# 1. Introduction

# Project Title: TravelSphere – Your Smart Travel Companion

# Team Members:

# Shivang Yadav – Frontend Developer & Project Lead

# Shreya Kumari– Backend Developer

# Parth Singh – UI/UX Designer

# Sristi Shrivastav– Database Administrator

# 2. Project Overview

# Purpose:TravelSphere is a smart travel planning and booking platform that allows users to explore destinations, book trips, and receive real-time updates, recommendations, and support – all from one app.

# Features:

# User registration/login via email, Gmail, or LinkedIn

# Search and filter destinations

# Real-time travel suggestions using ML

# Travel package booking with payment gateway integration

# Chat support and user feedback system

# Admin dashboard for managing packages and users

# 3. Architecture

# Frontend: Built using React.js with component-based architecture. Implements routing via React Router and uses Redux for state management.

# Backend: RESTful API built with Node.js and Express.js. Contains routes for user management, bookings, destinations, chat, and admin features.

# Database: MongoDB (NoSQL) used for storing users, destinations, bookings, and chat data. Mongoose ORM used for schema and queries.

# 4. Setup Instructions

# Prerequisites:

# Node.js (v18+)

# MongoDB (local or cloud-based like MongoDB Atlas)

# Git

# Installation Steps:

# git clone https://github.com/<your-repo>/travelsphere.git

# cd travelsphere

# For Backend:

# cd server

# npm install

# cp .env.example .env # Fill in environment variables

# For Frontend:

# cd client

# npm install

# 5. Folder Structure

# Client (React Frontend):

# client/

# │

# ├── public/

# ├── src/

# │ ├── components/

# │ ├── pages/

# │ ├── assets/

# │ ├── redux/

# │ └── App.js

# └── package.json

# Server (Node Backend):

# server/

# │

# ├── controllers/

# ├── models/

# ├── routes/

# ├── middlewares/

# ├── config/

# ├── utils/

# └── server.js

# 6. Running the Application

# Frontend:

# cd client

# npm start

# Backend:

# cd server

# npm start

# 7. API Documentation

| Endpoint | Method | Description | Auth Required |
| --- | --- | --- | --- |
| /api/auth/register | POST | Register a new user | No |
| /api/auth/login | POST | Login user | No |
| /api/destinations | GET | Fetch all travel destinations | No |
| /api/booking | POST | Book a travel package | Yes |
| /api/chat | GET | Fetch chat messages | Yes |
| /api/admin/packages | CRUD | Manage travel packages | Admin only |

# 8. Authentication

# Strategy: JWT-based token authentication

# Flow:

# On login/registration, a token is issued.

# Protected routes require this token in headers.

# Admin routes use role-based access control (RBAC).

# 9. User Interface

# Clean, travel-themed UI using React, CSS Modules, and icons from FontAwesome.

# Responsive design for mobile and desktop.

# UI components:

# Search bar with filters

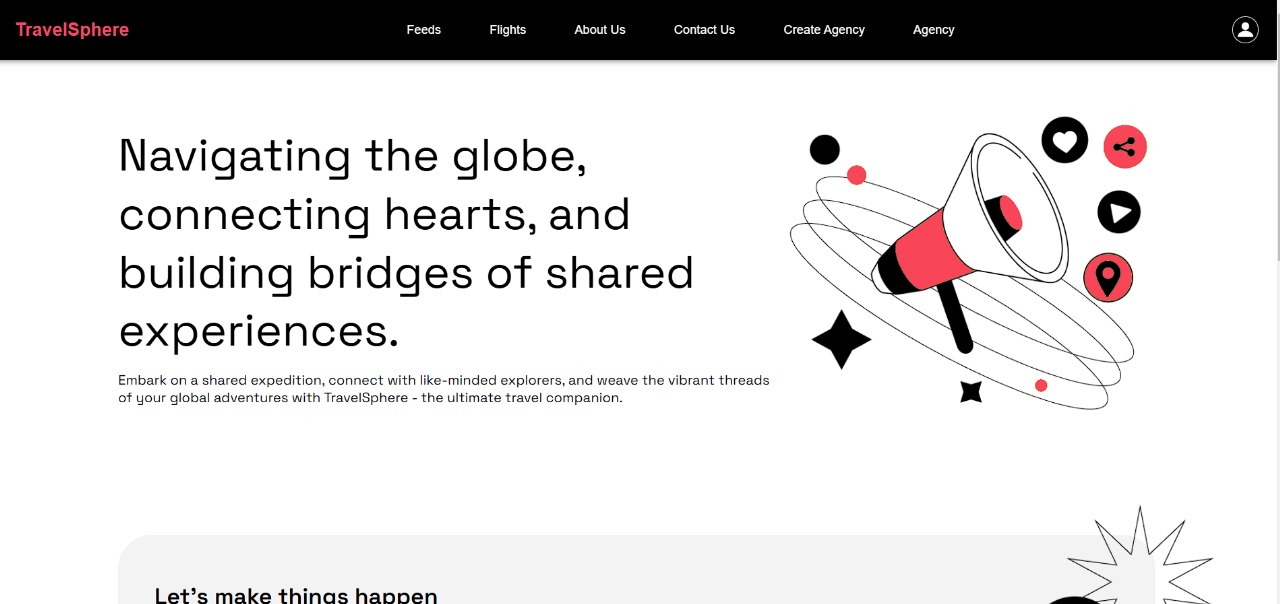
# Card view for packages

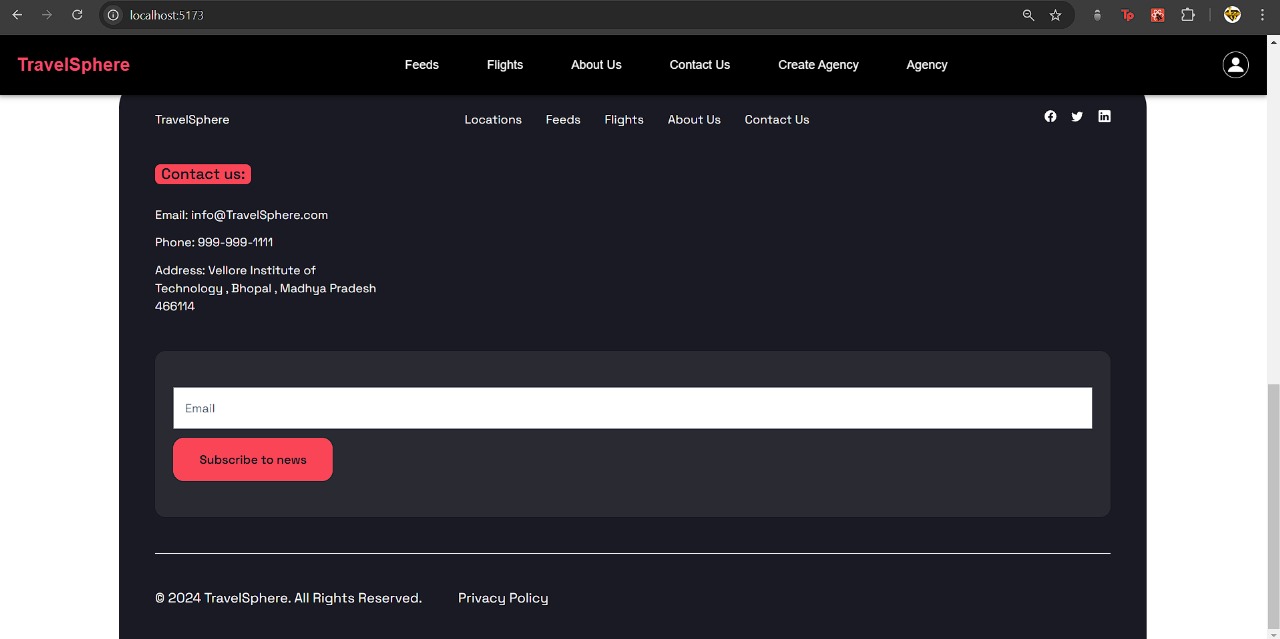
# Booking form

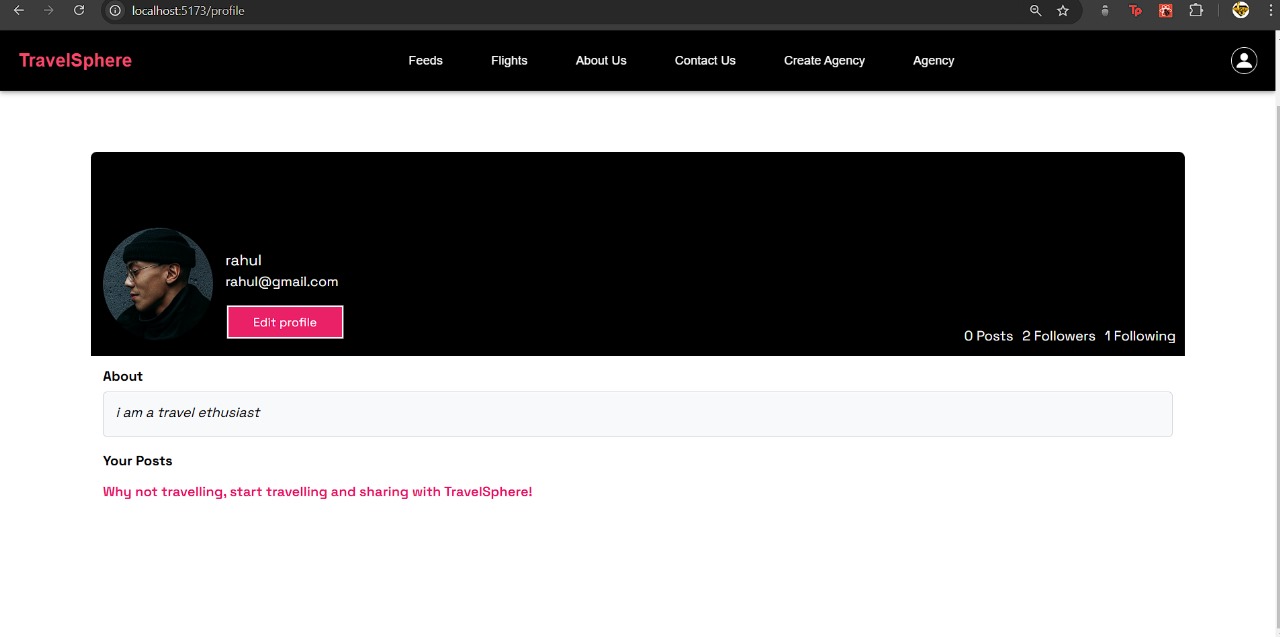
# Chat interface

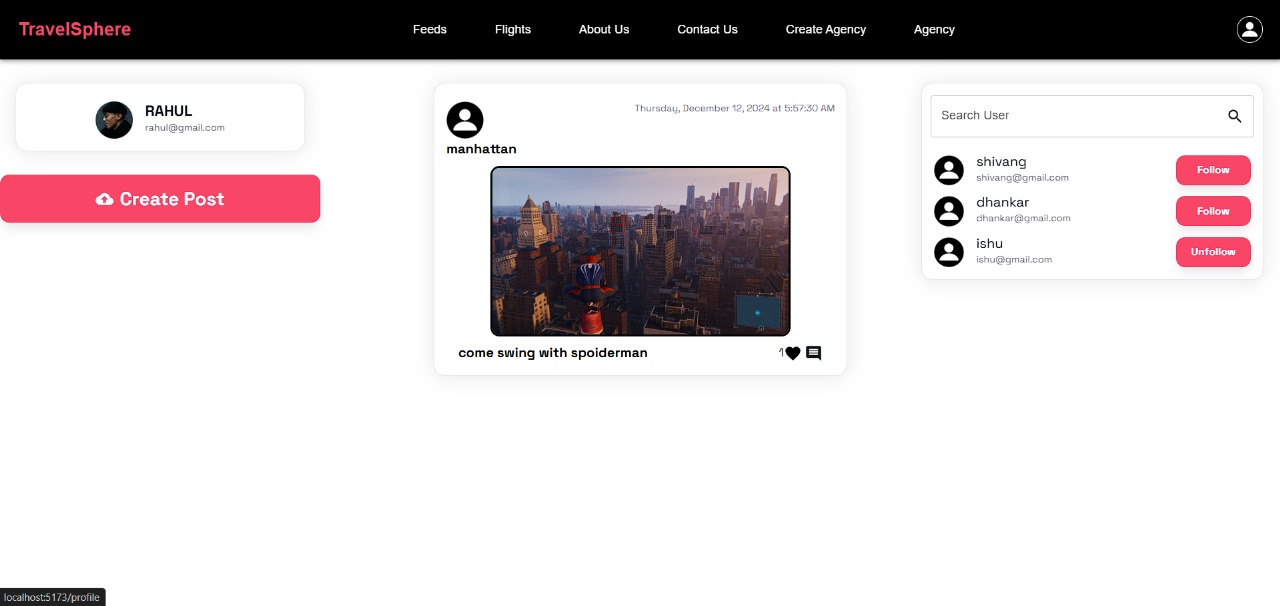
# *Screenshots to be added below.*

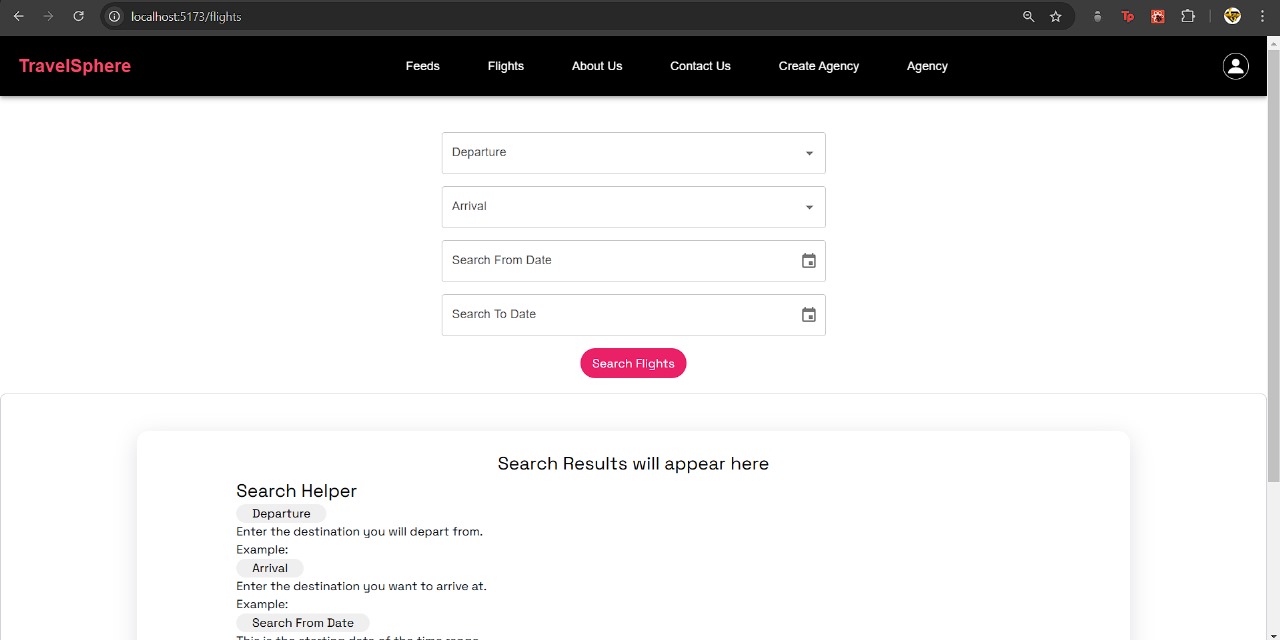
# 

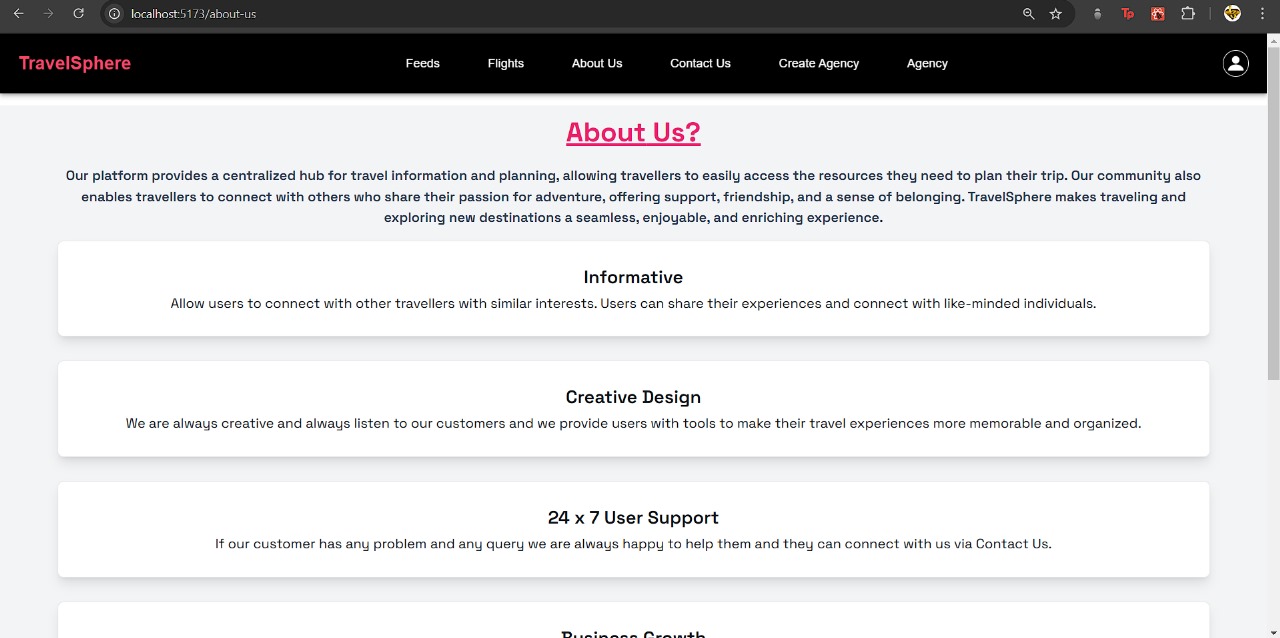
****

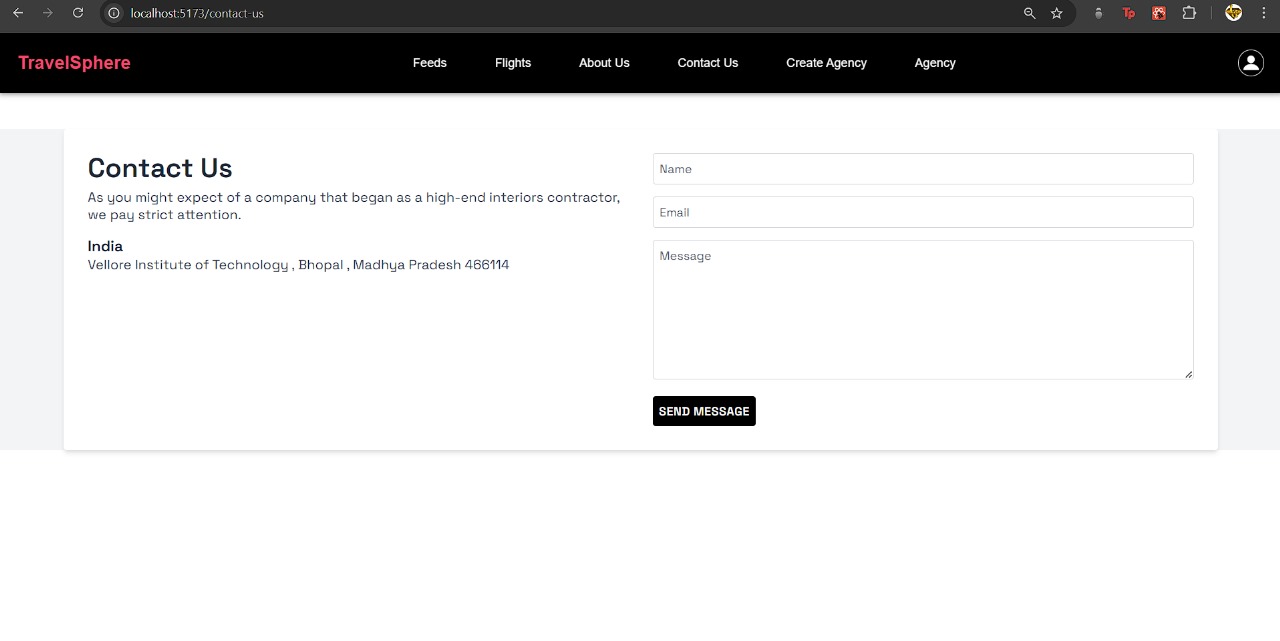
****

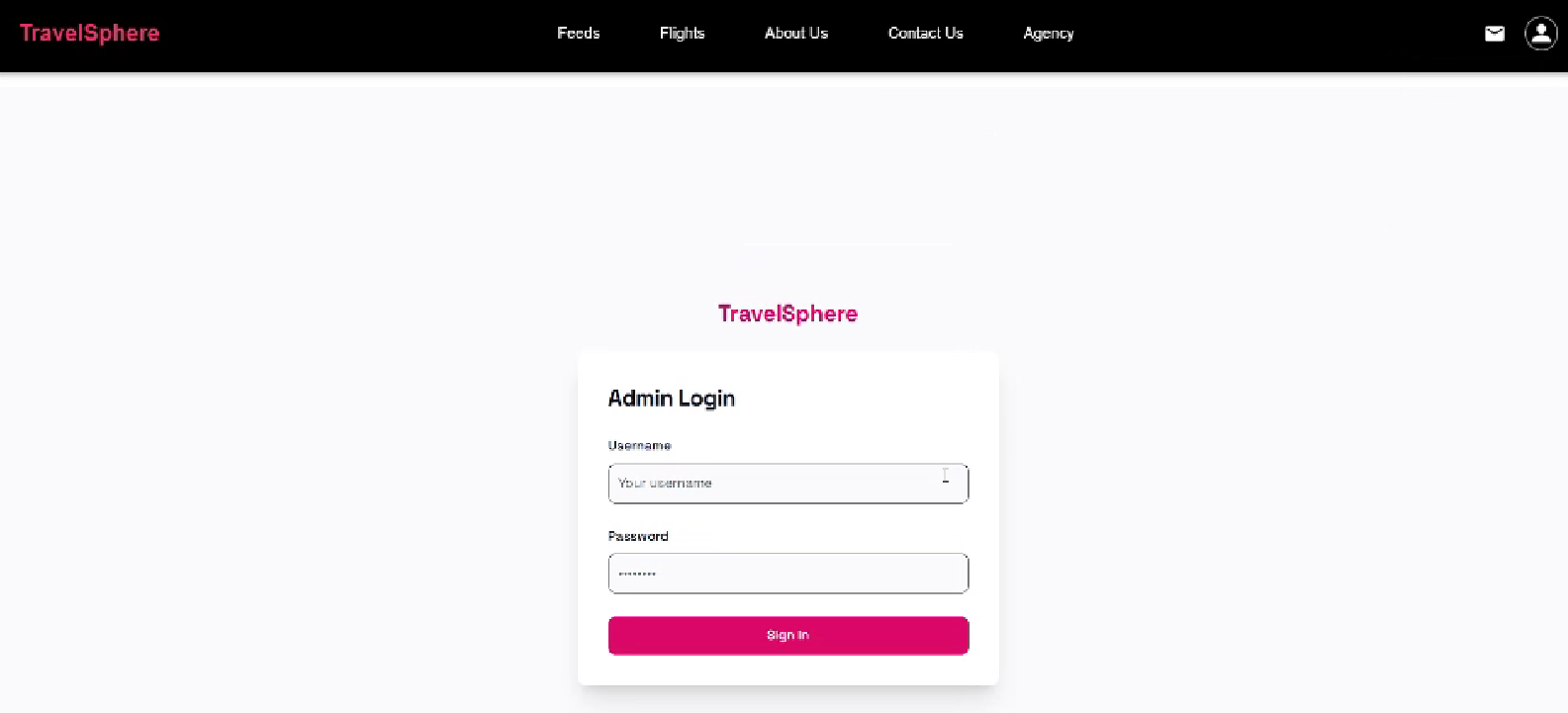
****

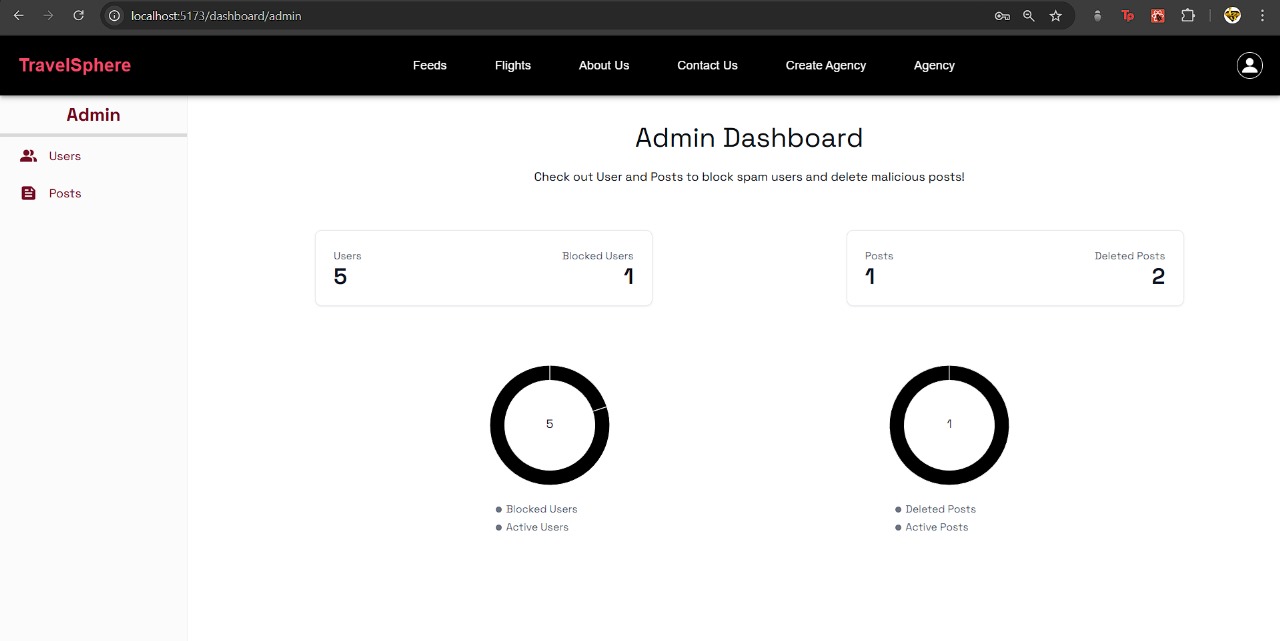
****

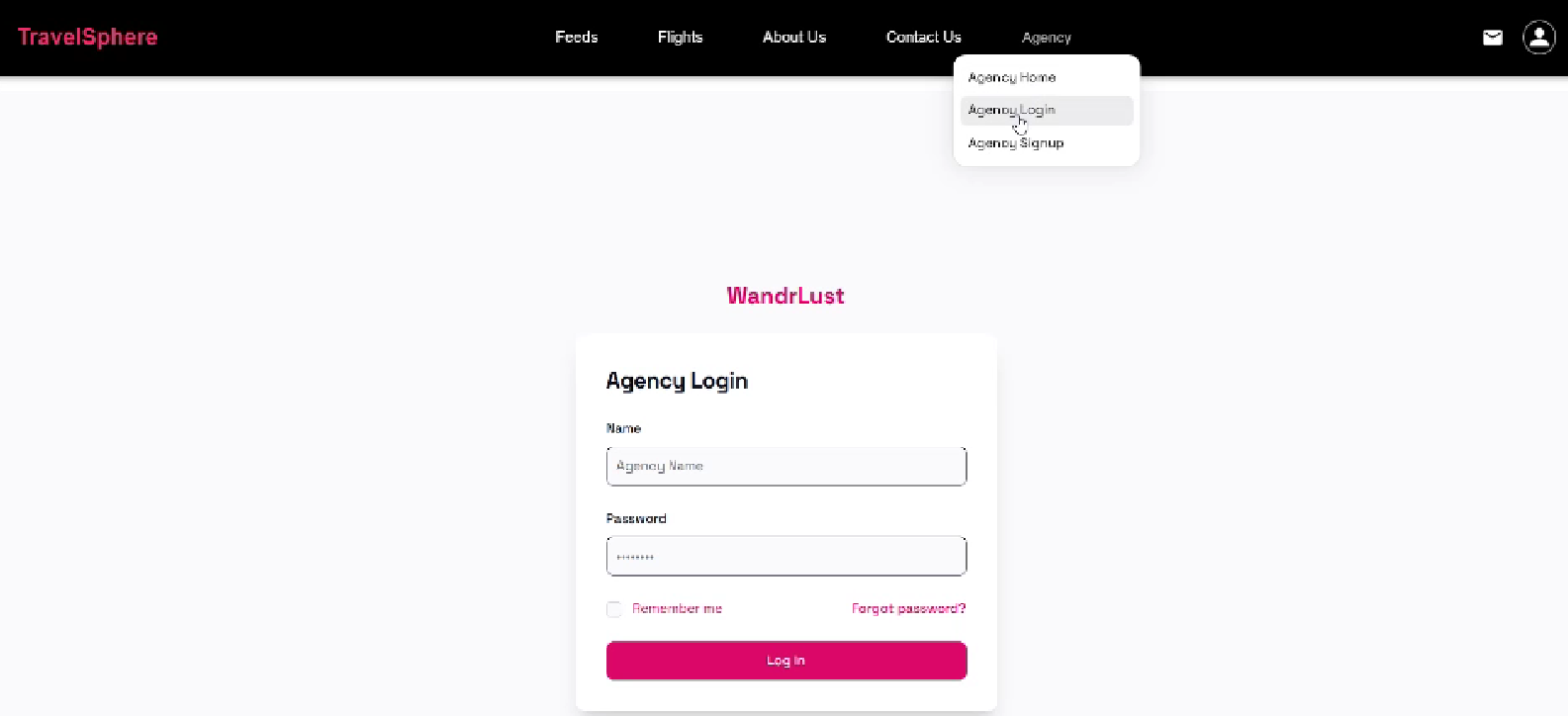
****

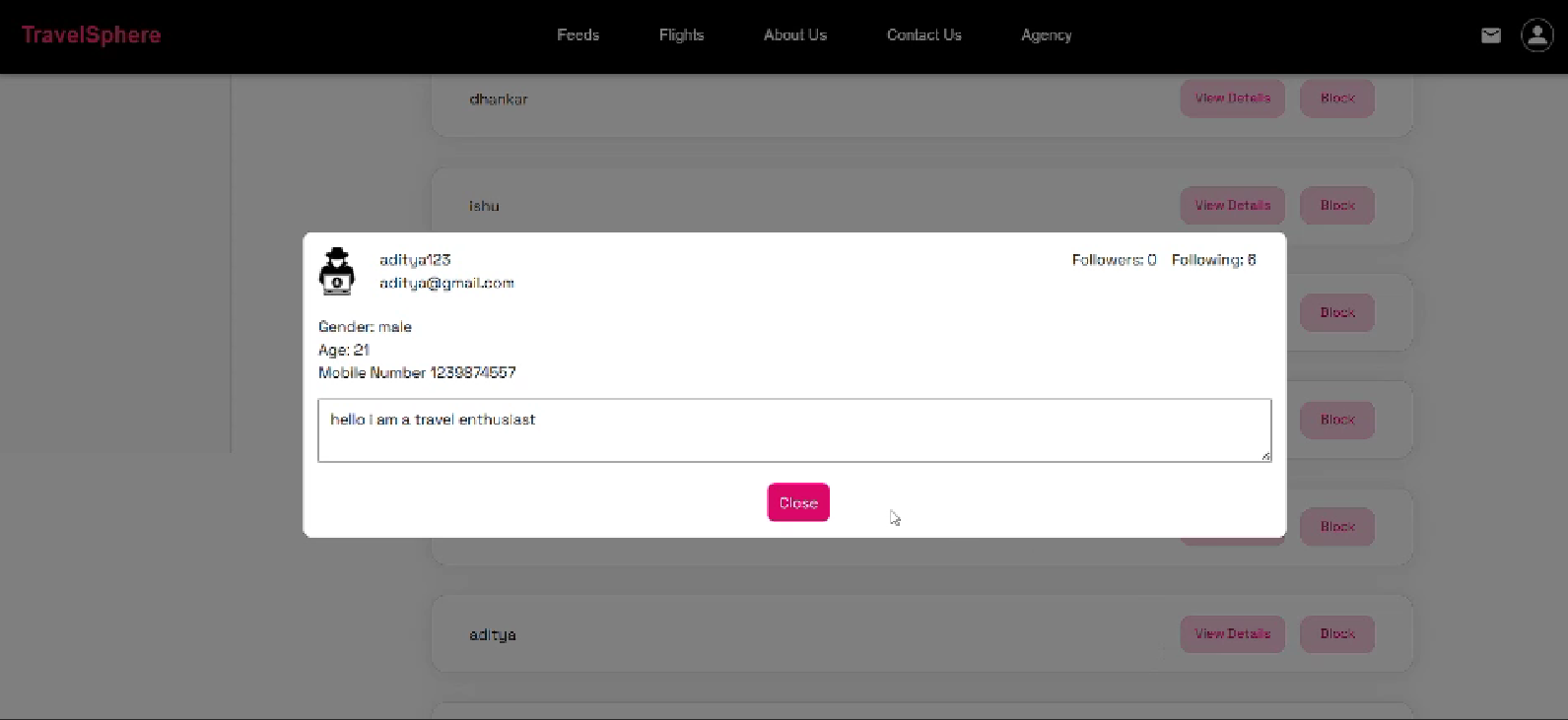
****

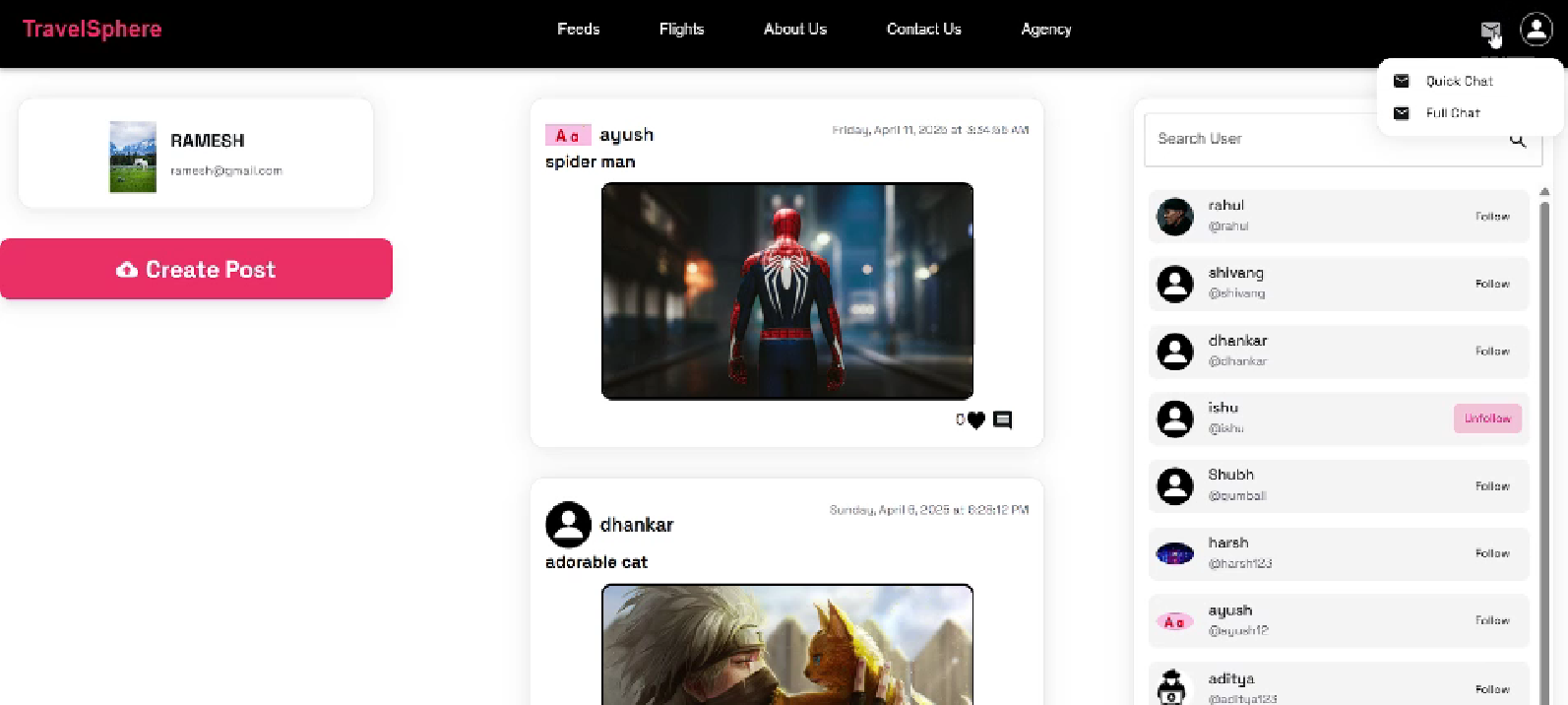
****

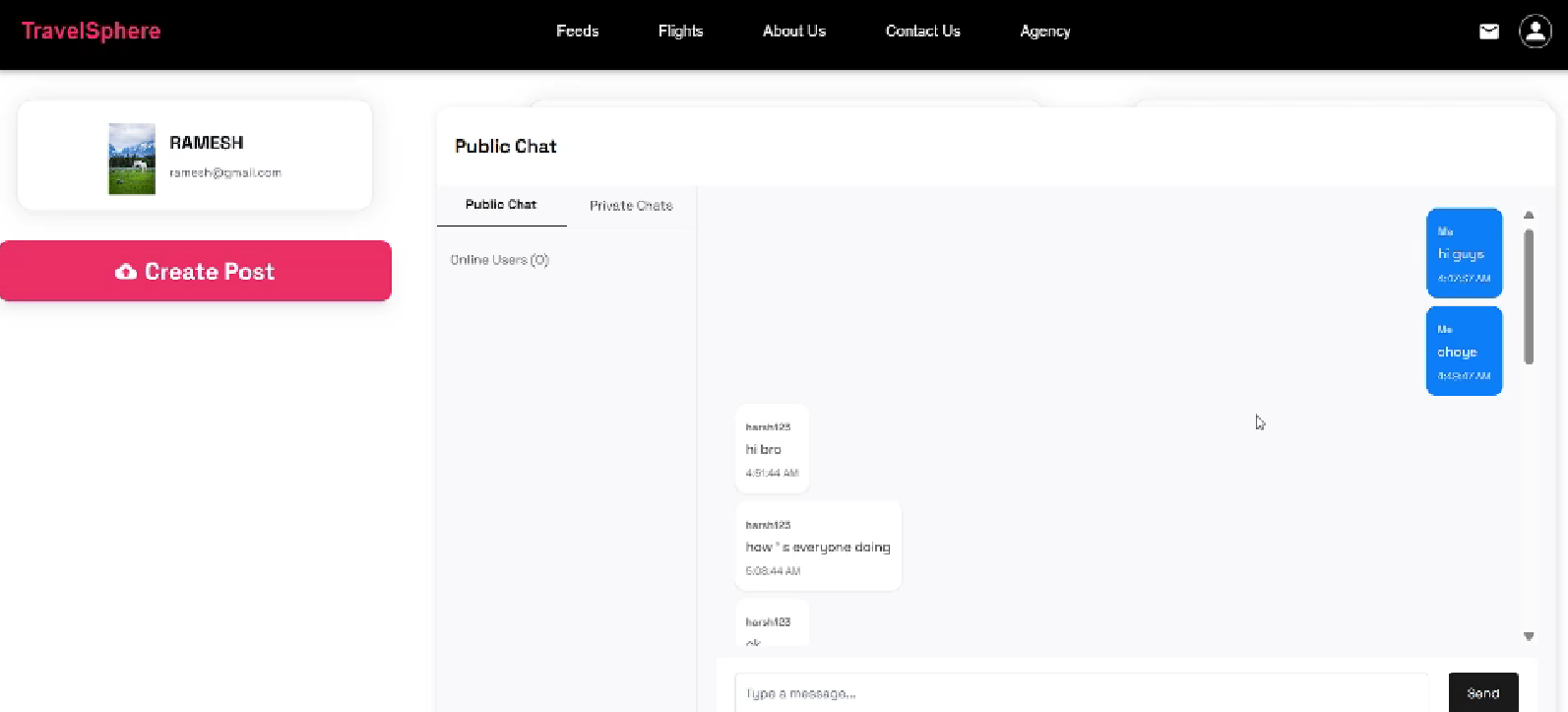
****

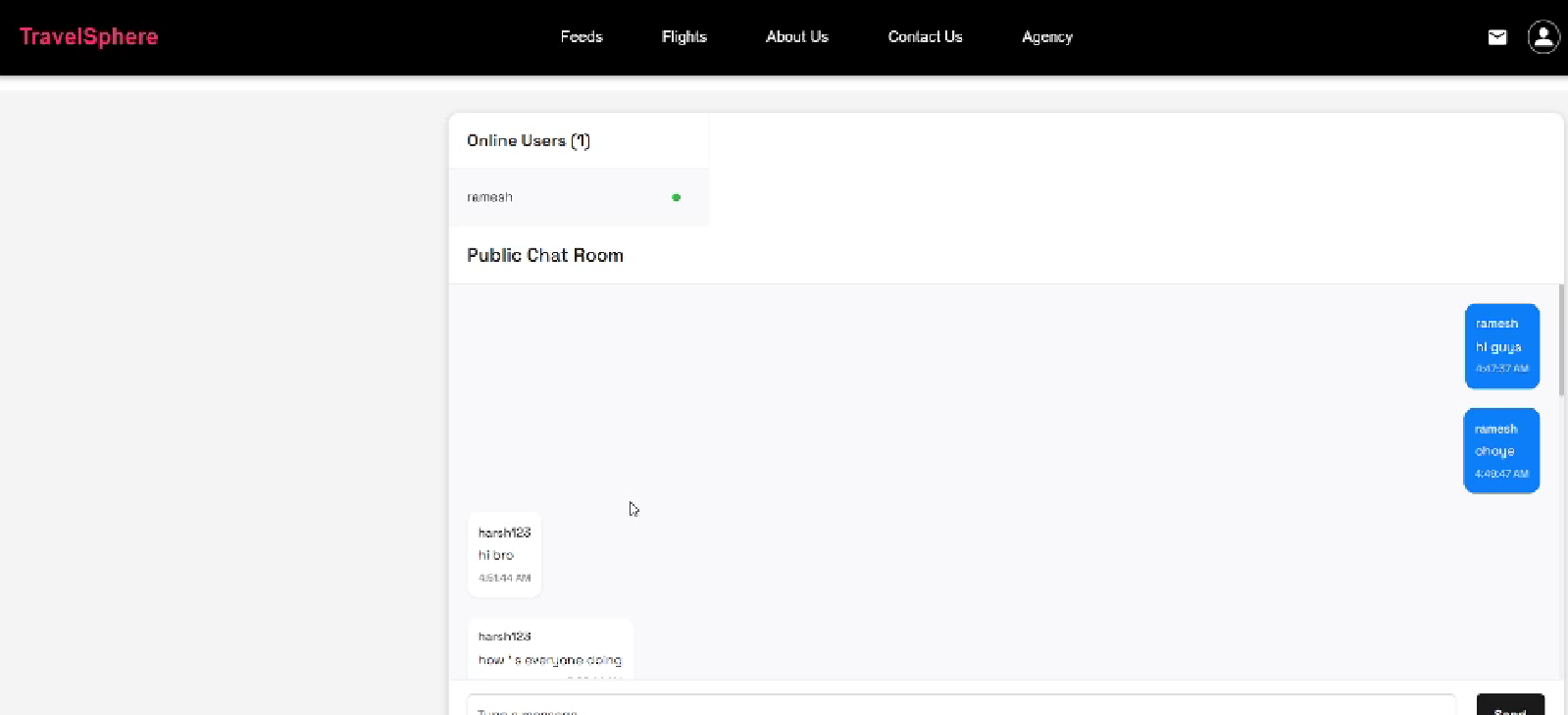
****

****

****

****

****

****

# 10. Testing

# Unit Testing: Jest + React Testing Library for frontend.

# API Testing: Postman & Mocha for backend endpoints.

# E2E Testing: Cypress (optional).

# 11. Screenshots or Demo

# [Add screenshots here or link to demo]

# Live Demo: https://travelsphere-demo.netlify.app

# 12. Known Issues

# Delays in loading search results on slow networks.

# Email confirmation may be delayed due to SMTP limits.

# Chat feature does not yet support group conversations.

# 13. Future Enhancements

# AI-based itinerary generation

# Offline booking capability

# Integration with hotel/flight APIs

# Multilingual support

# Voice assistant for travel planning