A PROJECT REPORT

**ON**

**( TravelMate AI )**

**Designed & Developed by:**

###### Mr. Satyam G. Agrahari

###### (Seat No. )

**Guided by:**

**Asst.****Prof: Ganesh Kumar R. Narayankar**

SUBMITTED IN PARTIAL FULFILLMENT OF ACADEMIC PROJECT

**(Bachelors of Science Computer Science)**



###### University of Mumbai

SHREE SHANKAR NARAYAN EDUCATION TRUST

**SHANKAR NARAYAN COLLAGE OF ARTS, COMMERCE & SCIENCE**

**Academic Year 2024-25**



#### ACKNOWLEDGEMENT

I express my sincere thanks to Principal Sir "**Dr. V. N. Yadav**", Shankar Narayan College, located in Bhayandar who has given me the opportunity to Pursue my Bachelor's Degree in Computer Science Department.

I also thank H.O.D “**Prof. Vaishali Kadam**" and other staff of the Computer Science Department. I would like to thank our guide.

**Prof: Ganesh Kumar R. Narayankar** ", for his encouragement and guidance, which helped me in completing this project. Finally, I would like to thank our colleagues and friends who helped me complete this project successfully.

I would also like to express my heartfelt gratitude to my parents, teachers, and friends for their direction, motivation, and selfless support.

Project Associate

"Satyam Agrahari"

# ABSTRACT

##### Abstract

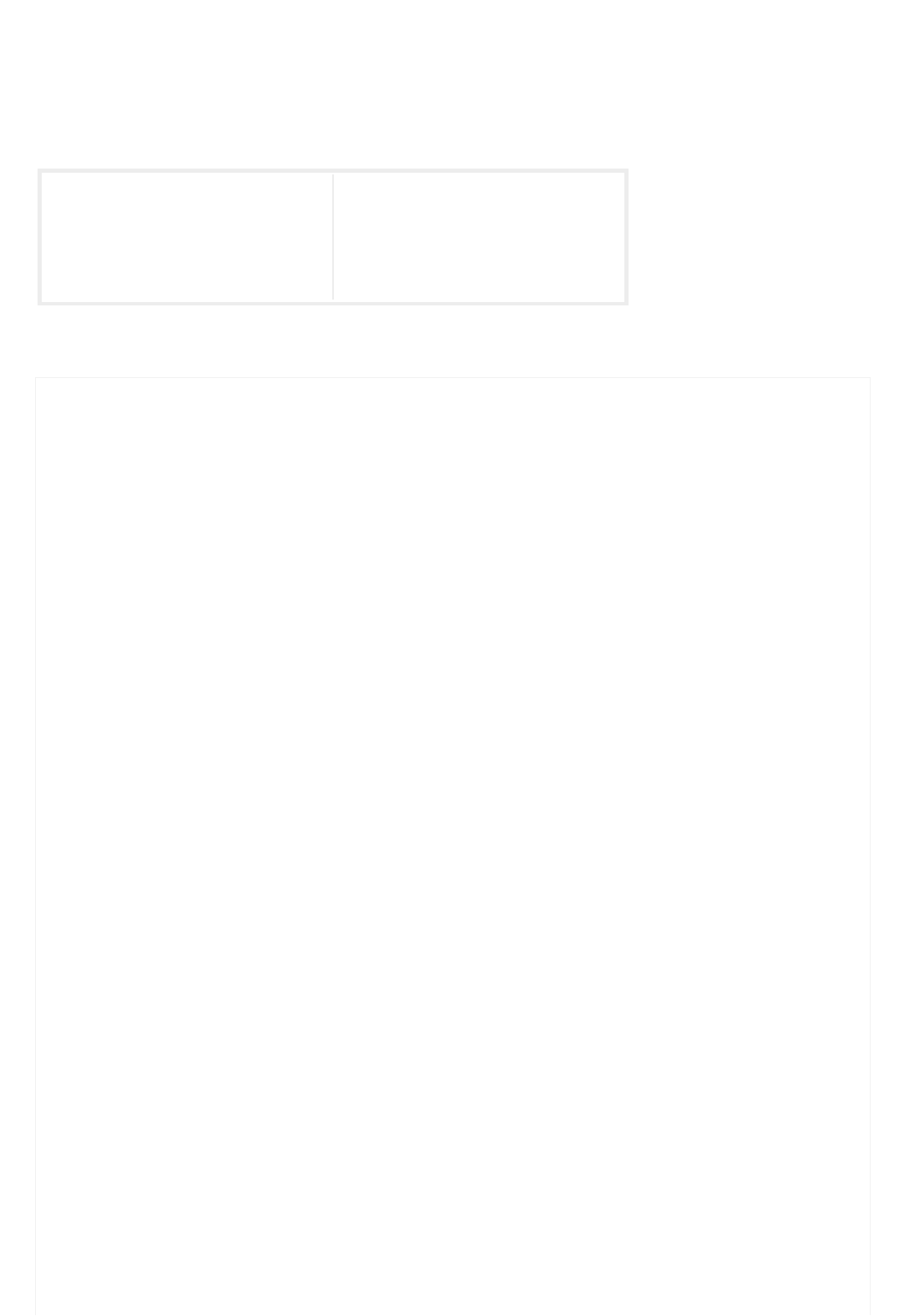
TravelMate AI steps in, transforming the way you plan your trips with the power of artificial intelligence.moreArranging a vacation may be a thrilling and daunting task. The details, which range from selecting the ideal location to scheduling travel and lodging, may quickly become overwhelming.

These cutting-edge technologies can help you find the greatest offers, create comprehensive itineraries, or get tailored recommendations. TravelMate AI may improve your trip experience by facilitating easier and pleasurable planning.

Arranging a vacation can be both exhilarating and overwhelming.

Our platform harnesses cutting-edge technology to make travel planning easier and more enjoyable. Built with a robust Node.js backend and a sleek React.js frontend, TravelMate AI offers a seamless and intuitive user experience.

Users can input their desired destinations, travel dates, budget constraints, and other preferences. The application then utilizes intelligent algorithms to curate personalized travel itineraries, accommodation suggestions, and activity recommendations tailored to their needs. By employing a modular architecture, TravelMate AI ensures scalability and facilitates future enhancements.



##### Plagiarism Report

Itinerary Building: Users can easily browse through the recommendations and visually build their itinerary using a

Travel Preferences: Users input their desired destinations, travel dates, budget, and any specific interests.

The workflow is as follows:

management.

preferences are securely stored, enabling personalized recommendations and seamless itinerary

trip data and preferences are securely stored, enabling personalized recommendations and seamless itinerarytrip data and

with a chosen travel booking API (e.g., Amadeus, Sabre) to fetch real-time data on accommodations, hotels and more. User

Backend: A robust backend powered by Node.js and Express.js works behind the scenes to process user requests, interface

preferences, explore a diverse range of destinations and activities, and visually build their ideal itineraries.

empowers travelers to effortlessly input their travel

process from destination selection to itinerary creation. Frontend: This user-friendly web interface, built with React.js,

TravelMate AI is engineered to revolutionize the way people plan their vacations, automating and personalizing the

suggestions can significantly enhance your trip.

and

mobPersonal Recommendations: AI technologies that use your input and historical information to provide tailored advice

Ease of Access: It’s critical to confirm the tool’s accessibility, including whether you can use a desktop browser or a

according to your tastes and requirements.

Level of Customization: The application must pose pertinent queries.and offer choices for customizing your schedule

in AI Travel Planner Tool

With TravelMate AI, you're not just Making a trip, you're embarking on a journey tailored to your dreams.Things to Look for

finding the most suitable flights, accommodations, and activities to craft your ideal itinerary.

sophisticated algorithms, fueled by a robust travel booking API [specify API used, e.g., Amadeus, Sabre], then go to work,

Through an intuitive interface powered by React.js, TravelMate AI allows you to effortlessly input your travel vision. Our

dates.

technologies to curate tailored travel itineraries, taking into account your unique preferences, budget, and desired travel

Gone are the days of tedious research and endless comparisons. This intelligent platform harnesses cutting-edge

TravelMate AI is a user-friendly, AI-powered application designed to simplify and personalize the way we plan our trips.

adventure. This is where TravelMate AI steps in, offering a breath of fresh air in the world of travel planning.

overwhelming options, and the pressure of finding the best deals can quickly overshadow the excitement of an upcoming

In an era defined by busy schedules and endless choices, planning a vacation can often feel like navigating a maze.

Content Checked For Plagiarism

Characters

6570

Plagiarised

Unique

Words

870

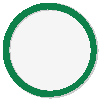
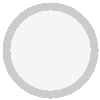
0%

100%

Date

2024-10-10

PLAGIARISM SCAN REPORT



**Name: – Satyam Agrahari Class: - T.Y. Computer Science**

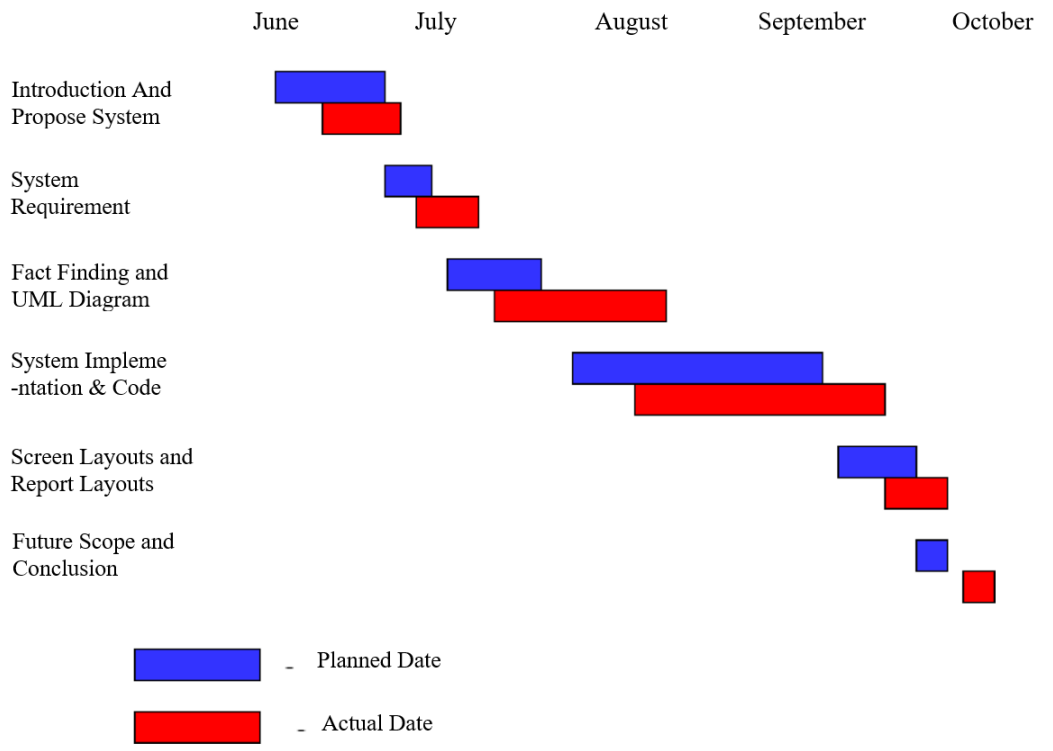
**Roll No.: - 61**

**Project Name: – TravelMate AI**

**Activity Sheet**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Content** | **Planned Date** | | **Actual Date** | | **Sign** |
| **Start**  **Date** | **End**  **Date** | **Start**  **Date** | **End**  **Date** |
| 1. | Introduction And  Propose System | 20-06-24 | 28-06-24 | 21-06-24 | 01-07-24 |  |
| 2. | System Requirement | 01-07-24 | 10-07-24 | 03-07-24 | 11-07-24 |  |
| 3. | Fact Finding And  UML Diagram | 11-07-24 | 29-08-24 | 12-07-24 | 30-07-24 |  |
| 4. | System Implementation And Code | 01-08-24 | 28-08-23 | 30-08-24 | 25-09-24 |  |
| 5. | Screen Layouts And  Report Layouts | 29-09-24 | 30-09-24 | 25-09-24 | 01-10-24 |  |
| 6. | Future Scope And Conclusion | 01-10-24 | 07-10-24 | 03-10-24 | 10-10-24 |  |

**Gantt chart**



**Index**

|  |  |  |
| --- | --- | --- |
| **1** | **Preliminary Investigation** | |
|  | 1.1 Introduction | 11 |
| 1.2 Features | 12 |
| 1.3 Propose System | 13 |
| 1.4 Fact finding and Question Answer | 14 |
| 1.5 Advantages & Disadvantages | 15 |
| **2** | **Requirements** | |
|  | 2.1 Requirements | 18 |
| 2.2 Prototype System | 19 |
| **3** | **System Design** | |
|  | 3.1 Stake Holder | 22 |
| 3.2 Event Table | 23 |
| 3.3 UML | 24 |
| **4** | **System Implementation & Code** | |
|  | 4.1 Modules | 35 |
| 4.2 Test Case | 37 |
| 4.3 System Code | 39 |
| **5** | **Screenshots and Repot Layout** | 58 |
| **6** | **Conclusion** | 63 |
| **7** | **Future Enhancement** | 65 |
| **8** | **References and Bibliography** | 66 |

## Preliminary Investigation

##### Introduction

In an era defined by busy schedules and endless choices, planning a vacation can often feel like navigating a maze. overwhelming options, and the pressure of finding the best deals can quickly overshadow the excitement of an upcoming adventure. This is where TravelMate AI steps in, offering a breath of fresh air in the world of travel planning.

TravelMate AI is a user-friendly, AI-powered application designed to simplify and personalize the way we plan our trips. Gone are the days of tedious research and endless comparisons. This intelligent platform harnesses cutting-edge technologies to curate tailored travel itineraries, taking into account your unique preferences, budget, and desired travel dates.

Through an intuitive interface powered by React.js, TravelMate AI allows you to effortlessly input your travel vision. Our sophisticated algorithms, fueled by a robust travel booking API [specify API used, e.g., Amadeus, Sabre], then go to work, finding the most activities to craft your ideal itinerary.

With TravelMate AI, you're not just Making a trip, you're embarking on a journey tailored to your dreams.

##### Implementation

TravelMate AI utilizes a modern and efficient technology stack to deliver a seamless travel planning experience. The user interface, designed with React.js, provides an intuitive and interactive platform for users to input their travel preferences, explore destinations, and build personalized itineraries. The application leverages Firebase local storage for persistent data handling, allowing users to save and retrieve their travel plans effortlessly. To provide a rich selection of travel options,

##### Features

When looking for an AI travel planner tool, there are several key features to consider that can enhance your planning experience:

1. **Level of Customization:** The application must pose pertinent queries.and offer choices for customizing your schedule according to your tastes and requirements.
2. **Ease of Access:** It’s critical to confirm the tool’s accessibility, including whether you can use a desktop browser or a mobile app.
3. **Collaborative Features:** If you plan a trip with others, specific applications let you work together to develop itineraries with travel companions. Personal Recommendations: AI technologies that use your input and historical information to provide tailored advice and suggestions can significantly enhance your trip.
4. **Smart Destination Search**:Keyword-based: Allow users to search by city, country, landmarks, or even vague descriptions .
5. **Personalized Trip Recommendations**

#### Proposed System

TravelMate AI is engineered to revolutionize the way people plan their vacations, automating and personalizing the process from destination selection to itinerary creation. The system is structured around two key components:

**Frontend:** This user-friendly web interface, built with React.js, empowers travelers to effortlessly input their travel preferences, explore a diverse range of destinations and activities, and visually build their ideal itineraries.

**Backend:** A robust backend powered by Node.js and Express.js works behind the scenes to process user requests, interface with a chosen travel booking API (e.g., Amadeus, Sabre) to fetch real-time data on accommodations, hotels and more. User trip data and preferences are securely stored, enabling personalized recommendations and seamless itinerary management.

The workflow is as follows:

**Travel Preferences:** Users input their desired destinations, travel dates, budget, and any specific interests.

**Itinerary Building:** Users can easily browse through the recommendations and visually build their itinerary using a drag-and-drop interface. They can add, remove, or rearrange items, customizing the trip to their liking.

**Real-time Plan :** As the itinerary takes shape, TravelMate AI can provide real-time plan on place status, pricing and other relevant travel information, allowing users to adjust their plans if needed.

#### Fact finding & Question Answer

**Q.What is the primary goal of the PlanIt AI vacation planner website? Ans:**The primary goal of PlanIt AI is to streamline the vacation planning process by leveraging artificial intelligence to provide personalized travel recommendations and itineraries based on user preferences and data.

**2.How does the website aim to enhance the vacation planning for users? Ans:**The website enhances the planning process by offering tailored suggestions for destinations, activities, accommodations, and travel arrangements. It uses AI to analyze user preferences and provide optimized travel plans, saving users time and effort.

**3.Who is the target audience for the PlanIt AI vacation planner website?**

**Ans:**The target Ans:audience includes tech-savvy travelers, busy professionals, families, and anyone looking for a personalized and efficient way to plan their vacations.

**4.How will these technologies contribute to the functionality of the website?**

**Ans**:Front-end technologies will ensure a responsive and engaging user interface, while back-end technologies will handle data processing and integration with external services. AI frameworks will enable advanced recommendation algorithms and data analysis.

**5.How will the website ensure a seamless and intuitive user experience?**

**Ans:**The website will feature a clean, user-friendly interface with intuitive navigation and clear instructions. It will continuously improve usability and provide a seamless planning experience.

**6.What design principles will be applied to make the website user-friendly? Ans:**Design principles include simplicity, clarity, consistency, and accessibility. The website will prioritize ease of use, ensuring that users can quickly find and utilize the features they need.

#### Advantages & Disadvantages

**Advantages**:

**1. Personalized Travel Experience**

**Tailored Recommendations:** The AI analyzes user preferences, past travels, and interests to suggest destinations and activities that best match their unique needs.

**Customized Itineraries:** Users receive optimized itineraries that take into account their schedules, preferences, and budgets, making travel planning more efficient and enjoyable.

**2. Planning Platform**

Seamless integration with APIsensures access to a wide range of travel options and services.

**3. Enhanced User Experience**

Real-Time Updates: Users receive real-time notifications and updates helping them stay informed and adjust plans as needed.

**4. Advanced Technology**

AI Integration: Leveraging Gemini AI a enhances the platform's ability to provide intelligent recommendations and real-time itinerary optimization.

**5.Increased User Engagement**

Personalization through user preferences keeps users engaged with the platform, as they are presented with options that reflect their tastes and preferences.

**6.Time-Saving:**

Eliminates hours spent browsing multiple websites and comparing options. The AI does the heavy lifting, presenting tailored choices.

**7.24/7 Accessibility:**

Plan anytime, anywhere, without being restricted to travel agency hours.

**Disadvantages:**

**1.Over-reliance on data:**

AI relies heavily on past data. Unexpected events like weather changes, political instability, or pandemics can disrupt travel plans and the AI may not adapt well.

**2. Limited creativity and personalization:**

While AI can offer suggestions based on preferences, it might struggle to provide truly unique and personalized experiences that cater to individual whims and spontaneity.

**3. Technical glitches and downtimes:**

As with any website, technical issues, server downtimes, or bugs can disrupt user experience and affect travel plans.

**4. Data security and privacy concerns:**

Storing and processing user data, including travel preferences and payment information, raises concerns about data security breaches and privacy violations.

**5. Complexity** **of Real-Time Updates:**

Ensuring the accuracy of real-time updates is crucial for user trust and satisfaction, but it can be difficult to guarantee.

**6.Overwhelming choices:** The sheer volume of information and options generated by AI can be overwhelming for some users, leading to decision fatigue.

**7.Trust and transparency:** Users might be hesitant to trust AI-generated itineraries without understanding the underlying logic and decision-making process.

# Requirements

##### System & Hardware Requirement

**Hardware Requirement:**

* **Processor:** Any modern processor (i5 or equivalent) with decent processing power will suffice for development work.
* **Memory (RAM):** 8GB of RAM is recommended for a smooth development experience, especially when dealing with data visualization libraries.
* **Storage:** Enough storage space to accommodate your development environment and project files. An SSD (Solid State Drive) is preferred for faster loading times.
* **Operating System:** You can choose between Windows, macOS, or Linux based on your preference. Most popular development tools work well on all three platforms.
* **Internet Connection:** A reliable internet connection is crucial for both development and deployment.

**Software Requirement:**

**Frontend Libraries:**

* **React.js:** (version 18.x or later): Used to build the dynamic and responsive user interface.
* **Tailwind CSS:** For styling the user interface.
* **React Router:** For managing routing between pages.

**Backend Development:**

* **Node.js:** (version 14.x or later): Used for building the backend services.
* **Express.js:** A web framework used for handling API requests.
* **Gemini API:** (Specify version or SDK if applicable) Interface with the Gemini API .
* **Firebase Local Storage**: Use browser's local storage for caching small amounts of data.

**Prototype System**

**Frontend Development:**

* **JavaScript Frameworks:**
* **React.js:** (version 18.x or later) Primary framework for building a dynamic and responsive user interface. Handles UI components, state management, and user interactions.
* **UI Library:**
* **Tailwind CSS:** For styling the user interface. Provides utility-first CSS classes for rapid and customizable design.
* **Routing:**
* **React Router (for React.js):** For managing navigation and routing logic between different pages/views within your React application.
* **API Interaction:**
* **Fetch API**: Built-in browser API for making HTTP requests to backend APIs (e.g., your Node.js server or external APIs like Gemini).

**Backend Development:**

* **Runtime Environment:**
* **Node.js**: (version 14.x or later) Provides the runtime environment for executing your backend JavaScript code.
* **External API Integration:**
* **Gemini API:** (Specify version or SDK if applicable) Interface with the Gemini API to access travel-related data (flights, hotels, etc.). Ensure you have necessary API keys and authentication mechanisms in place.
* **Authentication and Authorization:**
* **Google Authentication:** Integrate Google Sign-In for secure user authentication. Leverage Firebase Authentication for managing user accounts linked to Google.
* **Data Storage and Management:**
* **Firebase:**
* **Authentication:** Use Firebase Authentication for managing user accounts and authentication flows.
* **Firestore (recommended):** NoSQL document database for storing and syncing data in real-time. Consider Firestore for travel plans, user preferences, and dynamic content.
* **Local Storage**: Use browser's local storage for caching small amounts of data and improving performance (e.g., user preferences).

**Development Tools:**

**Visual Studio Code:** A popular code editor for both backend and frontend development.

**Postman:** Useful for testing APIs and ensuring smooth data flow between frontend and backend.

**Vite:** A frontend build tool used for faster development in React.js

### System Design

##### Stakeholder

**1. Primary Users**

**Roles:**

**End Users (Travelers)**

**Interests:**

Seamless and intuitive user experience.

Personalized vacation recommendations aligned with preferences and budget.

Easy access to booking services, travel tips, and real-time information.

**2.Development and Technical Teams**

**Roles:**

Platform Developers (software engineers, designers, and developers)

Ensuring consistent API usage and maintaining a positive working relationship.

**Product Owner/Project Manager:** Responsible for overall vision, strategy, and execution of the TravelMate project.

**Development Team:**

**Frontend Developers**: Build and maintain the user interface and user experience (React, Vue, Tailwind CSS).

**Backend Developers: C**reate and manage the server-side logic, databases, and API integrations (Node.js, Express.js).

**Data Providers:**

Gemini API (and similar): Sources of travel-related data such as flight details, hotel information, attractions, and travel recommendations.

Mapping and Navigation Services: Integration with map services (e.g., Google Maps) for location-based features and route planning.

**Event Table**

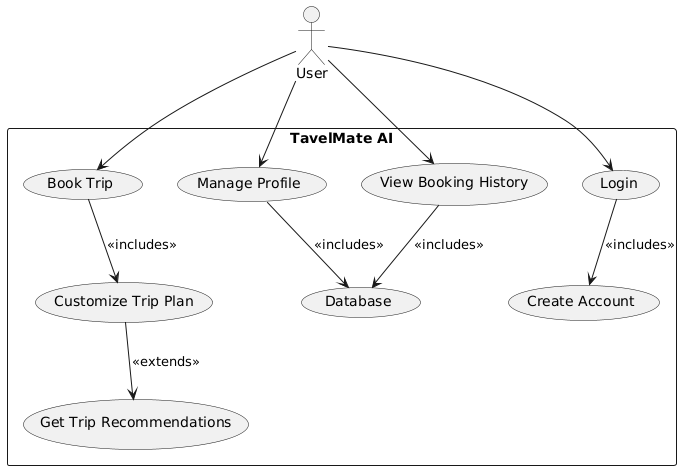
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Trigger** | **Source** | **Description** | **Response** |
| **Log in** | User acces the login | User | Authenticate user | User log in or error message |
| **User select preference** | User selects trip preferences on the Ul | User | Fetch selected trip preferences from the database | Display selected trip preferences on the Ul |
| **User books a trip** | User selects and books a recommended trip | User | Store booking information in database | User/Booking History |
| **Data aggregation refresh** | manual trigger | System | Fetch and clean new data from external sources and store in the database | Updated dataset in the database |
| **Generate trip itinerary** | User selects trip and requests itinerary | User | Compile trip details into a structured itinerary. | itinerary |
| **Trip history** | Save the info | System | database and system configurations. | Save in the database |
| **System backup** | manual backup | User | Backup database | Backup storage |
| **User logs out** | User clicks logout button | User | Terminate user session and clear session data | User is logged out |

### UNIFIED MODELING LANGUAGE DIAGRAM

**(UML Diagram)**

#### Use Case Diagram

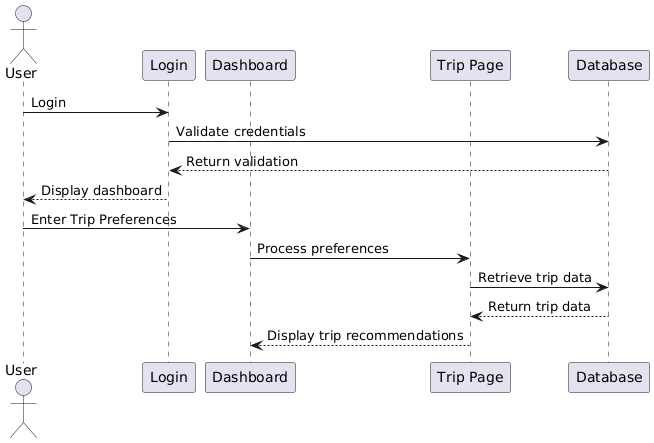
UML use case diagrams are the primary form of system/software requirements for new and undeveloped software programs. A use case specifies the expected behavior (what) rather than the exact way (how) to achieve it. Once a use case is specified, it can be referenced in both textual and visual representations (use case diagrams). A key concept of use case modeling is that it helps you design your system from the end user's perspective. This is an effective technique for communicating system behavior in user terms by specifying all externally visible system behavior.



#### Sequence Diagram

Sequence diagrams are a popular dynamic modeling solution in UML. This is because it focuses specifically on lifelines, or processes and objects that coexist at the same time, and the messages exchanged between them to perform a function before the lifeline terminates. Use this guide along with your UML diagramming tool to learn everything you need to know about UML sequence diagrams.

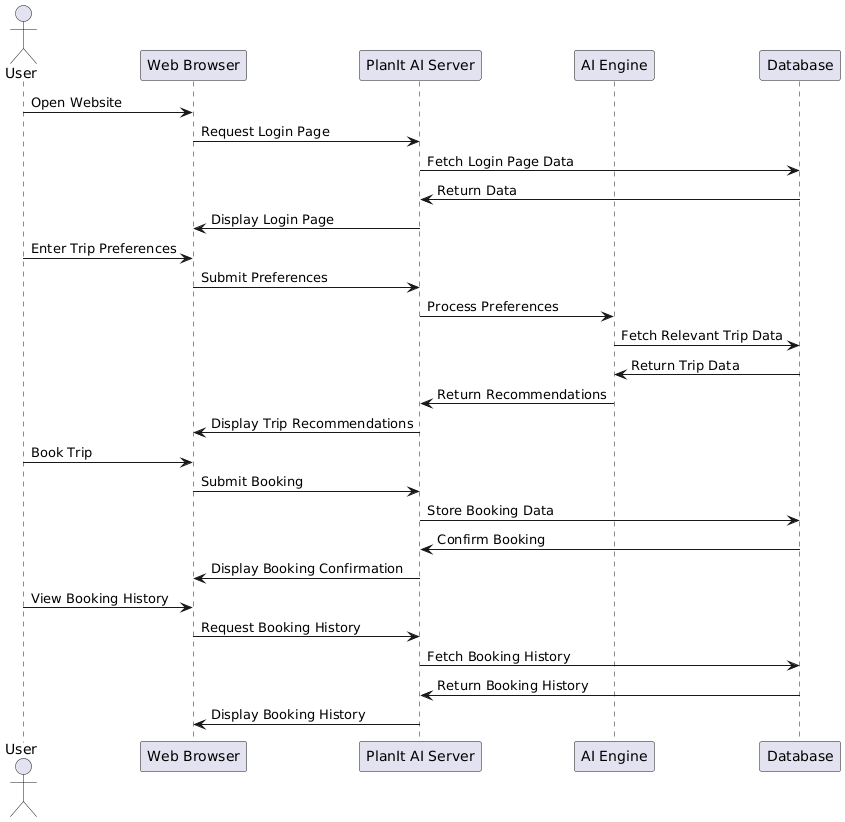
A sequence diagram is a type of interaction diagram because it describes how a group of objects work together and in what order. These diagrams are used by software developers and business professionals to understand new system requirements and to document existing processes. Sequence diagrams are sometimes called event diagrams or event scenarios.



#### System Sequence Diagram

A system diagram is a simple and very general description of an existing system or a system that needs to be built. It is a simple diagram that can be created in a short time. It helps teams to share and understand their systems clearly and completely.

The main enemy of system architects and software engineers is complexity. Complexity leads to a lack of understanding of the system and can make it very difficult to modify or use the system. The complexity of the system is reflected in its description and documentation. The task of explaining a system to a non-technical audience can be a daunting task. A system diagram provides an easy way to solve this problem.

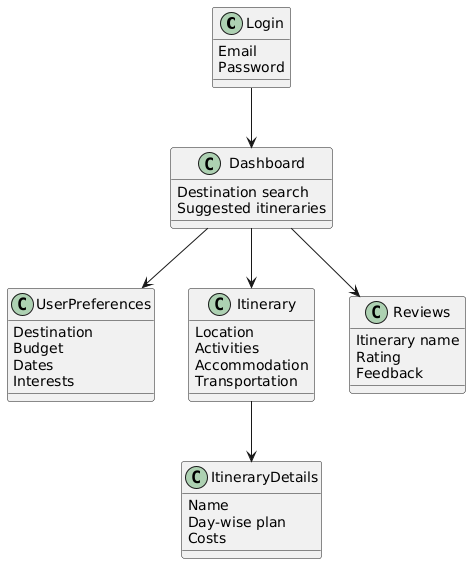


##### Class Diagram

A class is a blueprint for an object. Objects and classes are closely related. You can't talk about the other without talking about the other. Also, object-oriented design is about classes, not objects. Because you use classes to create objects. So a class describes what an object will be, but it's not the object itself.

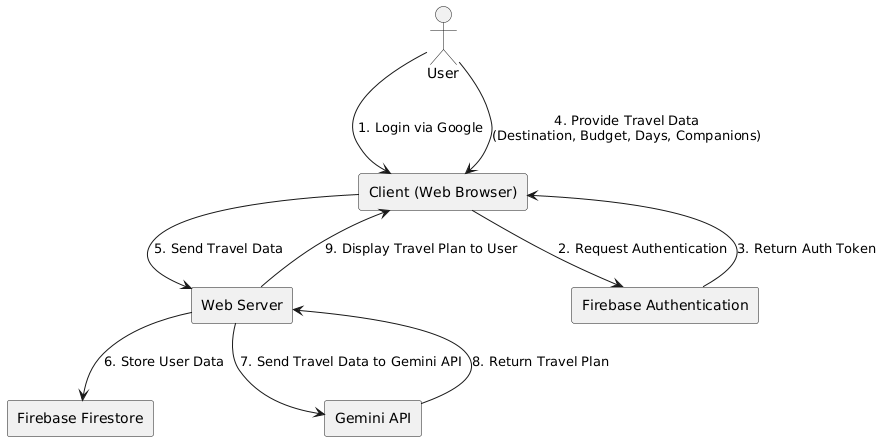
A class describes a type of object, but an object is a usable instance of a class. Each object is created from the same blueprint and therefore contains the same components (properties and methods). In the traditional sense, objects are instances of classes and objects. Objects have state and behavior.

-



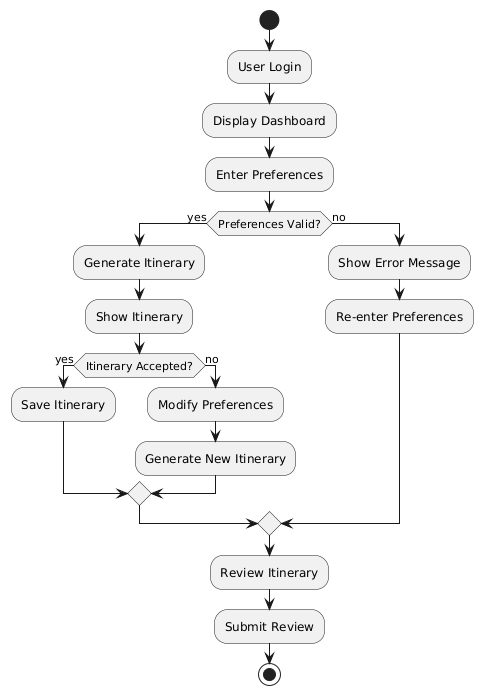
#### Collaboration Diagram

Collaboration diagrams are used to show the relationships between objects in the system. Both sequence diagrams and collaboration diagrams show the same information, but they are different. Based on object-oriented programming, it shows the architecture of the objects that reside in the system rather than showing message flow. An object consists of several properties. Several objects in the system are connected to each other. A collaboration diagram, also called a communication diagram, serves to represent the architecture of the objects in the system.



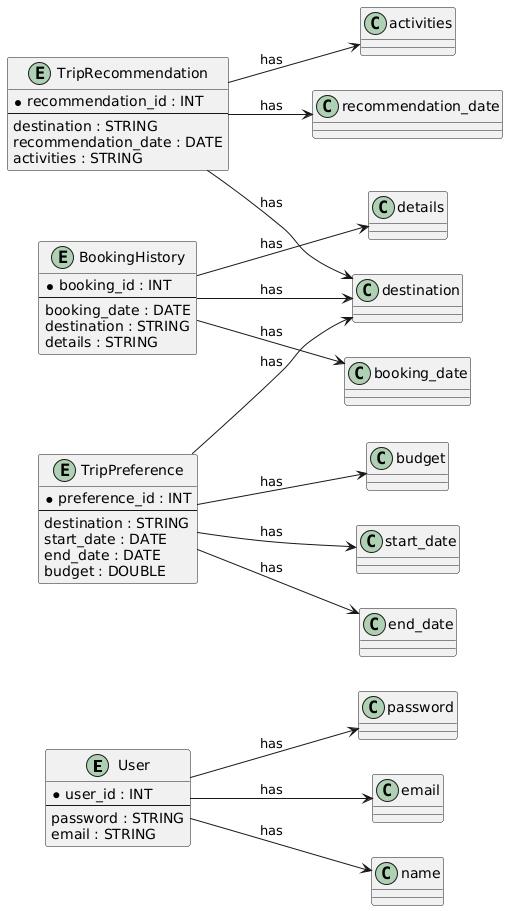
#### Flow chart Diagram

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

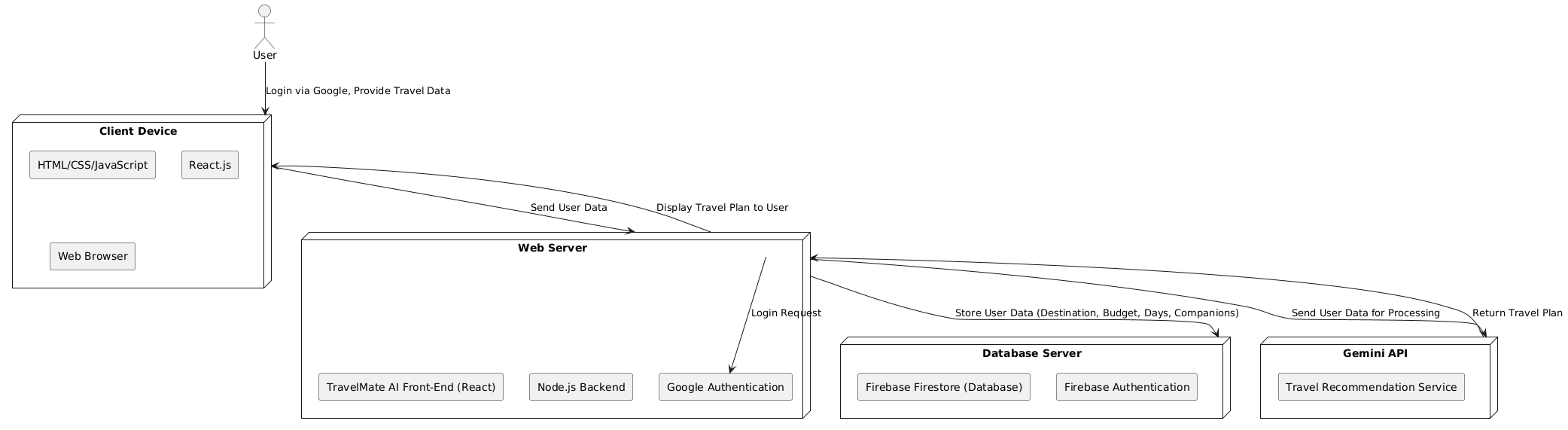


##### ER Diagram

An entity relationship diagram (ER diagram or ERD) is a pictorial or visual representation of a taxonomy of groups or entities of common interest and a definition of the relationships between these groups.



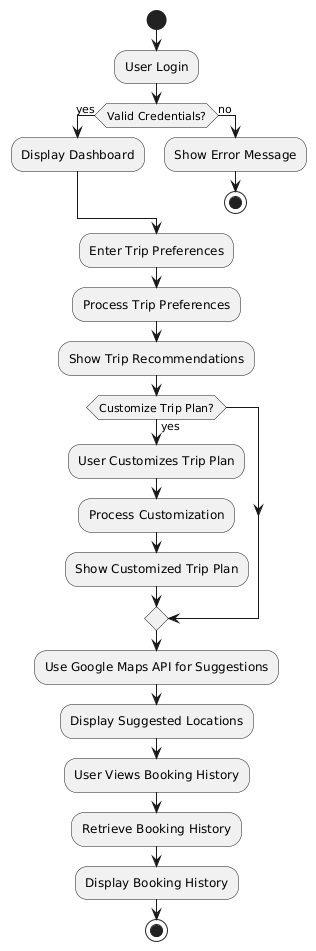
**Deployment Diagram**



**Activity diagram**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent

.



## System Coding & Implementation

#### Modules

A module is a collection of source files and build settings that allow you to divide your project into discrete units of functionality. Your project can have one or many modules, and one module may use another module as a dependency. We can independently build, test, and debug each module.

**Frontend:**

* **React.js:** For building a responsive and dynamic user interface.
* **Redux:** For efficient state management.
* **Material-UI**: Provides a consistent and modern design.
* **Axios:** To handle API requests.

**Backend:**

* **Runtime Environment:**
* **Node.js**: (version 14.x or later) Provides the runtime environment for executing your backend JavaScript code.
* **External API Integration:**
* **Gemini API:** (Specify version or SDK if applicable) Interface with the Gemini API to access travel-related data (flights, hotels, etc.). Ensure you have necessary API keys and authentication mechanisms in place.
* **Authentication and Authorization:**
* **Google Authentication:** Integrate Google Sign-In for secure user authentication. Leverage Firebase Authentication for managing user accounts linked to Google.
* **Data Storage and Management:**
* **Firebase:**
* **Authentication:** Use Firebase Authentication for managing user accounts and authentication flows.
* **Firestore (recommended):** NoSQL document database for storing and syncing data in real-time. Consider Firestore for travel plans, user preferences, and dynamic content.
* **Local Storage**: Use browser's local storage for caching small amounts of data and improving performance (e.g., user preferences).

**Development Tools:**

**• Visual Studio Code:** A popular code editor for both backend and frontend

development.

**• Postman:** Useful for testing APIs and ensuring smooth data flow between

frontend and backend.

**• Vite:** A frontend build tool used for faster development in React.js

###### Test Case

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Objective** | **Precondition** | **Step Description** | **Input Data** | **Expected Result** | **Actual Result** | **Status** | **Remark** |
| TC001 | Verify Google Login functionality | User is on the login page | Click on "Login with Google" | User clicks the login button | User is redirected to Google authentication page | same | pass | Success |
| TC002 | Verify successful login with valid credentials | User is logged into Google | Enter valid credentials and click "Continue" | Valid email and password | User is redirected to the travel data input page | same | pass | Success |
| TC003 | Verify input validation for travel data | User is logged in and on the travel data page | Fill in the travel data form with invalid data | Invalid destination, budget (negative) | Error message is displayed for invalid input | same | pass | Success |
| TC004 | Verify data submission to Firestore | User has filled out the travel data form | Click on "Submit" | Valid destination, budget, days | Data is stored in Firebase Firestore successfully | same | pass | Success |
| TC005 | Verify interaction with the Gemini API | User has submitted travel data | System sends data to the Gemini API | Valid travel data | API returns a travel plan | same | pass | Success |
| TC006 | Verify display of travel plan to the user | User has received a response from the API | Check the travel plan output page | N/A | Travel plan is displayed correctly | same | pass | Success |
| TC007 | Verify user logout functionality | User is logged in | Click on "Logout" | User clicks the logout button | User is logged out and redirected to the login page | same | pass | Success |
| TC008 | Verify error handling for API failure | User is logged in and has submitted data | Simulate Gemini API failure | N/A | Error message is displayed for API failure | saem | pass | Success |
| TC009 | Verify data retrieval from Firestore | User is logged in and has submitted data | Retrieve the user's travel data from Firestore | User's ID | Correct travel data is retrieved | same | pass | Success |
| TC010 | Verify session management | User is logged in | Leave the page idle for a set time | N/A | User session expires after a defined timeout | same | pass | Success |

**Codes**

**Homepage.jsx :**

import React, { useEffect, useState } from "react";

import { debounce } from "lodash";

import { Button } from "../ui/button";

import { toast } from "sonner";

import { chatSession } from "@/services/AImodel";

import { useNavigate } from "react-router-dom";

import {

Card,

CardContent,

CardDescription,

CardFooter,

CardHeader,

CardTitle,

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

AI\_PROMPT,

SelectBudgetOptions,

SelectTravelerList,

} from "@/constants/options";

import { doc, setDoc } from "firebase/firestore";

import { db } from "@/services/fireBaseConfig";

import { AiOutlineLoading3Quarters } from "react-icons/ai";

export const HomePage = () => {

const [placeSuggestions, setPlaceSuggestions] = useState([]);

const [selectedPlace, setSelectedPlace] = useState("");

const [formData, setFormData] = useState({

noOfDays: "",

budget: "",

people: "",

location: "",

});

const [openDialog, setOpenDialog] = useState(false);

const [loading, setLoading] = useState(false);

// Fetch places from RapidAPI

const fetchPlaceSuggestions = async (query) => {

if (!query) return;

try {

const response = await fetch(

`https://google-map-places.p.rapidapi.com/maps/api/place/autocomplete/json?input=${query}&radius=50000&strictbounds=true&offset=3&location=40,-110&origin=40,-110&components=country:us&language=en®ion=en`,

{

method: "GET",

headers: {

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

"x-rapidapi-host": "google-map-places.p.rapidapi.com",

"x-rapidapi-key": "29d8a67e4emsh4383f68fa8f089cp10a24djsn9e0bd0ab7d23", // Replace with your actual RapidAPI key

},

}

);

const data = await response.json();

console.log("Fetched place suggestions:", data); // Check the response

setPlaceSuggestions(data.predictions || []);

} catch (error) {

console.error("Error fetching place suggestions:", error);

setPlaceSuggestions([]);

}

};

const fetchPlaceSuggestionsDebounced = debounce(fetchPlaceSuggestions, 300);

const handleInputChanges = (name, value) => {

setFormData({

...formData,

[name]: value,

});

fetchPlaceSuggestionsDebounced(value); // Call the debounced function

};

const onGenerateTrip = async () => {

const user = localStorage.getItem("user");

if (!user) {

setOpenDialog(true);

return;

}

if (

formData.noOfDays > 15 ||

formData.noOfDays < 1 ||

!formData.location ||

!formData.budget ||

!formData.people

) {

toast("Please fill all details!");

return;

}

setLoading(true); // Indicate the process is starting

toast(

"Patience is a virtue, and awesome things take time. We'll get you there soon!"

);

try {

const FINAL\_PROMPT = AI\_PROMPT.replace(

"{noOfDays}",

formData.noOfDays

)

.replace("{people}", formData.people)

.replace("{location}", formData.location)

.replace("{budget}", formData.budget);

const result = await chatSession.sendMessage(FINAL\_PROMPT);

const responseText = await result?.response?.text();

const responseJSON = JSON.parse(responseText);

saveAiTrip(responseJSON);

} catch (error) {

console.error("Error during API call:", error);

toast("Something went wrong. Please try again later.");

} finally {

setLoading(false); // Set loading to false after the API call completes

}

};

const navigate = useNavigate();

const saveAiTrip = async (TripData) => {

setLoading(true);

const user = JSON.parse(localStorage.getItem("user"));

const docId = Date.now().toString();

await setDoc(doc(db, "AItrip", docId), {

userSelection: formData,

tripData: TripData,

userEmail: user?.email,

id: docId,

});

setLoading(false);

navigate("/view-trip/" + docId);

};

// Handle place selection

const handlePlaceSelection = (place) => {

setSelectedPlace(place.description);

handleInputChanges("location", place.description);

setPlaceSuggestions([]);

};

return (

<>

<div className="w-full overflow-hidden px-3 md:px-14 lg:px-14 xl:px-40 font-serif">

<Card className="mt-6 border-y-4 p-6">

<CardHeader>

<CardTitle className="pt-5 text-left text-lg md:text-2xl lg:text-3xl font-bold tracking-wider md:tracking-widest">

Please share your travel preferences with us🏕️🌴

</CardTitle>

<CardDescription className="pt-5 pb-3 text-justify md:text-left font-light text-sm md:text-lg lg:text-xl tracking-tighter md:tracking-widest">

Simply provide some basic information, and our trip planner will create a personalized itinerary tailored to your preferences.

</CardDescription>

</CardHeader>

<CardContent>

<form>

<div className="grid w-full items-center">

<div className="flex flex-col space-y-10">

{/\* Preferred Destination \*/}

<div className="space-y-2">

<Label htmlFor="location" className="text-base md:text-lg">

What is your preferred destination?

</Label>

<div className="relative">

<Input

id="location"

type="text"

placeholder="Type to search..."

value={selectedPlace}

onChange={(e) => {

setSelectedPlace(e.target.value);

fetchPlaceSuggestions(e.target.value);

}}

className="border-2 dark:border-customGreen border-blue-700"

/>

{placeSuggestions.length > 0 && (

<div className="absolute z-10 bg-black border border-gray-300 mt-2 max-h-60 overflow-auto w-full">

{placeSuggestions.map((suggestion, index) => (

<div

key={index}

onClick={() => handlePlaceSelection(suggestion)}

className="p-2 hover:bg-gray-100 cursor-pointer"

>

{suggestion.description} {/\* Display description \*/}

</div>

))}

</div>

)}

</div>

</div>

{/\* How Many Days \*/}

<div>

<Label htmlFor="days" className="text-base md:text-lg">

How many days do you plan to spend on your trip?

</Label>

<Input

id="days"

type="number"

placeholder="ex. 3"

min="1"

max="15"

value={formData.noOfDays}

onChange={(e) =>

handleInputChanges("noOfDays", e.target.value)

}

className="border-2 dark:border-customGreen border-blue-700 bg-white text-slate-800"

/>

</div>

{/\* Budget \*/}

<div>

<Label htmlFor="budget" className="text-base md:text-lg">

What's your spending limit?

</Label>

<div className="grid grid-cols-1 md:grid-cols-3 mt-5 cursor-pointer text-sm md:text-base lg:text-base items-center text-center">

{SelectBudgetOptions.map((item, index) => (

<div

key={index}

onClick={() => handleInputChanges("budget", item.title)}

className={`p-1 m-1 md:p-2 md:m-1 border-2 rounded-lg mb-3 hover:shadow-lg dark:border-customGreen border-blue-700 dark:hover:shadow-customGreen hover:shadow-blue-700 ${

formData.budget === item.title &&

`shadow-lg border-2 dark:shadow-customGreen shadow-blue-700`

}`}

>

<h2 className="font-bold text-md">

<span className="text-lg">{item.icon}</span>{" "}

{item.title}

</h2>

<h2 className="text-gray-600 dark:text-gray-400">

{item.des}

</h2>

</div>

))}

</div>

</div>

{/\* Who will you travel with? \*/}

<div>

<Label htmlFor="noOfPeople" className="text-base md:text-lg">

Who are you planning to travel with on your next adventure?

</Label>

<div className="grid grid-cols-2 mt-5 cursor-pointer text-[14px] md:text-base lg:text-base items-center text-center">

{SelectTravelerList.map((item, index) => (

<div

key={index}

onClick={() =>

handleInputChanges("people", item.people)

}

className={`p-1 m-1 md:p-4 md:m-3 border-2 mb-2 rounded-lg dark:border-customGreen border-blue-700 hover:shadow-lg dark:hover:shadow-customGreen hover:shadow-blue-700 ${

formData.people === item.people &&

`shadow-lg border-2 dark:shadow-customGreen shadow-blue-700`

}`}

>

<h2 className="font-bold">

<span className="text-xl">{item.icon}</span>{" "}

{item.title}

</h2>

<h2 className="text-gray-600 dark:text-gray-400">

{item.des}

</h2>

</div>

))}

</div>

</div>

</div>

</div>

</form>

</CardContent>

<CardFooter className="flex justify-end mt-5">

<Button

disabled={loading}

onClick={onGenerateTrip}

className="bg-blue-700 hover:bg-indigo-700 dark:text-white hover:shadow-xl transition ease-in-out delay-150 hover:-translate-y-1 hover:scale-100 md:text-lg"

>

{loading ? (

<AiOutlineLoading3Quarters className="h-7 w-7 animate-spin" />

) : (

"Generate Trip"

)}

</Button>

</CardFooter>

</Card>

</div>

</>

);

};

**LandingPage.jsx**

import { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import { AuthDialog } from "../AuthDialog/AuthDialog";

import { useGoogleAuth } from "@/services/Auth";

export default function LandingPage() {

const [openDialog, setOpenDialog] = useState(false);

const [loading, setLoading] = useState(false);

const navigate = useNavigate();

const login = useGoogleAuth(() => {

setOpenDialog(false);

navigate("/create-trip");

});

useEffect(() => {

const user = JSON.parse(localStorage.getItem("user"));

if (user) {

navigate("/create-trip");

}

}, [navigate]);

return (

<>

<div className="container flex flex-col items-center gap-9 -mt-14 md:mt-0">

<h1 className="container font-mono font-bold text-3xl md:text-4xl lg:text-[60px] p-1 text-center mt-16 ">

<span className="text-blue-700 dark:text-customGreen leading-tight">

Explore Your Next Adventure with AI:

</span>

<span className="leading-tight">

{" "}

Customized Itineraries at Your Fingertips

</span>

</h1>

<p className="text-sm md:text-lg lg:text-xl text-center p-1 text-gray-700 dark:text-gray-400">

Your dedicated travel companion, crafting personalized

journeys that match your passions and your pocket.

<img

className="xs:h-44 sm:h-52 md:h-60 lg:h-72 mx-auto mt-8 "

src="/landing.png"

/>

</p>

<button

onClick={() => {

const user = JSON.parse(localStorage.getItem("user"));

if (user) {

navigate("/create-trip");

} else {

setOpenDialog(true);

}

}}

className="font-sans font-medium transition ease-in-out delay-150 bg-blue-700

hover:-translate-y-1 hover:scale-110 hover:bg-indigo-600 text-white

duration-300 rounded-lg text-sm lg:text-lg -mt-5

px-4 items-center py-3 lg:py-3 ... focus:outline-none "

>

Get Started, It's Free!

</button>

</div>

<AuthDialog

open={openDialog}

loading={loading}

onLogin={login}

onClose={() => setOpenDialog(false)}

/>

</>

);

}

**Main.jsx**

iimport React from "react";

import ReactDOM from "react-dom/client";

import "./index.css";

import {

createBrowserRouter,

createRoutesFromElements,

Route,

RouterProvider,

} from "react-router-dom";

import Layout from "./Layout.jsx";

import LandingPage from "./components/LandingPage/LandingPage.jsx";

import { HomePage } from "./components/HomePage/HomePage";

import { GoogleOAuthProvider } from "@react-oauth/google";

import { ViewTrip } from "./viewTrip/ViewTrip";

import { MyTrips } from "./components/MyTrips/MyTrips";

import { AuthProvider } from "./Context/AuthContext";

import ProtectedRoute from "./Context/ProtectedRoute";

const router = createBrowserRouter(

createRoutesFromElements(

<Route path="/" element={<Layout />}>

<Route path="" element={<LandingPage />} />

<Route

path="/create-trip"

element={

<ProtectedRoute>

<HomePage />

</ProtectedRoute>

}

/>

<Route

path="/my-trips"

element={

<ProtectedRoute>

<MyTrips />

</ProtectedRoute>

}

/>

<Route

path="/view-trip/:tripId"

element={

<ProtectedRoute>

<ViewTrip />

</ProtectedRoute>

}

/>

</Route>

)

);

ReactDOM.createRoot(document.getElementById("root")).render(

<React.StrictMode>

<GoogleOAuthProvider

clientId='589136722646-s9m15id8pdilo7sl954toemaq9gmh0qh.apps.googleusercontent.com'

>

<AuthProvider>

<RouterProvider router={router} />

</AuthProvider>

</GoogleOAuthProvider>

</React.StrictMode>

);

**ViewTrip.jsx**

import { db } from "@/services/fireBaseConfig";

import { doc, getDoc } from "firebase/firestore";

import { useEffect, useState } from "react";

import { useParams } from "react-router-dom";

import { toast } from "sonner";

import { InfoSection } from "../components/Trip/InfoSection";

import { Hotels } from "../components/Trip/Hotels";

import { Itinerary } from "../components/Trip/Itinerary";

import { Card } from "@/components/ui/card";

export const ViewTrip = () => {

const { tripId } = useParams();

useEffect(() => {

if (tripId) {

getTripData();

}

}, [tripId]);

const [trip, setTrip] = useState([]);

const getTripData = async () => {

const docRef = doc(db, "AItrip", tripId);

const docSnap = await getDoc(docRef);

if (docSnap.exists()) {

setTrip(docSnap.data());

} else {

toast("No trip found");

}

};

return (

<Card className="container my-4 border-x-2 p-5">

<InfoSection trip={trip} />

<Hotels trip={trip} />

<Itinerary trip={trip} />

</Card>

);

};

**MyTrip.jsx**

import { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import { collection, query, where, getDocs } from "firebase/firestore";

import { db } from "@/services/fireBaseConfig";

import MyTripsCard from "./MyTripsCard";

import { Card } from "../ui/card";

export const MyTrips = () => {

const navigate = useNavigate();

const [userTrips, setUserTrips] = useState([]);

useEffect(() => {

const fetchUserTrips = async () => {

const user = JSON.parse(localStorage.getItem("user"));

if (!user) {

navigate("/create-trip");

return;

}

try {

const q = query(

collection(db, "AItrip"),

where("userEmail", "==", user.email)

);

const querySnapshot = await getDocs(q);

const trips = [];

querySnapshot.forEach((doc) => {

//console.log(doc.id, " => ", doc.data());

trips.push(doc.data()); // Accumulate trips

});

setUserTrips(trips); // Set all trips at once

} catch (error) {

console.error("Error fetching user trips: ", error);

}

};

fetchUserTrips();

}, [navigate]);

return (

<div className="container font-serif ">

<Card className="mt-5 border-x-2 p-2">

<h1 className="font-bold text-xl sm:text-2xl md:text-4xl lg:text-5xl mt-6 md:mt-10 lg:mt-10 mb-5 dark:text-customGreen text-blue-700">

My Trips 🏕️🌴

</h1>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-2 xl:grid-cols-2 text-justify gap-3 md:gap-6 xl:gap-6">

{userTrips?.length > 0

? userTrips.map((trip, index) => (

<MyTripsCard key={index} trip={trip} /> // Properly render each card with a key

))

: [1, 2, 3, 4, 5].map((item, index) => (

<div

key={index}

className="h-56 w-full bg-slate-300 dark:bg-slate-800 animate-pulse rounded-2xl p-2"

></div>

))}

</div>

</Card>

</div>

);

};

**AuthDialog.jsx**

import {

Dialog,

DialogContent,

DialogDescription,

DialogFooter,

DialogHeader,

} from "@/components/ui/dialog";

import { Button } from "@/components/ui/button";

import { FcGoogle } from "react-icons/fc";

export const AuthDialog = ({ open, loading, onLogin, onClose }) => {

return (

<Dialog open={open} onOpenChange={onClose}>

<DialogContent className="w-max rounded-2xl">

<DialogHeader>

<DialogDescription>

<img

className="w-44 sm:w-52 md:w-64 lg:w-72 xl:w-80 rounded-lg"

src="/travelMate.png"

/>

<h1 className="mx-auto mt-2 md:mt-4 text-left dark:text-green-400 text-green-600 font-semibold text-base md:text-2xl">

Sign In with Google

</h1>

<p className="text-slate-600 dark:text-slate-300 text-[0.6rem] sm:text-[0.7rem] md:text-base">

Sign in to the App with Google authentication

securely

</p>

</DialogDescription>

</DialogHeader>

<DialogFooter>

<Button

disabled={loading}

onClick={onLogin}

className="w-full pt-1 gap2 items-center text-base"

>

<FcGoogle className="h-7 w-7" /> Sign in with Google

</Button>

</DialogFooter>

</DialogContent>

</Dialog>

);

};

**eslint.config.js**

module.exports = {

root: true,

env: { browser: true, es2020: true },

extends: [

'eslint:recommended',

'plugin:react/recommended',

'plugin:react/jsx-runtime',

'plugin:react-hooks/recommended',

],

ignorePatterns: ['dist', '.eslintrc.cjs'],

parserOptions: { ecmaVersion: 'latest', sourceType: 'module' },

settings: { react: { version: '18.2' } },

plugins: ['react-refresh'],

rules: {

'react/jsx-no-target-blank': 'off',

'react-refresh/only-export-components': [

'warn',

{ allowConstantExport: true },

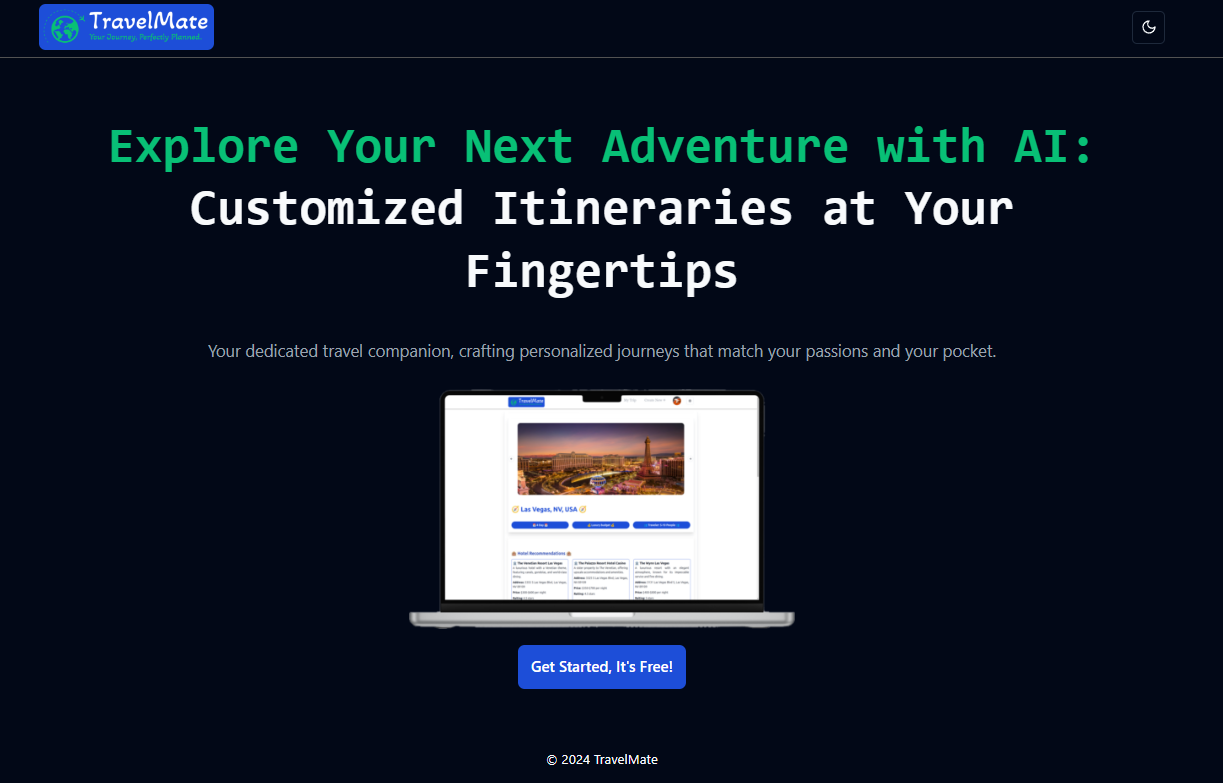
],

},

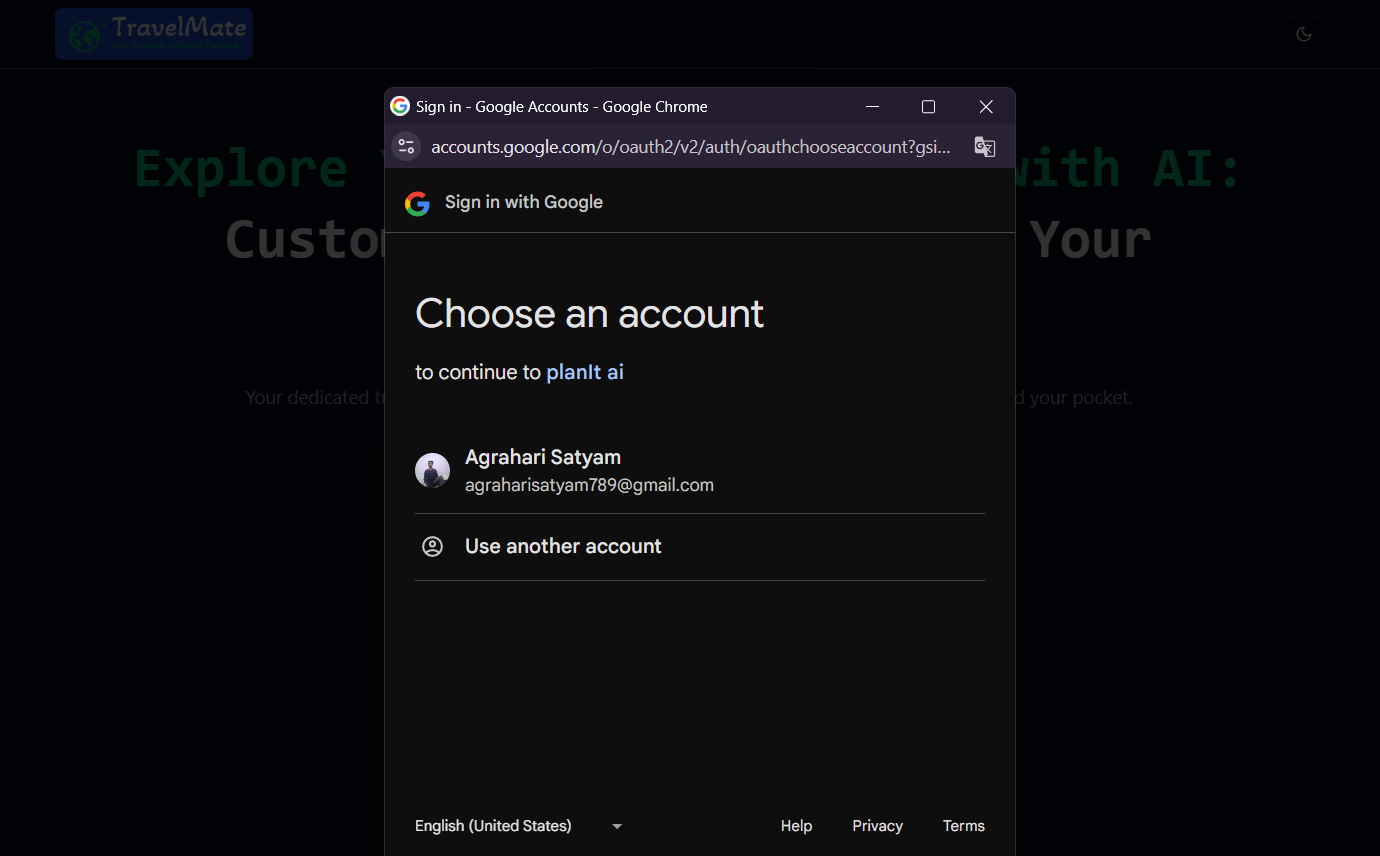
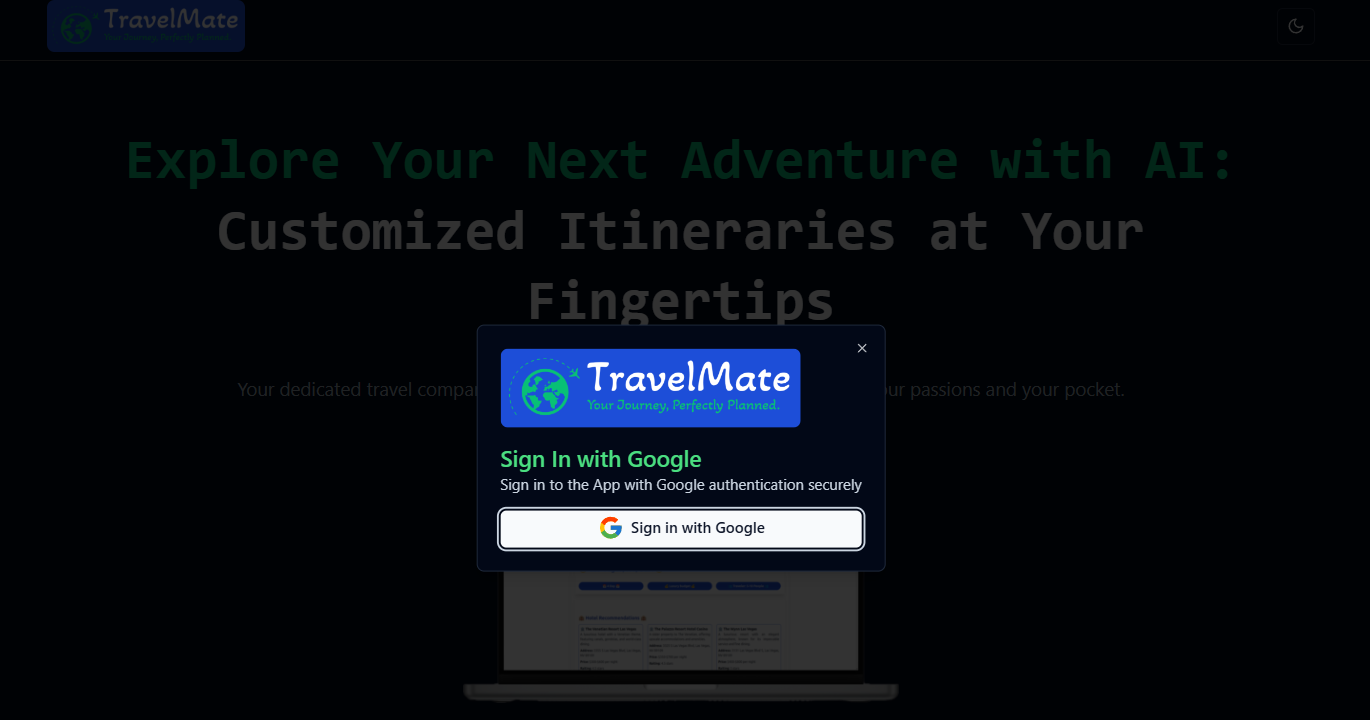
}

## Screenshots and Report Layout

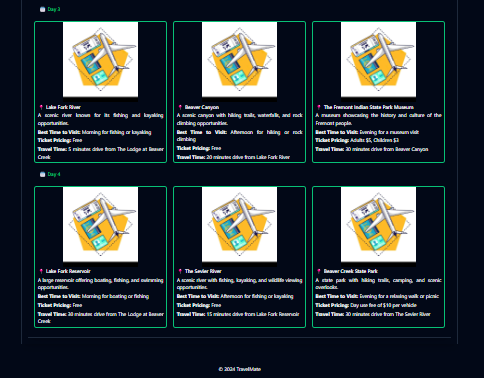
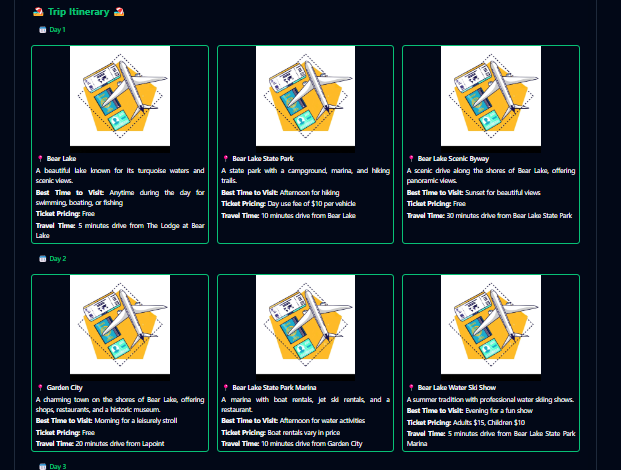
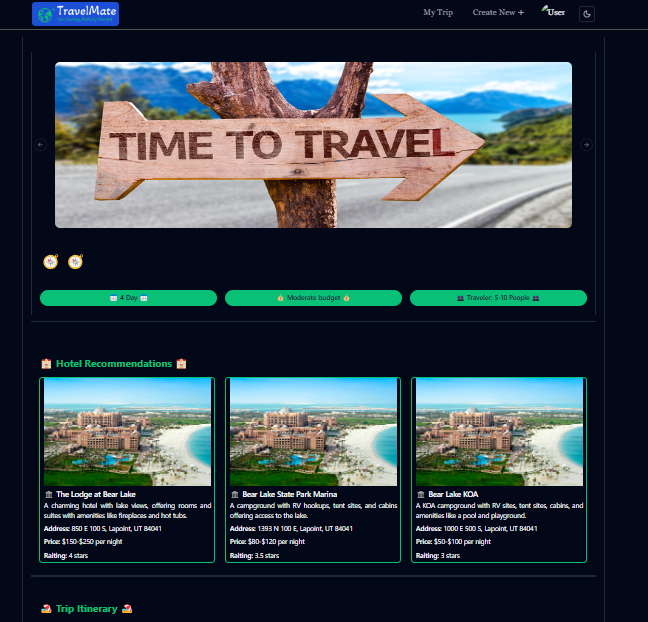
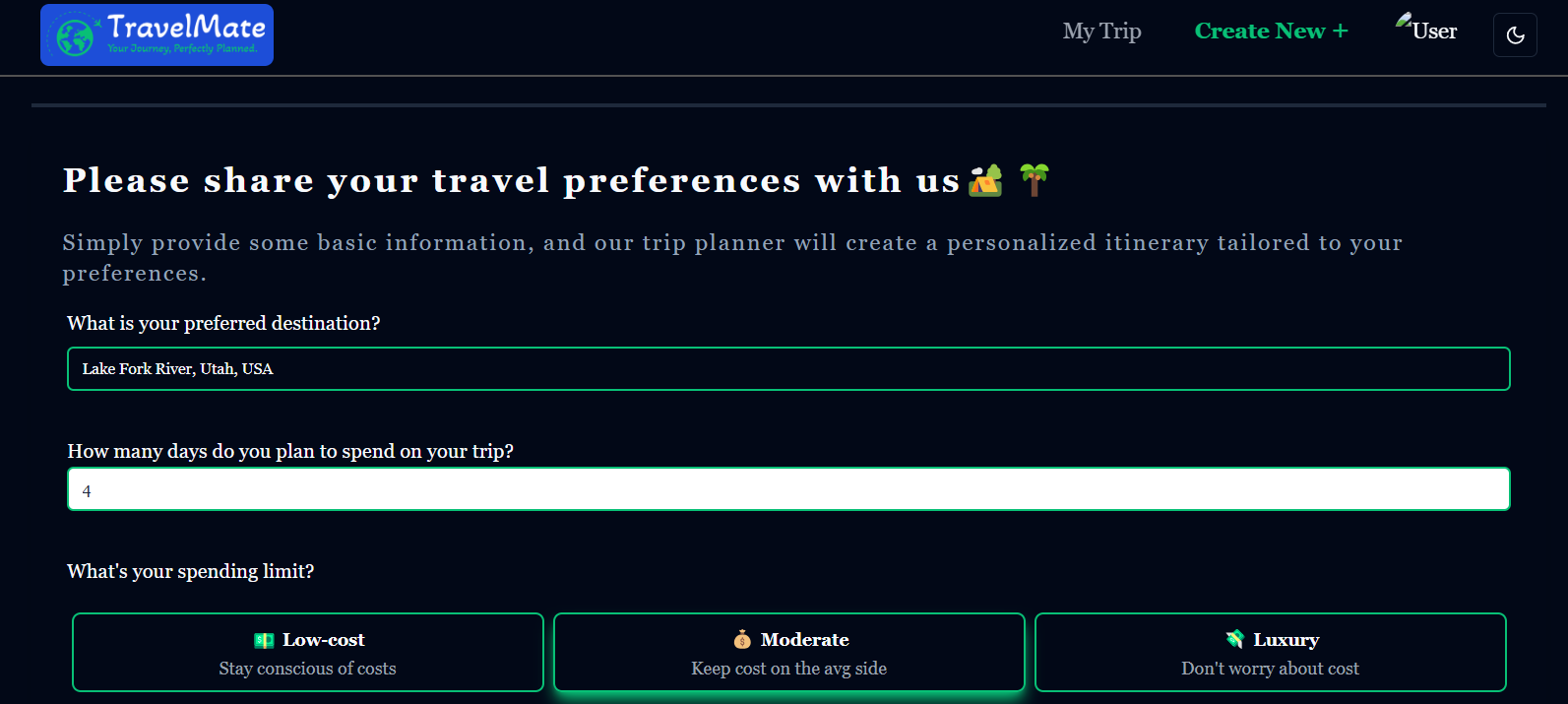
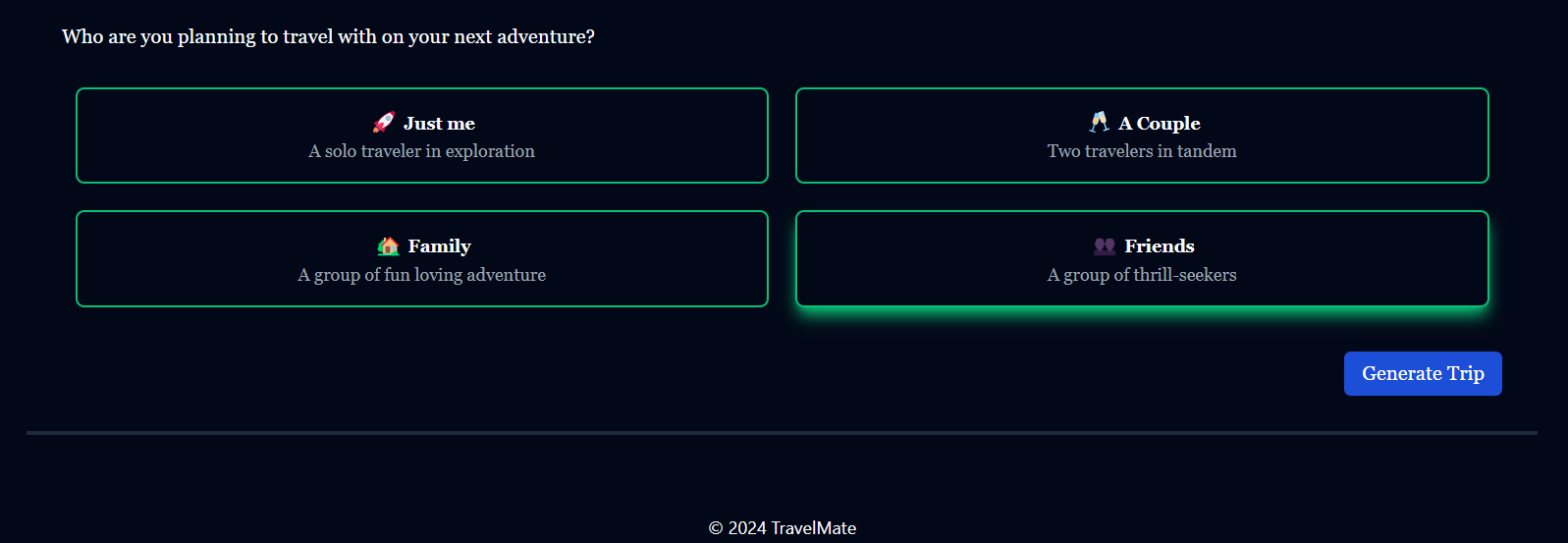
**Homepage**



**Authentication login**



**Trip Details**



#### Conclusion

The "TravelMate AI" project showcases the ability to simplify and personalize the process of travel planning through automation and smart algorithms. By integrating modern technologies such as Google Authentication, Firebase for data storage, and the Gemini API for personalized recommendations, the platform provides users with a streamlined experience for organizing their trips.

Currently, the system collects essential travel data such as destination, budget, number of days, and travel companions, and generates customized travel plans. the project has the potential to evolve with the inclusion of advanced AI-driven features, making it easier for users to plan vacations without the complexity of manual research and decision-making.

**Future Enhancements**

**Future Enhancements**

**Multi-Destination Planning**: Expanding the platform to allow users to plan multi-destination trips. This feature would let travelers design complex itineraries involving several locations, offering route optimization and cost-efficient travel suggestions.

**Social Integration**: Introducing a social aspect where users can share their travel plans with friends, receive recommendations from their network, or even collaborate on trip planning in real-time.

**Offline Mode:** Implementing an offline mode that allows users to access their travel itineraries and essential details without internet connectivity. This would be especially helpful for travelers in remote areas or places with limited internet access.

**Language Support for International Users:** Extending the platform to support multiple languages, ensuring that users from different regions can plan their trips seamlessly without language barriers.

**Customizable Itinerary Templates**: Providing users with a library of customizable travel itinerary templates based on popular destinations, themes (adventure, relaxation, culture), or trip length, making it easier for users to start planning without starting from scratch.

**User Reviews and Ratings:** Adding a feature where users can read and leave reviews and ratings for destinations, accommodations, and activities, helping future users make more informed decisions based on community feedback.

**Realtime image**: To provide real time info and the image of real time places.

##### References & Bibliography

**Google Links:**

* + 1. "Firebase Documentation," https://firebase.google.com/docs
    2. "Google Authentication Documentation," https://developers.google.com/identity
    3. "React.js Documentation," [https://reactjs.org](https://reactjs.org/)
    4. "Vite Documentation," [https://vitejs.dev](https://vitejs.dev/)
    5. "Gemini API Documentation," https://gemini.com/developers
    6. "Material UI Documentation," [https://mui.com](https://mui.com/)

**Books:**

* + 1. "Cloud Computing for Dummies" by Judith S. Hurwitz, published by Wiley.
    2. "The Road to React: Your Journey to Master Plain yet Pragmatic React.js" by Robin Wieruch,

published by Independently Published.

* + 1. "Firebase Essentials: Firebase Cloud Functions and Realtime Database" by Ashraff Hathibelagal, published by Packt Publishing.

**Videos:**

* + 1. "Firebase Tutorial for Beginners," <https://www.youtube.com/watch?v=9kRgVxULbag>
    2. "React Firebase Google Authentication," <https://www.youtube.com/watch?v=PKwu15ldZ7k>
    3. "Full Stack Travel Planner Website with React and Firebase," <https://www.youtube.com/watch?v=GDa8kZLNhJ4>
    4. "How to Use the Gemini API," <https://www.youtube.com/watch?v=rAxeTBlgKc>