Tourista AI' which says 'Online' and features a small robot icon. A message bubble from the bot says 'Hi 🤖 I'm Tourista Assistant. How can I help you?'. At the bottom right is a message input field with a placeholder 'Type a message...' and a blue send button.

## Admin Interfaces:

The screenshot shows the Admin Panel dashboard. The top navigation bar includes the 'Admin Panel' logo and a 'Logout' button. Below the navigation is a 'Dashboard' section with a 'Admin / Overview' link. The dashboard features four summary cards: 'Total Revenue \$18,500', 'Bookings 245', 'Users 500', and 'Active Packages 3'. To the right are two charts: 'User Interests' (a pie chart divided into three segments) and 'Package Performance' (a bar chart showing values for Basic, Premium, and Discount packages). The 'Basic' package has a value of 120, 'Premium' has 70, and 'Discount' has 50.



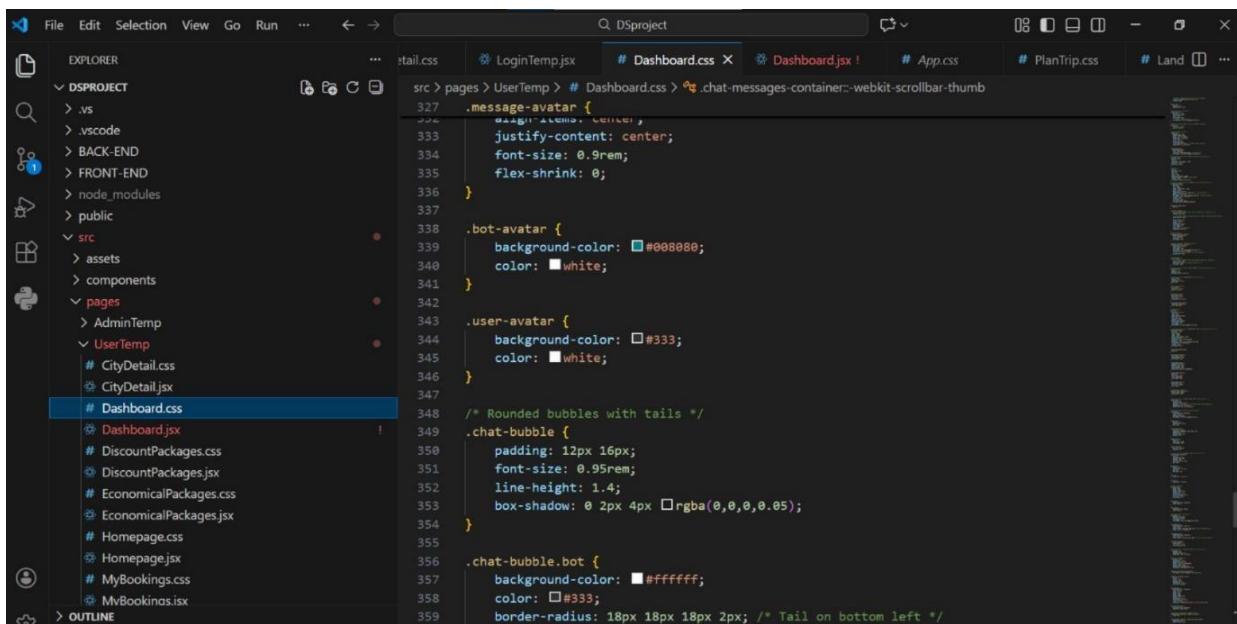
## Jsx Code:

A screenshot of a code editor window titled "DSProject". The left sidebar shows a project structure with "src" selected. The main pane displays the content of "Dashboard.jsx". The code is a React component definition with imports from "react", "react-icons/fa", and various local files like "UserTemp", "Dashboard.css", "PlanTrip", "Homepage", "EconomicalPackages", "PremiumPackages", "DiscountPackages", "Testimonials", and "MyBookings".

```
stail.css LoginTemp.jsx # Dashboard.css Dashboard.jsx ! # PlanTrip.css # LandingPage.css
File Edit Selection View Go Run ... ← → Q DSProject
src > pages > UserTemp > Dashboard.jsx > default
1 import React, { useState, useEffect, useRef } from "react";
2 import "./Dashboard.css";
3 import { FaSuitcase } from "react-icons/fa";
4
5 // Import your page components
6 import PlanTrip from "./PlanTrip";
7 import Homepage from "./Homepage";
8 import EconomicalPackages from "./EconomicalPackages";
9 import PremiumPackages from "./PremiumPackages";
10 import DiscountPackages from "./DiscountPackages";
11 import Testimonials from "./Testimonials";
12 import MyBookings from "./MyBookings";
13
14 import {
15   FaHome,
16   FaTags,
17   FaPlane,
18   FaStar,
19   FaMapMarkedAlt,
20   FaBars,
21   FaBell,
22   FaRobot,
23   FaPaperPlane,
24   FaUser
25 } from "react-icons/fa";
26
27 const Dashboard = () => {
28   const [selectedOption, setSelectedOption] = useState("homepage");

```

## Css Code:



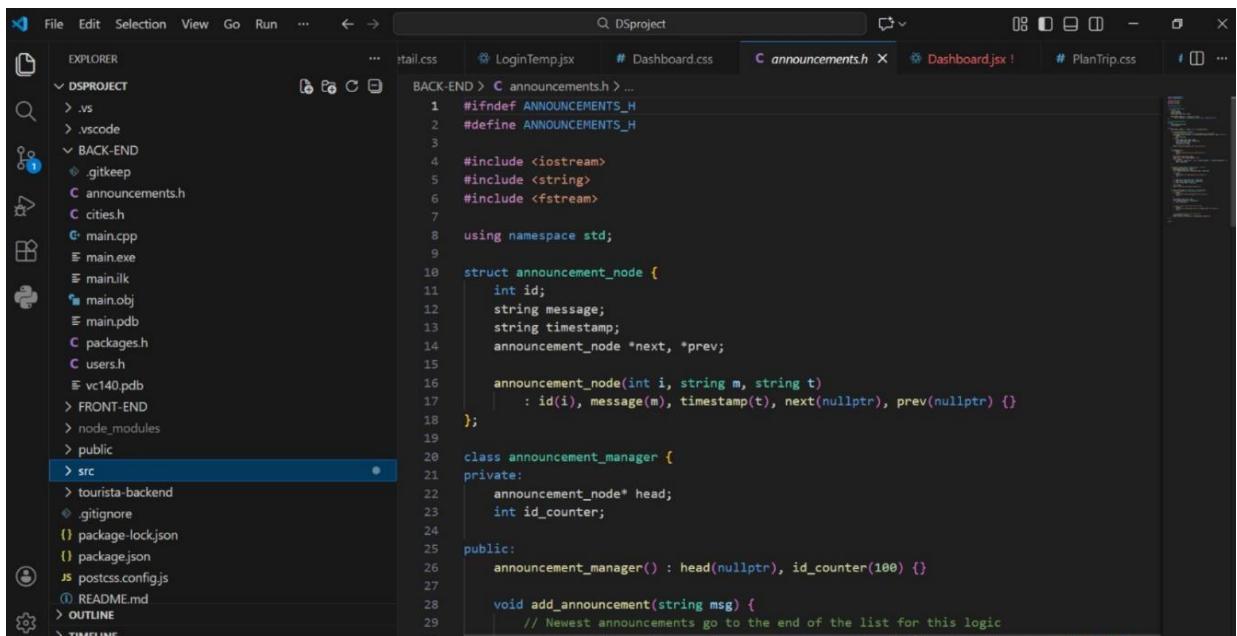
A screenshot of the Visual Studio Code interface. The left sidebar shows a project structure under 'DSPROJECT' with folders like BACK-END, FRONT-END, node\_modules, public, src, and pages. The 'src' folder contains files such as AdminTemp, CityDetail.css, CityDetail.jsx, Dashboard.css, Dashboard.jsx, DiscountPackages.css, DiscountPackages.jsx, EconomicalPackages.css, EconomicalPackages.jsx, Homepage.css, Homepage.jsx, MyBookings.css, and MyBookings.jsx. The right pane displays the content of 'Dashboard.css'. The code is a stylized version of the provided snippet, with color-coded syntax highlighting for CSS properties like background-color and font-size.

```
tail.css | LoginTemp.jsx | # Dashboard.css | Dashboard.jsx | App.css | PlanTrip.css | Land.css | ...
```

```
src > pages > UserTemp > # Dashboard.css > .chat-messages-container::webkit-scrollbar-thumb
```

```
  .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):



A screenshot of the Visual Studio Code interface. The left sidebar shows a project structure under 'DSPROJECT' with BACK-END, FRONT-END, node\_modules, and public. The 'src' folder contains files like announcements.h, cities.h, main.cpp, main.exe, main.lib, main.pdb, packages.h, users.h, and vc140.pdb. The right pane displays the content of 'announcements.h'. The code defines a struct 'announcement\_node' and a class 'announcement\_manager'.

```
tail.css | LoginTemp.jsx | # Dashboard.css | C announcements.h | Dashboard.jsx | # PlanTrip.css | ...
```

```
#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.