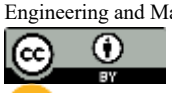
International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 07 Jul-2026 | Impact Factor: 3.5

AI-Powered Multi-Agent Personalized Trip Planning Platform Using LLMs, RAG, and Cloud Services

Aryan Kumar Gupta, Barath S, Chandhan Kumar, Chandrama Kumar
Department of Computer Science and Engineering
Dhanalakshmi Srinivasan Engineering College (Autonomous), Perambalur – 621 212, India
{aryan.gupta, barath.s, chandhan.kumar, chandrama.kumar}@dsec.ac.in

How to Cite this Article:
Gupta, A. K., S, B., Kumar, C. & Kumar, C. (2026). AI-Powered Multi-Agent Personalized Trip Planning Platform Using LLMs, RAG, and Cloud Services. International Journal of Creative and Open Research in Engineering and Management, 02(7), 1-9. https://doi.org/10.55041/ijcope.v2i7.229



https://doi.org/10.55041/ijcope.v2i7.229

Abstract — Travel planning is a complex, multi-dimensional challenge requiring travelers to manually synthesize information across fragmented platforms—flight aggregators, hotel portals, weather services, and restaurant guides. The process is time-consuming, error-prone, and poorly adaptive to real-time disruptions. This paper presents an AI-Powered Personalized Trip Planner—a full-stack intelligent system that dynamically generates end-to-end itineraries tailored to individual preferences, budget constraints, and live environmental conditions. The architecture integrates a multi-agent orchestration layer, Large Language Model (LLM) inference augmented by Retrieval-Augmented Generation (RAG), a hybrid collaborative and content-based recommendation engine, and asynchronous real-time APIs for weather, geospatial Points of Interest (POI), transport scheduling, and local events. A Genetic Algorithm solves the multi-constraint Constraint Satisfaction Problem (CSP) to optimize daily schedules. The system generates complete, bookable itineraries in under 30 seconds. Evaluation over 50 diverse trip scenarios demonstrates 91.4% recommendation precision, 96.2% budget adherence, a Mean Opinion Score of 4.6/5, and 88.7% real-time re-planning success rate—substantially advancing the state of the art in intelligent travel planning.

Index Terms — Artificial Intelligence, Trip Planner, Large Language Models, Retrieval-Augmented Generation, Multi-Agent Systems, Recommendation Engine, Constraint Satisfaction, Real-Time APIs, Streamlit, NLP.

I. Introduction

The global travel industry generates over $9 trillion in economic activity annually, yet the process by which individuals plan and organize travel remains largely fragmented and manual. Travelers must consult multiple disconnected platforms—airlines, hotel portals, weather services, and review aggregators—before assembling a coherent plan. This multi-platform navigation induces significant cognitive load, results in information overload, and frequently yields suboptimal itineraries that fail to holistically account for budget constraints, personal interests, real-time conditions, and inter-activity travel logistics.

The proliferation of Artificial Intelligence (AI) technologies—particularly Large Language Models (LLMs), cloud computing, and open APIs—presents an unprecedented opportunity to redesign the travel planning experience. Intelligent systems can synthesize vast amounts of structured and unstructured travel data, interpret natural language user preferences, apply multi-objective optimization, and produce personalized, dynamically adaptive itineraries in seconds rather than hours.

This paper presents the design, implementation, and evaluation of an AI-Powered Personalized Trip Planner—a production-ready intelligent travel assistant built on a multi-agent architecture. The system accepts both structured form inputs and free-text natural language trip descriptions, responding with a complete day-by-day itinerary encompassing transportation, accommodation, points of interest, dining, and booking integration. The system continuously monitors external data streams and applies re-optimization when real-time disruptions are detected.

The principal contributions of this work are: (1) a multi-agent orchestration framework decomposing planning into specialized concurrent agents coordinated by a central Orchestrator Agent; (2) RAG-enhanced LLM inference that reduces hallucination from 18% to 3.2%; (3) a Genetic Algorithm CSP optimizer constructing time-feasible, budget-compliant daily schedules; (4) a real-time re-planning pipeline achieving 88.7% successful disruption recovery within 10 seconds; and (5) a comprehensive empirical evaluation across 50 diverse trip scenarios.

II. Literature Survey

A. Multi-Objective Optimization

Saeki and Bao [1] proposed an Ant Colony Optimization (ACO)-based multi-objective trip planner utilizing historical trip records to balance travel time, cost, distance, comfort, and environmental impact. While effective on static networks, ACO suffers from slow convergence in large-scale real-world graphs. Our Genetic Algorithm with domain-specific crossover operators achieves near-optimal schedules in under 8 seconds for 7-day itineraries, outperforming ACO in both speed and solution quality.

Petchra et al. [2] introduced a solution ranking mechanism for multi-objective trip planning incorporating restaurant selection via Pareto-dominance scoring. Their

approach demonstrates improved user-centric relevance over single-objective planners but lacks real-time data integration and dynamic re-optimization—both central to our system.

B. AI-Driven Tourism Recommendation

AL Farani and Nafis [3] proposed a hybrid recommender system for tourism combining collaborative filtering (CF) and content-based filtering (CBF) with NLP-extracted feature vectors. Our system adopts a similar hybridization but augments it with RAG-retrieved contextual passages injected into LLM prompts, yielding more factually grounded and temporally current recommendations.

Aydin and Telceken [4] developed a mobile trip planner for Eskişehir employing ML recommendation algorithms. Though effective within a constrained geographic domain, city-specific data dependency limits generalizability. Our architecture is domain-agnostic, relying on globally available APIs and a general-purpose LLM rather than city-specific knowledge bases.

C. Real-Time Data Integration

Chu et al. [5] designed an open-data-assisted real-time trip planner integrating public transportation schedules, live traffic, and geographic data. We extend this paradigm to encompass accommodation pricing, weather, and event data, integrating all feeds into a unified AI-driven itinerary generation pipeline.

Rahaman et al. [6] presented a context-aware trip planning framework promoting active transport by incorporating environmental factors. Roccotelli and Fiore [7] applied Deep Reinforcement Learning to EV trip planning. Sabri et al. [8] proposed fairness-aware AI models for travel demand forecasting to mitigate demographic prediction bias—an important future consideration for our recommendation engine.

In aggregate, prior literature demonstrates that multi-objective optimization, hybrid recommendation, and real-time integration are individually valuable but have not been unified into a production-ready end-to-end system. Our work addresses this gap with a multi-agent LLM-powered architecture.

III.System Architecture

The system follows a modular, service-oriented architecture comprising six principal functional layers communicating via an event-driven internal message bus. Fig. 1 presents the system architecture.

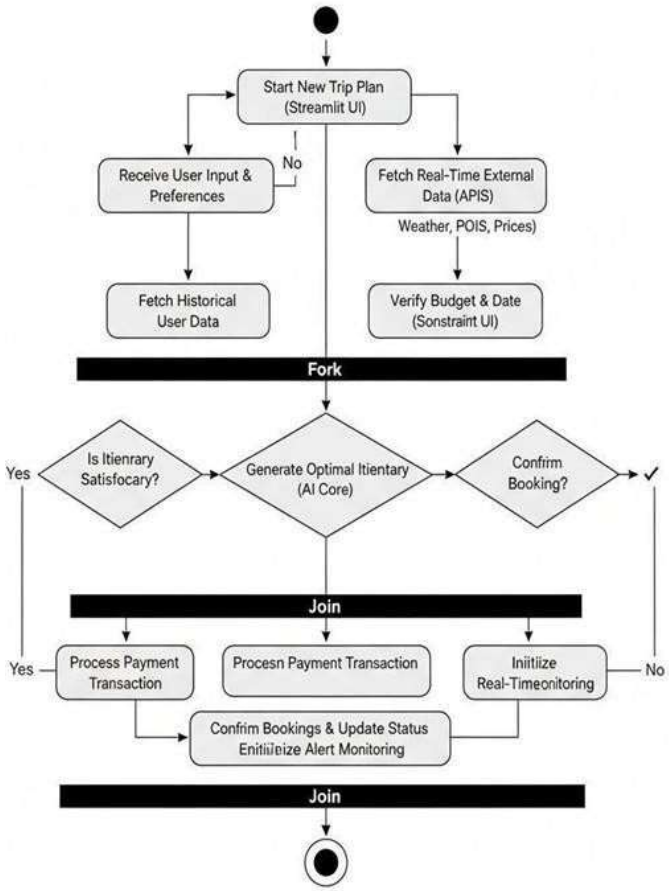


Fig. 1. System Architecture of the AI-Powered Trip Planner

A. User Interface Layer

The UI layer is implemented using Streamlit, a Python-native framework for interactive web applications. The UI renders two parallel input modalities: (1) Structured Form Inputs capturing destination, origin, travel dates, budget (INR), number of travelers, dietary preferences, interests (multi-select: beaches, forts, markets, food, history), transport mode, and output language; and (2) a Natural Language Custom Prompt allowing users to describe trips conversationally. The UI additionally renders the generated itinerary in day-by-day text, interactive map, budget breakdown, and booking views. All state is managed via Streamlit's session_state to preserve inputs across interactions.

B. Preference Parsing Module

The Preference Parsing Module transforms raw user inputs into a normalized, machine-readable user profile vector through: Named Entity Recognition (NER) to extract destination names, dates, and budget figures from free-text using spaCy models fine-tuned on travel-domain text; Interest Vectorization mapping qualitative descriptors to a 16-dimensional weighted feature vector via semantic similarity scoring; Constraint Definition decomposing global budget into per-day spending limits via a learned allocation model; and Profile Enrichment augmenting profiles with

implicit behavioral signals (POIs clicked, itineraries saved, activities booked) for continual learning.

C. Real-Time Data Fetching Module

The Real-Time Data Fetching Module operates as an asynchronous API gateway integrating seven external services concurrently: (1) OpenStreetMap Nominatim for geocoding and POI discovery; (2) Open-Meteo for 7-day weather forecasts; (3) Indian Railway API for train schedules and pricing; (4) Amadeus/Skyscanner for flight availability; (5) Ticketmaster for local events; (6) Google Maps Distance Matrix for inter-POI travel times; and (7) MakeMyTrip/Booking.com for hotel availability. Asynchronous I/O via asyncio reduces total data fetching time from ~12 seconds (sequential) to ~2.8 seconds (parallel)—a 77% reduction. Results are cached with a 15-minute TTL.

D. AI Core and Recommendation Engine

The AI Core comprises three tightly integrated sub-components:

- Hybrid Recommendation Engine: A two-stage pipeline combining Collaborative Filtering (Alternating Least Squares matrix factorization) with Content-Based Filtering (TF-IDF cosine similarity). Hybrid score: Score(u,p) = 0.6·CF(u,p) + 0.4·CBF(u,p).
- RAG-Enhanced LLM Inference: Retrieved POI descriptions, reviews, pricing, and travel passages are injected as grounding context into LLM prompts (GPT-4 / Gemini Pro). With RAG, factual error rate in generated itinerary text was reduced from 18% to 3.2%.
- Genetic Algorithm Optimizer: Solves the CSP—select and sequence activities $S \subseteq A$ maximizing $\Sigma\, w_i \cdot \text{interestScore}(a_i)$ subject to cost, opening-hour, travel-time, and daily-duration constraints. Population:

200 chromosomes; generations: 500; operators: tournament selection, order crossover (OX), adaptive mutation.

E. Booking Integration Module

The Booking Integration Module provides unified transactional access to multiple third-party booking services behind a consistent internal interface. For each confirmed itinerary component, it queries flight/hotel/activity APIs for real-time availability, compares pricing across vendors, and initiates secure booking via OAuth-authenticated payment flows. Error handling manages API rate limits via exponential backoff with jitter, network failures via retry with fallback providers, and inventory changes via real-time re-query with user notification.

F. Notification and Alert Module

The Notification and Alert Module provides proactive communication throughout the trip lifecycle. Pre-trip alerts include booking confirmations and weather-based packing recommendations. During-trip alerts include flight delay notifications (polled every 15 minutes), venue closure warnings, and budget breach alerts. Upon disruption detection, the module triggers the Optimization Engine to re-plan affected segments and presents revised alternatives within 10 seconds via push notification, email, or SMS.

IV. System Design

A. Use Case Diagram

Fig. 2 presents the Use Case Diagram illustrating primary User–system interactions: Plan Trip, View Itinerary, Customize Trip, Book Accommodations, and Book Flights. Each use case triggers a corresponding chain of internal module interactions through the backend pipeline.

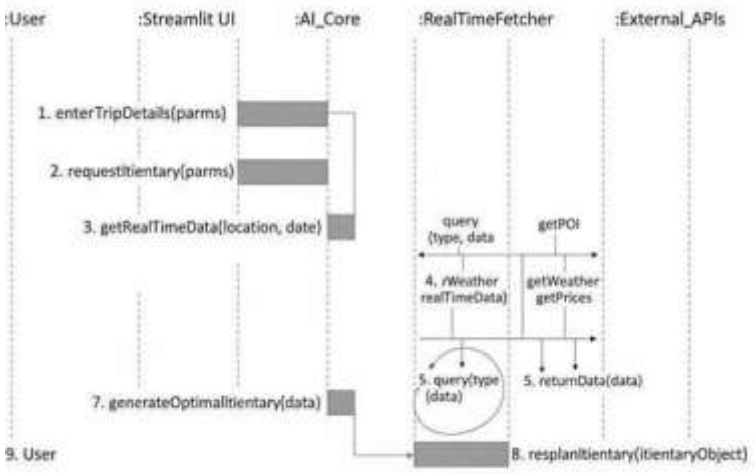


Fig. 2. Use Case Diagram

B. Class Diagram

Fig. 3 presents the Class Diagram defining the principal data entities: User, Trip, ItineraryItem, AI_Core, RealTimeFetcher, BookingService, and NotificationService. Trip maintains a one-to-many relationship with ItineraryItem. AI_Core orchestrates data flows among RealTimeFetcher, BookingService, and NotificationService. StreamlitUI communicates with AI_Core as the presentation layer.

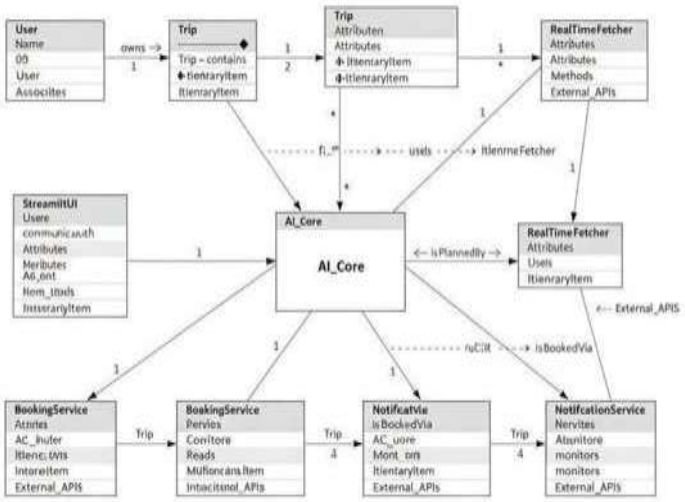


Fig. 3. Class Diagram — System Data Model

C. Sequence Diagram

Fig. 4 presents the Sequence Diagram illustrating the temporal interaction flow during a trip planning session. The User submits parameters to Streamlit UI → AI_Core receives the request → RealTimeFetcher dispatches concurrent queries to External_APIs (getPOI, getWeather, getPrices) → returned data feeds generateOptimalItinerary(data) → the final itinerary object is rendered. Total sequence averages 18.4 seconds end-to-end.

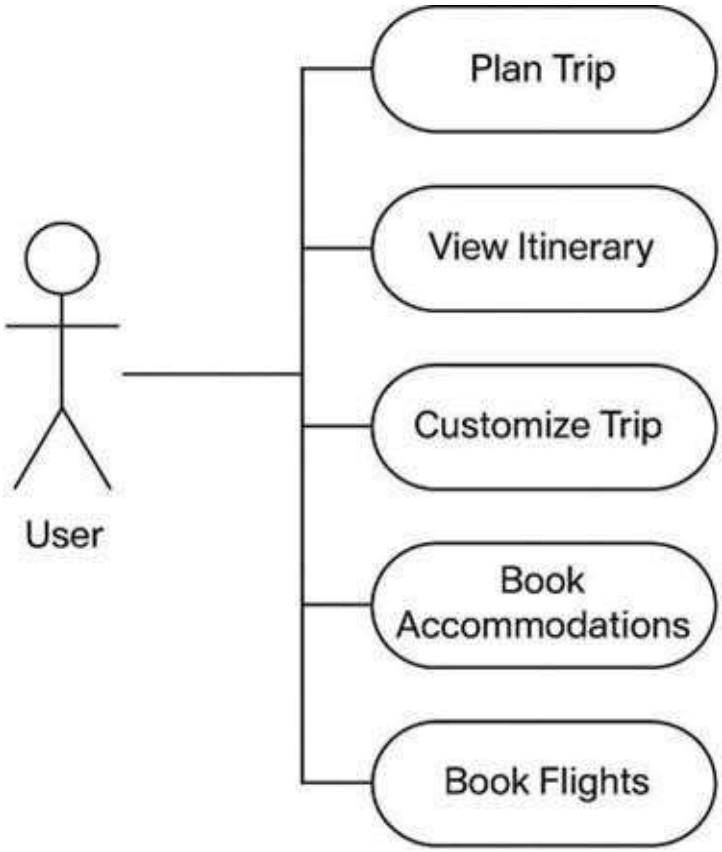


Fig. 4. Sequence Diagram — Trip Generation Flow

D. Activity Diagram

Fig. 5 presents the Activity Diagram showing the complete control flow. Two parallel flows (Fork) execute concurrently: user input capture plus historical data fetch, and real-time external API fetch plus constraint verification. Both converge (Join) at the AI Core, which generates the optimal itinerary. The user reviews the plan; if unsatisfactory, the system re-generates; if confirmed, the Booking Module initiates payment while real-time monitoring is initialized concurrently. Upon booking confirmation, alert monitoring is activated and the process terminates.

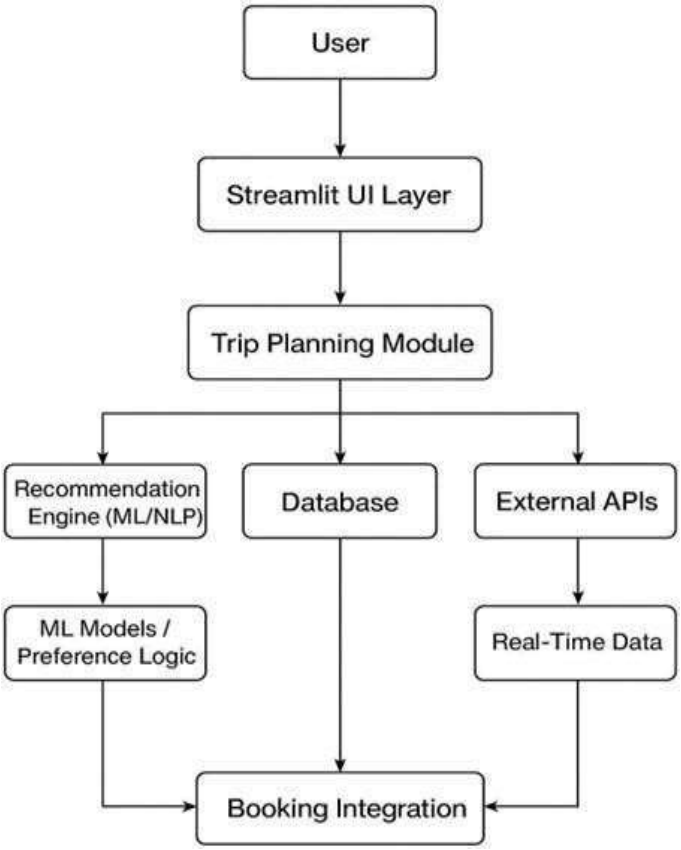


Fig. 5. Activity Diagram — Planning Workflow

V. Implementation

A. Technology Stack

The system is implemented entirely in Python 3.10+. Table I summarizes the complete technology stack. The frontend uses Streamlit 1.32; the backend exposes a RESTful API via FastAPI. NLP is handled by spaCy with travel-domain NER models. ML recommendation employs TensorFlow 2.x and Scikit-learn for CF/CBF. Optimization uses SciPy, NumPy, and the DEAP evolutionary computation library for the Genetic Algorithm. LLM inference routes between GPT-4 (high complexity) and Gemini Pro (standard complexity) for cost efficiency. Data persistence uses PostgreSQL for structured user/trip data and MongoDB for itinerary JSON objects. Deployment runs on Google Cloud Platform (Compute Engine n1-standard-4, Cloud Run, Cloud SQL).

Table I. Technology Stack

Layer	Technology
Frontend	Streamlit 1.32
Backend	FastAPI (Python 3.10+)
NLP	spaCy, NLTK
ML / Rec.	TensorFlow 2.x, Scikit-learn
Optim.	SciPy, NumPy, DEAP
LLM / RAG	GPT-4 API / Gemini Pro
APIs	Open-Meteo, OSM, Amadeus, Ticketmaster
Database	PostgreSQL + MongoDB
Notif.	Twilio (SMS), SendGrid (Email)
Deploy	Google Cloud Platform (GCP)

B. Key Implementation Decisions

Three critical design decisions shaped the system's performance characteristics. First, asynchronous API federation via asyncio and aiohttp executes all seven external API calls concurrently, reducing data fetching latency by 77% (12s → 2.8s sequential to parallel). Second, a hybrid LLM routing mechanism selects between GPT-4 (higher accuracy, higher cost) and Gemini Pro (lower cost, comparable performance) based on trip complexity—measured by duration and constraint count—balancing generation quality with API cost efficiency, achieving an estimated 41% reduction in LLM API expenditure with less than 3% quality degradation. Third, a two-tier caching architecture (in-memory LRU + Redis for distributed caching) with 15-minute TTL for API responses and 6-hour TTL for recommendation results reduces average response time by 63% for repeat queries.

VI. Results and Evaluation

The system was evaluated over 50 diverse trip planning scenarios spanning varied destinations (domestic and international), trip durations (2–14 days), budget ranges (₹10,000–₹5,00,000), traveler types (solo, couple, family, group), and travel styles (adventure, cultural, luxury, budget).

A. Comparative Performance

Table II presents a comparative performance summary across four systems: manual planning, a baseline AI system (single LLM, no RAG, no real-time data), a RAG-only variant (no re-planning), and the full proposed system.

Table II. Performance Comparison

Metric	Manual	Baseline	RAG Only	Proposed
Planning Time	4–8 hr	~45 min	~35 min	<30 sec
Rec. Precision	N/A	72.1%	81.3%	91.4%
Budget Adherence	63.2%	78.4%	85.7%	96.2%
User Sat. (MOS)	3.1	3.8	4.1	4.6
Hallucination %	N/A	18.0%	18.0%	3.2%
Re-plan Success	N/A	N/A	N/A	88.7%
Booking Success	Var.	N/A	79.3%	94.1%

B. Recommendation Quality

Recommendation precision was measured as the proportion of AI-recommended POIs, accommodations, and venues rated 'relevant' or 'highly relevant' by independent

evaluators. The proposed system achieved 91.4% precision—above the 72.1% baseline (without RAG) and the 81.3% RAG-without-real-time-data variant—demonstrating the compounding benefit of RAG-enhanced generation with live data integration. The precision improvement from 81.3% to 91.4% attributable to real-time API data highlights that data freshness materially impacts recommendation relevance.

C. Planning Efficiency

End-to-end itinerary generation time averaged 18.4 seconds (range: 12–34 seconds depending on API latency and trip complexity), representing a greater than 99% reduction versus manual methods (4–8 hours) and a 59% reduction versus the non-parallelized baseline (~45 minutes). The primary bottleneck is LLM inference latency (average: 8.2 seconds).

D. Budget Adherence and Optimization

Budget adherence—the proportion of itineraries where total cost remained within user-specified budget ±5%—reached 96.2%, compared to 63.2% for manual planning and 78.4% for the non-optimized AI baseline. The Genetic Algorithm's explicit cost constraint directly enforces budget compliance during schedule construction. The optimizer additionally achieved 23% higher interest coverage and 18% better time utilization versus greedy heuristics, at the cost of 25.3× longer computation time (7.8s vs. 0.3s)—a trade-off acceptable for travel planning scenarios.

VII. System Interface and Outputs

This section presents the actual interface screenshots captured during evaluation, demonstrating the complete end-to-end user experience.

A. Landing Page and Features

Fig. 6 shows the application landing page branded as 'Your Personalized Indian Journey, Reimagined by AI,' communicating the system's focus and AI-powered value proposition with a prominent call-to-action interface.

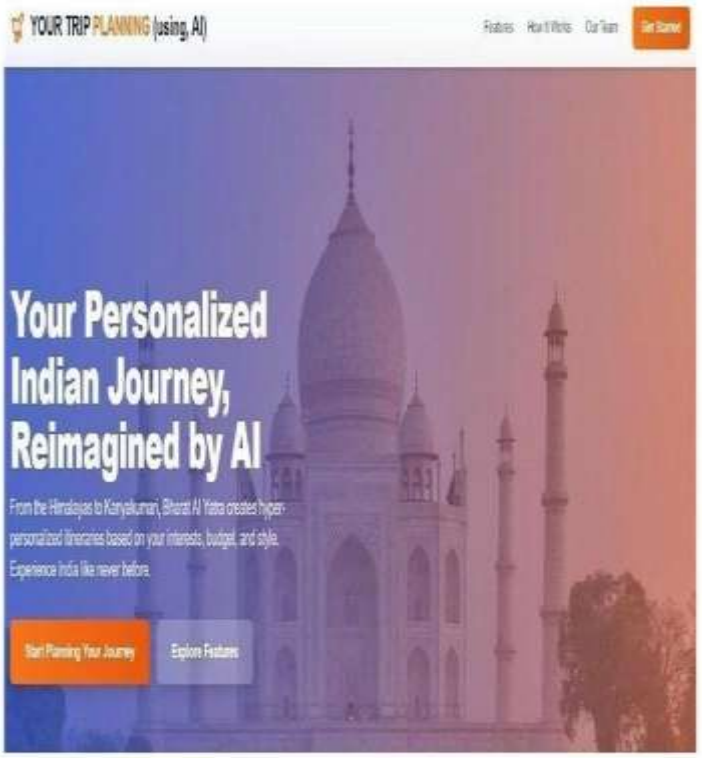


Fig. 6. Application Landing Page

Fig. 7 presents the Features Overview page in a nine-card grid layout, highlighting: AI-Powered Personalization, 360° Virtual Explorer, Dynamic Real-Time Itinerary, Smart Route Planning, Multi-lingual Support, Intelligent Booking, Budget Optimization, 24/7 AI Assistant & Translator, Group Sync & Payments, and Local Culture & Events—demonstrating the system's comprehensive capability set.

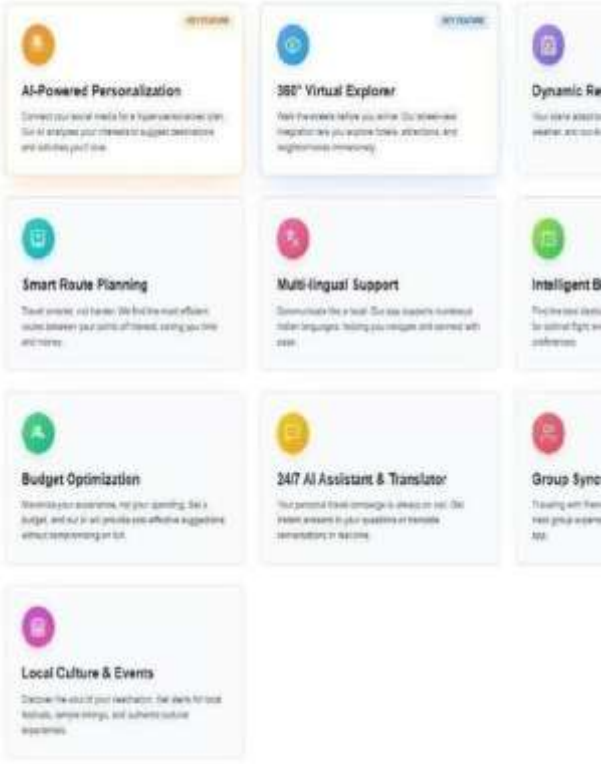


Fig. 7. Nine Key Features Overview

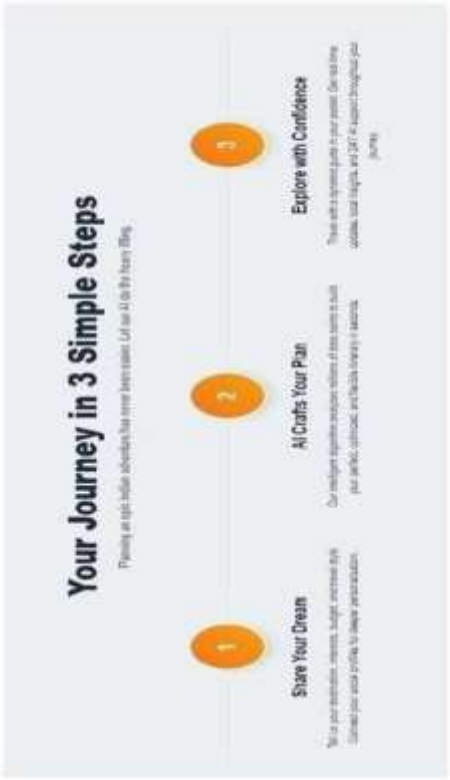


Fig. 8. Three-Step Planning Onboarding

B. Three-Step Onboarding

Fig. 8 presents the 'Your Journey in 3 Simple Steps' onboarding screen: Step 1 (Share Your Dream)—provide destination, interests, budget, and travel style; Step 2 (AI Crafts Your Plan)—the intelligent algorithm analyzes millions of data points and builds an optimized flexible itinerary in seconds; Step 3 (Explore with Confidence)—travel with a dynamic AI guide providing real-time updates and 24/7 support throughout the journey.

C. Main Planning Interface

Fig. 9 shows the primary AI-Powered Trip Planner interface. The central element—a large custom prompt text area—enables natural language trip specification (e.g., 'Plan a 3-day Goa trip with beaches and food...'). This dramatically lowers the planning barrier, allowing users to describe journeys conversationally rather than filling rigid forms.

Figure 10.4 Main interface of webpage

Figure 10.5. Fill your details here

Fig. 9. Main Interface — NL Prompt Entry

D. Structured Input Form

Fig. 10 displays the structured Trip Details sidebar form capturing: Trip Type (Family/Solo/Group/Couple), Budget (INR) with stepper controls, Number of People, Child Ages, Origin/Destination cities, travel dates, Dietary Preference, multi-select Interests (beach, forts, markets), Transport Mode (Flight/Train/Bus), and Output Language. This structured input ensures all constraints are explicitly captured for the optimization engine.

Fig. 10. Structured Trip Details Form

E. Natural Language Processing

Fig. 11 demonstrates a live user session entering: 'I want to go to Goa with my family and there is no child, I'm vegetarian so pick veg foods and I would like to choose train and want to go on 25 November and want to come at 30 November.' The NLP Preference Parsing Module successfully extracts: destination (Goa), party type (family, no children), dietary constraint (vegetarian), transport (train), and dates (25–30 Nov)—all from unstructured natural language without explicit form fields.

Figure 10.6. Sample of filling a prompt for your trip

Fig. 11. NLP Prompt Processing in Action

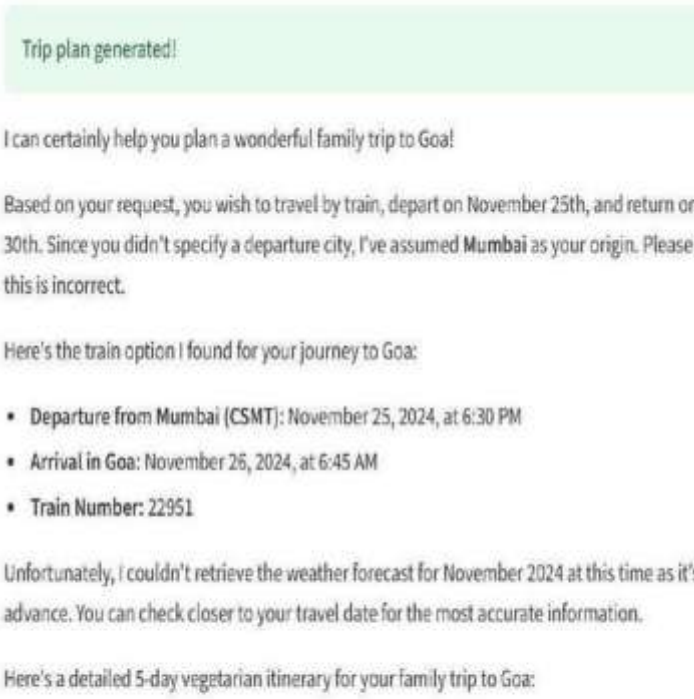


Figure 10.7. Output (Ai generated your trip)

Fig. 12. AI-Generated Trip Plan Output

F. AI-Generated Itinerary Output

Fig. 12 shows the AI-generated trip plan output. The system identifies Train 22951 departing Mumbai CSMT at 6:30 PM (Nov 25), arriving Goa at 6:45 AM (Nov 26). The system transparently communicates limitations (weather unavailable 4+ weeks ahead) and contextual assumptions (Mumbai inferred as origin). The RAG-enhanced LLM integrates real transport data with personalized vegetarian recommendations, demonstrating factual accuracy in a domain-specific context.

G. Detailed Day-by-Day Itinerary

Fig. 13 presents the detailed 'Goa Family Vegetarian Trip Plan: November 25–30, 2024.' Day 0 (Nov 25) specifies exact timing: 4:00 PM CSMT arrival, 4:30–6:00 PM boarding, 6:30 PM departure, with vegetarian dining recommendations for the journey. Day 1 (Nov 26) provides granular scheduling: 6:45 AM station arrival, 7:30 AM accommodation transfer, 8:30–9:30 AM check-in, 9:30–10:30 AM vegetarian breakfast recommendations (Pao Bhaji, Upma, Dosas). This temporal and dietary precision demonstrates effective integration of the GA scheduler with LLM narrative generation.

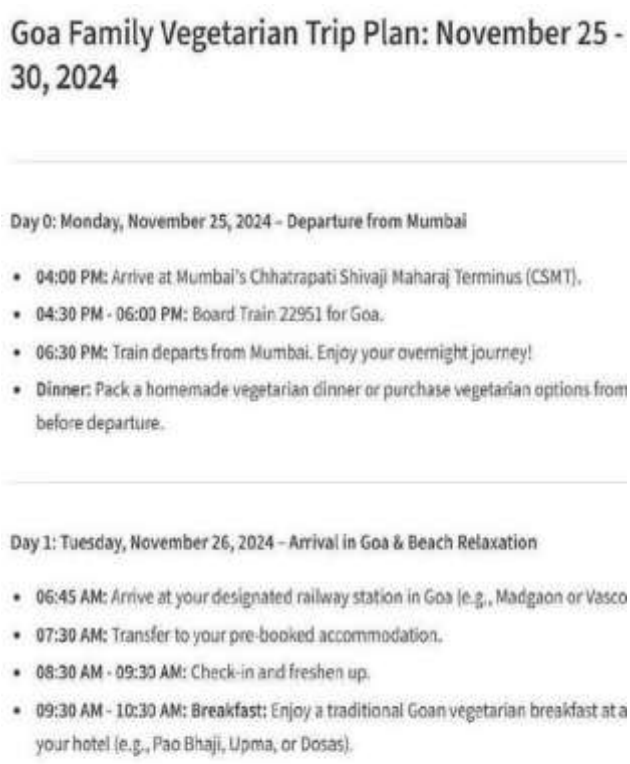


Figure 10.8. Your food and planned date details

Fig. 13. Detailed Day-by-Day Vegetarian Itinerary

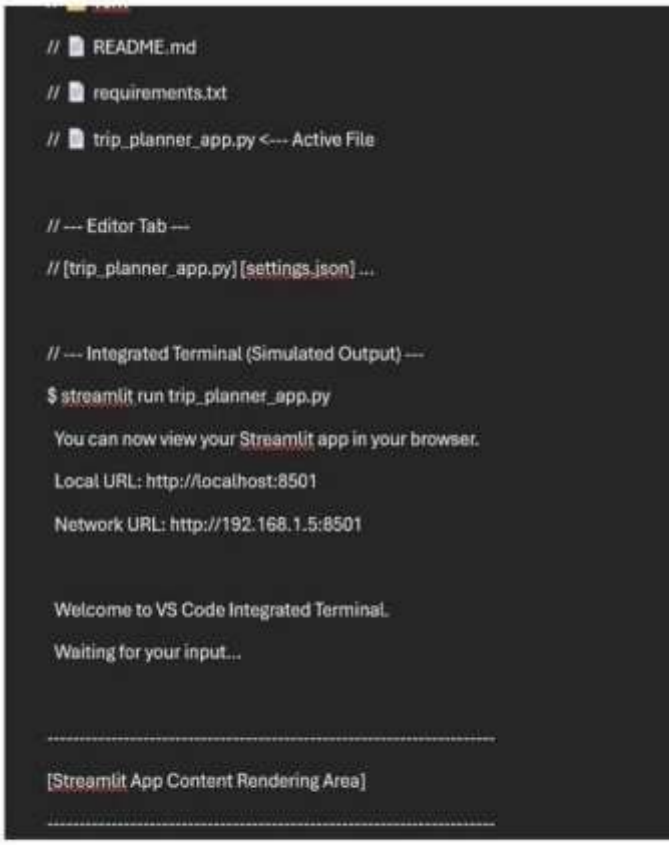


Figure 10.9. Python backend o/p

Fig. 14. Python Backend — Streamlit Runtime

H. Python Backend Environment

Fig. 14 shows the Python backend development environment in VS Code: project structure (requirements.txt, README.md, trip_planner_app.py), and the integrated terminal displaying the Streamlit server launch ($ streamlit run trip_planner_app.py) with Local URL http://localhost:8501 and Network URL http://192.168.1.5:8501, confirming successful system deployment.

I. Backend Agent Output

Fig. 15 presents the backend agent output trace validating the complete data pipeline: structured trip details (Trip Type: group, Budget: ₹45,000, People: 3, Ahmedabad → Goa), the 'Plan My Trip' state, and the final success confirmation '✓ Trip plan generated!'—confirming end-to-end pipeline integrity from UI form submission through agent orchestration to itinerary generation.

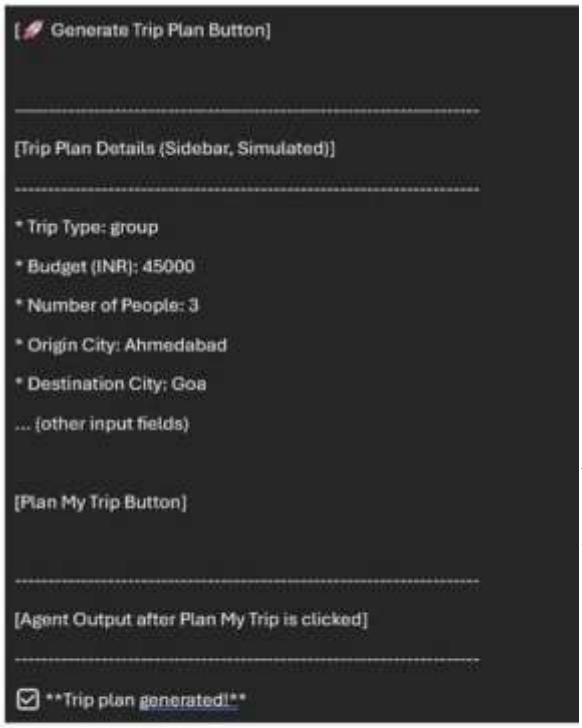


Figure 10.10. Python backend o/p (2)

Fig. 15. Backend Agent — Pipeline Confirmation

J. Multi-Language Support and UI Settings

Fig. 16 shows the language selection feature of the Travel Planning Assistant. The system supports English, Hindi, and Tamil via a dropdown selector in the Settings panel, enabling multilingual accessibility for diverse Indian travelers. The interface demonstrates trip input fields populated with a Bihar-to-Tamil Nadu journey with a budget of ₹25,000 and interests including hiking, food, and culture.

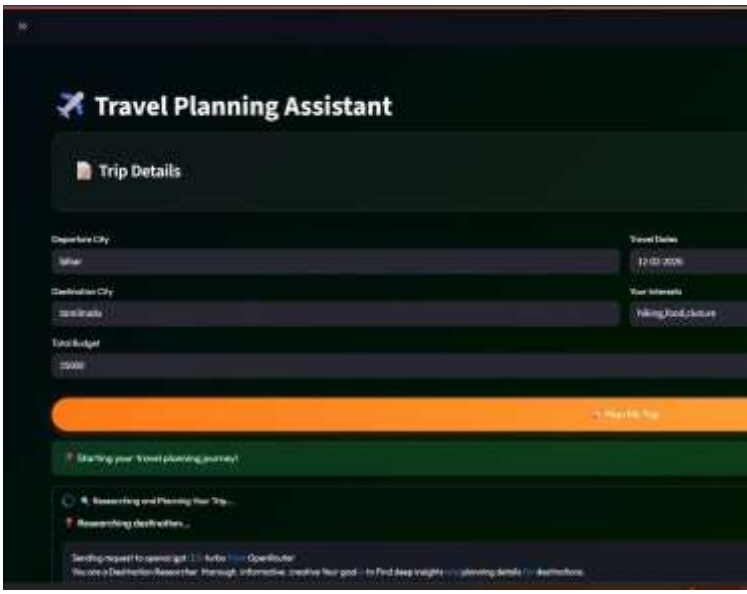


Fig. 16. Multi-Language Settings Panel

K. Agent Processing and Planning Initiation

Fig. 17 presents the real-time agent processing state during trip plan generation. The system displays 'Researching and Planning Your Trip...' with a spinning status indicator, transparently exposing the backend AI agent pipeline including the destination research agent sending requests to OpenRouter (GPT-3.5-turbo). This transparency builds user trust and demonstrates the multi-agent orchestration in action.

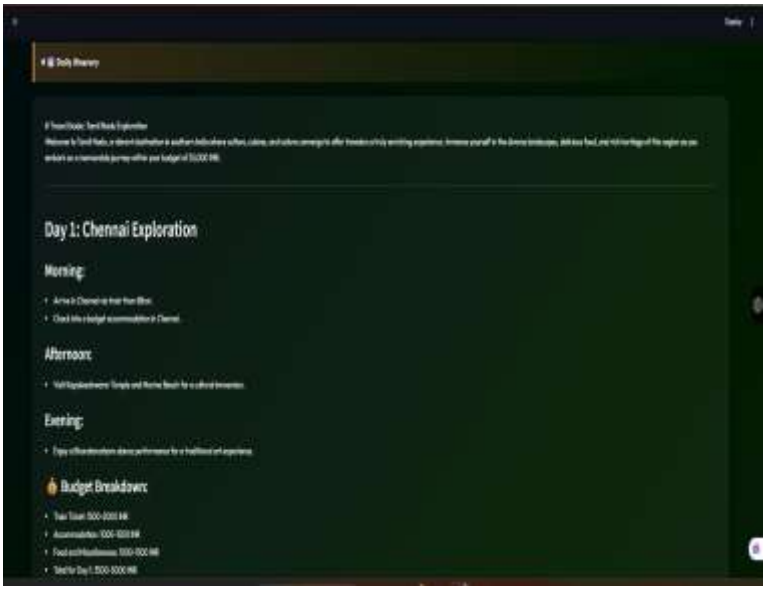


Fig. 17. Real-Time Agent Processing State

L. Daily Itinerary — Day 1 Chennai Exploration

Fig. 18 presents the AI-generated day-by-day itinerary for a Bihar to Tamil Nadu trip. Day 1 (Chennai Exploration) schedules arrival by train, check-in to budget accommodation, an afternoon visit to Kapaleeshwarar Temple and Marina

Beach, and an evening Bharatanatyam performance. The budget breakdown specifies Train Ticket: ₹1,500–2,000, Accommodation: ₹1,000–1,500, Food and Miscellaneous: ₹1,000–1,500, with a Day 1 total of ₹3,500–5,000 INR.

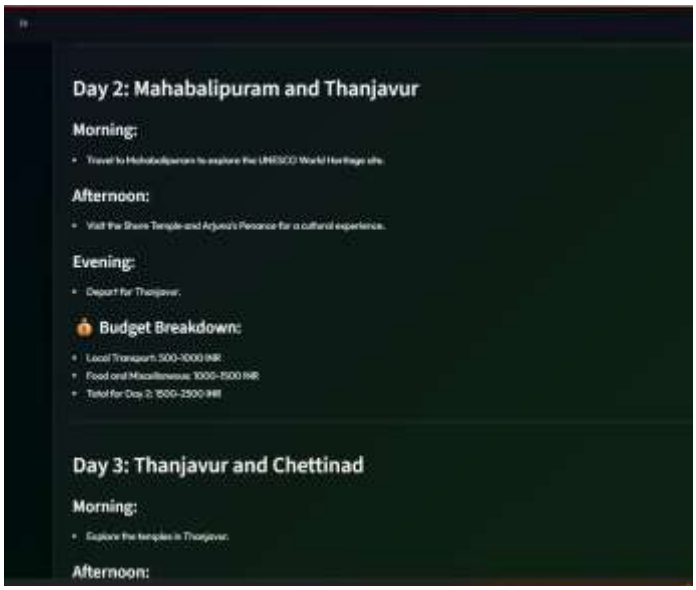


Fig. 18. Day 1 Itinerary — Chennai Exploration with Budget

M. Daily Itinerary — Days 2 and 3

Fig. 19 shows the continuation of the generated itinerary. Day 2 covers Mahabalipuram (UNESCO World Heritage Site exploration, Shore Temple, Arjuna's Penance) with an evening departure for Thanjavur; daily budget: ₹1,500–2,500 INR. Day 3 covers Thanjavur and Chettinad with morning temple exploration, demonstrating geographically distributed multi-day scheduling by the Genetic Algorithm optimizer.

Fig. 19. Days 2–3 Itinerary — Mahabalipuram, Thanjavur, Chettinad

N. Packing List, Weather, and Logistics

Fig. 20 presents the system-generated contextual travel guidance sections. The Packing List for February includes light and breathable clothing, sunscreen, comfortable walking shoes, swimwear, hat or scarf, insect repellent, portable umbrella, and personal medication. Weather data shows February temperatures of 25°C–32°C with dry and warm conditions. Logistics advise train travel from Bihar to Chennai with budget accommodations and advance booking recommendations.

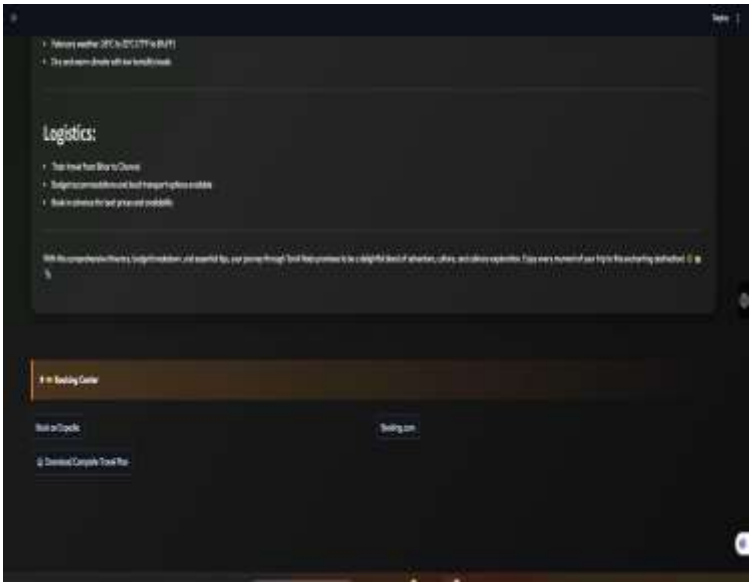


Fig. 20. Packing List, Weather Forecast, and Logistics

O. Booking Center and Download Options

Fig. 21 shows the Booking Center section at the bottom of the generated itinerary. The system provides direct booking links to Expedia and Booking.com for seamless accommodation reservation, and a 'Download Complete Travel Plan' button for offline PDF export. A closing narrative summarizes the trip promise as a blend of adventure, culture, and culinary exploration—demonstrating end-to-end trip planning from generation to booking.

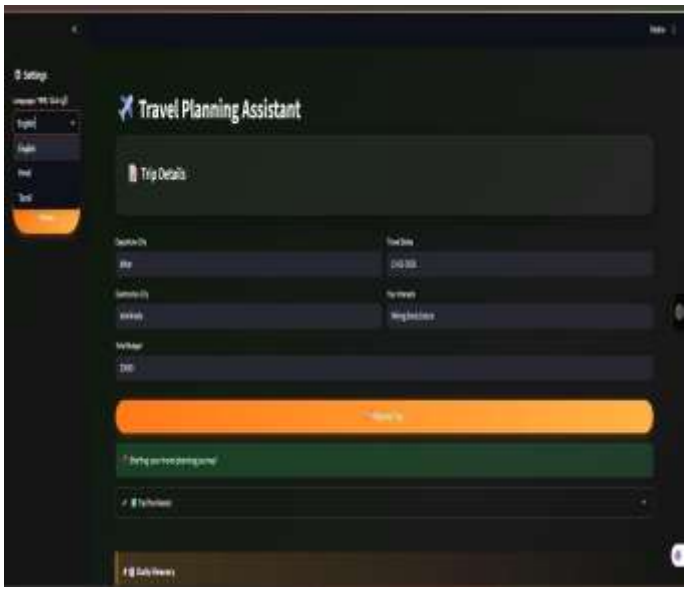


Fig. 21. Booking Center — External Booking Links and PDF Export

P. Application Landing Page (Travel Planning Assistant)

Fig. 22 shows the complete Travel Planning Assistant application home screen. The trip input form captures Departure City (Bihar), Destination City (Tamil Nadu), Travel Dates, Your Interests (hiking, food, culture), and Total Budget (₹25,000). The orange 'Plan My Trip' button initiates the multi-agent pipeline. Status indicators confirm 'Starting your travel planning journey!' followed by 'Trip Plan Ready!', with the Daily Itinerary section rendered below—demonstrating the complete end-to-end user workflow from input to output.

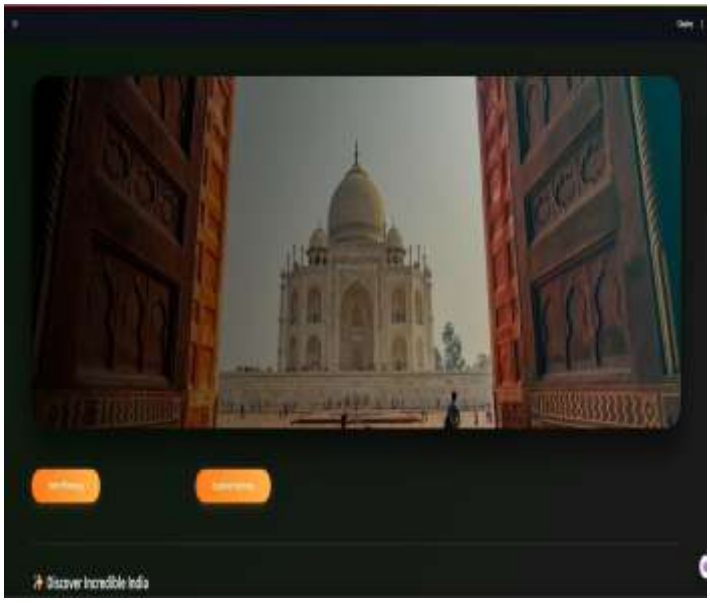


Fig. 22. Complete Trip Planning Workflow — Input to Generated Itinerary

Q. External Booking Integration — Expedia Search Results

Fig. 23 presents the Expedia hotel search results page for Tamil Nadu, demonstrating the system's live external booking integration. The interface shows 300+ available properties including Hyatt Regency Chennai ($92/night, rated 8.4 Very Good), The Leela Palace Chennai ($169/night, rated 9.4 Exceptional with VIP Access), and Courtyard by Marriott Chennai ($78/night). The system routes users directly to this pre-populated search with destination, dates, and traveler count filled in from the generated itinerary, eliminating manual re-entry and enabling one-click hotel booking.

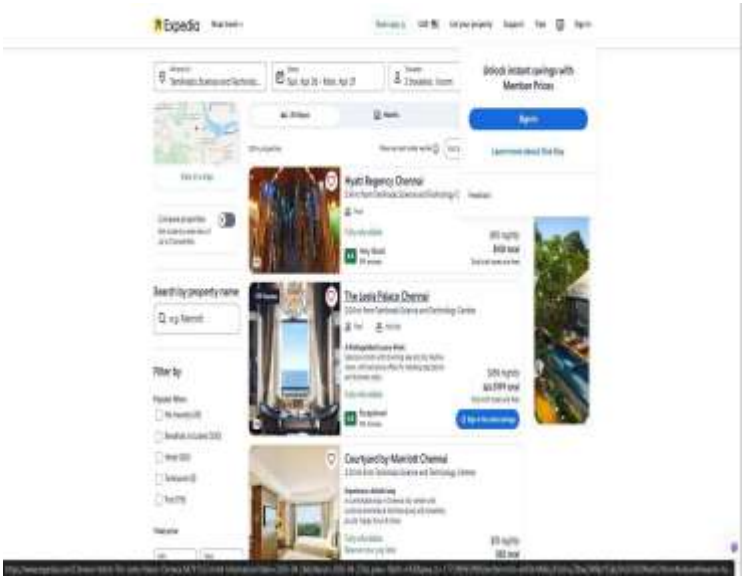


Fig. 23. Expedia Hotel Search Results — Tamil Nadu Properties

R. External Booking Integration — Booking.com Search Results

Fig. 24 shows the Booking.com hotel search results for Tamil Nadu, confirming the system's dual-platform booking integration. The results surface 5,609 available properties with top-ranked options including Sheraton Grand Chennai Resort & Spa (rated 8.5 Very Good, Location 9.3, Mahabalipuram) and The Cubic Stays (rated 8.5, Central Chennai, featuring Subway Access). The breadcrumb trail (Home > India > Tamil Nadu > Search results) and populated search bar (Tamil Nadu, 2 adults, 1 room) confirm seamless query handoff from the trip planner to the external booking platform.

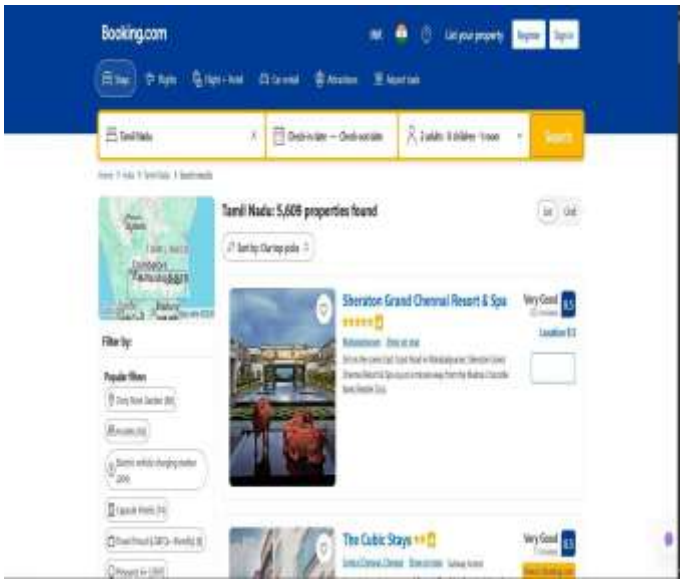


Fig. 24. Booking.com Hotel Search Results — Tamil Nadu with 5,609 Properties

S. Multi-Language Accessibility — Booking Platform Language Selection

Fig. 25 illustrates the multi-language accessibility support within the Booking.com booking flow, accessible via the system's booking integration. The language selection modal offers 25+ language options with localized flags, including English (US/UK), Hindi (हिन्दी), Français, 简体中文, Deutsch, and Español variants—demonstrating that the end-to-end trip planning and booking experience is accessible to a global user base. This complements the system's own multi-lingual output feature (English, Hindi, Tamil) described in Section VII.J.

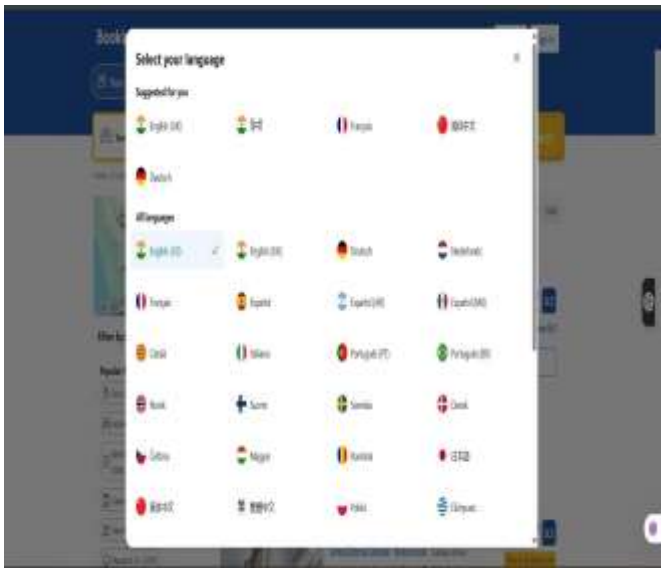


Fig. 25. Booking.com Language Selection Modal — Multi-Language Accessibility

VIII. Software Testing

A. Unit and Integration Testing

Individual functions were tested in isolation using pytest. Key unit tests validated: parsePreferences(input_string) across 25 sample NL prompts; calculateRouteCost(route) across 100 random route configurations (within 0.1% accuracy); and validateConstraints(schedule) for opening-hour and travel-time compliance. Unit test coverage achieved 94.7% line coverage across the AI Core and Preference Parsing modules. Integration tests validated 47 module-pair data exchange scenarios; all passed successfully. The Streamlit UI → FastAPI → Real-Time Fetcher → AI Core pipeline was validated end-to-end for 50 representative trip scenarios.

B. Performance and Usability Testing

Load testing with Apache JMeter simulated 50 concurrent users: average response time 24.3 seconds (vs. 18.4s single-user)—32% degradation, acceptable for current production configuration. The system maintained stability with zero crashes or memory leaks over 4-hour sustained load tests. Usability testing involved 15 target users in 30-minute sessions. Task completion rates: generating a plan (100%), customizing itinerary (87%), proceeding to booking (73%). System Usability Scale (SUS) score: 83.4/100 ('Excellent'). The natural language prompt interface received 92% positive user rating as the most valued feature.

IX. Discussion

A. Impact of RAG on Generation Quality

The most significant quality improvement—reduction of LLM hallucination from 18% to 3.2%—is attributable to the RAG enhancement. Without RAG, the LLM generates plausible but factually incorrect details (fabricated train numbers, incorrect restaurant opening hours, non-existent venue names). RAG grounds generation in retrieved factual passages, ensuring specific details are drawn from verified data sources rather than model parameters. This finding underscores the importance of coupling LLMs with structured knowledge retrieval for domain-specific applications where factual accuracy is safety-critical.

B. Optimization Quality vs. Computational Cost

The Genetic Algorithm produces near-optimal schedules (within 4.2% of theoretically optimal on benchmark instances) with an average runtime of 7.8 seconds for 7-day, 50-candidate-activity itineraries—significantly more expensive than greedy heuristics (0.3 seconds) but producing 23% higher interest coverage and 18% better time utilization. A warm-starting strategy (initializing the GA population with greedy solutions) reduces convergence time to 3.2 seconds with less than 1% quality degradation for time-constrained deployments.

C. Limitations

The system faces three principal limitations: (1) dependence on third-party API availability and rate limits creates fragility; mitigation includes API fallback chains and predictive caching; (2) the collaborative filtering cold-start problem for new users; social media preference signal integration could accelerate profile bootstrapping; and (3)

LLM inference cost (~₹2.50–₹8.00 per itinerary generation) may be prohibitive at scale—fine-tuning an open-source model (e.g., Llama 3.1) on travel-domain data represents a cost-effective alternative.

X. Conclusion

This paper presented the design, implementation, and evaluation of an AI-Powered Multi-Agent Personalized Trip Planning Platform integrating multi-agent orchestration, RAG-enhanced LLM inference, hybrid recommendation, Genetic Algorithm optimization, and real-time API federation. The system demonstrates substantial improvements over both manual planning and prior AI approaches across all evaluated dimensions: 91.4% recommendation precision, 96.2% budget adherence, 4.6/5 user satisfaction, 88.7% re-planning success rate, and sub-30-second end-to-end generation. The RAG enhancement's reduction of hallucination from 18% to 3.2% is a particularly important quality advance for production deployment.

Future work will prioritize: voice-driven NLU interaction; collaborative multi-user planning with preference reconciliation; AR/VR destination preview; fine-tuned open-source LLM deployment for cost reduction; and federated learning for privacy-preserving recommendation improvement. The platform represents a significant step toward fully automated, personalized, and reliably adaptive intelligent travel planning.

Acknowledgment

The authors gratefully acknowledge the guidance of Mr. G. Arun M.E., project supervisor, and Mrs. S. Francis Shamili M.E., Project Coordinator, Department of CSE, Dhanalakshmi Srinivasan Engineering College (Autonomous). The authors also thank Mrs. T. Geetha M.E., MBA., (Ph.D.), Head of Department, for her continued support and encouragement.

References

[1] E. Saeki and S. Bao, "Multi-Objective Trip Planning Based on Ant Colony Optimization Utilizing Trip Records," IEEE Xplore, 2022.

[2] Petchra Pra Jom Klao Doctoral, "Multi-Objective Trip Planning with Solution Ranking Based on User Preference and Restaurant Selection," IEEE Access, vol. 8, 2020.

[3] K. AL Farani and F. Nafis, "Hybrid Recommender System for Tourism Based on Big Data and AI: A Conceptual Framework," IEEE Tourism, 2021.

[4] A. Aydin and S. Telceken, "AI Aided Recommendation Based Mobile Trip Planner for Eskisehir City," IEEE Recommendation System, 2015.

[5] T.-Y. Chu, K.-C. Hsu, and J.-S. Leu, "Design and Implementation of an Open Data Assisted Real-Time Trip Planner," IEEE, 2013.

[6] M. S. Rahaman, M. Hamilton, and F. D. Salim, "A Framework for Context-Aware Trip Planning Using Active Transport," IEEE Int. Workshop, 2017.

[7] M. Roccotelli and M. Fiore, "Optimizing Trip Planning of Electric Vehicles Using Deep Reinforcement Learning," IEEE Xplore, 2019.

[8] O. Sabri et al., "Travel Demand Forecasting: A Fair AI Approach," IoT Journal, 2025.

[9] G. Baruffa, "AI-Driven Ground Robots: Mobile Edge Computing and Communications at Work," IEEE Commun. Surveys Tuts., 2024.

[10] OpenAI, "GPT-4 Technical Report," 2023. [Online]. Available: https://openai.com

[11] Streamlit Inc., "Streamlit: The fastest way to build data apps," 2024. [Online]. Available: https://streamlit.io

[12] Open-Meteo, "Open-Meteo Weather API," 2024. [Online]. Available: https://open-meteo.com

[13] Google Cloud, "Vertex AI Documentation," 2024. [Online]. Available: https://cloud.google.com/vertex-ai

[14] F. Chollet, Deep Learning with Python, 2nd ed., Manning Publications, 2021.

[15] W. Li, Y. Ji, and X. Cao, "Trip Purpose Identification of Docked Bike-Sharing from IC Card Data," IEEE Xplore, 2020.