AI Agent based itinerary planner

An Internship Report

Submitted by

Khushkumar Patel
211310132075

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

INFORMATION & COMMUNICATION TECHNOLOGY
DEPARTMENT OF ADANI INSTITUTE OF INFRASTRUCTURE ENGINEERING AHMADABAD – 382421

Gujarat Technological University, Ahmedabad

[April, 2025]

Adani Institute of Infrastructure Engineering
Ahmadabad - 382421

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **AI Agent based itinerary planner** has been carried by **Khushkumar Patel** out under my guidance in partial fulfillment for the degree of Bachelor of Engineering in, **Information & Communication Technology** 8th Semester of Gujarat Technological University, Ahmedabad during the academic year 2024-25.

Dr. Monalisa Ghosh
Internal Guide

Dr. Ajaykumar Vyas
Head of Department

CERTIFICATE BY THE COMPANY

nowonline

Date: 20th April 2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Patel Khushkumar V.** has successfully completed an internship as a **Data Scientist Intern** at **NowOnline Tech India Pvt. Ltd.** from 20th January 2025 to 19th April 2025.

During his tenure with us, he demonstrated exceptional dedication and skill in various aspects of AI/ML.

He consistently showed a strong work ethic, creativity, and a proactive approach to problem-solving.

We commend him for his hard work and dedication during the internship and wish him all the best in his future endeavors. We are confident that he will continue to excel in his professional journey.

Authorized Signature:

NOWONLINE TECH INDIA PVT. LTD. GANDHINAGAR

Namrata M Joshi
Sr.HR Generalist
NowOnline Tech India Pvt. Ltd.

NowOnline Tech India Private Limited.
701 to 704, 7th Floor, X-Change Plaza of DSCCSL(53E) Road 5C, Block 53, Opp WTC Building, Zone 5, DTA, Gift City, Gandhinagar 382355, Gujarat, India.
Email: info@nowonlinetech.com / Contact No: +91-7600235007

Adani Institute of Infrastructure Engineering
Ahmadabad - 382421

Declaration

We hereby declare that the Internship report submitted along with the Internship entitled **AI Agent based itinerary planner** submitted in partial fulfillment for the degree of Bachelor of Engineering in Information & Communication Technology to Gujarat Technological University, Ahmedabad, is a bona fide record of original project work carried out by me at NowOnline Tech India PVT LTD. under the supervision of Mr. Nilesh Dhorajiya and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

_____

Khushkumar Patel

Acknowledgment

I appreciate the investment of my time and energy in this project. It would not have been possible to finish the project without the help of a huge number of people and organization. To each and every one of them, I am profoundly grateful.

I wouldn't be where I am now without Dr. Monalisa Ghosh They have guided me, offered constant oversight, and briefed me on key aspects of the project. I'd like to use this opportunity to express my appreciation to everyone who helped get the ball rolling on this conversation and provided really helpful review input, as well as to those who provided consistent backing and assistance as we worked to complete our project

NowOnline Tech India PVT LTD. at which I am employed has been quite helpful in giving me both the tools and atmosphere I need to do my job properly.

I'd like to thank the Head of Department, Dr. Ajaykumar Vyas a for his kind assistance and support during this project.

Khushkumar Patel

Abstract

AI-powered travel planning solutions are becoming increasingly essential in the digital tourism landscape, but creating personalized itineraries remains time-consuming and often fails to capture individual preferences effectively. This project aims to develop an automated travel planning application using CREWAI framework and large language models for creating customized travel itineraries. The process involves implementing a multi-agent system with specialized roles for place selection, place exploration, and trip planning, connected through a carefully designed flow architecture. The application features a web-based interface built with Flask that allows users to input travel preferences, modify agent outputs, and obtain comprehensive travel plans including flight information through Amadeus API integration. Conducted as part of an internship at NowOnline, this project demonstrates the potential of AI agent collaboration to streamline the travel planning process while maintaining high levels of personalization and user control.

List of Figures

List of Tables

Abbreviations

ATS - Applicant Tracking System

UI - User Interface

AI - Artificial Intelligence

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

API - Application Programming Interface

iOS - iPhone Operating System

QA - Quality Assurance

UAT - User Acceptance Testing

SEO - Search Engine Optimization

JSON - JavaScript Object Notation

LLM - Large Language Model

VSCode - Visual Studio Code

OS - Operating System

RL - Reinforcement Learning

GPT - Generative Pre-trained Transformer

ML - Machine Learning

NLP - Natural Language Processing

PDF - Portable Document Format

CLI - Command Line Interface

IDE - Integrated Development Environment

Table of Contents

1. Overview of The Company

1.1 History :

- **About us :**

  We advise, build and create custom software. Digital solutions that let your organization grow. With a team of Experts, we build custom digital solutions that advance and differentiate your organization. Whether it's software, mobile apps, websites or data security, our team of experts will make sure your business is ready for the future.

  Together, we create innovative solutions that inspire you and bring you closer to your goals. Whether it's custom software, mobile apps, secure websites, careers website with ATS or smart data solutions-our goal is to work with you to build Digital solutions that grow your organization.

  Our team of over 50 experts will work with you from the initial consultation to delivery and further development. We believe that your challenges require a customized approach. That is why we ensure that every solution fits your organization and goals. Our expertise and tools are always up-to-date. From user-friendly UIs, modern development techniques, AI experts and our own ethical hackers and pentesters.

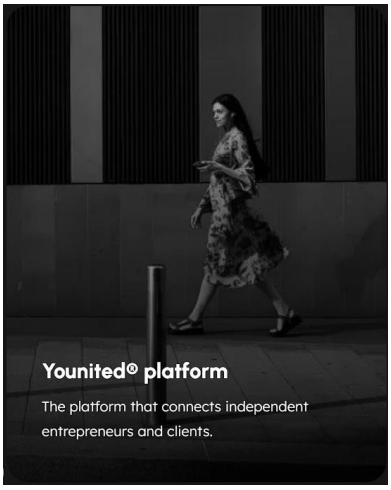
- **Our Motto :**

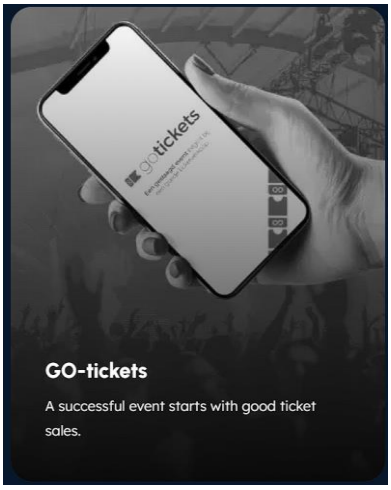
  'Create Together, inspire the World", That is our motto and our mission. Our people work closely with your organization. We advise, think along and look beyond the initial question. This is how we make the difference together with you.

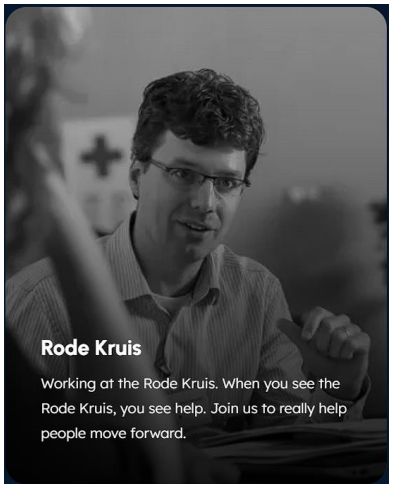
- **Our Location**

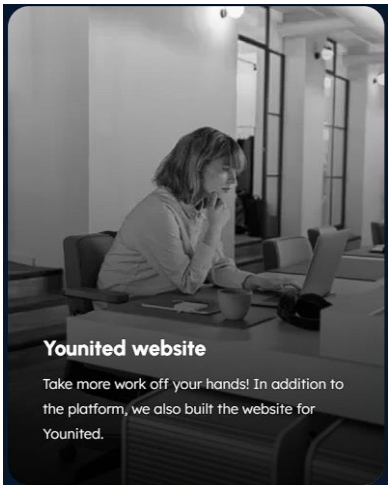
  We are located in the Innovation Valley building in Doetinchem (The Netherlands). An inspiring environment where cooperation and innovation go hand in hand. Innovation Valley offers not only modern office space, but also a unique working environment that promotes health, creativity and sustainability.

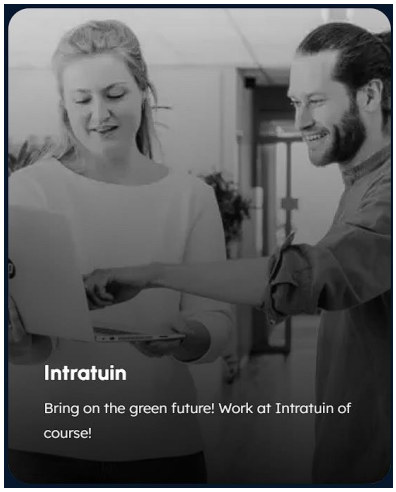
1.2 Different Products :

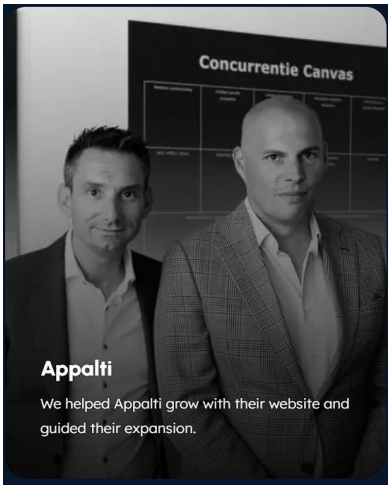












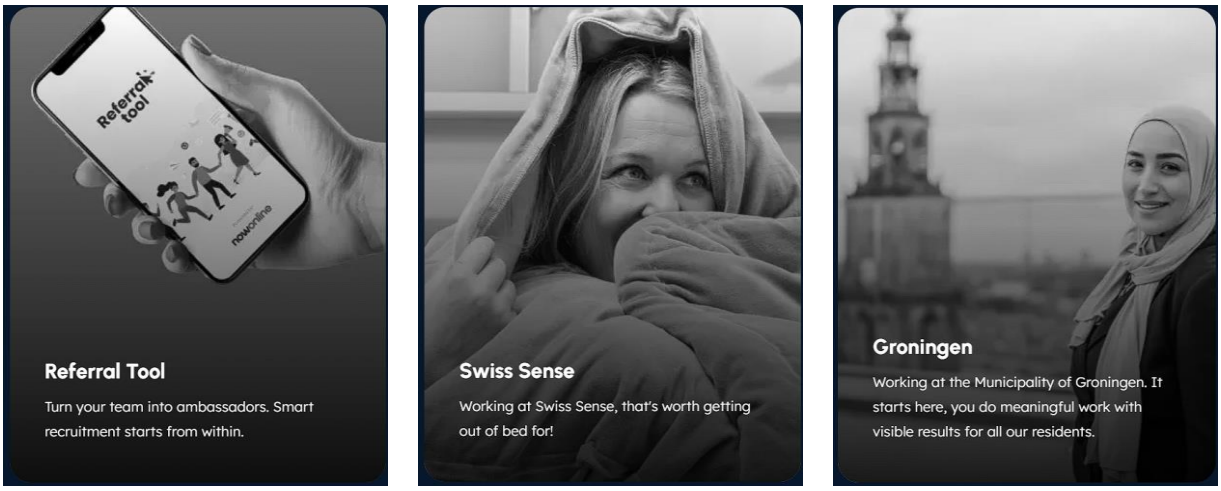


Fig 1.2.1 Products / Services provided by NowOnline

1.3 Organization Chart :

Solutions Provided at NowOnline

1) **Software :**
   - NowOnline advises, builds, and creates custom software. Digital solutions that help your organization grow. Solutions where functionality, efficiency, and innovation take center stage. With user-friendly environments tailored for your target audience.
2) **Websites :**
   - NowOnline designs, develops, and manages custom websites that tell your unique story. Whether your website is meant to inspire, inform, attract new clients, or all of the above: together, we'll create a website that achieves your online goals.
3) **Career Websites**
   - NowOnline designs, builds, and manages (customized) careers websites. Your careers website will, of course, be integrated with the ATS you use. Together, we create a careers website that targets your audience and potential candidates. We showcase company culture, who you are as an employer, and who future colleagues are. Enrich vacancies with storytelling, beautiful imagery, and detailed background information. This way, we make the difference in attracting talent and ensuring your organization attracts the right people.
4) **(Mobile) Apps :**
   - NowOnline advises, designs, and develops mobile apps that help your business grow. Apps that lead the way in user-friendliness and innovation and make a difference.

5) **Content :**
   - At NowOnline, we create content and copywriting that help your organization grow. From SEO-optimized texts to engaging social posts and blogs—together, we tell the story that elevates your brand to the next level.

6) **Data Security**
   - At NowOnline, we offer advanced data security solutions to keep your organization protected. From black box hacking to comprehensive pentests, we identify vulnerabilities and secure your systems. Our goal? To safeguard your data and that of your customers, so you can operate worry-free.

2. Overview of Different Department

2.1 Details About Work Carried out in each Department :

1) **Planning and Setup :**
   - The planning and setup phase involves defining the project scope, creating a roadmap, setting milestones, and identifying key requirements. It includes setting up development environments, configuring servers, version control, and establishing communication channels. Resource allocation, risk assessment, and creating detailed timelines are also handled to ensure smooth execution.
2) **FrontEnd Development :**
   - FrontEnd development involves designing and implementing the user interface using technologies like HTML, CSS, JavaScript, and frameworks such as React or Angular. It includes creating responsive layouts, handling user interactions, and ensuring cross-browser compatibility. Focus is placed on performance, accessibility, and delivering a smooth user experience through intuitive design and fast-loading components.
3) **BackEnd Development :**
   - BackEnd development focuses on creating the server-side logic using frameworks like .NET. It involves developing APIs, setting up database structures, handling authentication and authorization, and managing data flow between the client and server. Scalability, security, and data integrity are key priorities during implementation.
4) **Android Development :**
   - Android development involves creating mobile applications using Kotlin or Java. It includes building UI components, integrating APIs, managing user authentication, and ensuring app compatibility across different Android versions. Performance optimization, data security, and user experience are prioritized to ensure smooth functionality on various devices.
5) **IOS Development :**
   - iOS development involves building mobile applications using Swift or Objective-C. It includes designing user interfaces, integrating third-party libraries, and handling user authentication and data storage. Compatibility with different iOS versions, performance optimization, and compliance with Apple's guidelines are essential for successful deployment on the App Store.
6) **Testing and Quality Assurance :**
   - Testing and Quality Assurance (QA) involve ensuring that the application meets functional, performance, and security requirements. It includes unit testing, integration testing, system testing, and user acceptance testing (UAT) using tools like Selenium, JUnit, and Postman. QA focuses on identifying and fixing bugs,

ensuring cross-platform compatibility, and validating user experience. Performance testing ensures the system can handle expected load, while security testing identifies vulnerabilities. Continuous feedback loops and automated test cases are implemented to maintain high code quality and reliability.

7) Artificial Intelligence and data analyst :

- AI and data analysis involve building models using TensorFlow, PyTorch, or Scikit-learn to extract insights from data. Tasks include data preprocessing, training models, and evaluating performance. AI models may handle tasks like recommendation systems, image recognition, and natural language processing. Data visualization and reporting provide actionable insights for business decisions.

3. Introduction to Project

3.1 Project / internship summary :

Planning a well-structured travel itinerary is often a time-consuming and complex task, requiring users to manually research destinations, optimize schedules, and account for travel constraints. Traditional trip planning methods lack automation and personalization, leading to inefficient schedules and missed opportunities. This project aims to develop an AI-based Itinerary Planner that automates the process of creating personalized travel plans using Python and the CREWAI framework. The primary objective is to generate structured, time-optimized itineraries based on user preferences, ensuring a seamless travel experience.

The project began with the development of a multi-agent system, where each agent is responsible for a specific aspect of itinerary planning. The system consists of three AI agents: one for selecting places based on user interests, another for gathering detailed information about the selected locations, and a third for generating a well-structured, day-wise itinerary with estimated time allocations. This structured approach enables dynamic and intelligent trip planning that adapts to various user needs. However, initial development revealed key challenges such as dependency on prompt engineering, delays in processing due to computational constraints, and the need for better communication between agents to ensure coherence in the generated itineraries.

To address these challenges, the project is currently focused on refining prompts for improved accuracy, optimizing inter-agent communication for smoother data flow, and enhancing processing efficiency. Future improvements include integrating a real-time flight API to provide accurate travel updates, implementing an intuitive user interface for seamless interaction, and allowing users to review and modify the generated itineraries for greater flexibility.

By combining AI-driven automation with structured itinerary generation, this project seeks to provide an intelligent and scalable trip-planning solution. The expected outcomes include reduced manual effort, enhanced travel efficiency, and a more personalized travel experience. Through the integration of Python, CREWAI, and AI-driven decision-making, this project aims to revolutionize the way users plan and organize their trips.

3.2 Purpose :

The purpose of this project is to develop an AI-based itinerary planning system that automates the creation of personalized travel schedules. Traditional trip planning methods require users to manually research destinations, optimize routes, and allocate time efficiently, making the process tedious and time-consuming. This project aims to leverage AI agents within the CREWAI framework to streamline the planning process by dynamically selecting destinations, gathering relevant details, and generating structured day-wise itineraries based on user preferences.

By utilizing AI-driven decision-making, the system will ensure optimized scheduling while minimizing user effort. The project focuses on enhancing inter-agent communication, refining prompt accuracy, and integrating real-time data sources such as flight information to improve travel recommendations. Ultimately, the goal is to provide an intelligent and adaptive itinerary planner that enhances travel efficiency, reduces planning complexity, and delivers a seamless user experience.

3.3 Objective :

The objective of this project is to develop an AI-based itinerary planning system that automates the trip-planning process by leveraging AI agents within the CREWAI framework. The key objectives are:

- **To analyze user preferences and travel constraints -** The system will gather user preferences such as destination interests, budget, travel duration, and activity preferences. This data will be processed to ensure a personalized and optimized itinerary generation.
- **To implement a structured AI-driven itinerary generation process -** Develop AI agents that handle different aspects of trip planning, including place selection, travel route optimization, and scheduling, to ensure a well-structured day-wise itinerary.
- **To enhance inter-agent communication for better decision-making -** Improve the flow of information between AI agents to ensure seamless coordination in selecting places, gathering details, and finalizing a travel schedule.
- **To refine prompt engineering for improved accuracy -** Continuously optimize AI-generated responses by refining prompts, enabling more accurate and context-aware itinerary recommendations.
- **To integrate real-time travel data sources -** Incorporate external APIs for flight details, local transportation options, and weather conditions to enhance itinerary relevance and practicality.
- **To provide a user-friendly interface for reviewing and modifying plans -** Develop a UI-based solution where users can review AI-generated itineraries, make modifications, and customize their travel plans as needed.
- **To evaluate and improve system efficiency -** Assess the performance of the itinerary planning system based on factors like response time, accuracy of recommendations, and user satisfaction to refine the model further.

3.4 Scope :

The scope of this project defines the boundaries and limitations of the AI-based itinerary planning system developed using Python and the CREWAI framework. The system aims to automate trip planning by leveraging AI agents to generate optimized travel schedules based on user preferences.

1) What the System Can Do:

- **User Preference Analysis** – The system can collect and analyse user preferences such as travel interests, budget, duration, and activity choices to create personalized itineraries.

- **AI-Driven Place Selection** – The system can intelligently select travel destinations and points of interest based on user preferences and relevance.
- **Detailed Itinerary Generation** – The system can generate structured day-wise travel plans, including time-based scheduling of activities, ensuring a well-organized itinerary.
- **Inter-Agent Communication** – The system enables AI agents to collaborate, improving decision-making in selecting places, refining schedules, and optimizing trip plans.
- **Scalability** – The system is designed to accommodate various types of trips, from single-day outings to multi-day travel plans, and can be expanded to include additional features.

2) What the System Can't Do:

Real-Time Travel Booking – The system does not include direct booking functionalities for flights, hotels, or activities.

Live Traffic and Route Optimization – The system will not provide real-time navigation updates or suggest alternative routes based on live traffic data.

Weather Adaptation – The system does not factor in real-time weather conditions when generating itineraries.

Multilingual Support – The initial implementation will support itinerary generation in English only, without multilingual capabilities.

Direct Integration with External Platforms – The system does not integrate directly with travel platforms like Google Maps, TripAdvisor, or Booking.com for automated reservations or live data retrieval.

3.5 Technology and Literature Review :

Technological Review :

This project utilizes a combination of artificial intelligence, web development, and API-based data retrieval to build an AI-powered itinerary planning system. The key technologies involved include Python, Flask, CREWAI, and various external APIs for real-time travel data. The technological foundation integrates AI-driven decision-making, data processing, and web-based interactions to provide a seamless travel planning experience. Below is a detailed review of the technologies used:

1) **AI Framework: CREWAI**
   - ➢ CREWAI is an AI agent framework designed to facilitate structured and goal-oriented decision-making using multiple agents. In this project, CREWAI is used to create and manage AI agents that handle different aspects of itinerary planning.
   - ➢ Key Features of CREWAI:
     - Multi-agent collaboration for stepwise itinerary generation.
     - Task-specific AI agents for place selection, exploration, and itinerary structuring.
     - Seamless integration with external APIs for dynamic data retrieval.
     - Modular and scalable design to expand itinerary customization capabilities.
   - ➢ CREWAI ensures that the system generates optimized itineraries based on user preferences, providing personalized travel plans efficiently.

2) Programming Language: Python

- ➢ Python is the core programming language for this project due to its flexibility, extensive library support, and ease of integration with AI models and web frameworks.
- ➢ Relevant Libraries :
  - **Flask** – A lightweight web framework for handling user requests, API interactions, and serving the front end.
  - **Requests** – Used to fetch data from external travel APIs.
  - **JSON** – Essential for structuring and exchanging data between AI agents and the backend system.
  - **Ollama** – A platform for running large language models (LLMs) locally to enhance AI-driven itinerary generation.
- ➢ Python's simplicity and extensive ecosystem allow for rapid development, testing, and deployment of AI-based itinerary planning.

3) APIs & External Tools:

- ➢ The system integrates multiple APIs to fetch real-time travel data, ensuring up-to-date recommendations for flights, hotels, and local attractions.
- ➢ Key APIs Used:
  - **Amadeus API** – Provides real-time flight information, allowing users to explore travel options based on availability and pricing.
  - **Hotel & Accommodation APIs** – Fetches lodging options to suggest suitable stays based on user budget and location preferences.
  - **Serper API** – Enables fast and efficient Google search-based data retrieval to enhance place exploration and itinerary refinement.
- ➢ These APIs help bridge the gap between AI-generated travel plans and real-world data, improving the reliability of suggested itineraries.

4) Web Development & Deployment

- ➢ Flask as the Backend Framework:
  - Flask is used to manage user interactions, API requests, and itinerary generation logic.
  - It serves as the intermediary between AI agents and external data sources.
- ➢ Development Platforms:
  - **VSCode** – Used for local development, debugging, and deploying the backend services.
- ➢ **Operating System:** Windows is used for local testing and development before cloud deployment.

5) System Limitations & Future Enhancements

➢ **Current Limitations:**

- The system does not handle real-time hotel bookings but provides available options for user reference.
- The implementation is currently **English-only**, with no multilingual support.
- While real-time traffic updates are integrated for route optimization, the system does not yet account for sudden disruptions such as accidents or road closures.
- The system relies on external APIs for data retrieval and does not support offline itinerary planning.
- The AI agent does not handle **currency conversion** for cost calculations.
- **Locally hosted Ollama-based LLMs** are slower and less accurate compared to cloud-based models like OpenAI and Claude.

➢ **Potential Future Enhancements:**

- Enhancing AI agents with **adaptive learning** to refine travel suggestions based on user preferences and feedback.
- Expanding **multilingual support** to improve accessibility for a global audience.
- Integrating a **hotel booking feature**, allowing users to make reservations directly through the platform.
- Incorporating **predictive analytics** to anticipate congestion patterns and suggest alternate routes ahead of time.
- Implementing **currency conversion**, enabling the AI agent to generate cost estimates in the user's preferred currency.
- Replacing **local LLMs with cloud-based LLMs** to improve response speed and accuracy.

Literature Review :

Trip planning has evolved significantly over the years, from traditional guidebooks and manual research to AI-driven itinerary planners. Early travel recommendation systems relied on rule-based approaches, where predefined heuristics suggested destinations based on user preferences (Ricci et al., 2002). However, these systems lacked personalization and adaptability to real-time conditions.

The advent of recommender systems, leveraging collaborative filtering and content-based filtering (Adomavicius & Tuzhilin, 2005), improved trip suggestions by analyzing historical data and user preferences. However, these methods struggled with cold-start problems and failed to incorporate real-time factors like weather, traffic, and availability.

Recent advancements in AI and deep learning have introduced more sophisticated itinerary planners. Knowledge-based systems (Gavalas et al., 2014) enhanced personalization by integrating

user constraints, but they were limited by static data sources. Reinforcement Learning (RL) models, such as those by Chen et al. (2018) and Li et al. (2021), demonstrated improved adaptability by dynamically optimizing travel plans based on evolving constraints. However, these models required extensive training data and faced challenges in generalizing across different travel scenarios.

Real-time data integration has become crucial in modern itinerary planning. APIs for flights (e.g., Amadeus API) and accommodations (e.g., Hotel Booking APIs) provide up-to-date availability, allowing AI agents to generate feasible plans (Tussyadiah & Miller, 2020). Additionally, traffic-aware routing has gained prominence, with real-time traffic updates enabling optimized travel schedules (Zhang et al., 2022). Despite these advancements, many existing systems still struggle with sudden disruptions such as accidents and road closures.

The rise of Large Language Models (LLMs) has further transformed AI-driven trip planning. Platforms like OpenAI's GPT and local models such as Ollama allow natural language interactions for more user-friendly itinerary generation. However, locally hosted LLMs often suffer from slower inference speeds and lower accuracy compared to cloud-based models, limiting their effectiveness in real-time applications.

This project addresses these challenges by integrating real-time data from APIs, AI-driven adaptive learning, and LLM-based itinerary generation. By leveraging the CREWAI framework for AI agent coordination and optimizing route selection with real-time traffic data, the system aims to provide a more dynamic, efficient, and personalized travel planning experience. Future advancements could focus on improving multilingual support, predictive analytics for congestion forecasting, and seamless integration with hotel booking platforms to enhance usability and accuracy.

3.6 Project / Internship Planning :

3.6.1 Project Approach and Justification :

Approach

- **Preparation and Learning Phase :** The initial phase focused on understanding AI-based itinerary planning and familiarizing with the CREWAI framework. This included setting up Python, Flask, and the required AI libraries, along with studying different itinerary generation methods. The focus was on developing AI agents capable of selecting locations, analyzing real-time travel constraints, and structuring personalized trip plans. Since real-time traffic information was a critical component, APIs for traffic updates and place availability were explored and integrated.
- **Development Phase :** The system was built using a **multi-agent AI approach**. Three AI agents were designed:

- **Location Selection Agent:** Suggested places based on user interests, ensuring diversity and personalization.
- **Exploration Agent:** Provided additional details on selected places, such as descriptions, reviews, and opening hours.
- **Itinerary Generation Agent:** Created structured, time-optimized plans based on user preferences and real-time constraints, including traffic conditions.

- The **Flask-based front end** was developed for user interaction, allowing users to input their preferences and receive dynamically generated itineraries. **External APIs** were integrated for fetching place details.
- **Testing and Evaluation Phase :** The system was tested on different travel scenarios to ensure itinerary feasibility, **accuracy of recommendations, and adaptability to different user inputs**. Key evaluation metrics included:
  - **Itinerary coherence:** Assessing whether the generated plans follow a logical sequence of activities.
  - **User satisfaction:** Evaluating the relevance and diversity of recommended places.
  - **System efficiency:** Measuring response times and performance, particularly when using local vs. cloud-based LLMs.

Justification

- **AI-driven itinerary planning** was chosen over traditional static methods to provide **dynamic, personalized, and automated trip generation**. Unlike conventional travel planners that rely on **fixed datasets**, this system dynamically **selects and structures places** based on user preferences.
- **Multi-agent AI architecture** enabled structured **modular functionality**, where each agent specialized in a specific task, making the system **more scalable and efficient**. This approach ensured that **place selection, itinerary structuring, and trip customization** were handled separately for better adaptability.
- **Hybrid LLM usage** was necessary for balancing **cost, speed, and accuracy**. **Local models** were used for **basic itinerary structuring**, while **cloud-based LLMs (such as OpenAI's GPT or Claude)** were leveraged for **more complex itinerary refinements**. This approach ensured both **efficiency and affordability**.

3.6.2 Project Effort, Time and Cost estimation :

Activity	Estimated Duration
Project Initialization	1 day
Learning requirements	2 weeks
Project Design	2 weeks
Development	5 weeks
Testing & Evaluation	2 weeks
Fine-tuning & Optimization	2 weeks
Report Writing	3 days
Total Effort	**14 weeks (Approx. 3.5 months)**

Table 3.6.2 Internship Time

Cost Estimation :

- The cost estimation for the project involves various factors, including personnel expenses, software and hardware costs, and operational overheads. Personnel expenses will cover development time, including research, coding, testing, and optimization of AI agents for itinerary planning.
- **Software costs** will be minimal as most tools, including Python, Flask, and CREWAI, are open-source. The project also relies on free-tier cloud storage (Google Drive) and local computing resources, reducing licensing expenses.
- **Hardware costs** will depend on the computational requirements for running AI models efficiently. If processing large datasets or running AI agents locally, additional RAM, a capable CPU, or GPU upgrades may be necessary to improve performance.
- **Operational expenses** may include internet costs, temporary cloud storage for file processing, and hosting costs for deploying the system. Power consumption for running the model and maintaining uptime for web services should also be considered.
- A **contingency budget** may be required to accommodate unexpected expenses, such as upgrading temporary cloud storage, API rate limits, or additional debugging and optimization. A detailed cost analysis will ensure efficient financial planning and resource allocation throughout the project lifecycle.

3.6.3 Roles and Responsibilities :

1) **Project Manager :**
   - Oversee the overall project development and ensure that milestones are met on time.
   - Coordinate between different team members and manage resource allocation.
   - Monitor progress, address roadblocks, and ensure the project stays within scope and budget.
2) **AI / ML Developer :**
   - Develop and refine AI agents for itinerary planning.
   - Implement NLP techniques to process and understand user preferences.
   - Optimize agent performance by fine-tuning prompt structures and response generation.
   - Integrate AI models with the backend for seamless interaction.
3) **Frontend Developer:**
   - Design and develop a user-friendly web interface for itinerary planning.
   - Integrate AI-generated results with an interactive UI.
   - Ensure responsiveness and cross-platform compatibility.
   - Implement features like itinerary customization and report downloads.
4) **Data Analyst:**
   - Analyze user inputs and generated itineraries for quality improvement.
   - Evaluate AI agent performance based on relevance, accuracy, and efficiency.
   - Generate reports on system usage and identify areas for enhancement.
   - Provide insights to refine AI agent recommendations.
5) **Deployment Engineer:**
   - Deploy the web application on a suitable cloud platform.
   - Implement cloud storage integration for document management.
   - Optimize server performance for faster processing.
   - Ensure data security and manage temporary storage solutions.
6) **Technical Writer:**
   - Document the project development process, methodologies, and results.
   - Prepare project reports and presentations.
   - Ensure clear and structured documentation for future reference and scalability.

3.7 Project / Internship Scheduling (Gantt Chart):

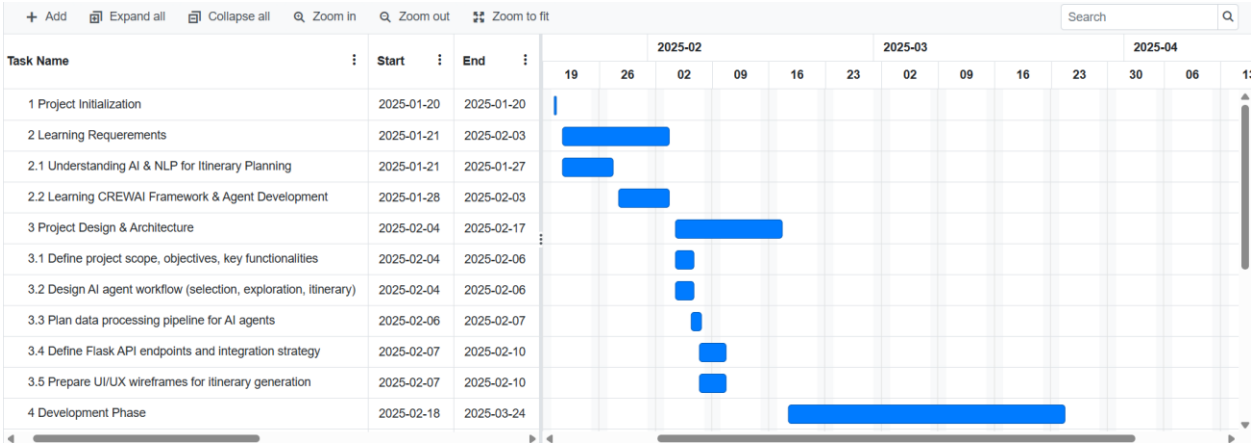


Fig 3.7.1 Gantt Chart

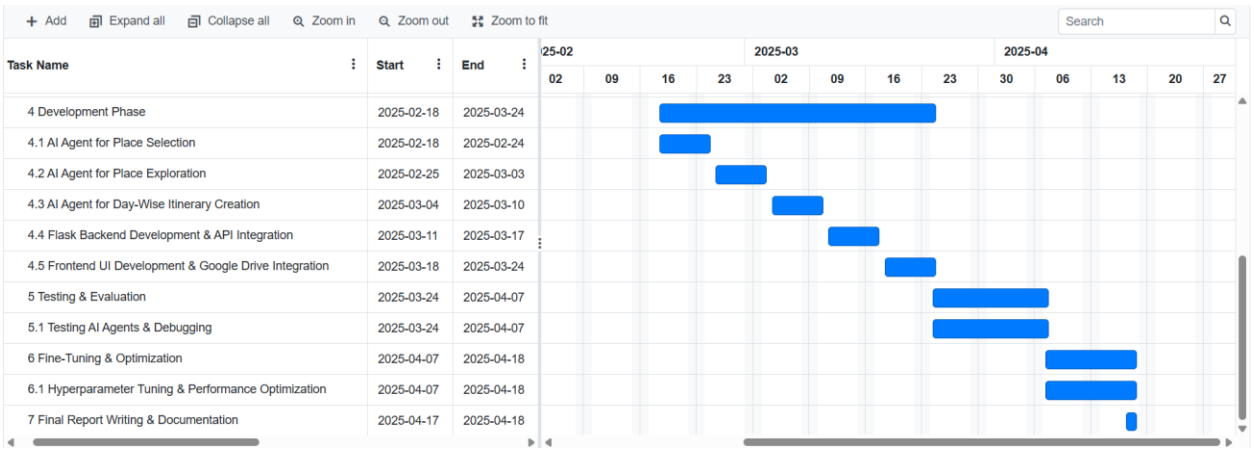


Fig 3.7.2 Gantt Chart

4. System Analysis

4.1 Study of Current System

Introduction

Itinerary planning is an essential aspect of travel, helping users organize their trips efficiently based on preferences, time constraints, and available attractions. Traditional trip planning methods primarily rely on manual research, static travel guides, and predefined tour packages. While these approaches have been effective in the past, they lack personalization, adaptability, and real-time optimization.

Manual Trip Planning

Many travelers still plan their trips manually using travel blogs, guidebooks, and personal recommendations. This process involves extensive research on destinations, transportation, accommodations, and scheduling. However, manual trip planning has several limitations:

- Time-consuming and requires significant effort.
- Difficult to optimize travel time and avoid unnecessary delays.
- Lack of flexibility to adjust itineraries based on user preferences dynamically.

Predefined Tour Packages

Travel agencies offer predefined tour packages that cover popular destinations with fixed schedules and routes. While these packages provide convenience, they often fail to meet the specific interests of individual travellers. Some common drawbacks include:

- Limited customization options for users.
- Fixed schedules that may not align with personal preferences.
- Inflexibility in modifying itineraries based on changing circumstances

Basic Travel Recommendation Systems

Some online platforms and travel apps suggest places to visit based on user ratings, reviews, and popularity. However, these systems are often **rule-based** and do not provide **AI-driven personalized** itineraries. Their main limitations are:

- Lack of advanced AI-based customization.
- No **context-aware** decision-making based on user interests.
- Inability to generate **optimized multi-day itineraries** with time allocation.

4.2 Problems and Weakness of Current System

Despite advancements in travel planning technology, existing trip planning systems still have several limitations that affect their ability to create optimal and personalized itineraries. These weaknesses arise due to the system's lack of real-time adaptability, inefficient recommendation strategies, and inability to handle user-specific preferences dynamically. These issues result in poor user experiences, inefficient time allocation, and suboptimal trip schedules.

1. Lack of Personalization and Dynamic Adaptability

One of the biggest limitations of current itinerary planning systems is their inability to generate truly personalized trip schedules dynamically.

1) **One-Size-Fits-All Approach :**
   - Many existing systems use **rule-based recommendations** that suggest popular places **without considering** personal interests.
   - Users often receive **generic** itinerary suggestions, which **do not align** with their specific preferences.
   - There is no **real-time itinerary adjustment** based on changing circumstances, such as delays or new interests.
2) **Rigid Itinerary Planning**
   - Most existing trip planners create **fixed schedules** that do not allow changes once the itinerary is generated.
   - Travelers may face issues such as:
     - **Time conflicts** between planned activities
     - **Unrealistic travel durations** between destinations
     - **No flexibility** for spontaneous changes or detours

2. Inefficient Recommendation Systems

Most current trip planners **fail to optimize the selection of attractions, restaurants, and activities** in an intelligent manner.

1) **Over-Reliance on Reviews and Ratings**
   - Many platforms rely only on user ratings (e.g., Google Reviews, TripAdvisor) to suggest places.
   - However, these do not consider factors such as travel time, budget, or personal preferences.
   - Some places may be highly rated but not suitable for the user's available time and schedule.
2) **Lack of Context Awareness**
   - **Existing systems do not adapt recommendations based on:**
     - **Weather conditions** (e.g., suggesting indoor activities during rain).
     - **User travel history** (e.g., avoiding places the user has already visited).
     - **Real-time events** (e.g., recommending special exhibitions or festivals).

3. Poor Time and Route Optimization

Many existing trip planners struggle to **efficiently allocate time** for activities and optimize routes.

1) **Inefficient Travel Time Estimation**
   - Some systems **underestimate or overestimate** travel times between locations, leading to:
     - **Rushed visits** to attractions.
     - **Unplanned delays** and missed activities.
2) **No Real-Time Traffic Consideration**
   - Many planners fail to account for real-time traffic, leading to poor route selection.
   - Users may spend excessive time commuting between destinations instead of enjoying activities.

4. Inability to Learn from User Behavior

Most traditional trip planning tools **do not have AI-driven learning mechanisms**.

1) **No Adaptive Learning from User Preferences.**
   - Existing systems do not remember user preferences from past trips.
   - The planner does not refine its suggestions based on user choices over time.
2) **No AI-Based Smart Itinerary Adjustments**
   - There is no mechanism to automatically adjust schedules based on real-time conditions.
   - Users must manually edit their itineraries when faced with unexpected delays or changes.

5. Inefficient Handling of Group Travel and Collaborations

Most systems are designed for solo travelers, making group itinerary planning complex and inefficient.

1) **No Multi-User Coordination.**
   - Many platforms lack features for group members to vote on places or suggest itinerary modifications collaboratively.

b) Poor Scheduling for Group Preferences

- There is no AI mechanism to balance different interests, budgets, and availability among group travelers.

4.3 Requirements of New System

To overcome the limitations of existing trip-planning solutions, the new AI-based itinerary planner must incorporate intelligent, real-time, and user-centric features. The system should be able to dynamically adapt, personalize travel plans, optimize routes, and enhance the overall trip experience. Below are the key requirements for the new system:

1) **AI-Based Dynamic Itinerary Adjustment**
   - Real-time adaptation of the itinerary based on changing conditions.
   - The system should ensure seamless adjustments without requiring manual modifications.

2) **Real-Time Data Collection and Processing**
   - The system must **gather real-time data** from various sources:
     - **Weather forecasts** to suggest suitable activities.
     - **Event data** (e.g., festivals, concerts, exhibitions).
   - This data should be **processed using AI** to generate the most efficient trip schedules.
3) **Intelligent Destination and Activity Recommendation**
   - The planner should **use AI-based recommendation algorithms** to:
     - Suggest attractions based on user interests.
     - Filter recommendations by budget, travel time, and activity type.
     - Prioritize must-visit places while maintaining travel efficiency.
   - The system should be capable of learning from user feedback to refine future recommendations.
4) **Personalized User Experience**
   - The itinerary planner should provide:
     - Customizable trip preferences (e.g., food preferences, walking vs. taxi).
     - User profile-based recommendations (learning from past trips).
     - Flexible scheduling (allowing free time for exploration).
   - The system should be designed to enhance user satisfaction by offering a truly tailored travel experience.
5) **User-Friendly Interface:**
   - The planner should feature a **simple, intuitive interface** allowing users to:
     - **Easily modify** their schedules.
     - **View trip details and travel routes** in a clean, organized manner.
     - **Access their itinerary offline** in case of network issues.

4.4 System Feasibility

4.4.1 Does the system contribute to the overall objectives of the organization?

Yes, the proposed system significantly contributes to the overall objective of improving traffic efficiency and reducing congestion. By dynamically adjusting traffic signal timings based on real-time traffic data and predicted future states, the system aims to minimize average waiting time, improve traffic flow, and reduce vehicle idling time. This will result in lower fuel consumption and reduced emissions, aligning with broader goals of enhancing urban mobility and environmental sustainability. Additionally, improved traffic management will lead to better driving experiences and reduced frustration among commuters, thereby increasing public satisfaction with transportation infrastructure.

4.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints?

Yes, the system is implemented using open-source tools and free-tier APIs, making it cost-effective. It is developed in Python with Flask for backend functionality and CREWAI for AI-driven itinerary generation. Free-tier APIs, including Amadeus for flights, hotel APIs for lodging, and Serper API for travel-related searches, enable real-time data retrieval. Ollama allows LLMs to run locally, reducing dependency on cloud-based services. Development is done in VSCode on Windows, ensuring a practical and budget-friendly implementation. The 14-week timeline is achievable with structured development phases.

4.4.3 Can the system be integrated with other systems which are already in place?

Yes, the system seamlessly integrates with real-time travel services through standardized APIs. It fetches flight details, hotel availability, and relevant search results while ensuring smooth communication between AI agents using JSON. The use of Ollama enables offline AI processing, enhancing privacy and performance. The modular architecture supports future integrations with additional services such as car rentals and local transport systems, ensuring scalability and adaptability.

4.5 Purposed System

The AI-based Itinerary Planner is an intelligent system that automates and optimizes trip planning using AI agents and real-time travel data.

Key Components :

1) Data Collection & Processing:
    - Fetches **real-time flight details** using Amadeus API.
    - Uses **Serper API for fast travel-related searches**.
    - Processes **user preferences** for a customized itinerary.
2) AI-Based Itinerary Generation (CREWAI):
    - AI agents analyze travel options and generate an optimal schedule.
    - Dynamic updates allow itinerary modifications based on real-time events.
3) Route Optimization & Time Management:
    - AI optimizes travel routes and schedules to maximize efficiency.
    - Adjusts plans based on flight delays, hotel availability, or user changes.
4) User-Friendly Interface:
    - Developed using Flask for smooth API handling and backend processing.
    - Allows manual modifications to the generated itinerary.
    - Provides PDF and digital itinerary exports for offline use.
5) Scalability and Integration:
    - Designed to operate with any place for any number of travelers.
6) Monitoring and Control Interface:

- A user-friendly interface will allow to modify the content of an Agent before passing it to the next Agent.

5. System Design

5.1 System Design & Methodology

- The System Design & Methodology for this project follows an AI-driven, agent-based architecture, integrating the CREWAI framework with web technologies like Flask and external services such as the Amadeus API.
- The project is designed to create an intelligent travel itinerary planning system. It aims to enhance the user experience by leveraging collaborating AI agents for suggesting destinations, exploring place details, planning multi-day trips, and incorporating real-time flight information.
- Key techniques employed include multi-agent collaboration using CREWAI's flow concept, prompt engineering for refining agent outputs, API integration for external data retrieval, and developing an interactive web-based user interface for input collection and result presentation, including options for users to modify intermediate agent outputs.

5.2 System Architecture

The system is structured into the following core layers:

1) **User Interaction & Data Collection Layer (Frontend)**
    - This layer is responsible for collecting user travel requirements (like destination preferences, dates, interests, budget) through a web interface.
    - Flask framework is used to handle web requests and serve HTML templates.
    - HTML/CSS/JavaScript are used to build the user interface forms for input and display areas for results.
    - Includes functionality for users to review and modify agent-generated suggestions before they are passed to the next stage.
2) **Backend Orchestration & AI Agent Layer (Core Logic)**
    - This layer processes user inputs received from the frontend and manages the workflow between different AI agents.
    - Python serves as the primary backend language.
    - The CREWAI framework is used to define, configure, and run the AI agents (Place Selector, Place Explorer, Trip Planner) organized into separate crews communicating via a flow.
    - Flask manages the backend logic, routing requests to the appropriate CREWAI processes and handling data validation.
    - This layer orchestrates the sequence of agent tasks: place selection based on user input, exploring selected places, itinerary generation, and integration of flight data.
3) **External Services & Data Integration Layer**
    - This layer connects to external APIs to fetch supplementary data required for planning.

- The Amadeus API is integrated to search for real-time flight information based on the planned itinerary's origin, destination, and dates.
- Handles API request/response cycles, including authentication and error management.

4) **Result Presentation & Reporting Layer (Output)**
   - Presents the final generated travel itinerary, including place details, planned activities, and flight options, back to the user via the web interface.
   - Utilizes Flask to render the final results dynamically.
   - Features an interactive result display for itinerary visualization.
   - Includes mechanisms for notifications and user feedback.

5.2.1 System Architecture for AI Travel Planner using CREWAI & Flask

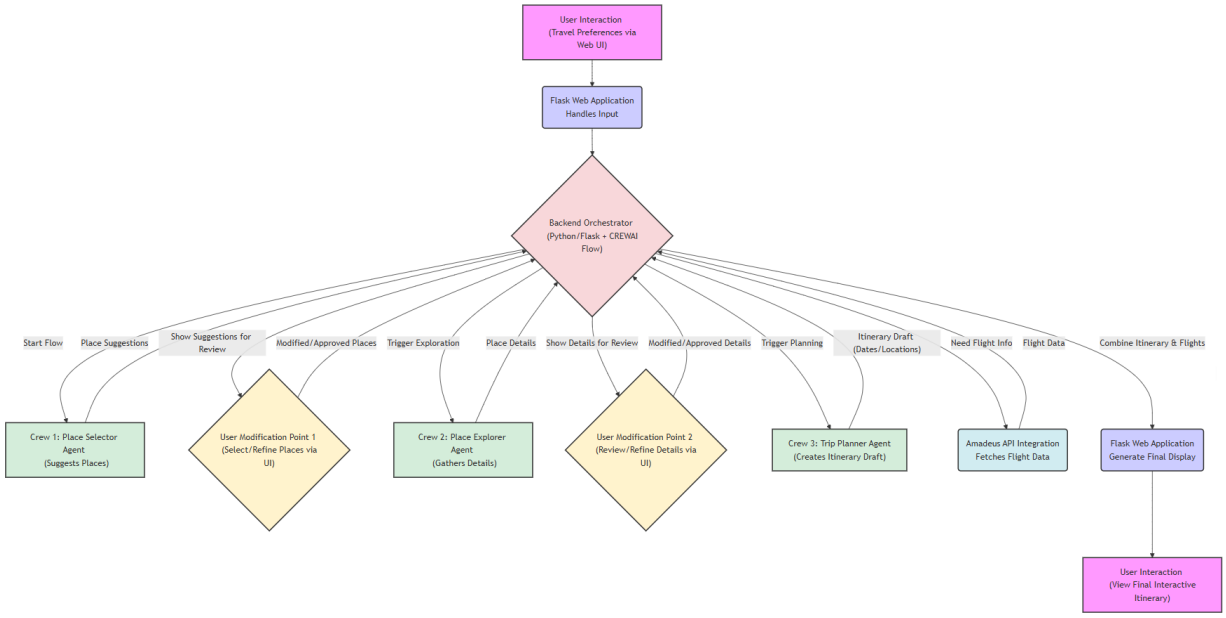


Fig 5.2.1 System Architecture

5.3 Methodology

The methodology employed in this project follows a structured, iterative development process focused on building an AI-powered travel planning application using the CREWAI framework. It includes the following key stages:

1) **Requirement Gathering & Research**
   - Understanding project goals and expectations through initial briefings.
   - Researching user needs for travel itinerary creation, defining necessary inputs, variables, and parameters.
   - Developing user stories, flow diagrams, and acceptance criteria to guide development.

- Conducting initial research into the CREWAI framework, AI agent architecture, capabilities, and best practices.

2) **Technology Setup & Environment Configuration**
   - Setting up the development environment with essential tools like VS Code, Python, and communication platforms.
   - Installing necessary libraries and dependencies for CREWAI implementation and web development.
   - Establishing connections and authentication for external services like the Amadeus API.
3) **AI Agent & Core System Implementation**
   - Learning the fundamentals of the CREWAI framework, including agent creation, tool integration, and memory mechanisms.
   - Implementing an initial crew structure with three distinct agents: **Place Selector**, **Place Explorer**, and **Trip Planner**.
   - Refactoring the architecture to a flow-based approach with separate crews for improved data handling and modularity.
   - Developing the backend logic using Python and Flask to manage agent orchestration and data flow.
   - Integrating the Amadeus API to fetch real-time flight data based on agent outputs.
4) **Frontend Development & User Interaction**
   - Designing UI wireframes and developing a web-based interface using the Flask framework to replace initial CLI interactions.
   - Implementing frontend components to collect user input and display agent-generated results.
   - Developing functionality to allow users to review and modify outputs generated by agents before passing them to subsequent steps.
   - Creating interactive elements for result display, itinerary visualization, notifications, and feedback.
5) **Testing, Refinement & Optimization**
   - Performing iterative testing of individual agents and the integrated crew/flow system via CLI and later the web interface.
   - Debugging data flow issues between agents and crews.
   - Conducting prompt engineering to analyse and refine agent instructions for improved output quality and consistency.
   - Enhancing UI/UX based on testing and feedback, including responsive design and improved error handling.
6) **Documentation & Finalization**
   - Documenting company policies, workflow procedures, and technical findings, including best practices and limitations discovered.

- Performing final code clean-up, comprehensive testing across various scenarios, and preparing final project documentation for handover.

6. Implementation

6.1 Implementation Platform / Environment

The development of the AI Travel Planner project was carried out primarily using **Visual Studio Code (VSCode)**, a versatile integrated development environment (IDE), facilitating efficient coding in **Python**. The core programming language used for backend logic, agent definition, and framework interaction was Python.

The application's backend and web interface were developed using the **Flask** web framework, allowing for rapid development and routing of user requests to the appropriate AI processing components. The core intelligence of the application relies on the **CREWAI** open-source framework, which was used to structure, manage, and orchestrate the AI agents responsible for travel planning tasks.

For the Large Language Model (LLM) capabilities required by the CREWAI agents, the system utilized **Ollama** to serve the **Llama 3.2** model locally. This setup provided the natural language understanding and generation capabilities needed for tasks like place selection and itinerary creation directly within the development environment.

External data integration was crucial. Real-time flight information was sourced using the **Amadeus API**, providing access to schedules and availability. General web research and data gathering for place exploration were powered by the **Serper API**, enabling agents to access up-to-date information from the internet.

The primary development and testing environment was based on **Windows OS**. Communication during development involved standard tools like Slack and email (as mentioned during onboarding).

6.2 Module Specification

The AI Travel Planner system consists of several interconnected modules designed to generate personalized travel itineraries based on user input, leveraging AI agents and external data sources. These modules include:

1) **User Interface & Input Module (Flask Frontend)**
   - This module provides the web-based interface for users to input their travel preferences, such as destination interests, dates, budget, and activity preferences. It uses Flask templates (HTML/CSS/JS) to render forms and collect data.
2) **Place Selection Module (CREWAI Agent 1)**
   - This module utilizes the first CREWAI agent, powered by the Llama 3.2 model via Ollama. It processes the user's initial input to research and suggest potential destinations or specific places of interest relevant to the user's request. It may leverage the Serper API for current information.
3) **Place Exploration Module (CREWAI Agent 2)**
   - Taking the selected places from the previous module (potentially refined by user feedback), this module employs the second CREWAI agent (using Llama 3.2 via

Ollama and the Serper API) to gather detailed information about each place, such as descriptions, key attractions, opening hours, and user reviews.

4) **User Modification & Feedback Module (Flask + CREWAI Interaction)**
   - This intermediary module allows the user to review the outputs from the Place Selection and Place Exploration agents via the Flask web interface. Users can approve, reject, or provide feedback to refine the selections before the final planning stage.
5) **Trip Planning Module (CREWAI Agent 3)**
   - The third CREWAI agent (using Llama 3.2 via Ollama) takes the finalized list of places and details, along with constraints like dates and duration, to construct a logical day-by-day itinerary structure, suggesting activity sequences and potential timings.
6) **Flight Integration Module (Amadeus API + Flask Backend)**
   - Based on the origin, destination, and dates determined by the Trip Planning Module, this module interacts with the Amadeus API via the Flask backend to search for relevant flight options.
7) **Itinerary Presentation Module (Flask Frontend)**
   - This module, part of the Flask application, takes the structured itinerary from the Trip Planner and the flight data from the Flight Integration Module to generate a final, interactive travel plan displayed to the user on the web interface.

6.3 Screenshots of the System

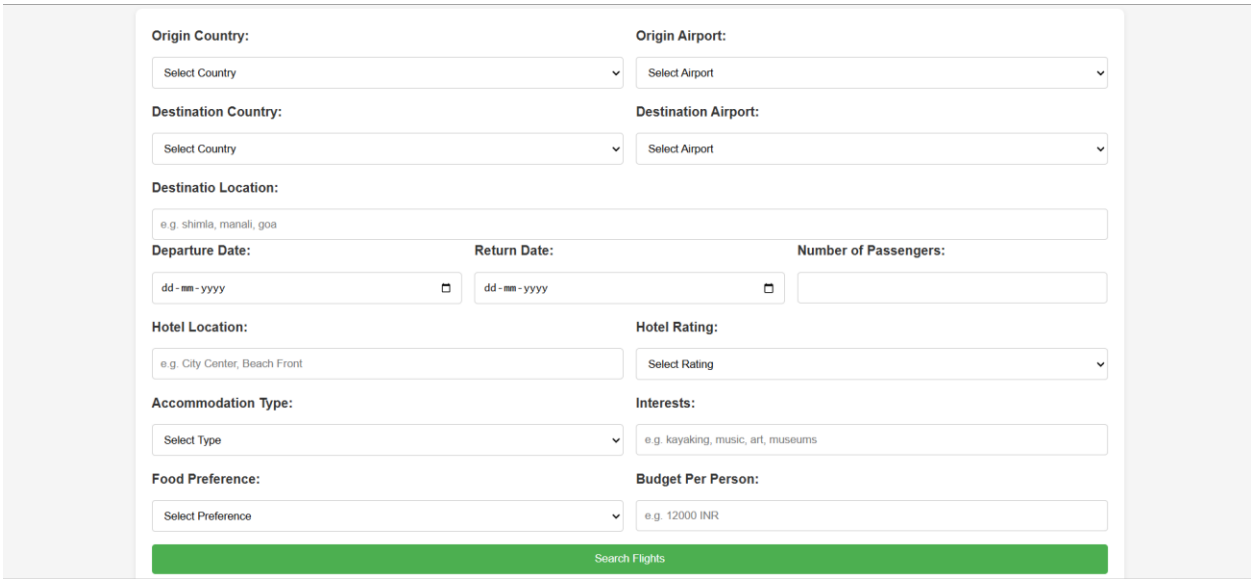


Fig 6.3.1 Home Page

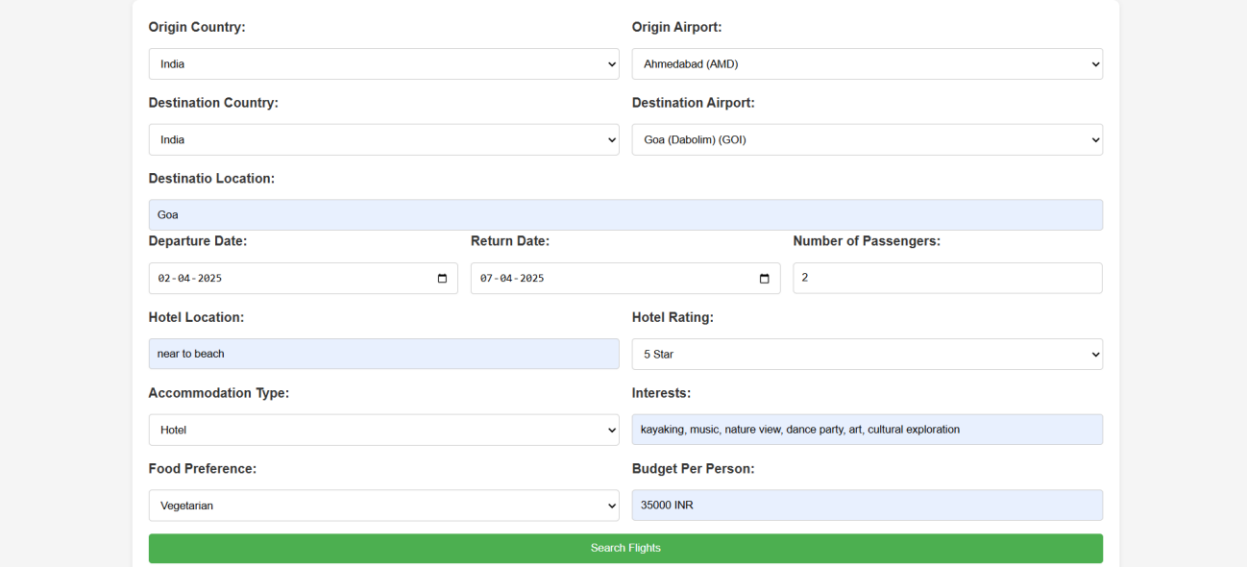


Fig 6.3.2 User Inputs

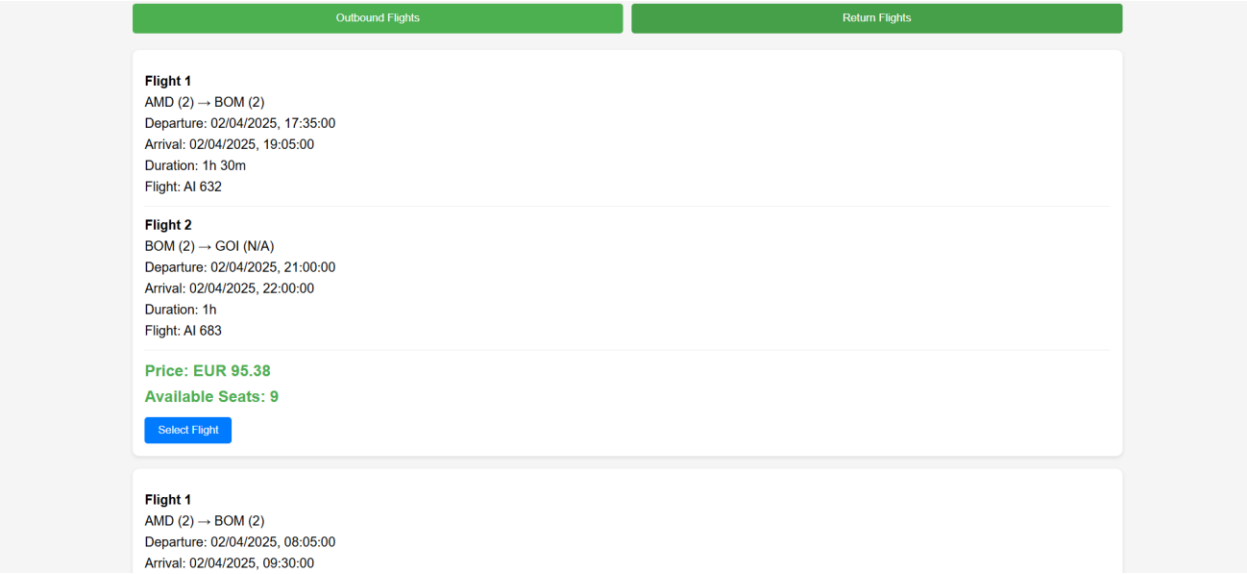


Fig 6.3.3 Selecting Flights

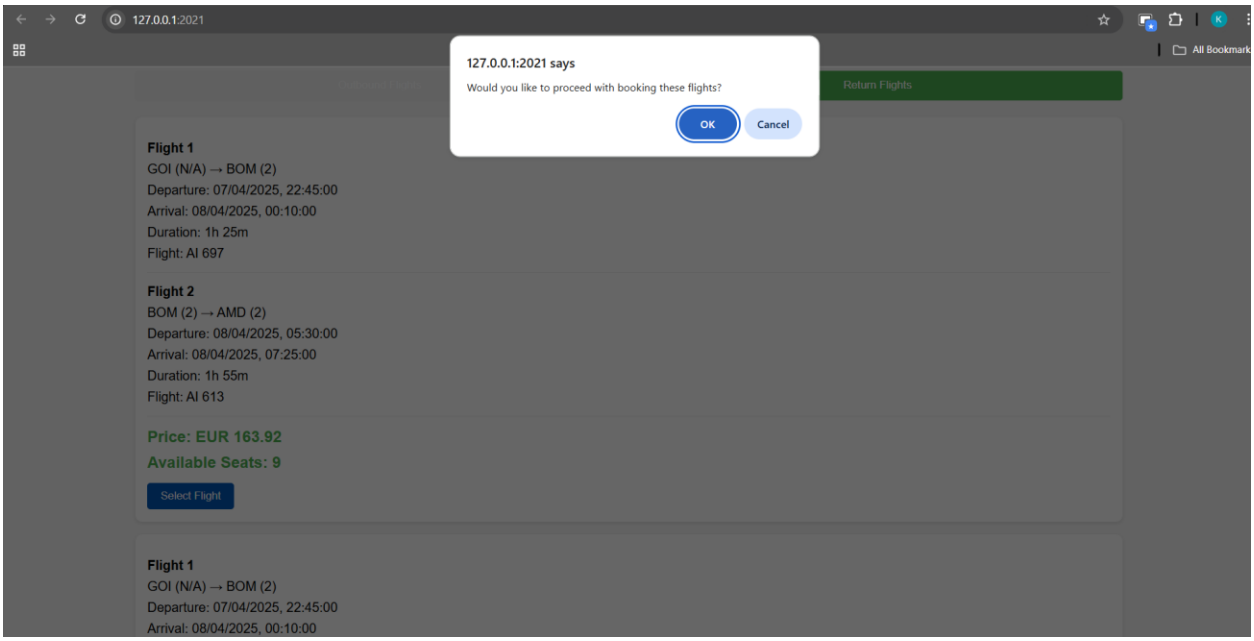


Fig 6.3.4 Getting confirmation of Flights

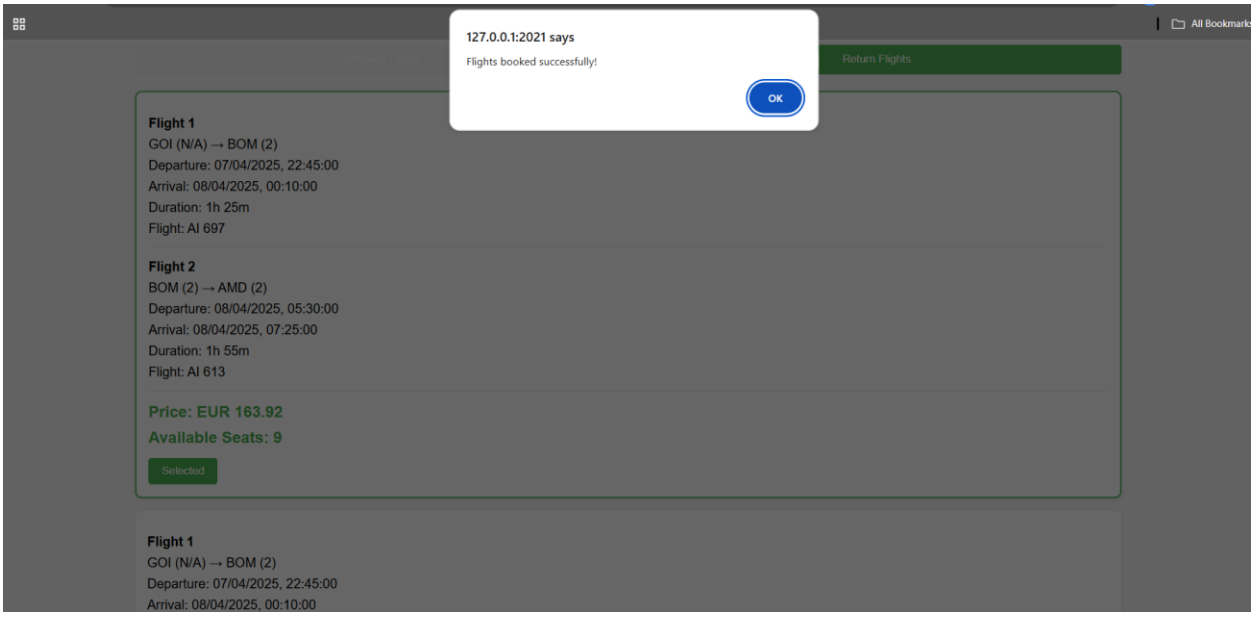


Fig 6.3.5 Conforming Flights

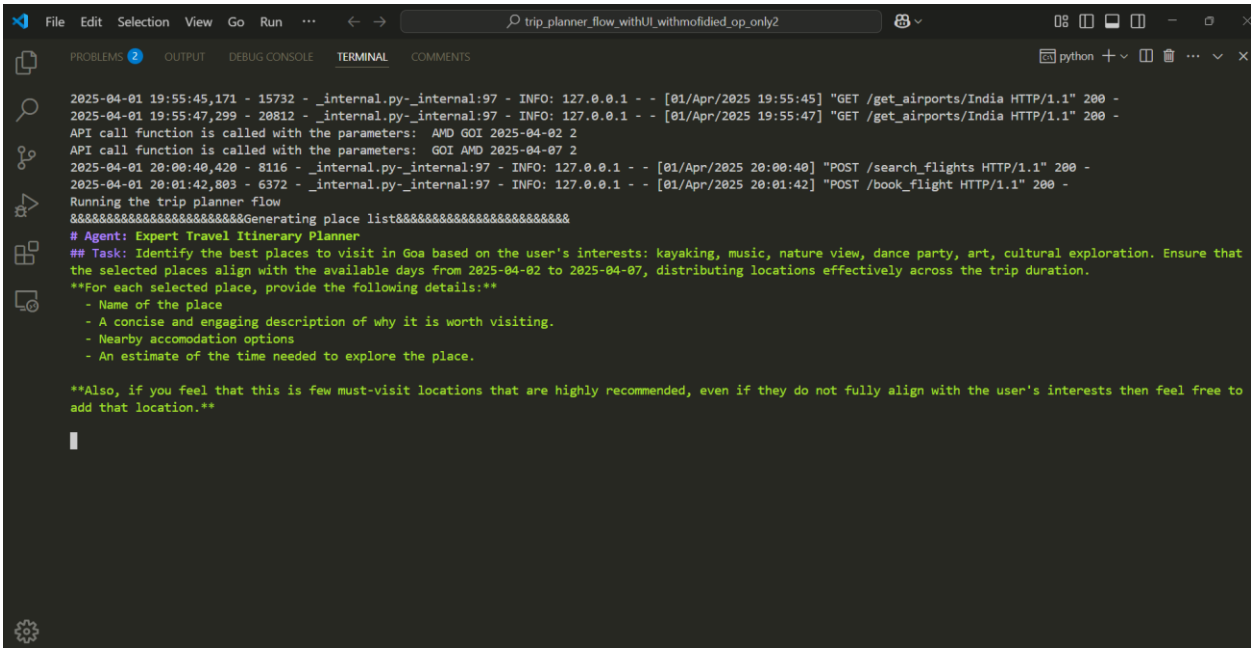


Fig 6.3.6 Agent working in Backend

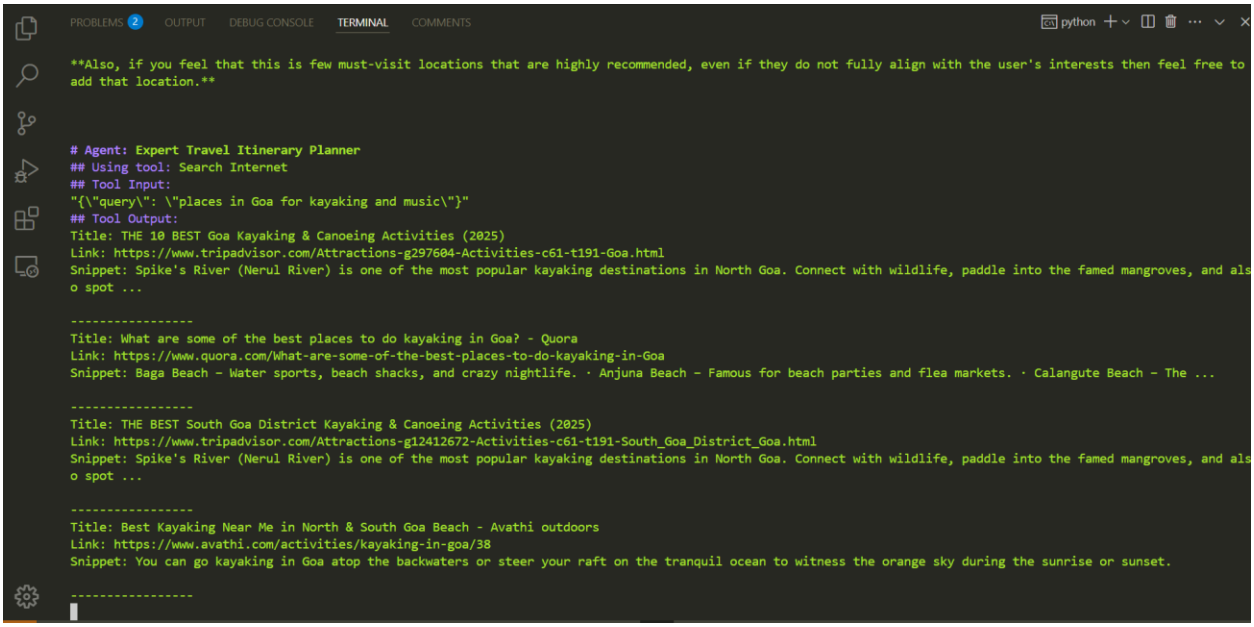


Fig 6.3.7 Agent exploring Internet

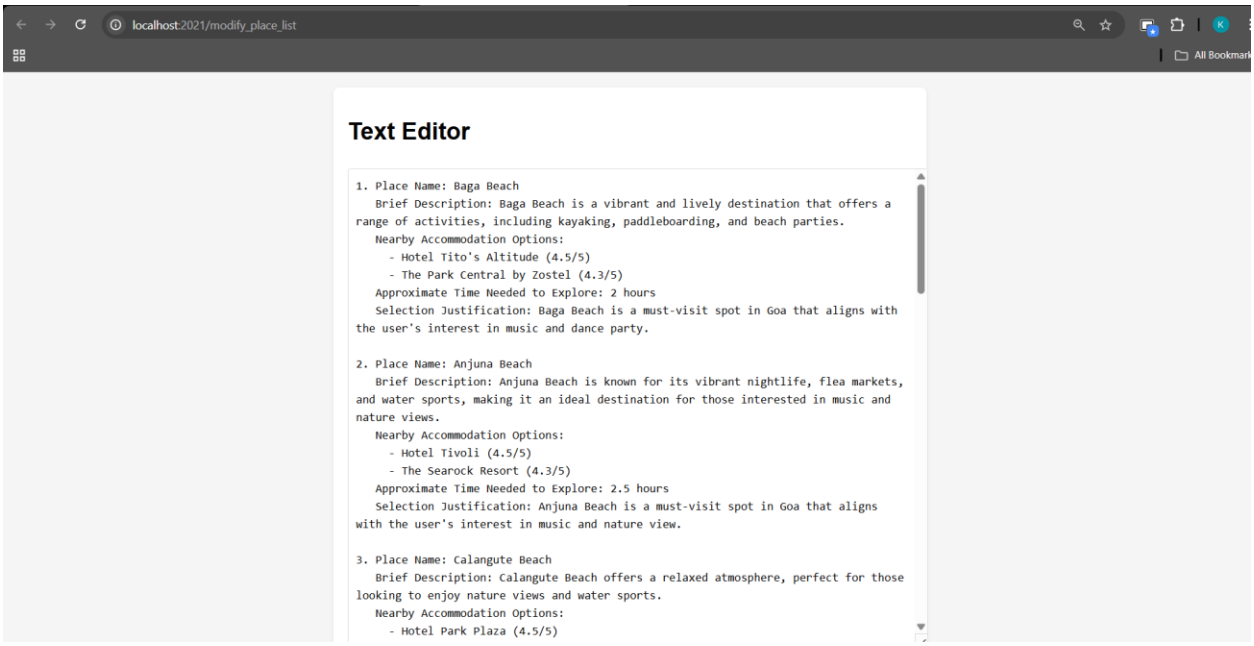


Fig 6.3.8 Allowing user to modify/verify intermediate result

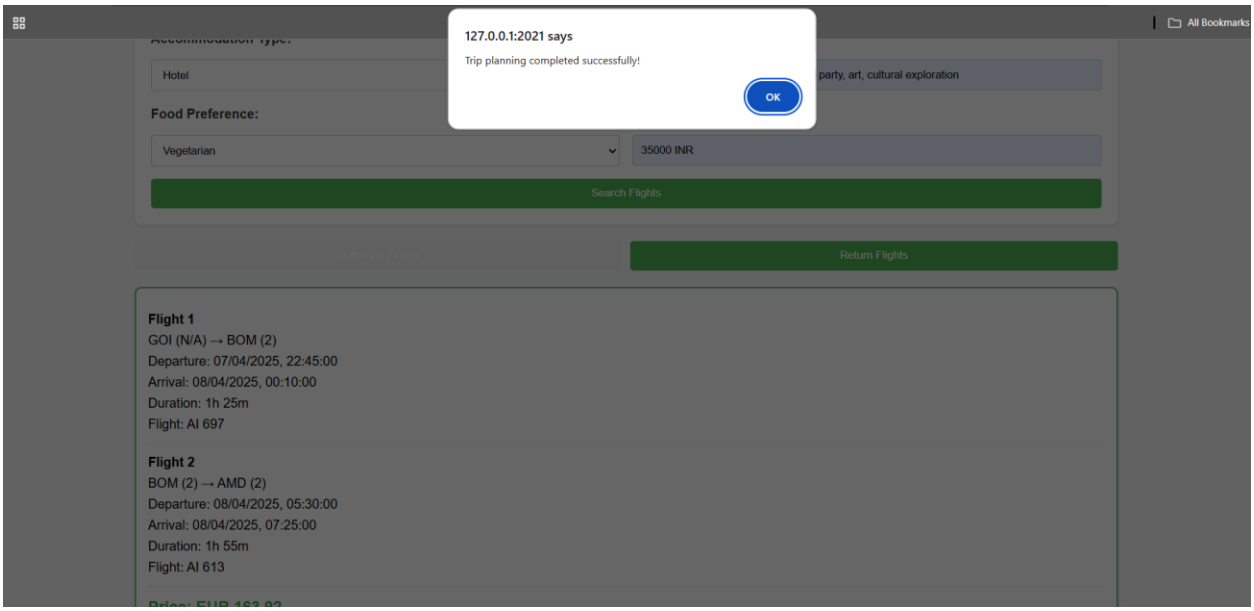


Fig 6.3.9 Getting Alert when trip planned successfully

- **Day 1:**
  - 9:00 AM - Breakfast at Café Veggie (€10)
  - 10:00 AM - Visit Vagator Canyon (transport: taxi €5, activity €20)
  - 1:00 PM - Lunch at Baga Beach Restaurant (vegetarian options available, €15)
  - 2:30 PM - Relax on Maganam Beach (transport: bus €2, activity free)
  - 6:00 PM - Dinner at Ashwem Beach Restaurant (vegetarian options available, €20)
- **Day 2:**
  - 9:00 AM - Breakfast at Hotel (€10)
  - 10:30 AM - Visit Vagator Beach (transport: bus €2, activity free)
  - 1:00 PM - Lunch at Sinquerim Beach Restaurant (vegetarian options available, €15)
  - 2:30 PM - Explore Ashwem Village (transport: taxi €5, activity free)
  - 6:00 PM - Dinner at Café Veggie (€20)
- **Day 3:**
  - 9:00 AM - Breakfast at Hotel (€10)
  - 10:00 AM - Visit Maganam Beach (transport: bus €2, activity free)
  - 1:00 PM - Lunch at Baga Beach Restaurant (vegetarian options available, €15)
  - 2:30 PM - Relax on Vagator Canyon (transport: taxi €5, activity free)
  - 6:00 PM - Dinner at Ashwem Beach Restaurant (vegetarian options available, €20)

Estimated Costs per Activity

- Breakfast: €10
- Lunch: €15
- Dinner: €20

Realistic Logistics for Seamless Transitions between Activities

- Transportation:
  - Taxi: €5-€10 per ride

Fig 6.3.10 Final result

```python
class TripFlow(Flow[TripState]):

    @listen(taking_user_inputs)
    def generate_place_list(self):
        print("&&&&&&&&&&&&&&&&&&&&&&&&Generating place list&&&&&&&&&&&&&&&&&&&&&&&&")
        result = (
            PlaceCrew()
            .crew()
            .kickoff(inputs={
        "destination_location" : self.state.destination_location,
        "interest" : self.state.interest,
        "food_preferences" : self.state.food_preferences,
        "travel_end_Date" : self.state.travel_end_Date,
        "travel_start_Date" : self.state.travel_start_Date,
    }))
        self.state.place_selection_op = result.raw
        return result.raw

    @listen(generate_place_list)
    def modify_place_list(self):
        try:
            original_content = self.state.place_selection_op
            print("Original place list:", original_content[:100] + "...")
```

Fig 6.3.11 Code for defining Crew-flow

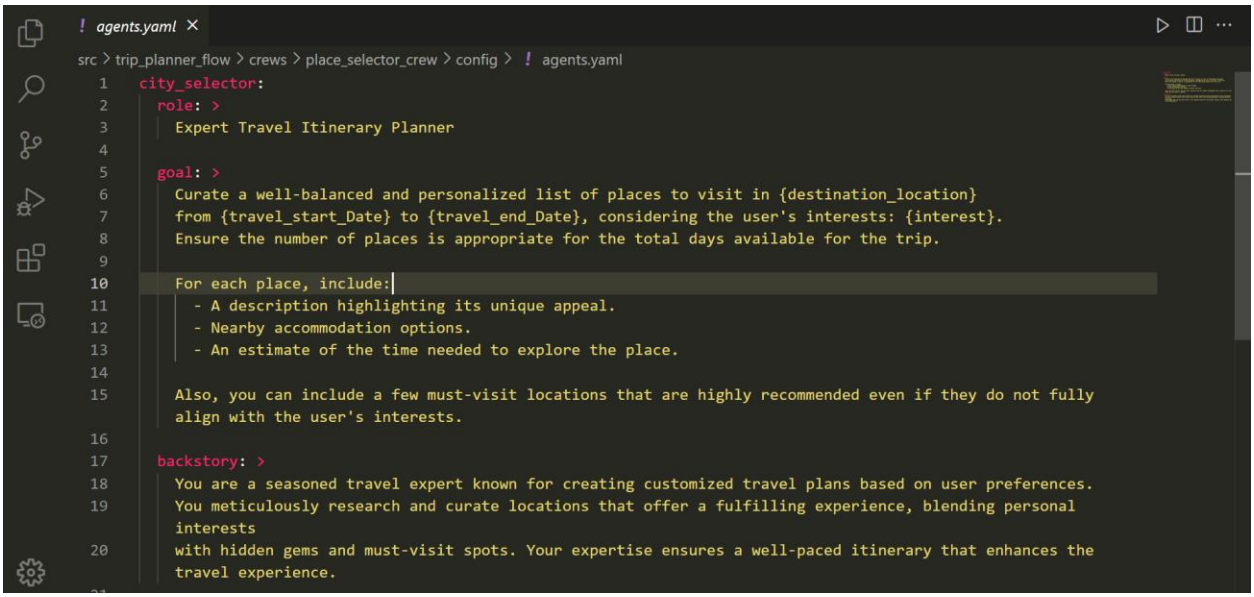


Fig 6.3.12 Agent 1 yaml file

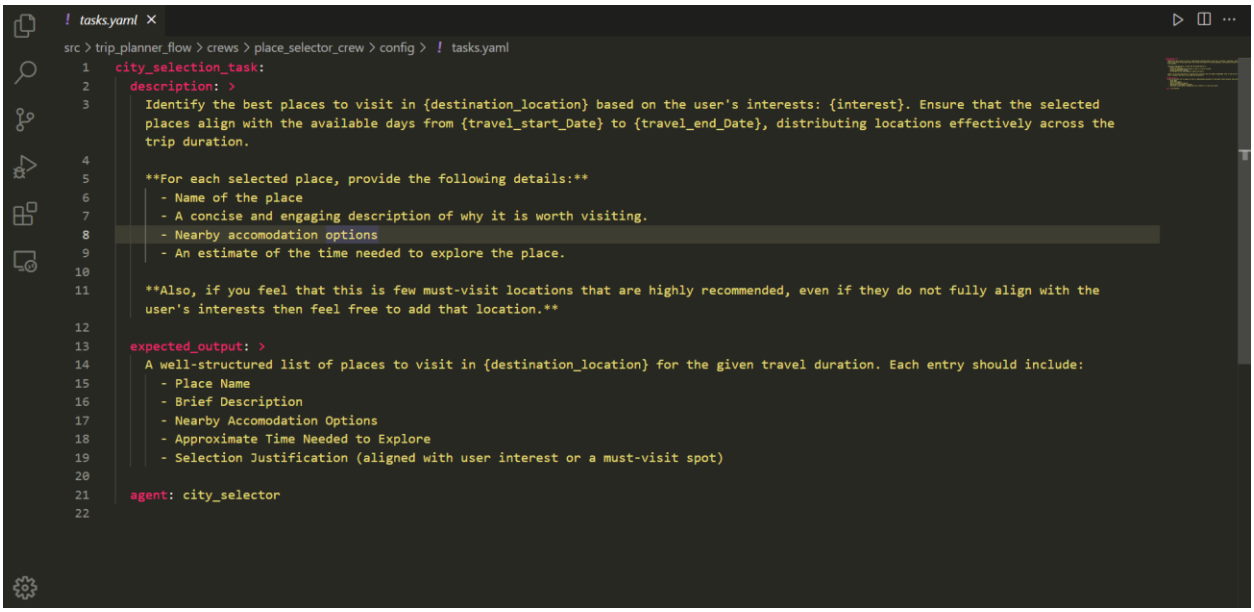


Fig 6.3.13 Agent 1tasks.yaml file

7. Testing

7.1 Testing Plan / Strategy

Once the source code for the AI Travel Planner has been developed, the software must undergo rigorous testing to identify and resolve errors before potential deployment or final handover. The goal is to design a structured testing process that ensures the system functions correctly across all components, from user input handling to final itinerary generation. Testing techniques provide a systematic approach to:

- Validating the internal logic and decision-making processes of the CREWAI agents (Place Selector, Place Explorer, Trip Planner).
- Ensuring data inputs (user preferences, API responses) and outputs (agent suggestions, final itinerary) are accurate and handled correctly, including interactions with Flask, CREWAI, Amadeus API, and Serper API.
- Checking system behaviour, agent response consistency, and performance under various travel planning scenarios and user inputs.

7.1.1 Testing Objectives

- Testing is performed to identify any potential errors, inconsistencies, or unexpected behaviours in the system's functionality or outputs.
- Well-designed test cases should maximize the chances of uncovering undetected issues within the agents, data flow, API integrations, or user interface.
- A successful test is one that identifies an issue (e.g., incorrect flight data integration, illogical itinerary step, poor agent response) and helps trace it back to its root cause for resolution.

7.2 Test Results and Analysis

7.2.1 Unit Testing

Unit testing is a critical software testing phase where individual components or units of the AI Travel Planner system are tested in isolation. This ensures that each small piece of code, such as specific Python functions, Flask route handlers, utility methods, or individual agent task logic, functions correctly on its own.

For this project, unit tests would focus on:

- Functions within the Flask application responsible for processing form data or rendering templates.
- Helper functions used for data manipulation or formatting.
- Code responsible for interacting with external APIs (Amadeus, Serper), often using mocks or stubs (Test Doubles) to isolate the test from the actual external service.
- Where feasible, testing specific methods within the CREWAI agent definitions.

Unit testing, often performed using automated testing frameworks like pytest or unittest in Python, helps developers confidently modify or refactor the source code without inadvertently impacting

other parts of the system. Once unit tests confirm that individual units work as expected, integration testing follows to evaluate how different components (like multiple CREWAI agents in a flow, or Flask interacting with CREWAI) interact with each other.

7.2.2 Integration Testing

Following unit testing, integration testing focuses on verifying the interactions and data flow between different components and modules of the AI Travel Planner system. This phase ensures that independently tested units work together correctly as subsystems or within the defined CREWAI agent flow. In this phase:

- The interaction between the **Flask frontend** (user input forms, display elements) and the **Flask backend** (request handling, routing) was tested to ensure data is passed correctly.
- The communication between the **Backend Orchestrator** and the **CREWAI framework** was validated, ensuring agents are triggered correctly within the defined flow.
- Data passing mechanisms **between consecutive CREWAI agents** (Place Selector -> Place Explorer -> Trip Planner) were tested to confirm outputs from one agent are correctly used as inputs for the next.
- Integrations with external APIs were validated:
    - Calls to the **Amadeus API** were tested for correct request formatting and successful retrieval and parsing of flight data.
    - Calls to the **Serper API** (used potentially by Place Selector/Explorer agents) were tested for reliable web search query execution and result processing.
- The **User Modification Module** was tested to ensure user feedback correctly influenced the data passed to subsequent agents in the flow.
- The final itinerary generation process was tested, verifying the correct combination of agent outputs and integrated flight data.

Successful integration testing confirms that key workflows and data transitions function as expected before evaluating the system as a whole.

7.2.3 System Testing

System testing evaluates the entire AI Travel Planner application from end-to-end to identify bugs, usability issues, and performance bottlenecks. It assesses the complete user experience, from initial input to viewing the final itinerary. During testing:

- The overall **responsiveness and usability** of the Flask web interface were evaluated across different browsers and potentially device sizes.
- End-to-end travel planning scenarios were executed with diverse inputs (various destinations, interests, durations, budgets) to validate the **accuracy, relevance, and coherence** of the generated itineraries.

- The reliability and correctness of integrated data from **Amadeus (flights)** and **Serper (web search results)** within the final output were verified.
- Consistency of **CREWAI agent responses** (powered by Ollama/Llama 3.2) was assessed across multiple runs with similar inputs.
- Performance under simulated typical user load was observed (if feasible within the project scope).
- Minor **UI/UX improvements** identified during testing were implemented and re-verified.

Bugs and issues discovered during system testing were documented, fixed, and subsequently retested to ensure application stability and correctness.

7.2.4 Recovery Testing

This phase tested the AI Travel Planner's ability to handle failures and unexpected conditions gracefully, evaluating its resilience and error-handling mechanisms. Simulations included:

- **External API failures:** Testing system behavior when Amadeus or Serper APIs are unavailable, return errors, or time out.
- **LLM Service interruption:** Assessing how the system handles potential errors or unavailability of the locally hosted Ollama/Llama 3.2 service.
- **Web Application errors:** Simulating internal errors within the Flask application.
- **Network connectivity issues:** Testing behavior during temporary network disruptions affecting API calls.

The system was evaluated on its ability to detect these failures, provide informative feedback to the user (e.g., "Could not fetch flights at this time"), and recover or fail gracefully without crashing or losing critical state where possible.

7.2.5 Security Testing

Security testing focused on identifying potential vulnerabilities and ensuring the protection of sensitive information, such as API keys. Key areas checked included:

- **API Key Management:** Ensuring that Amadeus and Serper API keys were securely stored on the backend and not exposed in client-side code or logs.
- **Input Validation:** Basic checks on user input handled by the Flask application to prevent potential issues if input were ever used directly in sensitive operations (though the risk profile is different when input is primarily processed by an LLM).
- **Dependency Security:** Checking for known vulnerabilities in project dependencies (Python libraries, Flask extensions).
- **(If Applicable) Access Control:** If any form of user authentication or session management was implemented, it would be tested here.

The goal was to ensure the application follows basic security best practices relevant to its architecture and data handling requirements.

7.2.6 Performance Testing

Performance testing assessed the speed, responsiveness, and efficiency of the AI Travel Planner system under typical usage scenarios encountered during development and testing. Tests included:

- Monitoring the response time for **CREWAI agent processing**, considering the local execution of the Llama 3.2 model via Ollama for tasks like place selection, exploration, and itinerary generation.
- Measuring the time taken for **external API calls** (Amadeus for flights, Serper for web search) and ensuring they completed within acceptable limits.
- Verifying that the **Flask web interface** remained responsive during backend processing, particularly while waiting for agent or API responses.
- Assessing the overall time required to generate a complete itinerary for various representative user requests.

Performance considerations were monitored throughout the development and system testing phases to ensure the application provides a reasonably efficient user experience within the constraints of the local LLM setup and external API dependencies.

8. Conclusion

The AI Travel Planner project represents an innovative approach to personalized itinerary creation, leveraging the power of Large Language Models and autonomous agent frameworks. This tool is designed to assist users by automating the complex tasks of researching destinations, exploring points of interest, structuring a logical travel plan, and integrating real-time flight information, moving beyond simple template generation to offer more dynamic and tailored suggestions.

With a web-based interface developed using Flask and core intelligence powered by collaborating AI agents built with the CREWAI framework, the system offers features like AI-driven place selection, detailed place exploration, multi-day itinerary planning, and user interaction points for refining suggestions. The integration of the locally hosted Llama 3.2 model via Ollama demonstrates the potential for sophisticated AI processing within a controlled environment, while connections to the Amadeus and Serper APIs provide essential real-world data for flights and web exploration.

By leveraging key technologies including **Python, Flask, CREWAI, Ollama, Llama 3.2, Amadeus API,** and **Serper API**, this project delivers a functional prototype of an intelligent travel planning assistant. The iterative development process, including structured agent design, prompt engineering, API integration, and user interface development, resulted in a system capable of handling complex travel planning queries.

Overall, this project serves as a valuable exploration into applying multi-agent AI systems to the domain of travel planning. Its interactive design, reliance on advanced AI capabilities for generation and research, and integration with external data sources make it a compelling alternative to manual planning methods, offering users a powerful tool to create customized and data-informed travel itineraries.

References

1. TravelAgent: An AI Assistant for Personalized Travel Planning
2. TRIP-PAL: Travel Planning with Guarantees by Combining Large Language Models and Automated Planners
3. Robots, Artificial Intelligence, and Service Automation in Travel Agencies and Tourist Information Centers