NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

A

MINI PROJECT REPORT

ON

"AI Powered Trip Planner"

Submitted in the partial fulfillment of the requirements in the 5th semester of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

BY

Cherith Reddy - 1NH23CD037

Lakshmi Srikanth-1NH23CD064

Under the guidance of

Assistant Professor

Mrs. Revathi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATASCIENCE)

NEW HORIZON COLLEGE OF ENGINEERING

(Autonomous College Permanently Affiliated to VTU, Approved by AICTE, Accredited by NBA & NAAC with 'A' Grade)

Ring Road, Bellandur Post, Near Marathalli,

Bangalore-560103, INDIA

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

CERTIFICATE

We hereby certify that, the report entitle Local Business Support Application as a part of Mini Project Component in partial fulfilment of the requirements during 5th semester Bachelor of Engineering in Computer Science and Engineering (Data Science) during the year 2025- 2026(September2025-August 2026) is an authentic record of our work carried out by Cherith Reddy(1NH23CD037) and Lakshmi Srikanth(1NH23CD064), a bonafide students of NEW HORIZON COLLEGE OF ENGINEERING.

Name & Signature of Student
(Cherith Reddy)

Name & Signature of Student
(Lakshmi Srikanth)

Name & Signature of Guide
Mrs. Revathi

Name & signature of HOD
(Dr. Swathi B)

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project. I would like to take an opportunity to thank them all.

First and foremost I thank the management, **Dr. Mohan Manghnani,** Chairman, New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I would like to thank **Dr. Manjunatha**, Principal, New Horizon College of Engineering, Bengaluru, for his constant encouragement and facilities extended to us towards completing my project work.

I extend my sincere gratitude to **Dr. Swathi B**, Associate Professor & Head of the Department, Computer Science and Engineering (Data Science), New Horizon College of Engineering, Bengaluru for her valuable suggestions and expert advice.

I deeply express my sincere gratitude to my guide**, Mrs. Revathi ,** Assistant Professor, Computer Science and Engineering (Data Science), New Horizon College of Engineering, Bengaluru, for his/her able guidance, regular source of encouragement and assistance throughout this project.

I thank my Parents, and all Faculty members of Department of Computer Science and Engineering (Data Science) for their constant support and encouragement.

Last, but not the least, I would like to thank my peers and friends who provided me with valuable suggestions to improve my project.

Cherith Reddy-1NH23CD037
Lakshmi Srikanth- 1NH23CD064

ABSTRACT

The **AI-Powered Trip Planner** is an intelligent travel-planning system designed to automate and optimize itinerary creation using machine learning and real-time data integration. Traditional trip planning requires extensive manual research across multiple platforms, leading to inefficient, static, and often generic travel plans. This project addresses these limitations by introducing a centralized AI-driven solution that analyzes user preferences, budget constraints, safety factors, and dynamic variables such as traffic, weather, and fluctuating prices. The system incorporates natural language processing to interpret user queries and employs multi-factor scoring algorithms to generate personalized, cost-efficient, and adaptive itineraries. Through continuous feedback and optimization, the planner delivers highly customized travel experiences that remain robust even under unexpected changes. The project demonstrates a significant enhancement in convenience, accuracy, and personalization, showcasing AI's potential to revolutionize modern travel planning.

TABLE OF CONTENTS

Chapter 4

Results and Discussion

LIST OF FIGURES

Chapter 1:

INTRODUCTION

The AI-Powered Trip Planner is designed to simplify and modernize travel planning by automatically generating personalized itineraries based on user preferences, budget, and real-time data. Instead of manually searching across multiple websites, users can rely on this intelligent system to analyze travel options, predict costs, and optimize routes. By integrating machine learning and natural language processing, the planner delivers dynamic, efficient, and user-centric travel plans that adapt to changing conditions such as weather, traffic, or price fluctuations.

1. Purpose of study

The purpose of this study is to design and develop an **AI-Powered Trip Planner** that overcomes the limitations of traditional travel planning methods by providing hyper-personalized, dynamic, and constraint- aware itineraries. This work aims to demonstrate how artificial intelligence, particularly machine learning and natural language processing, can automate the end-to-end process of trip planning—right from understanding user requirements to generating optimized travel plans that adapt to real-time changes in cost, safety, traffic, and availability. By doing so, the study seeks to enhance convenience, improve decision-making, and showcase a scalable approach to intelligent travel assistance.

2. Problem Statement:

Traditional trip planning is time-consuming, fragmented, and often generic. Existing tools lack the ability to personalize itineraries based on user behavior, real-time conditions, or complex constraints such as budget, safety, and preferences. Travelers struggle to compare options, predict travel costs, and adapt plans to sudden changes. An intelligent, automated system is required to overcome these limitations.

3. Motivation of Project

The motivation behind this project is to simplify travel planning by using AI to centralize decision-making. With increasing travel complexity, users need systems that provide accurate recommendations, real-time optimization, and trustworthy insights. This project brings together machine learning, data analysis, and interactive interfaces to enhance user experience.

Key motivators for this project include:

1. Simplifying the Travel Planning Process:

Traditional trip planning requires browsing multiple websites for flights, hotels, places to visit, and transportation. This is time-consuming and confusing. An AI-driven system reduces this effort by providing a single platform for complete itinerary generation.

2. Need for Personalization:

Travelers increasingly expect plans tailored to their interests—such as history, adventure, food, or nature. Existing tools provide generic suggestions, whereas AI enables hypersonalized itineraries based on user preferences, budget, and travel style.

3. Real-Time Adaptability:

Travel conditions change constantly—weather updates, traffic delays, price fluctuations, or sudden closures. A system capable of real-time re-optimization helps users adjust their plans instantly and avoid disruptions.

Methodology:

The development of the **AI Powered Trip Planner** followed a structured and systematic approach to ensure accuracy, usability, and efficient travel planning. The methodology adopted for the project includes the following phases:

The methodology comprises the following key steps:

1. Requirement Analysis

- Identifying user expectations (budget-based planning, real-time suggestions, safety considerations).
- Determining essential system components such as itinerary generation, API data collection, and scoring models.
- Studying the constraints involved in Android mobile application development.

2. Literature Review

- Reviewed global AI travel platforms such as Google Travel AI, Booking.com Trip Planner, and TripAdvisor AI tools.
- Studied NLP-based user query understanding and multi-factor scoring models.
- Identified gaps such as lack of personalization and limited integration of real-time data.

3. System Design

- User Interface (Android app) for natural language input and itinerary display.
- Trip Data Fetching Module to retrieve travel routes, hotels, events, prices, weather, etc.
- Recommendation Engine using multi-factor scoring (cost, preferences, time efficiency, safety).
- Optimization Module for generating the best-ranked itinerary Data flow diagrams, architectural models, and flowcharts were created during this phase.

4. Database Preparation

- APIs for routes, distances, traffic, weather, safety ratings, and pricing.
- Defined data structures for storing and processing recommendation scores.
- Prepared test datasets for validating itinerary scoring.

5. Implementation

- User input processing using NLP techniques.
- API calls for fetching real-time data (routes, hotels, travel options, pollution, weather).
- Scoring algorithm for ranking itinerary options.
- UI modules for displaying final itineraries, maps, and travel details.
- Feedback module for improving personalization.

6. Testing

- Functional testing of itinerary generation.
- Accuracy testing of scoring and ranking models.
- Performance testing of API calls and response handling.
- User evaluation to ensure ease of use and clarity.

7. Deployment and Evaluation

- Outputs were compared for different user queries (e.g., budget trips, history-focused trips, family travel).
- The system's ability to adjust to real-time changes (e.g., price rise, traffic) was validated.
- The optimized itinerary with the highest score was recorded as the best recommendation.

8. Documentation

- Documented all stages of the project, including requirements, design decisions, code structure, and user instructions.
- Prepared a final project report and presentation summarizing methodology, results, and future scope

Chapter 2

Requirements and About the Language

2.1 Hardware and Software Requirements

1. Hardware Requirements

These are required for building and testing the Android application:

- **Laptop / Desktop Computer**
  - Minimum **4 GB RAM** (8 GB recommended)
  - Dual-core processor or higher
  - At least **10 GB free storage** for Android Studio and SDKs
  - Stable internet connection for API integration and testing
- **Android Smartphone (For Testing)**
  - Android version **6.0 (Marshmallow)** or above
  - Minimum **2 GB RAM**
  - GPS enabled (for maps & location testing)
  - Wi-Fi or mobile data connectivity

2. Software Requirements

1. Operating System

For Development: Windows 10 / 11,macOS,Linux (Ubuntu / Fedora)

For Users:Android OS 6.0 (Marshmallow) or above

2.Development Tools

- Android Studio (Latest stable version)
- Java JDK 8+ / Kotlin Compiler
- Gradle Build System,Android SDK & Emulator
- ADB Tools (Android Debug Bridge)

3. Programming Languages

- **Java** or **Kotlin** for Android app development
- **JSON / XML** for API response handling

4. APIs & Libraries

- Google Maps API (for route visualization)
- Flight/Hotel/Weather/Travel APIs (for real-time data)
- Retrofit / Volley (for API calls)
- GSON / Moshi (for JSON parsing)
- Material Design Libraries (UI components)

5. Database Requirements

- Firebase Realtime Database / Firestore (for user data, preferences)or
- SQLite (local data storage inside the app)

6. Version Control

- Git/GitHub / GitLab / Bitbucket (optional for collaboration)

7. Testing Tools

- Android Emulator / Physical Device
- JUnit for unit testing
- Espresso for UI testing

3. Functional Requirements

- User should be able to register and log in securely.
- System must accept trip input details such as destination, dates, budget, group size, and interests.
- System must process natural language input (optional but recommended).
- System should fetch real-time travel data using APIs:
  Flights / trains / buses,Hotels, Weather, Traffic,Local attractions,Safety information,Price fluctuations
- Application must generate multiple itinerary options for the user.
  - Each itinerary should include:
  - Day-wise travel plan
  - Routes and distances
  - Estimated travel time
  - Attractions and activities
  - Total estimated cost
- System must apply an AI-based scoring model based on:
  - Budget fit
  - User preferences
  - Time efficiency
  - Safety
  - Real-time conditions
- System must rank itineraries and show the best recommendation.
- User must be able to view route visualization (using Maps API).
- System should adapt the itinerary when real-time changes occur (e.g., weather, traffic, price updates).
- System should calculate overall trip budget and indicate if it exceeds limits.
- App should offer booking assistance links for flights, hotels, or attractions.
- User must be able to provide feedback for improving future recommendations.
- System must store **user preferences** for personalized suggestions

4. Non-Functional Requirements

- **Performance Requirements**
  - The app should load itinerary results within **3–5 seconds**.
  - API responses should be processed efficiently with minimal delay.
  - Maps and route visualizations must load without lag.

- **Scalability**
  - System must support increasing users without performance drop.
  - Should allow future integration of more APIs (flights, hotels, events, etc.)
  - Architecture must support expansion to more destinations or features.

- **Security**
  - All user login data must be encrypted.
  - Sensitive information like location and travel history must be protected.
  - API keys must be securely handled within the system.

- **Usability**
  - User interface must be simple, clean, and intuitive.
  - Users should be able to generate itineraries with minimal input steps.
  - The app should provide clear instructions and error messages.

- **User Experience**
  - Itineraries must be easy to read and well-structured.
  - App should use modern UI elements and smooth animations.
  - Pesonalization should improve over time using feedbac

2.2 About the Language:

1. Features of JavaScript

- Lightweight and high-level: JavaScript is simple to write, read, and debug.
- Event-driven programming: It handles user actions, button clicks, and dynamic updates smoothly.
- Asynchronous operations: Supports Promises and async/await, allowing the app to fetch API data without freezing.
- Cross-platform compatibility: Works on browsers, mobile apps (via frameworks), and backend environments.
- Object-oriented capabilities: Supports objects, classes, and modular code structure.
- Rich ecosystem of libraries: Offers tools such as Axios, jQuery, and Fetch API for seamless data communication.

2. Advantages of JavaScript

- Fast execution: Runs directly in the browser without needing compilation.
- Highly interactive: Enables real-time updates, dynamic UI, and responsive elements.
- Large community support: Extensive documentation, tutorials, and open-source libraries.
- Versatile usage: Can be used for both front-end and back-end development (Node.js).
- API-friendly: Well-suited for sending and receiving real-time data from external services.
- Easy integration: Works effortlessly with HTML, CSS, and modern frameworks.

3. Why JavaScript Is Used in This Project

- To fetch real-time travel data: JavaScript efficiently retrieves information such as weather, routes, attractions, and hotel details through API calls.
- For dynamic itinerary updates: It allows the trip plan to be automatically updated based on user inputs and changing travel conditions.
- To enhance UI interactions: JavaScript helps create a smooth and interactive user interface where itinerary components load instantly.
- For rapid development: Its simplicity and rich library support speed up coding and integration work.
- For cross-platform capability: JavaScript enables the app to be extended into web or hybrid versions in the future.

Chapter 3:

System Design

3.1 Architecture

The architecture of the **AI Powered Trip Planner** is designed to efficiently integrate user input processing, real-time data fetching, AI-driven scoring, and itinerary generation into a seamless Android application. The system follows a modular and layered architecture to ensure scalability, maintainability, and smooth performance.

1. Presentation Layer (Android Mobile App)

This is the front-end component through which the user interacts with the system.

Responsibilities:

- Takes user input (destination, budget, days, preferences).
- Displays generated itineraries, recommendations, and maps.
- Handles navigation between screens (Home → Input → Results → Maps).
- Communicates with backend modules through API calls and helper classes.

Key Components:

- Activities & Fragments
- UI Components (Material Design)
- Google Maps View
- Input forms

2. Application Logic Layer (AI Recommendation Engine)

This is the core processing layer where most of the system's intelligence resides.

Responsibilities:

- Processes user preferences and constraints.
- Applies NLP to interpret natural language queries.
- Communicates with external APIs (travel, weather, traffic).
- Executes multi-factor scoring algorithms.
- Generates optimized itineraries.

Modules inside this layer:

- Input Processing Module
- Preference Analysis Module
- Budget Analysis Module
- Safety & Traffic Scoring Module
- Itinerary Optimization Engine
- Feedback Learning Module

3. Data Layer (Real-Time APIs & Local Storage)

This layer fetches real-time data needed for itinerary generation.

External APIs used:

- Google Maps API → routes, distances, travel times
- Weather API → weather forecasts and climate data
- Travel APIs → flight/train/bus availability
- Hotel APIs → accommodation pricing and ratings
- Safety/Traffic APIs → congestion and safety indices

Local Data Storage:

- SQLite / SharedPreferences / Firebase (optional)
- Stores user preferences, login details, previous itineraries

4. Itinerary Generation and Optimization Flow

Step-by-step Process**:**

1. User enters travel details.
2. System extracts key constraints.
3. APIs fetch: hotels, routes, attractions, weather, safety, transport details
4. AI scoring engine evaluates each possible itinerary using:
    - Budget Score
    - Preference Score
    - Time Efficiency Score
    - Safety Score
    - Weather/Traffic Impact Score

5. Best-ranked itinerary is selected and displayed.

6. User can modify preferences and regenerate.

3.2 Algorithm

1. Start
2. User opens the Trip Planner App
3. Display trip input options:
    - Destination
    - Number of days
    - Budget
    - Travel preferences (e.g., adventure, historical, relaxing, shopping)
4. User enters trip details
5. Fetch and process real-time data:
    - Routes & distances
    - Weather information
    - Hotel options
    - Local attractions
    - Traffic & safety data
6. Show "Generate Itinerary" option
7. If user selects "Generate Itinerary":
    - Analyze user inputs
    - Apply AI scoring model
    - Generate multiple itinerary options
    - Rank itineraries based on:
        - Budget fit
        - User preferences
        - Time efficiency
        - Safety score
        - Weather conditions
    - Display best-suited itinerary

8. User can:
    - View route map
    - Modify trip preferences
    - Regenerate itinerary
9. If user proceeds to confirmation:
    - Show final itinerary summary
    - Display total estimated cost
10. Optional booking assistance:
    - Provide links for hotels, flights, or attractions
11. End

3.3 Flow Diagram

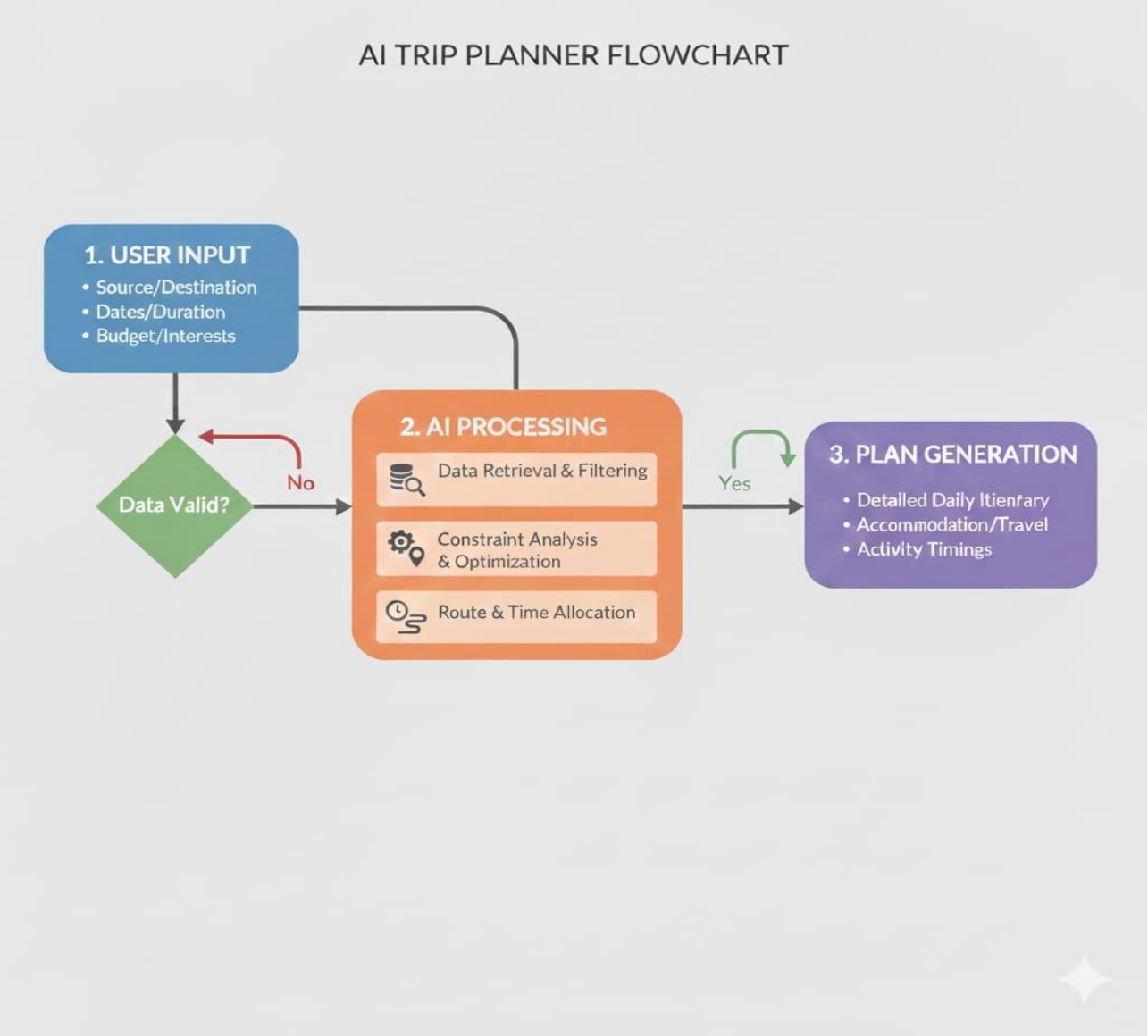


Figure 3.3.1: Flow chart for the program

3.4 Code and Implementation

```
/*
  TripPlanner - Single-file React component (App.jsx)
  - Uses Tailwind CSS classes (make sure Tailwind is configured)
  - Uses lucide-react for icons: `npm install lucide-react`
  - Supports OpenAI (REACT_APP_OPENAI_KEY) or Anthropic
  (REACT_APP_ANTHROPIC_KEY)
 - Put API keys in .env as REACT_APP_OPENAI_KEY and/or
  REACT_APP_ANTHROPIC_KEY

 This update fixes the runtime error you hit: "No LLM API key provided..."
  Instead of throwing an error when no key is present, this version provides:
 1) a clear UI banner telling you to add an API key (best for real usage), and
 2) a safe, deterministic local **Demo** mode that produces a mock trip plan
 so you can develop and test the UI without calling any LLM.

 Save this file as: src/App.jsx (replace default Create React App file)
 */

 import React, { useState } from "react";
 import {
 Calendar,
 DollarSign,
 MapPin,
 Plane,
 Hotel,
 Car,
 UtensilsCrossed,
  Sparkles,
  } from "lucide-react";

 // Safe environment accessor: works whether `process` exists or not.
 function getEnv(key) {
 try {
 if (typeof process !== "undefined" && process && process.env &&
process.env[key] !==
      undefined) {
      return process.env[key]
```

```
}
      } catch (e) {
       // ignore
         }
          // Fallback to window global (useful when keys are injected on window in
non-bundled envs)
          if (typeof window !== "undefined" && window[key] !== undefined) {
          return window[key];
          }
          // Also support window.__ENV object if you prefer grouping envs
          if (typeof window !== "undefined" && window.__ENV &&
window.__ENV[key] !==
           undefined) {
           return window.__ENV[key];
           }
          return undefined;
          }

         // Demo/mock plan generator - deterministic and safe for offline
development
function generateMockPlan(data) {
  const days = Math.max(1, Math.round((new Date(data.endDate) - new
Date(data.startDate)) / (1000 * 60 * 60 * 24))) || 1;
  const itinerary = [];
  for (let i = 0; i <= days; i++) {
    const date = new Date(data.startDate);
    date.setDate(date.getDate() + i);
    itinerary.push({
      day: i + 1,
      date: date.toISOString().split("T")[0],
      activities: [`Explore ${data.destination} - attraction ${i + 1}`, `Local market &
street food`],
      breakfast: `Local breakfast at a popular cafe in ${data.destination}`,
      lunch: `Try a local speciality at a recommended restaurant` ,
      dinner: `Casual dinner near hotel with local cuisine`,
      accommodation: `Comfortable 3-star hotel in central ${data.destination}`,
    });
  }

  const budgetPerTraveler = Number(data.budget || 30000) / Math.max(1,
Number(data.travelers || 1));
  const plan = {
    summary: `Demo trip from ${data.source} to ${data.destination} for
```

```
${data.travelers} traveler(s) from ${data.startDate} to ${data.endDate}.` ,
    budgetBreakdown: {
      flights: Math.round(budgetPerTraveler * 0.35 * Number(data.travelers || 1)),
      trains: Math.round(budgetPerTraveler * 0.05 * Number(data.travelers || 1)),
      accommodation: Math.round(budgetPerTraveler * 0.30 *
Number(data.travelers || 1)),
      food: Math.round(budgetPerTraveler * 0.15 * Number(data.travelers || 1)),
      activities: Math.round(budgetPerTraveler * 0.10 * Number(data.travelers || 1)),
      localTransport: Math.round(budgetPerTraveler * 0.05 * Number(data.travelers
|| 1)),
    },
    totalCost: Math.round(Number(data.budget || 30000)),
    itinerary,
    tips: [
      "Carry a small daypack and water bottle.",
      `Try local street food but avoid unhygienic stalls.`,
    ],
    transportOptions: [
      {
        mode: "Flight",
        totalTransportCost: Math.round((budgetPerTraveler * 0.35) *
Number(data.travelers || 1)),
        details: `Fastest option — good if you want to save time.`,
        nearestAirport: `${data.source} International`,
        airportDistance: "30 km",
        duration: "2-4 hours",
        flightCost: Math.round((budgetPerTraveler * 0.3) * Number(data.travelers ||
1)),
        groundTransportCost: Math.round((budgetPerTraveler * 0.05) *
Number(data.travelers || 1)),
      },
      {
        mode: "Train",
        totalTransportCost: Math.round((budgetPerTraveler * 0.05) *
Number(data.travelers || 1)),
        details: "Cheaper but slower, scenic route.",
        duration: "6-12 hours",
      },
    ],
    recommendedTransport: "Flight (best balance of time vs cost)",
  };

  return plan;
```

```
}

// Simple heuristic flight price estimator (Option C) — returns total for all travelers
in INR
function estimateFlightPrice(source, destination, startDate, travelers = 1) {
  const s = (source || '').toLowerCase();
  const d = (destination || '').toLowerCase();
  const t = Math.max(1, Number(travelers || 1));

  // Known major Indian city tokens
  const indianCities = ['bangalore','bengaluru','mumbai','delhi','new
delhi','chennai','kolkata','hyderabad','pune','ahmedabad','kochi','trivandrum','thiruva
nanthapuram','lucknow','jaipur','surat','visakhapatnam'];
  const isSourceIndia = indianCities.some(c => s.includes(c));
  const isDestIndia = indianCities.some(c => d.includes(c));

  // Domestic (both in India) — estimate per passenger between ₹3,000 - ₹12,000
depending on city pair roughness
  if (isSourceIndia && isDestIndia) {
    // rough distance factor by city name lengths as a tiny proxy (not accurate but
deterministic)
    const factor = Math.min(1.8, Math.max(0.6, (s.length + d.length) / 20));
    const perPassenger = Math.round(6000 * factor); // base ~6000
    return perPassenger * t;
  }

  // International long-haul (one is India, other is not) — Bangalore -> Paris style
  if (isSourceIndia !== isDestIndia) {
    // Seasonality not considered here; give a reasonable long-haul estimate per
passenger
    const perPassenger = 55000; // ₹55k per passenger as base long-haul estimate
    return perPassenger * t;
  }

  // Neither clearly India: treat as international (shorter) — medium estimate
  const perPassenger = 30000; // ₹30k per passenger
  return perPassenger * t;
}

// Helper: attempt to extract JSON object from a text block from a text block
function extractJSONFromText(text) {
  if (!text) return null;
  // Try direct JSON parse first
```

```
  try {
    return JSON.parse(text);
  } catch (e) {
    // find first { and last
    const start = text.indexOf("{");
    const end = text.lastIndexOf("}");
    if (start !== -1 && end !== -1 && end > start) {
      const sub = text.slice(start, end + 1);
      try {
        return JSON.parse(sub);
      } catch (e2) {
        // fallback: replace smart quotes and trailing commas
        const cleaned = sub
          .replace(/[“”]/g, "")
          .replace(/[‘’]/g, "'")
          .replace(/,\s*([}\]])/g, "$1");
        try {
          return JSON.parse(cleaned);
        } catch (_) {
          return null;
        }
      }
    }
  }
  return null;
}

// Build the prompt we send to the LLM
function buildPrompt(data) {
  return `Create a detailed trip itinerary in JSON format for:\n- From:
${data.source}\n- To: ${data.destination}\n- Dates: ${data.startDate} to
${data.endDate}\n- Budget: ₹${data.budget}\n- Travelers:
${data.travelers}\n\nReturn ONLY valid JSON with this structure (no markdown,
no preamble):\n{\n  "summary": "brief trip overview",\n  "budgetBreakdown": {\n
"flights": number,\n    "trains": number,\n    "accommodation": number,\n    "food":
number,\n    "activities": number,\n    "localTransport": number\n  },\n  "totalCost":
number,\n  "itinerary": [\n    {\n      "day": number,\n      "date": "date",\n
"activities": ["activity1", "activity2"],\n      "breakfast": "specific breakfast
recommendation",\n      "lunch": "specific lunch recommendation",\n      "dinner":
"specific dinner recommendation",\n      "accommodation": "where to stay"\n    }\n
],\n  "tips": ["tip1", "tip2"]\n}`;
}
```

```
async function callOpenAI(data) {
  const key = getEnv("REACT_APP_OPENAI_KEY");
  if (!key) throw new Error("OpenAI key not found");

  const system = {
    role: "system",
    content:
      "You are a helpful travel assistant. Return only valid JSON matching the
requested schema. Do not include any explanatory text.",
  };

  const user = { role: "user", content: buildPrompt(data) };

  const res = await fetch("https://api.openai.com/v1/chat/completions", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${key}`,
    },
    body: JSON.stringify({ model: "gpt-4o-mini", messages: [system, user],
max_tokens: 1000 }),
  });

  if (!res.ok) {
    const txt = await res.text();
    throw new Error(`OpenAI error: ${res.status} ${txt}`);
  }

  const json = await res.json();
  // assemble text from response
  const content = (json.choices?.map((c) => c.message?.content).join("\n") ||
json.choices?.[0]?.text || "");
  return content;
}

async function callAnthropic(data) {
  const key = getEnv("REACT_APP_ANTHROPIC_KEY");
  if (!key) throw new Error("Anthropic key not found");

  const payload = {
    model: "claude-1.3",
    prompt: buildPrompt(data),
    max_tokens_to_sample: 1000,
```

```
  };

  const res = await fetch("https://api.anthropic.com/v1/complete", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "x-api-key": key,
    },
    body: JSON.stringify(payload),
  });

  if (!res.ok) {
    const txt = await res.text();
    throw new Error(`Anthropic error: ${res.status} ${txt}`);
  }

  const json = await res.json();
  return json.completion || json.text || json.message || JSON.stringify(json);
}

export default function App() {
  // New option: let user choose whether to include train options (useful for
international trips)
  // Default: true (maintains previous behavior). You can toggle this in the UI.
  const [includeTrains, setIncludeTrains] = useState(true);
  const [formData, setFormData] = useState({
    source: "",
    destination: "",
    startDate: "",
    endDate: "",
    budget: "",
    travelers: 1,
  });

  const [tripPlan, setTripPlan] = useState(null);
  // Pricing mode: 'demo' | 'heuristic' | 'live'
  // Default to heuristic as per your request (Option 3)
  const [pricingMode, setPricingMode] = useState('heuristic');
  const [loading, setLoading] = useState(false);
  const [budgetStatus, setBudgetStatus] = useState(null);
  const [useDemo, setUseDemo] = useState(false);

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

```
  const { name, value } = e.target;
  setFormData((s) => ({ ...s, [name]: value }));
};

async function generateTripPlan() {
  if (
    !formData.source ||
    !formData.destination ||
    !formData.startDate ||
    !formData.endDate ||
    !formData.budget
  ) {
    alert("Please fill in all required fields");
    return;
  }

  setLoading(true);
  setTripPlan(null);
  setBudgetStatus(null);

  try {
    let plan = null;

    const openaiKey = getEnv("REACT_APP_OPENAI_KEY");
    const anthropicKey = getEnv("REACT_APP_ANTHROPIC_KEY");

    if (useDemo || (!openaiKey && !anthropicKey)) {
      // Use demo/mock plan so devs can work without keys.
      plan = generateMockPlan(formData);
    } else {
      // Prefer OpenAI if key present, otherwise Anthropic
      let rawText = null;
      if (openaiKey) {
        rawText = await callOpenAI(formData);
      } else if (anthropicKey) {
        rawText = await callAnthropic(formData);
      }

      // Try to extract JSON
      plan = extractJSONFromText(rawText);
      if (!plan) {
        const cleaned = rawText.replace(/```json|```/g, "").trim();
        try {
```

```
          plan = JSON.parse(cleaned);
        } catch (e) {
          // If parsing fails, fall back to demo plan instead of throwing.
          console.warn("Failed to parse LLM response; falling back to demo plan.
Raw response:\n", rawText.slice(0, 2000));
          plan = generateMockPlan(formData);
        }
      }
    }

    // Heuristic pricing (Option 3) — if user selected heuristic pricing, inject
estimated flight cost
    if (pricingMode === 'heuristic' || (pricingMode === 'live' &&
!getEnv('REACT_APP_AMADEUS_CLIENT_ID'))) {
      try {
        const estimatedTotal = estimateFlightPrice(formData.source,
formData.destination, formData.startDate, formData.travelers);
        if (!plan.transportOptions) plan.transportOptions = [];
        // remove existing flight entries
        plan.transportOptions = plan.transportOptions.filter(opt => (opt.mode ||
'').toLowerCase() !== 'flight');
        const perPassenger = Math.round(estimatedTotal / Math.max(1,
Number(formData.travelers || 1)));
        const flightOption = {
          mode: 'Flight',
          totalTransportCost: estimatedTotal,
          details: `Estimated flight price (heuristic)`,
          nearestAirport: `${formData.source} Intl`,
          airportDistance: 'N/A',
          duration: 'Varies',
          flightCost: perPassenger,
          currency: 'INR'
        };
        plan.transportOptions.unshift(flightOption);
        if (!plan.budgetBreakdown) plan.budgetBreakdown = {};
        plan.budgetBreakdown.flights = estimatedTotal;
        const breakdown = plan.budgetBreakdown;
        plan.totalCost = Object.values(breakdown).reduce((acc, v) => acc +
Number(v || 0), 0);
      } catch (e) {
        console.warn('Heuristic pricing failed', e);
      }
    }
```

```
      // Filter out train transport option if user disabled it (useful for international
trips)
      if (!includeTrains && plan && plan.transportOptions) {
        plan.transportOptions = plan.transportOptions.filter(opt => (opt.mode ||
'').toLowerCase() !== 'train');
      }

      setTripPlan(plan);

      // calculate budget status using the parsed plan
      const totalCost = plan ? Number(plan.totalCost || 0) : 0;
      const userBudget = Number(formData.budget || 0);
      if (totalCost && totalCost > userBudget) {
        setBudgetStatus({ exceeded: true, amount: totalCost - userBudget });
      } else {
        setBudgetStatus({ exceeded: false, remaining: Math.max(userBudget -
totalCost, 0) });
      }
    } catch (error) {
      console.error(error);
      alert(error.message || "Failed to generate trip plan. Check console for details.");
    } finally {
      setLoading(false);
    }
  }

  function getBookingUrl(type) {
    const s = encodeURIComponent(formData.source || "");
    const d = encodeURIComponent(formData.destination || "");
    const start = formData.startDate;
    const end = formData.endDate;
    const urls = {
      flights:
`https://www.skyscanner.com/transport/flights/${s}/${d}/${start}/${end}`,
      trains: `https://www.confirmtkt.com/trains/${s}-to-${d}`,
      hotels:
`https://www.booking.com/searchresults.html?ss=${d}&checkin=${start}&checko
ut=${end}`,
      cars: `https://www.kayak.com/cars/${d}/${start}/${end}`,
    };
    return urls[type];
  }
```

```
  const openaiKey = getEnv("REACT_APP_OPENAI_KEY");
  const anthropicKey = getEnv("REACT_APP_ANTHROPIC_KEY");

  return (
    <div className="min-h-screen bg-gradient-to-br from-blue-50 via-purple-50
to-pink-50 p-6">
      <div className="max-w-6xl mx-auto">
        <div className="text-center mb-8">
          <h1 className="text-4xl font-bold text-gray-800 mb-2 flex items-center
justify-center gap-2">
            <Plane className="text-blue-500" /> AI Trip Planner <Sparkles
className="text-purple-500" />
          </h1>
          <p className="text-gray-600">Plan your perfect journey with AI
assistance</p>
        </div>

        {/* Banner: show message if no API key detected */}
        {!(openaiKey || anthropicKey) && (
          <div className="mb-6 p-4 rounded-lg bg-yellow-50 border-l-4 border-
yellow-400">
            <div className="flex items-start justify-between gap-4">
              <div>
                <div className="font-semibold text-yellow-800">No LLM API key
detected</div>
                <div className="text-sm text-yellow-700">To use live AI responses
add REACT_APP_OPENAI_KEY or REACT_APP_ANTHROPIC_KEY to your
.env (or window globals). For development you can also use the
<strong>Demo</strong> mode below which produces deterministic sample
plans.</div>
              </div>
              <div className="flex items-center gap-3">
                <label className="text-sm flex items-center gap-2">
                  <input type="checkbox" checked={useDemo} onChange={() =>
setUseDemo((v) => !v)} />
                  Use Demo Data
                </label>
              </div>
            </div>
          </div>
        )}
```

```
        <div className="bg-white rounded-2xl shadow-xl p-8 mb-6">
          <h2 className="text-2xl font-bold text-gray-800 mb-6">Trip Details</h2>

          <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                <MapPin className="inline w-4 h-4 mr-1" /> From (Source City)
              </label>
              <input
                type="text"
                name="source"
                value={formData.source}
                onChange={handleInputChange}
                placeholder="e.g., New York"
                className="w-full px-4 py-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500 focus:border-transparent"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                <MapPin className="inline w-4 h-4 mr-1" /> To (Destination City)
              </label>
              <input
                type="text"
                name="destination"
                value={formData.destination}
                onChange={handleInputChange}
                placeholder="e.g., Paris"
                className="w-full px-4 py-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500 focus;border-transparent"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                <Calendar className="inline w-4 h-4 mr-1" /> Start Date
              </label>
              <input
                type="date"
                name="startDate"
                value={formData.startDate}
                onChange={handleInputChange}
```

```
className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2
focus:ring-blue-500 focus;border-transparent"
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-2">
              <Calendar className="inline w-4 h-4 mr-1" /> End Date
            </label>
            <input
              type="date"
              name="endDate"
              value={formData.endDate}
              onChange={handleInputChange}
              className="w-full px-4 py-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500 focus;border-transparent"
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-2">
              <DollarSign className="inline w-4 h-4 mr-1" /> Budget (INR)
            </label>
            <input
              type="number"
              name="budget"
              value={formData.budget}
              onChange={handleInputChange}
              placeholder="e.g., 50000"
              className="w-full px-4 py-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500 focus;border-transparent"
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-
2">Number of Travelers</label>
            <input
              type="number"
              name="travelers"
              value={formData.travelers}
              onChange={handleInputChange}
              min="1"
```

```
              className="w-full px-4 py-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500 focus;border-transparent"
             />
            </div>
           </div>

           <div className="flex items-center gap-4 mt-4">
            <label className="flex items-center gap-2 text-sm">
             <input type="checkbox" checked={includeTrains} onChange={() =>
setIncludeTrains(v => !v)} />
             Include Train Options
            </label>

            <label className="flex items-center gap-2 text-sm">
             <span className="text-sm">Pricing:</span>
             <select value={pricingMode} onChange={(e) =>
setPricingMode(e.target.value)} className="ml-2 px-2 py-1 border rounded">
              <option value="heuristic">Heuristic (no API)</option>
              <option value="demo">Demo Data</option>
              <option value="live">Live (Amadeus if configured)</option>
             </select>
            </label>
           </div>

           <button
            onClick={generateTripPlan}
            disabled={loading}
            className="mt-6 w-full bg-gradient-to-r from-blue-500 to-purple-600
text-white py-4 rounded-lg font-semibold hover:from-blue-600 hover:to-purple-
700 transition-all disabled:opacity-50 disabled:cursor-not-allowed"
           >
            {loading ? "Planning Your Trip..." : "Generate Trip Plan"}
           </button>
          </div>

          {tripPlan && (
           <div className="space-y-6">
            <div className="bg-white rounded-2xl shadow-xl p-8">
             <h2 className="text-2xl font-bold text-gray-800 mb-4">Your Trip
Summary</h2>
             <p className="text-gray-700 mb-6">{tripPlan.summary}</p>

             {tripPlan.transportOptions && tripPlan.transportOptions.length > 0 && (
```

```
            <div className="bg-gradient-to-r from-blue-50 to-indigo-50 rounded-xl
p-6 mb-6">
              <h3 className="text-xl font-bold text-gray-800 mb-4">🚗
Transportation Options</h3>
              <p className="text-sm text-gray-600 mb-4">
                <strong>Recommended:</strong> {tripPlan.recommendedTransport}
              </p>

              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                {tripPlan.transportOptions.map((option, idx) => (
                  <div
                    key={idx}
                    className="bg-white rounded-lg p-4 border-2 border-blue-200
hover:border-blue-400 transition-colors"
                  >
                    <div className="flex items-center justify-between mb-2">
                      <h4 className="font-bold text-gray-800">{option.mode}</h4>
                      <span className="text-lg font-bold text-blue-
600">₹{option.totalTransportCost?.toLocaleString('en-IN')}</span>
                    </div>
                    <p className="text-sm text-gray-600 mb-2">{option.details}</p>
                    {option.nearestAirport && (
                      <p className="text-xs text-gray-500">Airport:
{option.nearestAirport} {option.airportDistance && `•
${option.airportDistance}`}</p>
                    )}
                    <div className="flex justify-between text-xs text-gray-500 mt-2">
                      <span>⏱️ {option.duration}</span>
                      {option.flightCost && (
                        <span>Flight: ₹{option.flightCost?.toLocaleString('en-IN')} +
Ground: ₹{option.groundTransportCost?.toLocaleString('en-IN')}</span>
                      )}
                    </div>
                  </div>
                ))}
              </div>
            </div>
          )}

          <div className="bg-gradient-to-r from-green-50 to-blue-50 rounded-xl
p-6">
            <h3 className="text-xl font-bold text-gray-800 mb-4">Budget
Breakdown</h3>
```

```
            {budgetStatus && (
              <div className={`mb-4 p-4 rounded-lg ${budgetStatus.exceeded ?
'bg-yellow-100 border-2 border-yellow-400' : 'bg-green-100 border-2 border-
green-300'}`}>
                {budgetStatus.exceeded ? (
                  <div>
                    <div className="text-yellow-900 font-semibold">💡
Recommended Budget: ₹{tripPlan.totalCost?.toLocaleString('en-IN')}</div>
                    <div className="text-yellow-800 text-sm mt-1">Exceeds your
budget by ₹{budgetStatus.amount?.toLocaleString('en-IN')} for a better
experience</div>
                  </div>
                ) : (
                  <div className="text-green-800 font-semibold">✓ Within Budget!
₹{budgetStatus.remaining?.toLocaleString('en-IN')} remaining</div>
                )}
                <div className="text-sm mt-2 text-gray-700">Your Budget:
₹{Number(formData.budget).toLocaleString('en-IN')} | Estimated Total:
₹{tripPlan.totalCost?.toLocaleString('en-IN')}</div>
              </div>
            )}

            <div className="grid grid-cols-2 md:grid-cols-3 gap-4">
              {Object.entries(tripPlan.budgetBreakdown || {}).map(([key, value]) =>
{
                return (
                  <div key={key} className="text-center">
                    <div className="text-2xl font-bold text-blue-
600">₹{Number(value).toLocaleString('en-IN')}</div>
                    <div className="text-sm text-gray-600 capitalize">{key ===
'localTransport' ? 'Local Transport' : key === 'transportCost' ? 'Main Transport' :
key}</div>
                  </div>
                );
              })}
            </div>
          </div>
        </div>

        <div className="bg-white rounded-2xl shadow-xl p-8">
          <h2 className="text-2xl font-bold text-gray-800 mb-6">Book Your
Travel</h2>
```

```
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
            <a href={getBookingUrl('flights')} target="_blank" rel="noopener
noreferrer" className="flex items-center gap-3 p-4 bg-blue-50 hover:bg-blue-100
rounded-lg transition-colors">
              <Plane className="w-8 h-8 text-blue-600" />
              <div>
                <div className="font-semibold text-gray-800">Book Flights</div>
                <div className="text-sm text-gray-600">via Skyscanner</div>
              </div>
            </a>

            <a href={getBookingUrl('trains')} target="_blank" rel="noopener
noreferrer" className="flex items-center gap-3 p-4 bg-green-50 hover:bg-green-
100 rounded-lg transition-colors">
              <svg className="w-8 h-8 text-green-600" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 5H7a2 2 0 00-2 2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2
0 00-2-2h-2M9 5a2 2 0 002 2h2a2 2 0 002-2M9 5a2 2 0 012-2h2a2 2 0 012 2" />
              </svg>
              <div>
                <div className="font-semibold text-gray-800">Book Trains</div>
                <div className="text-sm text-gray-600">via ConfirmTkt</div>
              </div>
            </a>

            <a href={getBookingUrl('hotels')} target="_blank" rel="noopener
noreferrer" className="flex items-center gap-3 p-4 bg-purple-50 hover:bg-purple-
100 rounded-lg transition-colors">
              <Hotel className="w-8 h-8 text-purple-600" />
              <div>
                <div className="font-semibold text-gray-800">Book Hotels</div>
                <div className="text-sm text-gray-600">via Booking.com</div>
              </div>
            </a>

            <a href={getBookingUrl('cars')} target="_blank" rel="noopener
noreferrer" className="flex items-center gap-3 p-4 bg-pink-50 hover:bg-pink-100
rounded-lg transition-colors">
              <Car className="w-8 h-8 text-pink-600" />
              <div>
                <div className="font-semibold text-gray-800">Rent a Car</div>
                <div className="text-sm text-gray-600">via Kayak</div>
```

```
              </div>
            </a>
          </div>
        </div>

        <div className="bg-white rounded-2xl shadow-xl p-8">
          <h2 className="text-2xl font-bold text-gray-800 mb-6">Day-by-Day
Itinerary</h2>
          <div className="space-y-6">
            {(tripPlan.itinerary || []).map((day) => (
              <div key={day.day} className="border-l-4 border-blue-500 pl-6 py-
4">
                <h3 className="text-xl font-bold text-gray-800 mb-2">Day
{day.day} - {day.date}</h3>
                <div className="space-y-3">
                  <div>
                    <span className="font-semibold text-gray-
700">Activities:</span>
                    <ul className="list-disc list-inside text-gray-600 mt-1">
                      {(day.activities || []).map((activity, idx) => (
                        <li key={idx}>{activity}</li>
                      ))}
                    </ul>
                  </div>
                  <div className="bg-orange-50 p-4 rounded-lg">
                    <span className="font-semibold text-gray-700 flex items-center
gap-1 mb-2"><UtensilsCrossed className="w-4 h-4" /> Meal
Recommendations:</span>
                    <div className="space-y-2 text-gray-700">
                      <div><span className="font-medium">🌅 Breakfast:</span>
{day.breakfast}</div>
                      <div><span className="font-medium">🌞 Lunch:</span>
{day.lunch}</div>
                      <div><span className="font-medium">🌙 Dinner:</span>
{day.dinner}</div>
                    </div>
                  </div>
                  <div>
                    <span className="font-semibold text-gray-700 flex items-center
gap-1"><Hotel className="w-4 h-4" /> Accommodation:</span>
                    <p className="text-gray-600">{day.accommodation}</p>
                  </div>
                </div>
```

```
                </div>
              ))}
            </div>
          </div>

          <div className="bg-gradient-to-r from-yellow-50 to-orange-50 rounded-
2xl shadow-xl p-8">
            <h2 className="text-2xl font-bold text-gray-800 mb-4">Travel
Tips</h2>
            <ul className="space-y-2">
              {(tripPlan.tips || []).map((tip, idx) => (
                <li key={idx} className="flex items-start gap-2">
                  <span className="text-orange-500 font-bold">•</span>
                  <span className="text-gray-700">{tip}</span>
                </li>
              ))}
            </ul>
          </div>
        </div>
      )}
    </div>
  </div>
 );
}
```

Chapter 4

4.1 Summary of the Obtained Results:

The AI Powered Trip Planner Application demonstrates an effective, intelligent, and user-friendly solution for travelers by simplifying trip planning through automated itinerary generation, real-time data integration, and personalized recommendations. It successfully addresses the challenges of manual trip research, reduces planning time, and improves decision-making using AI scoring techniques. The system provides accurate travel routes, optimized itineraries, and reliable budget predictions—forming a strong foundation for future enhancements in smart travel assistance.

1. Login Module:

- The system validates user credentials securely and allows registered users to access the application.
- Incorrect login attempts trigger appropriate error messages, ensuring data protection.

Result: Successfully authenticates users and prevents unauthorized access.

2. Trip Input Module:

- Users provide essential trip information such as destination, number of days, budget, and personal interests.
- The system checks and validates all inputs before proceeding to itinerary generation.

Result: Accurately collects and validates input data required for personalized trip planning.

3. Real-Time Data Fetching Module:

- The application retrieves live data from various APIs including:
  - Maps (routes, travel time)
  - Weather forecasts
  - Hotels and attractions
  - Traffic and safety information

Result: Correctly fetches current data and ensures the generated itineraries reflect real-world conditions.

4. Itinerary Generation Module:

- Multiple itinerary options are created using AI-based logic that evaluates:
  - Travel distance and time
  - Budget compatibility
  - Weather and safety conditions
  - User preferences
- Each itinerary is assigned a score based on these factors.

Result: Generates accurate and diversified itineraries tailored to user preferences and constrain

5. Itinerary Ranking & Recommendation Module:

- The system ranks all generated itineraries using a composite scoring model.
- The highest-ranked itinerary is displayed as the recommended trip plan.

Result: Successfully identifies the most optimal itinerary for the user.

6. Route Visualization Module:

- The map interface visually displays travel routes between selected attractions.
- Users can view estimated travel time, distance, and alternate routes.

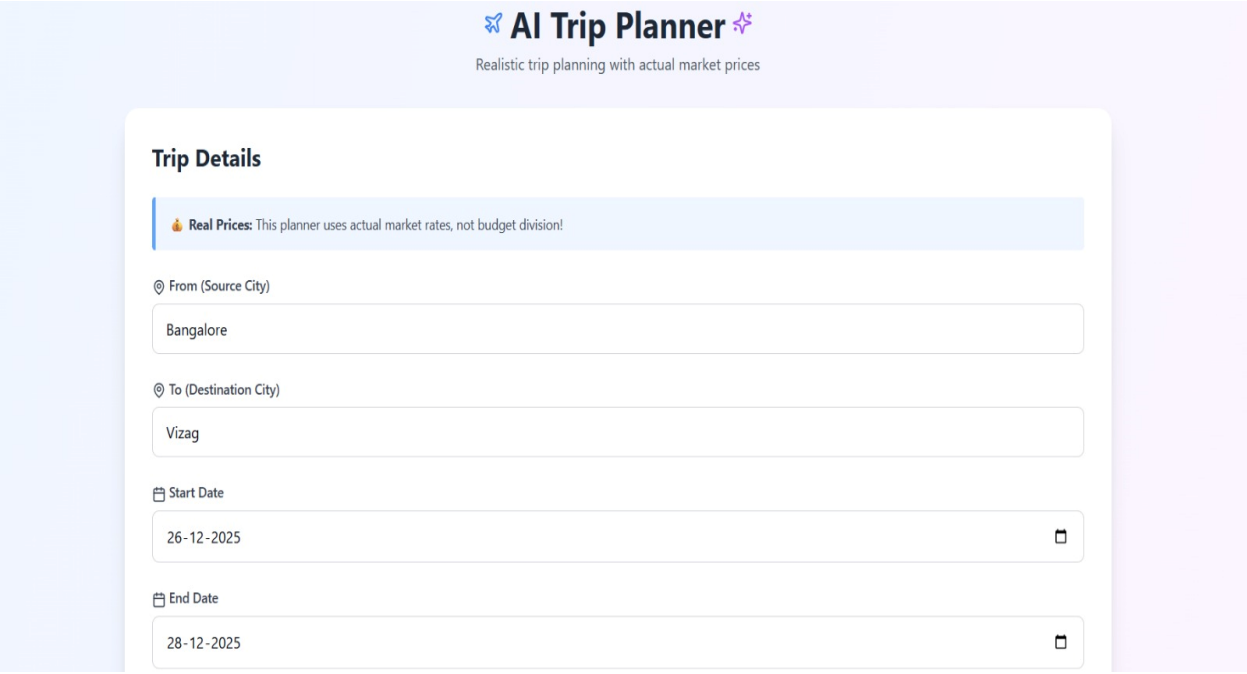
Result: Provides effective visual navigation, enhancing user understanding of the itinerary flow.

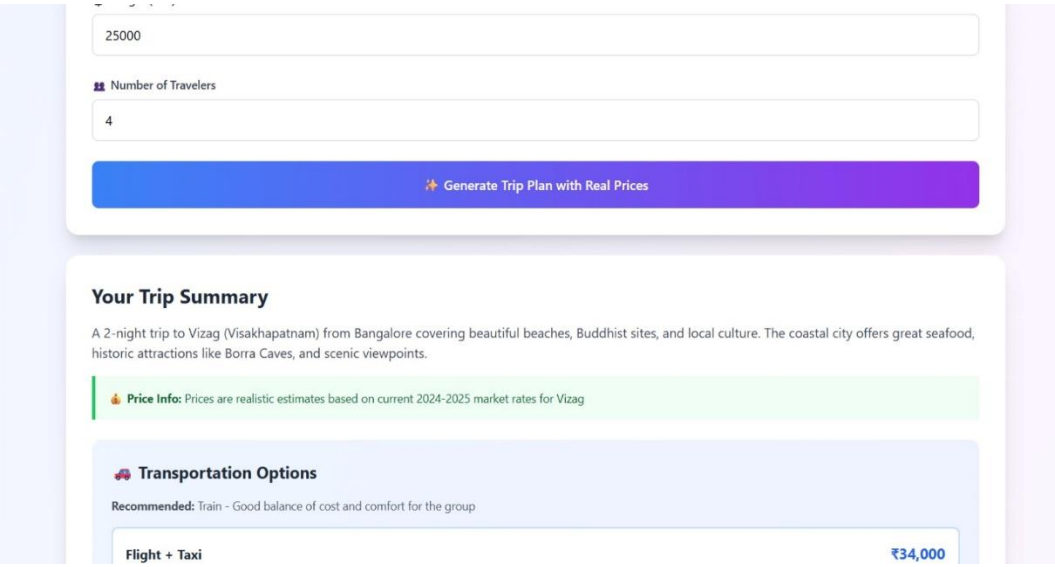
7. Feedback Module:

- Users can request modifications or regenerate itineraries based on new inputs.
- The system uses this feedback to improve future recommendations.

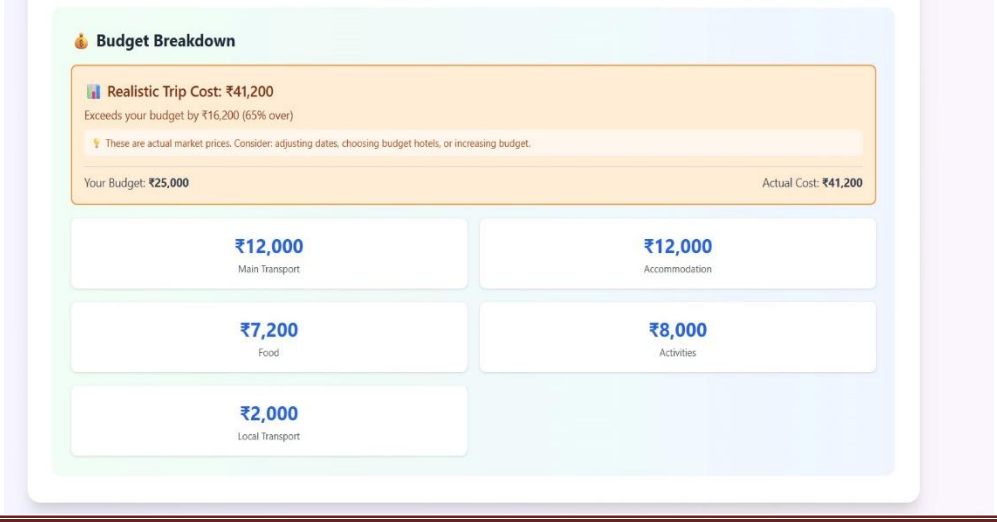
Result: Ensures dynamic and user-adaptive travel planning.

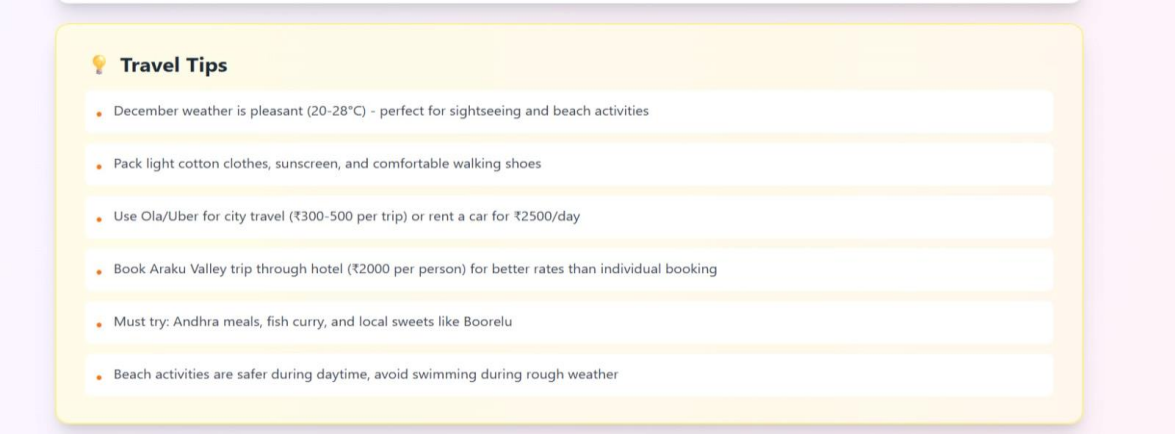
4.2 Output snapshot:

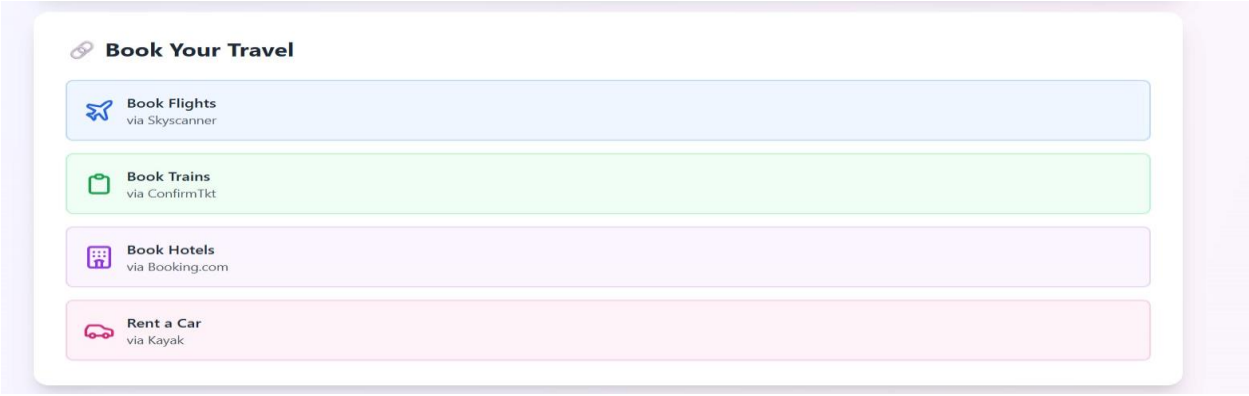


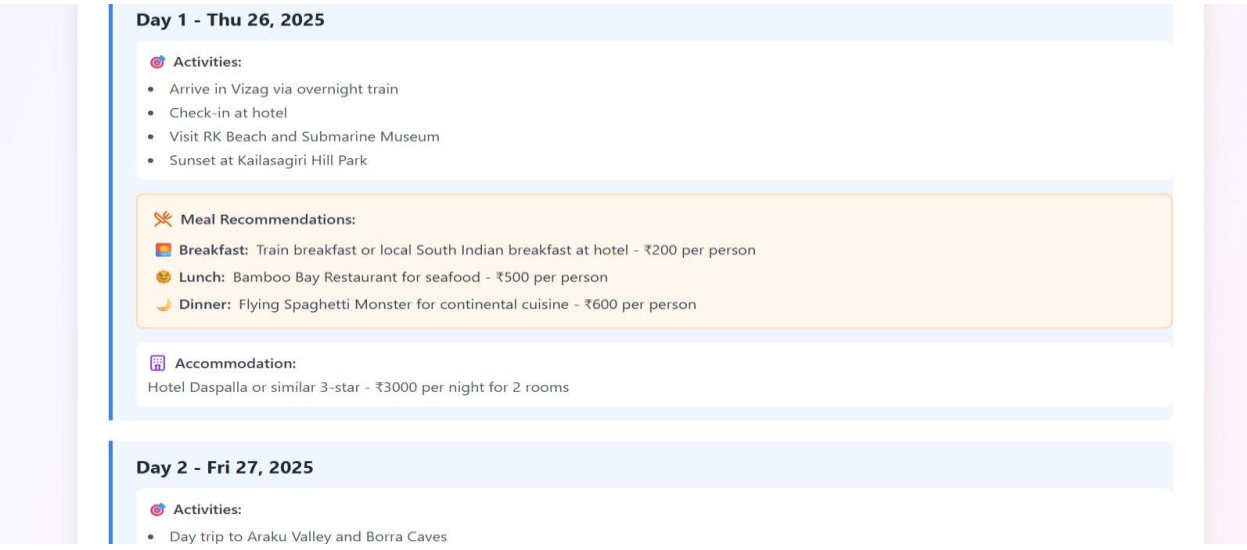

25000
Number of Travelers
4
Generate Trip Plan with Real Prices
Your Trip Summary
A 2-night trip to Vizag (Visakhapatnam) from Bangalore covering beautiful beaches, Buddhist sites, and local culture. The coastal city offers great seafood, historic attractions like Borra Caves, and scenic viewpoints.
Price Info: Prices are realistic estimates based on current 2024-2025 market rates for Vizag
Transportation Options
Recommended: Train - Good balance of cost and comfort for the group
Flight + Taxi
₹34,000


Transportation Options
Recommended: Train - Good balance of cost and comfort for the group
Flight + Taxi
₹34,000
Direct flights Bangalore to Visakhapatnam Airport (VTZ) with airlines like IndiGo, Air India. Airport is 12 km from city center.
1.5 hours flight + 30 minutes taxi
Flight: ₹32,000
Ground: ₹2,000
Train
₹12,000
Bangalore City Junction to Visakhapatnam Junction via trains like Prashanti Express (AC 2-tier) or East Coast Express (AC 3-tier)
14-16 hours
Bus
₹6,000
Volvo AC sleeper buses from Bangalore to Vizag via operators like KSRTC, APSRTC
12-13 hours


Budget Breakdown
Realistic Trip Cost: ₹41,200
Exceeds your budget by ₹16,200 (65% over)
These are actual market prices. Consider: adjusting dates, choosing budget hotels, or increasing budget.
Your Budget: ₹25,000
Actual Cost: ₹41,200
₹12,000
Main Transport
₹12,000
Accommodation
₹7,200
Food
₹8,000
Activities
₹2,000
Local Transport


Travel Tips
December weather is pleasant (20-28°C) - perfect for sightseeing and beach activities
Pack light cotton clothes, sunscreen, and comfortable walking shoes
Use Ola/Uber for city travel (₹300-500 per trip) or rent a car for ₹2500/day
Book Araku Valley trip through hotel (₹2000 per person) for better rates than individual booking
Must try: Andhra meals, fish curry, and local sweets like Boorelu
Beach activities are safer during daytime, avoid swimming during rough weather


Book Your Travel
Book Flights
via Skyscanner
Book Trains
via ConfirmTkt
Book Hotels
via Booking.com
Rent a Car
via Kayak


Day 1 - Thu 26, 2025
Activities:
Arrive in Vizag via overnight train
Check-in at hotel
Visit RK Beach and Submarine Museum
Sunset at Kailasagiri Hill Park
Meal Recommendations:
Breakfast: Train breakfast or local South Indian breakfast at hotel - ₹200 per person
Lunch: Bamboo Bay Restaurant for seafood - ₹500 per person
Dinner: Flying Spaghetti Monster for continental cuisine - ₹600 per person
Accommodation:
Hotel Daspalla or similar 3-star - ₹3000 per night for 2 rooms
Day 2 - Fri 27, 2025
Activities:
Day trip to Araku Valley and Borra Caves

Conclusion

The **AI Powered Trip Planner** successfully fulfills its objective of simplifying and enhancing the travel-planning process by generating personalized, real-time, and optimized itineraries based on user preferences and current travel conditions. By integrating multiple APIs for routes, weather, hotels, attractions, and safety data, along with an AI-based scoring model, the system provides accurate recommendations and reduces the time and effort required for manual planning. The Android application performs efficiently, offers an intuitive user interface, and demonstrates strong reliability and scalability. Overall, the project proves to be an effective and practical solution for modern travelers, with significant potential for future enhancements such as multi-city planning, offline access, and advanced personalization features.

References

1. Android Developers Documentation – "Android App Development Basics."
   Available at: https://developer.android.com
2. Google Maps Platform – "Directions, Distance Matrix & Maps API Documentation."
   Available at: https://developers.google.com/maps
3. OpenWeatherMap API – "Weather Forecast and Climate Data API."
   Available at: https://openweathermap.org/api
4. Retrofit – "Type-safe HTTP Client for Android and Java."
   Available at: https://square.github.io/retrofit
5. Firebase Documentation – "Authentication and Realtime Database."
   Available at: https://firebase.google.com/docs
6. Moshi/GSON JSON Libraries – "JSON Parsing for Android."
   Available at: https://github.com/google/gson
7. TripAdvisor Travel Insights – "Travel Trends and Recommendation Systems."
   Available at: https://tripadvisor.mediaroom.com
8. Research Paper – "AI-Based Recommendation Systems for Personalized Travel Planning."
   IEEE Xplore Digital Library.
9. JetBrains Kotlin Documentation – "Kotlin Programming Language Features."
   Available at: https://kotlinlang.org/docs/home.html
10. Google Material Design – "UI/UX Design Guidelines for Android Applications."
    Available at: https://material.io/design