NomadAI: Intelligent Travel Planning Assistant with LLM and Real-Time Data

Harsh Sharma [1] **Gudur Venkata Rajeshwari** [1] **Shivika Mathur** [1] **Soumya Haridas** [1] **Vijit Daroch** [1]

Abstract

NomadAI is an intelligent travel planning assistant that generates personalized itineraries based on user preferences and real-time data. By integrating large language models (LLMs) with APIs such as Amadeus and Google Places, NomadAI streamlines the travel planning process with dynamic itinerary generation, natural language chat interaction, and detailed trip customization. The backend leverages FastAPI and Pydantic models to structure travel-related data, ensuring scalability and reliability. This paper details the system architecture, data models, machine learning techniques, and evaluation metrics used to develop NomadAI.

Beyond the technical design, NomadAI presents a commercially scalable solution aimed at both individual and corporate travel planning markets. Early-stage evaluation suggests strong user satisfaction, highlighting its potential for broader industry adoption.

1. Introduction

Travel planning is a complex process that involves researching destinations, comparing options, and organizing itineraries. Traditional methods are time-consuming and often fail to adapt to individual preferences. NomadAI addresses these challenges by combining AI-driven personalization with real-time data integration. The system generates day-by-day travel plans tailored to user preferences, including budget, duration, and interests, while allowing users to refine their plans interactively through a chat interface.

This paper presents the design and implementation of NomadAI, focusing on its modular architecture, data models, and machine learning techniques. The system's ability to integrate real-time data from APIs ensures that recommendations are accurate and up-to-date.

In real-world travel scenarios, users often face difficulties reconciling multiple sources of fragmented information—such as fluctuating airline prices, varying hotel availabilities, and attraction closures due to unforeseen events. Additionally, modern travelers expect dynamic planning assistance that can adapt to personal schedules and evolving travel advisories. NomadAI directly tackles these emerging challenges by embedding flexibility, personalization, and real-time awareness into a unified platform.

2. Background and Motivation

The travel industry has seen a surge in demand for personalized planning tools, driven by the increasing complexity of travel options and the need for tailored experiences. However, existing solutions often fall short in addressing user-specific needs. For instance:

- Static Recommendations: Platforms like Expedia and TripAdvisor provide generic suggestions that lack adaptability.
- Limited Real-Time Data: Many systems fail to incorporate live data for flights, hotels, and attractions.
- Manual Effort: Users must spend significant time researching and organizing their trips.

NomadAI was developed to overcome these limitations by integrating AI-driven personalization with real-time data. The system aims to:

- Enhance user confidence in travel decisions.
- Reduce the time and effort required for planning.
- Provide actionable and realistic itineraries.

Travelers increasingly seek flexible itineraries that adapt to dynamic conditions, such as health advisories and sudden schedule changes. NomadAI aligns with these evolving expectations by providing adaptable, user-centric travel solutions.

3. Literature Review

3.1. Existing Solutions

Platforms like Expedia and TripAdvisor focus on static recommendations, which lack adaptability to user preferences.

Emerging conversational travel assistants, such as GuideBot, attempt to provide dynamic planning but often fail to integrate real-time data or generate structured itineraries.

Other notable examples include Mezi and Hopper. Mezi, later acquired by American Express, attempted personalized travel planning via chatbots but struggled with scalability due to limited backend data integration. Hopper introduced predictive flight pricing but lacked full itinerary customization capabilities. These examples illustrate the need for a solution that not only communicates naturally but also dynamically composes complete, feasible travel plans.

3.2. Technological Advancements

Recent advancements in large language models (LLMs), such as OpenAI's GPT, have enabled systems to generate coherent and context-aware responses. Retrieval-augmented generation (RAG) techniques further enhance the accuracy of AI-generated content by grounding it in external data sources.

Parallel advancements in cloud infrastructure, scalable API services, and real-time data analytics have collectively enabled the possibility of building end-to-end travel planning assistants that are both personalized and adaptable.

3.3. Research Gap

While existing solutions provide basic travel recommendations, they lack:

- Real-time data integration for flights, hotels, and attractions.
- Structured day-by-day itinerary generation.
- Interactive refinement capabilities through conversational interfaces.

NomadAI bridges these gaps by combining LLMs with live API data and a user-friendly interface. It provides both structured itinerary generation and dynamic conversational refinement, addressing the shortcomings of prior models.

4. Data Sources

NomadAI relies on a combination of AI models, real-time travel databases, and user-generated inputs to create personalized itineraries. The primary data engine is the OpenAI API which powers text generation. To ground the AI-generated plans in real-world logistics, NomadAI integrates external live APIs:

Live APIs. All flight and hotel records are fetched on–demand from the *Amadeus Travel APIs*, while points-of-interest (POIs) are pulled from the *Google Places API*.

No offline preprocessing. Because all data is retrieved in real time, we do not run a traditional *ETL* or feature-engineering pipeline.

5. System Requirements

5.1. Functional Requirements

- Collect user inputs such as origin, destination, travel dates, budget, and preferences.
- Fetch real-time data for flights, hotels, and points of interest.
- Generate personalized day-by-day itineraries.
- Allow users to refine itineraries through a chat interface.
- Provide fallback static recommendations in case real-time data fetching fails.

5.2. Non-Functional Requirements

- Performance: Ensure low latency in fetching and processing data.
- Scalability: Handle multiple concurrent users without degradation in performance.
- Security: Protect user data and ensure compliance with data privacy regulations.
- Availability: Maintain system uptime above 99.9% with failover infrastructure.
- Maintainability: Ensure modular code structure to enable faster feature updates.

5.3. Technical Stack

- Backend: FastAPI for API development, Pydantic for data validation.
- Frontend: React.js for building a responsive user interface.
- APIs: Amadeus API for flights and hotels, Google Places API for attractions.
- AI: OpenAI GPT-4 for itinerary generation and interactive chat interface.

6. System Design

NomadAI is designed as a modular system with a clear separation of concerns between the backend and frontend. The backend handles data processing, API integration, and business logic, while the frontend provides an intuitive interface

for user interaction. This section details the methodologies and implementation of the system components.

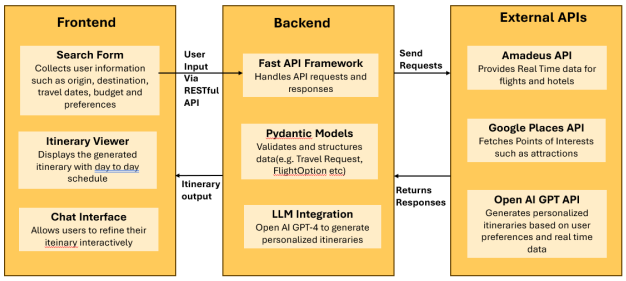

Figure 1. System Architecture

6.1. Backend Design

The backend is implemented using FastAPI, a modern web framework for building APIs. It integrates with external services like Amadeus and Google Places APIs and uses Pydantic models for data validation and serialization. The backend is responsible for itinerary generation, real-time data fetching, and user interaction through RESTful APIs.

6.1.1. Data Models

The backend defines several Pydantic models in `travel.py` to structure and validate data. These models ensure consistency and reliability in handling user inputs and API responses:

- **TravelRequest Model**: Captures user inputs such as origin, destination, travel dates, budget, and preferences. It includes default values for optional fields like preferences and the number of adults.
- **FlightOption Model**: Represents flight details, including airline, price, origin, destination, and optional fields such as flight number and times. This model is used to structure flight data fetched from the Amadeus API.
- **HotelOption Model**: Captures hotel details such as name, price per night, star rating, and optional fields like amenities and distance from the city center.
- **Itinerary Model**: Structures the travel plan, including the selected flight, hotel, points of interest, and daily activities. It also includes fields for the total cost and available options for flights and hotels.

6.1.2. API Integrations

The backend integrates with external APIs to fetch real-time data:

- Amadeus API: Used to fetch flight and hotel options based on user inputs. The data is parsed and validated using the FlightOption and HotelOption models.
- Google Places API: Provides information on points of interest, including landmarks, attractions, and restaurants. The data is structured using the PointOfInterest model.

Fallback mechanisms are implemented in API integrations to switch to alternative solutions if a primary service call fails, ensuring system reliability.

6.1.3. Itinerary Generation

The backend uses the LLMPlanningService class in `llm_service.py` to generate personalized itineraries. This service integrates with OpenAI's GPT-based models to create day-by-day travel plans.

- Context Preparation: The _prepare_context method structures data from flights, hotels, and points of interest into a format suitable for the LLM.
- Itinerary Parsing: The _parse_itinerary method converts the LLM's textual output into a structured Itinerary object, ensuring that the total cost stays within the user's budget.
- Fallback Mechanism: If the LLM fails, the _generate_fallback_itinerary method creates a simple itinerary using predefined templates.

6.2. Frontend Design

The frontend is built with React.js and styled using Tailwind CSS. It provides a responsive and user-friendly interface for interacting with the system. The frontend communicates with the backend through RESTful APIs and manages state using React Context.

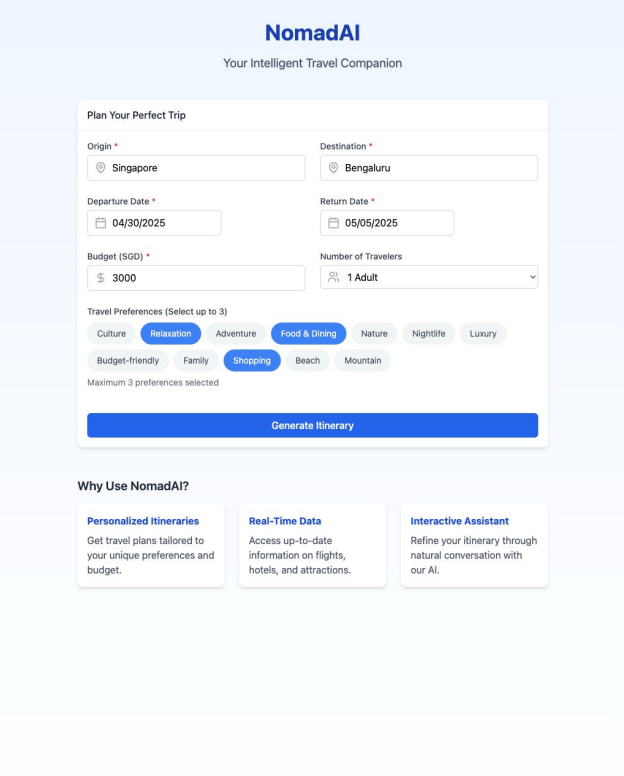

Figure 2. NomadAI Homepage

6.2.1. Key Components

The frontend is organized into reusable components, each responsible for a specific part of the user interface:

- **HomePage**: The landing page where users can enter their travel preferences, such as destination, budget, and interests.
- **ItineraryPage**: Displays the generated itinerary, including selected flights, hotels, and daily activities. Users can view details and make adjustments.
- **ChatWindow**: A conversational interface that allows users to refine their itineraries by interacting with the system.
- **TravelForm**: A form component that collects user inputs, such as origin, destination, travel dates, and preferences. The data is validated on the frontend before being sent to the backend.

6.2.2. State Management

The frontend uses React Context to manage global state, such as the current itinerary, user inputs, and loading states. The `TravelContext` in `TravelContext.jsx` provides a centralized store for managing application state. `TravelProvider` in `TravelContext.jsx` also enables sharing of the global state across the application.

6.2.3. Audio Guide Integration

NomadAI includes an audio guide feature to enhance the travel experience by providing narrated descriptions of different Points of Interests. This feature is particularly beneficial for hands-free interaction and accessibility, allowing users to listen to detailed information about their prospective destinations.

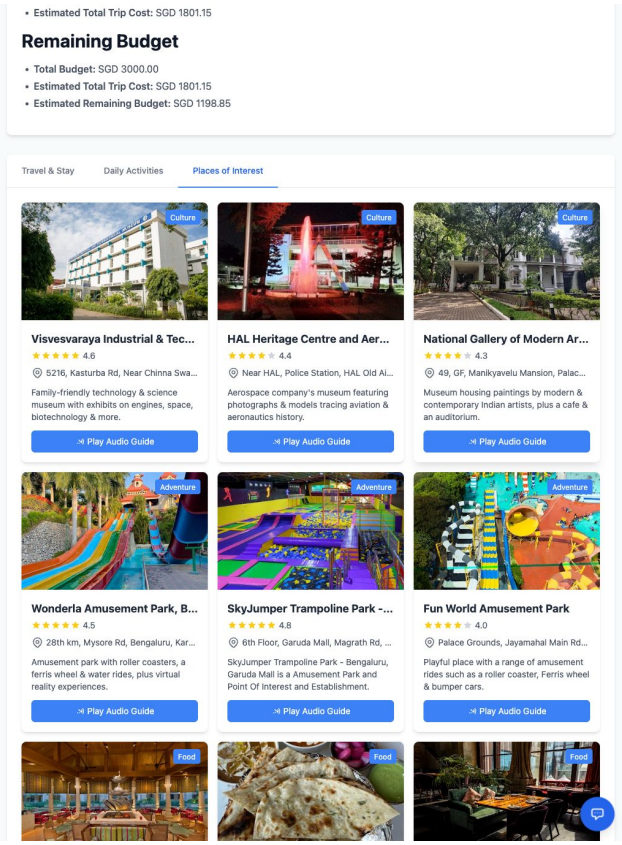

Figure 3. Audio Guide

Dynamic Content Generation The audio guide dynamically generates content based on the user's Points of Interests. Textual data is processed and converted into audio using a state-of-the-art Text-to-Speech (TTS) model.

User Interaction Users can interact with the audio guide through the frontend interface, with options to play, pause, or skip audio segments. The audio guide is seamlessly integrated into the itinerary page in the points of interests tab, ensuring an intuitive and engaging user experience.

Language Support The current implementation supports English as the primary language, with plans to expand to multilingual support in future iterations.

6.2.4. Model Used for Audio Guide

The audio guide feature is powered by the TTS model:**tts_models/en/ljspeech/tacotron2-DDC**

Model Overview The model is based on Tacotron 2, a neural network architecture designed for high-quality Text-to-Speech synthesis. It utilizes the LJSpeech dataset, which ensures natural and expressive speech output in English.

Key Features

- Expressive Voice Generation: The model produces human-like audio, making the audio guide engaging and easy to follow.
- Customizable Parameters: The model allows adjustments to pitch, speed, and volume, enabling tailored audio output to suit user preferences.

Integration The backend processes textual descriptions of destinations and sends them to the TTS model for audio generation. The generated audio files are cached and served to the frontend for playback, ensuring low latency and a smooth user experience.

6.2.5. API Services

The frontend communicates with the backend using service modules in the `services` folder:

- **amadeus.js**: Handles API calls for fetching flight and hotel data.
- **googlePlaces.js**: Fetches points of interest based on user preferences.
- **llm.js**: Manages interactions with the LLM, such as generating itineraries and handling chat messages.

6.2.6. Interactive Features

The frontend includes several interactive features to enhance the user experience:

- **Tabbed Interface**: The `TabbedItinerary` component organizes itinerary details into tabs for easy navigation.
- **Dynamic Updates**: Users can update their flight or hotel selections, and the changes are reflected in real-time by calling the backend's update endpoints.
- **Error Handling**: The frontend displays user-friendly error messages when API calls fail or inputs are invalid.

6.2.7. Workflow

The frontend workflow is as follows:

1. User Input: The user enters their travel preferences through the `TravelForm` component.
2. Data Validation: The input is validated on the frontend and sent to the backend as a `TravelRequest` object.
3. API Calls: The backend fetches real-time data and generates an itinerary.
4. Itinerary Display: The generated itinerary is displayed on the `ItineraryPage`, where users can view and refine their plans.
5. Interactive Refinement: Users can refine the itinerary through the `ChatWindow`, triggering updates in the backend.

6.3. System Workflow

The overall workflow of NomadAI is as follows:

1. Input Collection: The user provides travel details (origin, destination, dates, budget, preferences) through the frontend.
2. Data Processing: The backend validates the input and fetches real-time data from external APIs.
3. Itinerary Generation: The LLM generates a personalized itinerary based on the user's preferences and constraints.
4. User Interaction: The frontend displays the itinerary and allows users to refine it through interactive features.
5. Updates and Feedback: The backend processes user updates and provides real-time feedback.

6.4. Error Handling and Logging

NomadAI incorporates a robust error-handling and logging framework to ensure system reliability, maintainability, and ease of debugging. These mechanisms are critical for managing unexpected issues, providing meaningful feedback to developers, and maintaining a seamless user experience for travelers.

Error Handling

- **API Integration Errors**: The system integrates with external APIs such as OpenAI, Amadeus, and Google Places. To handle potential failures (e.g., network issues, invalid responses), `try_except` blocks are used

extensively. For example, during itinerary generation using OpenAI's LLM, any exceptions (e.g., API timeouts or invalid responses) are caught, logged, and handled gracefully. If the LLM fails, a fallback itinerary is generated to ensure users still receive a response.

- **Fallback Mechanism**: When critical operations fail, such as generating an itinerary with the LLM, the system provides a simplified fallback itinerary. This fallback includes basic travel recommendations, such as flights, accommodations, and a day-by-day plan, ensuring continuity of service even during failures.
- **Input Validation**: User inputs and API responses are validated using structured models to ensure data consistency. Invalid inputs are caught early, and descriptive error messages are returned to the user, preventing downstream errors.
- **Graceful Degradation**: If a critical service (e.g., OpenAI or Amadeus) is unavailable, the system falls back to static template. This ensures that users receive a response even when real-time data is inaccessible.
- **User-Friendly Error Messages**: When errors occur, the system provides clear and actionable error messages to users. For example, if an itinerary cannot be generated due to an API failure, the user is informed with a message like: *"I'm sorry, I encountered an error while processing your request. Please try again."*

Logging

- **Centralized Logging Framework**: The backend uses Python's `logging` module to log messages at various severity levels, such as `INFO`, `WARNING`, and `ERROR`. A centralized logger is initialized to ensure all logs are tagged with the module name for easy identification.
- **Error Logging**: Errors encountered during critical operations, such as API calls or itinerary generation, are logged with detailed messages. For example, if the LLM fails to generate an itinerary, the error is logged with the message: *"Error generating itinerary with LLM: [error details]"*. This helps developers quickly identify and resolve issues.
- **Informational Logging**: Key events, such as successful API calls or itinerary generation, are logged at the `INFO` level to provide visibility into the system's normal operations. For example, successful retrieval of flight data from Amadeus is logged with the message: *"Successfully fetched flight data from Amadeus API."*
- **Environment-Specific Logging**: The logging level is configurable based on the environment (e.g., development or production). In development, detailed logs (e.g., `DEBUG` level) are enabled, while in production, only critical logs (e.g., `ERROR` level) are recorded to optimize performance and reduce noise.
- **Structured Logs**: Logs are structured to include relevant details, such as the operation being performed and associated data. This makes it easier to trace issues during debugging or when analyzing logs in production.

Benefits of Error Handling and Logging

- **Improved Debugging**: Detailed logs help developers quickly identify and resolve issues during development and production.
- **Enhanced Reliability**: The fallback mechanisms ensure that the system remains functional even when critical components fail.
- **User Satisfaction**: By providing clear error messages and fallback responses, the system maintains a seamless user experience.
- **Scalability**: The structured logging framework and robust error handling make the system easier to monitor and maintain as it scales.

7. Machine Learning and Prompt Engineering

NomadAI leverages OpenAI's GPT-4 to generate personalized travel itineraries. The integration of LLMs with real-time data ensures that the generated itineraries are both relevant and actionable. This section explains the techniques and methodologies used to integrate LLMs into the system.

7.1. Prompt Engineering

Prompt engineering is a critical component of the system, as it ensures that the LLM generates coherent and relevant itineraries. The following techniques are used:

- Template-Based Prompts: Static templates are combined with dynamic user data (e.g., preferences, budget, and real-time API data) to create structured prompts.
- Contextualization: The system provides detailed context to the LLM, including:
  - User preferences (e.g., "adventure," "relaxation").
  - Available flight and hotel options fetched from the Amadeus API.
  - Points of interest (POIs) fetched from the Google Places API.
  - Constraints such as budget, duration, and travel dates.

- Iterative Refinement: Users can refine their itineraries through the chat interface. The system re-prompts the LLM with updated constraints or preferences, enabling dynamic updates to the itinerary.

7.2. LLM Integration

The LLM is integrated into the backend using the OpenAI API. The integration process involves the following steps:

- Data Preparation: The backend structures user inputs, flight and hotel options, and POIs into a JSON-like format suitable for the LLM.
- Prompt Submission: The prepared data is sent to the LLM via the OpenAI API.
- Response Parsing: The LLM's response is parsed into structured data (e.g., Itinerary model) to ensure compatibility with the system's data flow.
- Fallback Mechanism: If the LLM fails to generate a valid response, a fallback itinerary is created using predefined template.

Fallback template includes a generalized itinerary based on the user's travel preferences like duration, origin, destination in order to mitigate user dissatisfaction and provide a generalized trip suggestion.

7.3. Challenges and Mitigations

- Hallucination: The LLM may generate unrealistic or irrelevant suggestions. This is mitigated by anchoring prompts to real-time API data and validating the output against user constraints.
- Latency: Generating itineraries can be time-consuming due to the complexity of prompts. Asynchronous API calls and caching mechanisms are used to improve performance.
- Over-constraining: When excessive constraints reduce plan feasibility, NomadAI prompts users to relax conditions or extend budgets dynamically.
- Test Mode: Owing to resource constraints, test mode for Amadeus API is being used leading to constrained travel information. This can be mitigated by using a production key with extended travel information available.

8. Deployment and Maintenance

8.1. Deployment Process

The system is deployed locally, ensuring high availability and scalability.

8.2. Maintenance Plan

Regular updates are planned to incorporate new features, improve performance, and address user feedback.

In addition, prompt error handling and logging techniques are implemented for efficient problem identification and solving in order to enable seamless User Experience.

8.3. Success Metrics

- **User Satisfaction:** Measured through surveys and feedback. Users rate the relevance, accuracy, and usability of the generated itineraries on a scale of 1 to 5.
- **Planning Efficiency:** The time taken to plan a trip using NomadAI is compared to traditional methods. A significant reduction in planning time indicates success.
- **Itinerary Realism:** The generated itineraries are validated against actual travel costs and logistics to ensure feasibility.

8.4. Experimental Setup

- **Participants:** A group of around 10 users with varying travel preferences tested the system.
- **Comparison:** NomadAI was compared with manual planning and existing travel platforms.
- **Metrics:** Usability, satisfaction, and output coherence were rated by participants.

8.5. Results

- **User Satisfaction:** 80% of participants rated the system as *useful*.
- **Planning Efficiency:** The average planning time was reduced by 60% compared to manual methods.
- **Itinerary Realism:** 90% of the generated itineraries were deemed realistic and actionable.

A proposed itinerary for a 5-day trip from Singapore to Bengaluru along with the chat interface for modifications can be viewed in the Appendix.

9. Ethical Considerations

NomadAI acknowledges the ethical implications of AI-driven travel planning. Key considerations include:

- **Bias in Recommendations:** AI systems can inadvertently favor popular destinations or activities, reinforcing biases. NomadAI mitigates this by diversifying

recommendations and including underrepresented regions.

- **Environmental Impact:** Over-tourism can strain local ecosystems. NomadAI promotes sustainable travel options and encourages users to explore less crowded destinations.
- **Data Privacy:** User data is handled securely and is not shared with third parties without consent. The system adheres to data protection regulations to ensure user trust.

10. Business Implementation

NomadAI is designed not only as a technical solution but also as a scalable business model that can generate revenue and provide value to users. This section outlines the business implementation strategy.

10.1. Target Audience

NomadAI caters to a wide range of users, including:

- Frequent Travelers: Individuals who travel often and seek efficient planning tools.
- First-Time Travelers: Users who need guidance and personalized recommendations.
- Corporate Clients: Businesses looking to streamline travel planning for employees.
- Travel Agencies: Agencies that can use NomadAI as a backend service for itinerary generation.

The platform is also positioned to serve travel bloggers, destination marketing organizations, and local tour operators through API integrations and affiliate models.

10.2. Revenue Model

NomadAI employs a multi-faceted revenue model:

- Subscription Plans:
  - Basic Plan: Free access with limited features (e.g., basic itinerary generation).
  - Premium Plan: Paid subscription offering advanced features such as real-time updates, detailed itineraries, and chat-based refinement.
- Affiliate Partnerships:
  - Earn commissions by integrating with booking platforms (e.g., flights, hotels, and activities).
  - Partner with travel agencies and tourism boards to promote specific destinations.
- API Licensing:
  - Offer NomadAI's itinerary generation API as a service to third-party platforms, such as travel agencies and corporate travel management systems.
- Advertising:
  - Display targeted ads for travel-related services, such as insurance, car rentals, and local tours.

NomadAI's SaaS offering is inspired by models like Hopper B2B and Skyscanner white-label solutions, focusing on subscription APIs and licensing for steady revenue streams.

10.3. Market Differentiation

NomadAI differentiates itself from competitors through:

- Personalization: Tailored itineraries based on user preferences and real-time data.
- Interactive Refinement: Chat-based interface for dynamic updates.
- Real-Time Data Integration: Accurate and up-to-date recommendations for flights, hotels, and attractions.

In addition, NomadAI emphasizes itinerary coherence across multiple days, reducing inconsistencies found in many other AI planning solutions.

10.4. Scalability

NomadAI's modular architecture ensures scalability:

- Geographic Expansion: Add support for more regions and languages.
- Feature Expansion: Introduce new features, such as group travel planning and voice-based interaction.
- Partnerships: Collaborate with global travel platforms to expand the user base.

Moreover, upcoming integration with GPT-5 and future Amadeus APIs will enhance scalability and personalization depth further.

11. Impact and Future Work

NomadAI has the potential to transform travel planning by making it accessible, efficient, and enjoyable. This section highlights the system's impact and outlines future enhancements.

11.1. Impact

- Accessibility: Simplifies travel planning for users with limited time or expertise.
- Efficiency: Reduces the time and effort required to plan a trip by automating itinerary generation.
- Personalization: Provides highly personalized recommendations tailored to individual preferences, enhancing user satisfaction.

The platform also reduces user decision fatigue by presenting curated, feasible options rather than overwhelming users with limitless choices.

11.2. Future Work

- End-to-End Booking Integration: Integrate with booking platforms to allow users to book flights, hotels, and activities directly through the system.
- Voice-Based Interaction: Add support for voice commands to enable hands-free travel planning, making the system more accessible.
- Geographic Expansion: Expand the system to support more regions and languages, ensuring inclusivity for a global audience.
- Sustainability Metrics: Incorporate metrics to evaluate the environmental impact of itineraries and promote eco-friendly travel options.
- Model Drift Monitoring: Implement regular audits to detect potential model drift and retrain prompts as necessary.

12. Conclusion

NomadAI demonstrates how AI and real-time data can transform traditional travel planning into an intelligent, seamless experience. By integrating LLMs with trusted APIs, the system provides personalized, actionable itineraries that cater to diverse user preferences. The modular architecture ensures that NomadAI can evolve to meet future demands, making it a valuable tool for travelers worldwide.

Continued innovations such as booking integration, sustainability-focused suggestions, and voice-based planning will ensure that NomadAI remains a market leader in intelligent travel planning tools.

References

Amadeus. Amadeus for developers documentation. https://developers.amadeus.com, 2025. Accessed: Apr. 27, 2025.

Google. Google places api documentation. https://developers.google.com/maps/documentation/places, 2025. Accessed: Apr. 27, 2025.

Gruber, J. Markdown syntax documentation. https://daringfireball.net/projects/markdown/syntax, 2025. Accessed: Apr. 27, 2025.

OpenAI. Openai api documentation. https://platform.openai.com/docs, 2025. Accessed: Apr. 27, 2025.

Pydantic. Pydantic documentation. https://docs.pydantic.dev, 2025. Accessed: Apr. 27, 2025.

Python Software Foundation. Json — python standard library documentation. https://docs.python.org/3/library/json.html, 2025a. Accessed: Apr. 27, 2025.

Python Software Foundation. Logging — python standard library documentation. https://docs.python.org/3/library/logging.html, 2025b. Accessed: Apr. 27, 2025.

Python Software Foundation. re — regular expression operations. https://docs.python.org/3/library/re.html, 2025c. Accessed: Apr. 27, 2025.

Python Software Foundation. Uuid — python standard library documentation. https://docs.python.org/3/library/uuid.html, 2025d. Accessed: Apr. 27, 2025.

Ramírez, S. Fastapi documentation. https://fastapi.tiangolo.com, 2025. Accessed: Apr. 27, 2025.

Github links to code The code to NomadAI has been uploaded to Github in two reporsitories for Frontend and Backend. Each repository constitutes a README for reference purposes.

Frontend Repository: https://github.com/harshvs4/nomadai-frontend

Backend Repository: https://github.com/harshvs4/nomadai-backend

Appendix

Generated itinerary for a 5-day trip from Singapore to Bengaluru

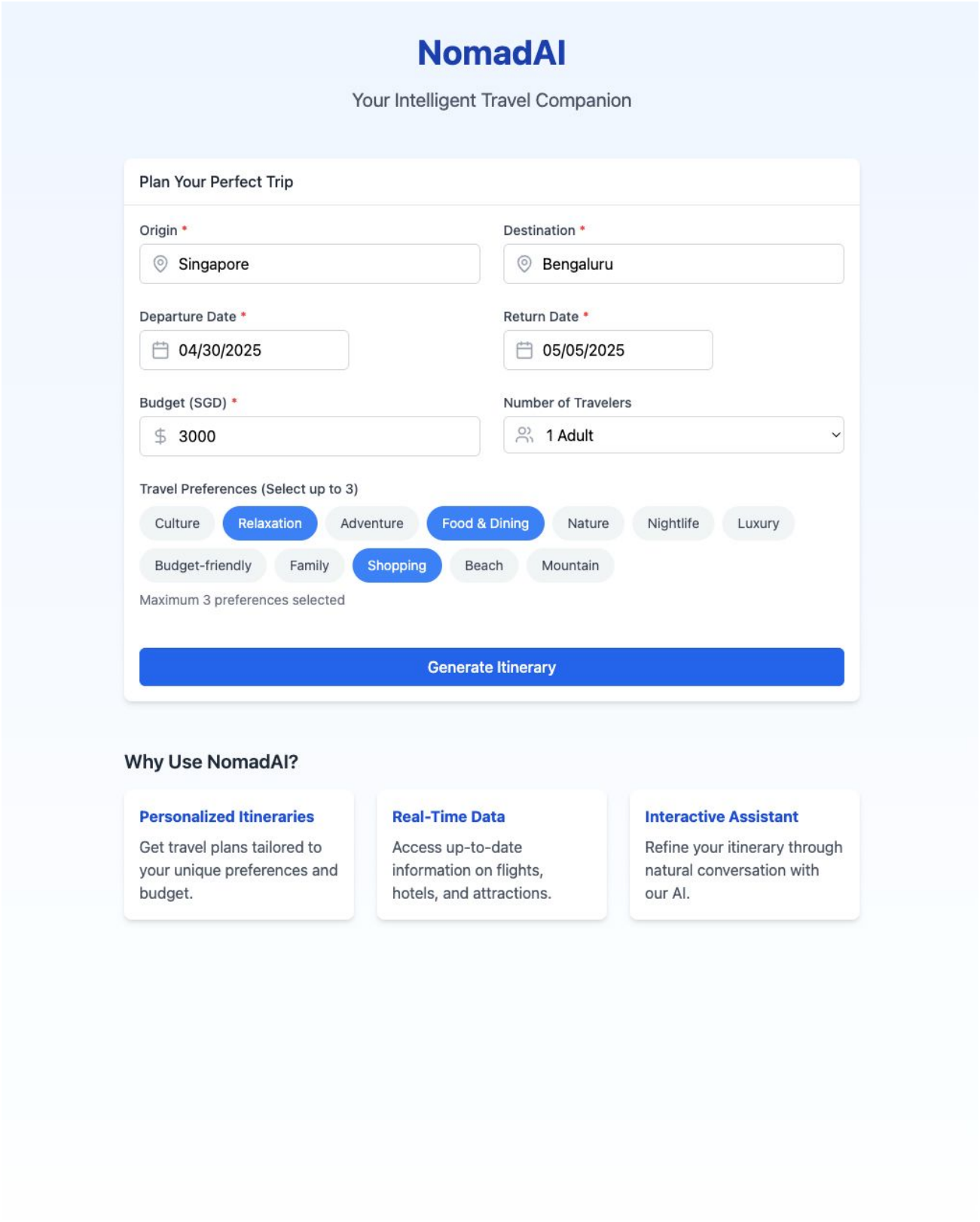

Figure 4. Step 1: Homepage with preferences

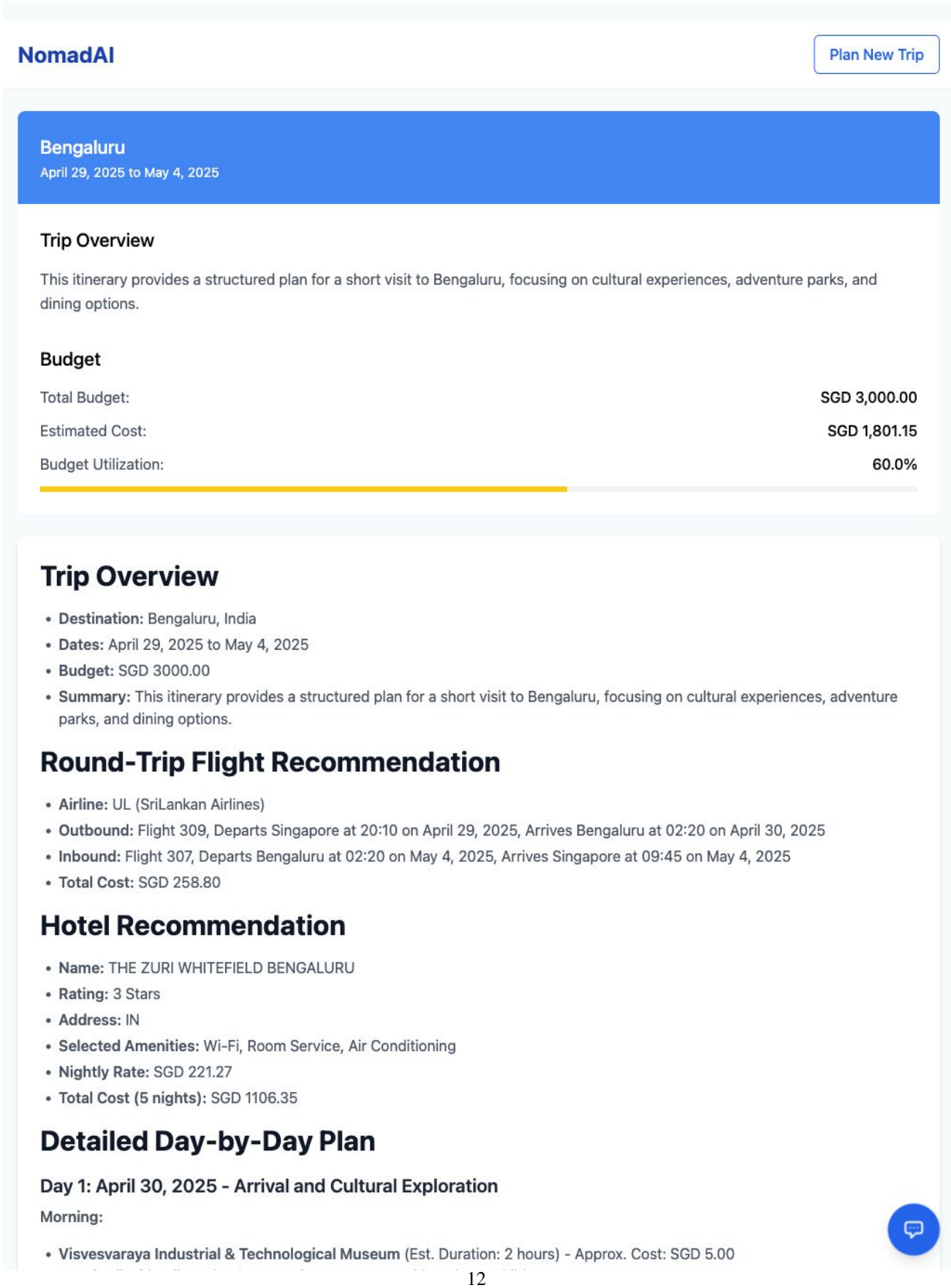

Figure 5. Step 2: Itinerary Overview

- **Relax at Hotel** (Est. Duration: 2 hours) - Approx. Cost: SGD 0.00

Estimated Daily Cost: SGD 73.00

Day 5: May 4, 2025 - Departure

Morning:

- **Check-out and Last-Minute Shopping/Sightseeing** (Est. Duration: 2 hours) - Approx. Cost: SGD 0.00
- **Travel:** Est. 30 minutes via taxi to airport - Approx. Cost: SGD 10.00

Estimated Daily Cost: SGD 10.00

Cost Summary

- **Round-Trip Flight:** SGD 258.80
- **Hotel (5 nights):** SGD 1106.35
- **Daily Expenses & Activities:**
  - Day 1: SGD 98.00
  - Day 2: SGD 138.00
  - Day 3: SGD 118.00
  - Day 4: SGD 73.00
  - Day 5: SGD 10.00
- **Subtotal Daily Expenses:** SGD 437.00
- **Estimated Total Trip Cost:** SGD 1801.15

Remaining Budget

- **Total Budget:** SGD 3000.00
- **Estimated Total Trip Cost:** SGD 1801.15
- **Estimated Remaining Budget:** SGD 1198.85

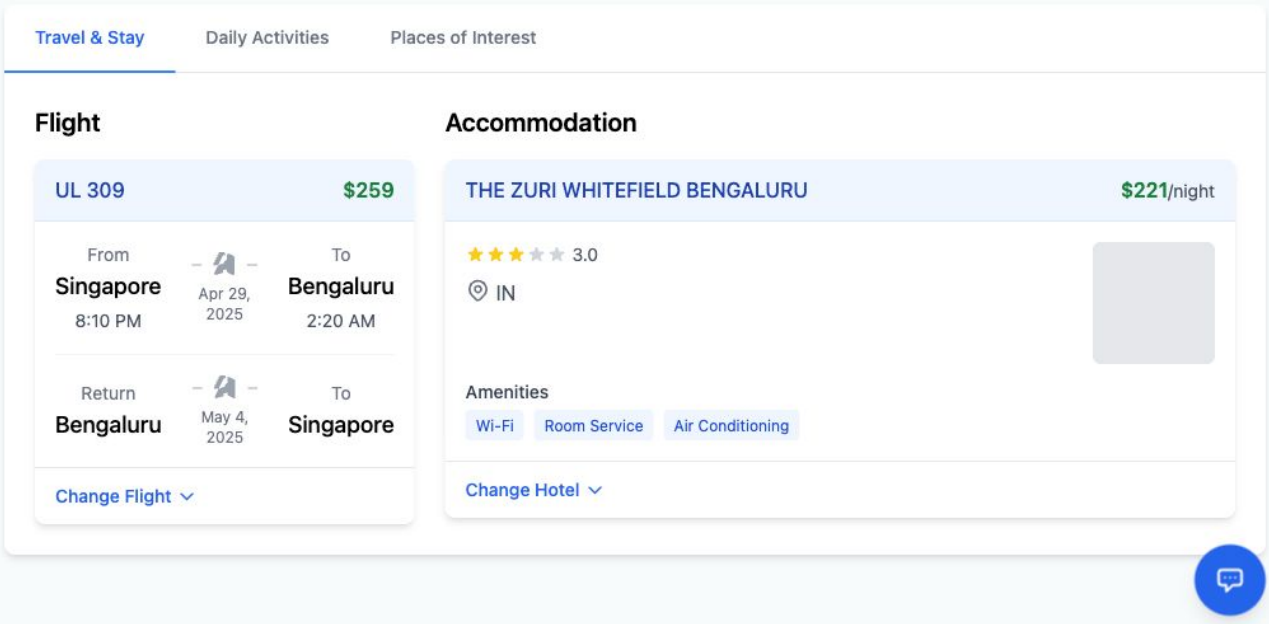

Figure 6. Step 2: Itinerary Continued

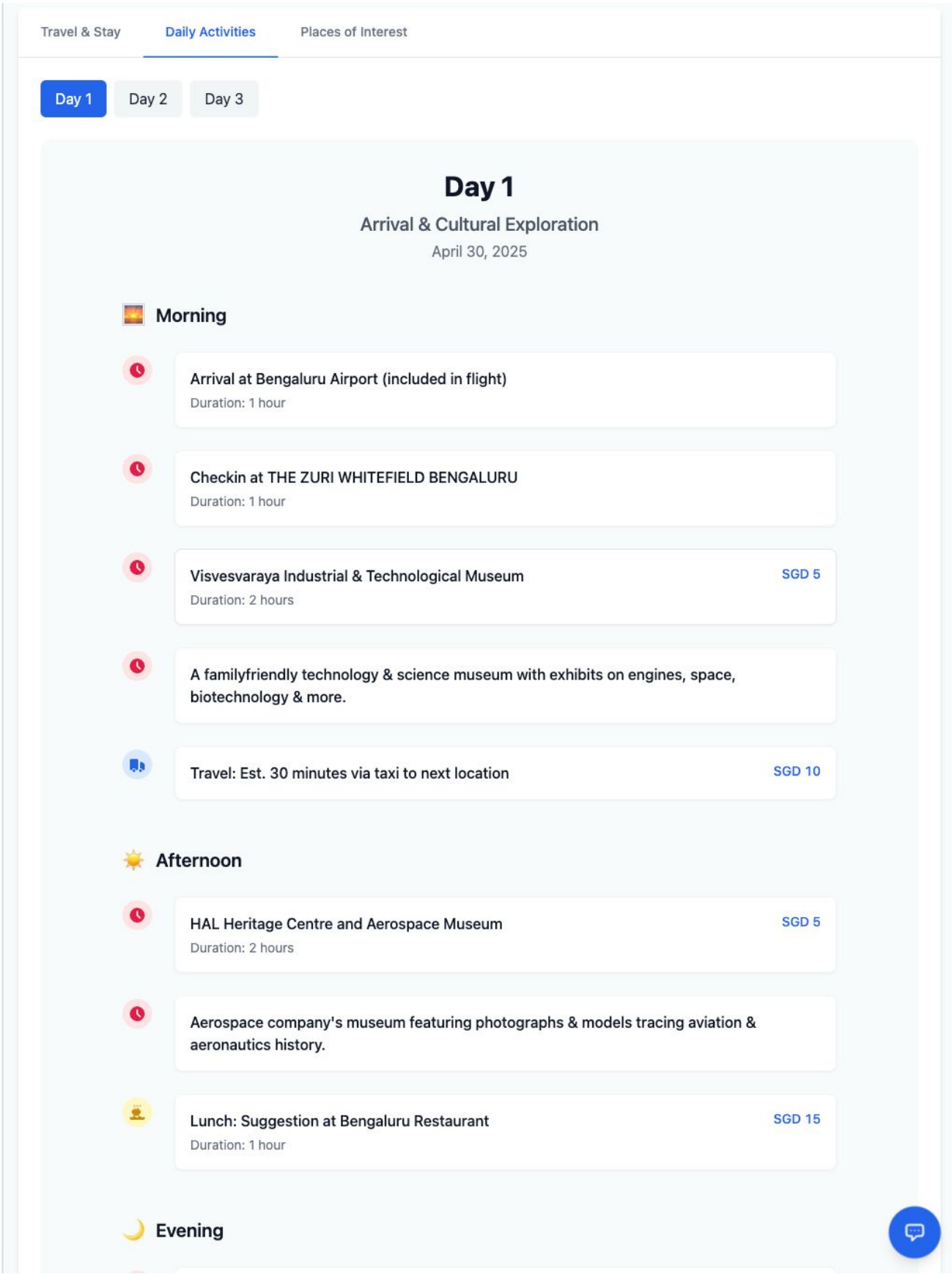

Figure 7. Day-to-Day Schedule

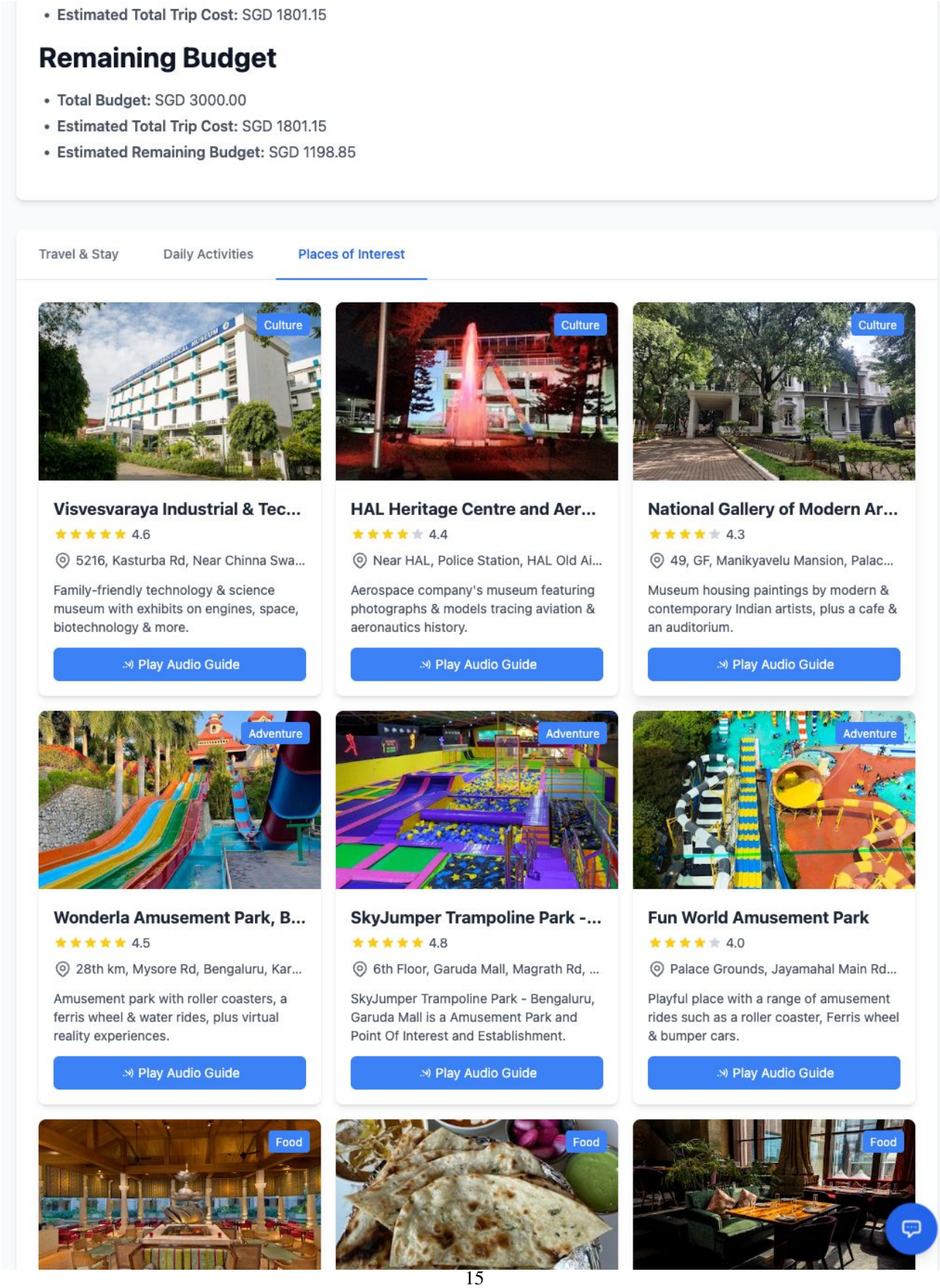

Figure 8. Audio Integration for POI

Afternoon:

- Lunch at *Bengaluru Restaurant* (Est. Duration: 1 hour) - Approx. Cost: SGD 15.00
- Visit *Visvesvaraya Industrial & Technological Museum* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to museum (Est. 30 minutes) - Approx. Cost: SGD 10.00

Evening:

- Dinner at *Lotus Pavilion, ITC Gardenia* (Est. Duration: 1.5 hours) - Approx. Cost: SGD 60.00
- Relax at the hotel or explore nearby areas (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 125.00

Day 2: May 1, 2025 - Cultural Exploration Morning:

- Breakfast at hotel (Included)
- Visit *HAL Heritage Centre and Aerospace Museum* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to museum (Est. 30 minutes) - Approx. Cost: SGD 10.00

Afternoon:

- Lunch at *Shiro Bengaluru* (Est. Duration: 1 hour) - Approx. Cost: SGD 50.00
- Visit *National Gallery of Modern Art* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to gallery (Est. 30 minutes) - Approx. Cost: SGD 10.00

Evening:

- Dinner at *Bengaluru Restaurant* (Est. Duration: 1.5 hours) - Approx. Cost: SGD 40.00
- Evening walk in nearby park (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 130.00

Day 3: May 2, 2025 - Adventure and Fun Morning:

- Breakfast at hotel (Included)
- Visit *Wonderla Amusement Park* (Est. Duration: 4 hours) - Approx. Cost: SGD 30
- *Travel:* Taxi to amusement park (Est. 1 hour) - Approx. Cost: SGD 25.00

Afternoon:

- Lunch at the park (Est. Duration: 1 hour) - Approx. Cost: SGD 20.00
- Continue at *Wonderla Amusement Park* (Est. Duration: 2 hours)

Evening:

- Dinner at *Lotus Pavilion, ITC Gardenia* (Est. Duration: 1.5 hours) - Approx. Cost: S
- Relax at the hotel (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 165.00

Day 4: May 3, 2025 - Departure Morning:

- Breakfast at hotel (Included)
- Last-minute shopping or exploring local markets (Est. Duration: 2 hours) - Free
- *Travel:* Taxi to airport (Est. 1 hour) - Approx. Cost: SGD 25.00

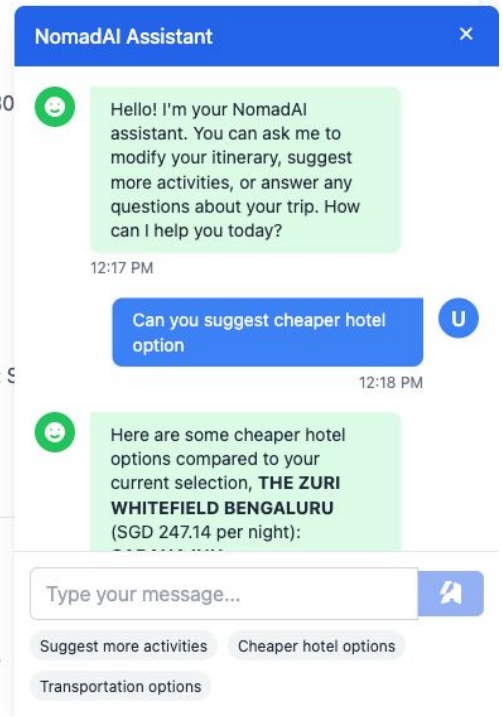

Figure 9. Chat Interface

Afternoon:

- Lunch at *Bengaluru Restaurant* (Est. Duration: 1 hour) - Approx. Cost: SGD 15.00
- Visit *Visvesvaraya Industrial & Technological Museum* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to museum (Est. 30 minutes) - Approx. Cost: SGD 10.00

Evening:

- Dinner at *Lotus Pavilion, ITC Gardenia* (Est. Duration: 1.5 hours) - Approx. Cost: SGD 60.00
- Relax at the hotel or explore nearby areas (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 125.00

Day 2: May 1, 2025 - Cultural Exploration Morning:

- Breakfast at hotel (Included)
- Visit *HAL Heritage Centre and Aerospace Museum* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to museum (Est. 30 minutes) - Approx. Cost: SGD 10.00

Afternoon:

- Lunch at *Shiro Bengaluru* (Est. Duration: 1 hour) - Approx. Cost: SGD 50.00
- Visit *National Gallery of Modern Art* (Est. Duration: 2 hours) - Approx. Cost: SGD 5.00
- *Travel:* Taxi to gallery (Est. 30 minutes) - Approx. Cost: SGD 10.00

Evening:

- Dinner at *Bengaluru Restaurant* (Est. Duration: 1.5 hours) - Approx. Cost: SGD 40.00
- Evening walk in nearby park (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 130.00

Day 3: May 2, 2025 - Adventure and Fun Morning:

- Breakfast at hotel (Included)
- Visit *Wonderla Amusement Park* (Est. Duration: 4 hours) - Approx. Cost: SGD 30
- *Travel:* Taxi to amusement park (Est. 1 hour) - Approx. Cost: SGD 25.00

Afternoon:

- Lunch at the park (Est. Duration: 1 hour) - Approx. Cost: SGD 20.00
- Continue at *Wonderla Amusement Park* (Est. Duration: 2 hours)

Evening:

- Dinner at *Lotus Pavilion, ITC Gardenia* (Est. Duration: 1.5 hours) - Approx. Cost: S
- Relax at the hotel (Est. Duration: 1 hour) - Free

Estimated Daily Cost: SGD 165.00

Day 4: May 3, 2025 - Departure Morning:

- Breakfast at hotel (Included)
- Last-minute shopping or exploring local markets (Est. Duration: 2 hours) - Free
- *Travel:* Taxi to airport (Est. 1 hour) - Approx. Cost: SGD 25.00

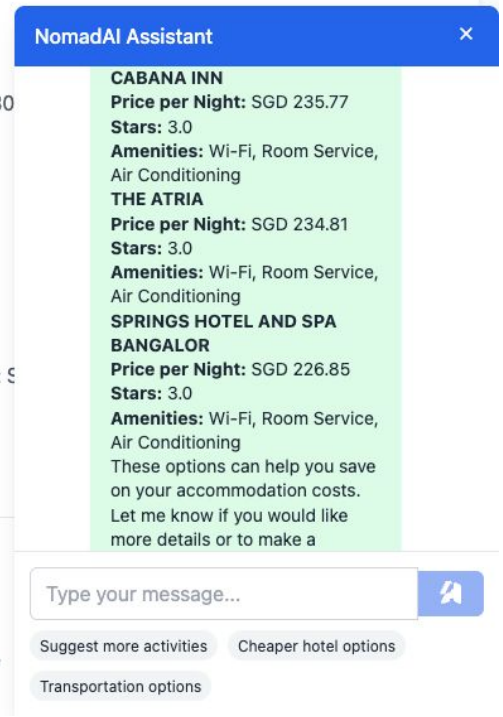

Figure 10. Chat Interface Continued