iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79415

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Travel Ease - An AI Agent for Smart Gate Travel Planning

Y. Nandini[1], M. Srinidhi[2], Mrs. Y. Ashwini[3]
[1, 2]*Students, Department of CSE, Methodist College of Engineering and Technology, Abids, Hyderabad, Telangana, 500001, India*
[3]*Assistant Professor, Department of CSE(AI&ML) , Methodist College of Engineering and Technology, Abids, Hyderabad, Telangana, 500001, India*

Abstract: ***TravelEase is an AI-based smart travel planning system designed to simplify and enhance the travel experience by providing intelligent and personalized recommendations. The system integrates a user-friendly frontend with a Node.js backend to process user inputs and manage travel operations efficiently. It utilizes AI-based planning logic to generate optimized travel itineraries based on user preferences such as destination, budget, and travel dates.The application stores and manages data using JSON-based datasets and cloud services, while Firebase is used for authentication and hosting. Additionally, the system integrates external APIs to fetch travel-related data such as flights and hotels. TravelEase reduces the need for multiple applications by offering an all-in-one platform for searching, planning, and booking travel. The system improves efficiency, saves time, and enhances user experience through intelligent automation and real-time-like data processing.***
Keywords: AI-based Travel Planning, Smart Itinerary Generation, Node.js Backend, Firebase , Personalized Recommendations, Travel Booking System, Authentication, API Integration, Web Application.

I. INTRODUCTION

Anyone who has planned a trip from scratch knows the problem. You start with a rough idea — a destination, some dates — and within an hour you have six browser tabs open, three of which have expired session timeouts, and you still haven't compared prices properly. The process is fragmented by design, because the tools were never meant to talk to each other.

This isn't a new observation. Researchers and developers have been chipping away at it for years: smarter booking interfaces, recommendation algorithms, route planners. Each one solves a piece of the problem. None of them solve the whole thing, because they weren't designed to. A hotel recommendation engine doesn't know about your flight. A flight aggregator doesn't care about your budget for accommodation.

The question this paper addresses is straightforward: what would a system look like if it actually handled all of it in one place? TravelEase is our answer to that. This paper walks through the design decisions behind it, what existing systems got right and wrong, and what the initial results look like.

II. LITERATURE SURVEY

AI has been reshaping travel planning steadily over the past few years — not in a single breakthrough, but through a series of incremental systems, each solving one piece while leaving others untouched. What follows is an honest look at where that work stands.

Honmane et al. (2025) combined ML, NLP, and real-time analytics into a single planning system with live route updates and booking hooks built in. User satisfaction scores improved noticeably. But three problems remained open at the end of the paper: data accuracy under load, API dependency chains, and what happens when the system tries to scale beyond a few hundred concurrent users. None were resolved.

Maheen et al. (2025) approached the problem differently — rather than planning, they focused on recommendation quality using collaborative and content-based filtering trained on historical user behavior. Accuracy improved meaningfully, but the system required substantial data volume to function well. Cold-start users with no history got generic results. The paper acknowledges this; it doesn't resolve it.

Londhe et al. (2024) built a conversational travel planner around ChatGPT, letting users describe preferences in plain language and receiving itineraries in return. Web scraping fed the recommendation engine with live data, and the system hit roughly 86% precision in personalization tasks. That's a solid number — but the architecture's dependence on external scraping means any upstream change can quietly break things. Stability was never fully addressed.

Manideep et al. (2024) took a similar conversational approach but leaned harder into generative AI for behavior analysis and route suggestions. The interaction felt more fluid than form-based alternatives. The cost was scalability: once datasets grew or real-time conditions changed, the system slowed. It handled the average case well and struggled at the edges.

The TravelAgent system (2024) is the most architecturally ambitious work in this space. It uses large language models with dedicated modules for memory, planning, recommendations, and tool use — essentially a multi-agent framework for handling complex trip constraints like overlapping budgets and time windows. It works impressively in controlled conditions. Real-world dynamic environments exposed computational limits the architecture wasn't built to handle.

Perhaps the most sobering finding in recent literature comes from the TravelPlanner benchmark (2024), which tested LLM-based agents — including GPT-4 — against genuine multi-constraint travel scenarios. The success rate across advanced models was 0.6%. Not 60%. Not 6%. That number is worth sitting with. It suggests that despite how capable modern language models appear in demos, structured planning under real constraints is still largely unsolved.

TravelSage (2024) brought chatbot-style interaction to itinerary generation with real-time booking API integration. Functionally it covered a lot of ground. The paper itself was thin on implementation detail and offered little in the way of concrete evaluation metrics, making it hard to assess how well it actually performed outside the authors' own testing.

Broader review studies from the same period point to three trends worth tracking: generative AI is becoming the default planning backbone, real-time API integration is now expected rather than novel, and collaborative planning — systems that adjust based on group preferences — is gaining traction. The persistent blockers remain the same: data privacy, integration complexity, and the difficulty of scaling personalized systems without degrading response quality.

Earlier platforms from 2023 tried to bundle route optimization, hotel search, and booking into unified tools. They worked for standard itineraries. Anything requiring adaptive decision-making — a delayed flight, a sold-out hotel, a budget overrun — exposed how rigid the underlying logic was. Personalization was shallow by design.

Work from that same year on generative AI specifically showed clear gains in itinerary optimization and cost efficiency. The concerns that came with it — algorithmic bias, opaque decision-making, data handling — were noted in paper after paper and addressed in none of them. They remain open problems now.

III. EXISTING MODELS

Travel planning software has gone through several phases, and understanding each one explains why a unified system is still needed.

A. Rule-Based Systems

Early tools ran on fixed logic — predefined rules mapped inputs to outputs. Destination search, hotel lookup, basic scheduling all worked through static decision trees. Predictable and easy to debug, but completely rigid. Users with preferences that didn't fit the expected pattern got results that technically answered the query but missed the point. No mechanism existed to recognize this or adapt.

B. Data-Driven Recommendation

Data-driven approaches improved things by learning from actual user choices rather than fixed rules. Londhe et al. (2024) built a conversational system using content-based filtering — extracting preferences like budget and travel interests to rank options. Accuracy improved over rule-based predecessors, but the system lived and died by its data. Sparse records or inconsistent inputs meant proportionally worse output, with no fallback.

C. Machine Learning Models

The next step was inferring preferences rather than asking for them. Maheen et al. (2025) used collaborative filtering and predictive analytics trained on behavioral data — search patterns, booking sequences, past interactions. It worked well for users with history. For new users, it fell back to generic suggestions no better than the systems it replaced. Cold-start was never solved; it was inherited.

D. Generative AI and NLP

More recent systems dropped structured inputs entirely. Manideep et al. (2024) built a generative AI planner that parsed plain-language queries and produced dynamic itineraries mid-conversation. More natural to use, but two problems appeared consistently: real-time adaptability broke down when conditions changed, and computational cost was rarely accounted for at scale.

E. LLM-Based Agents

The most advanced current approach treats planning as multi-step reasoning. TravelAgent (2024) split responsibilities across dedicated modules for memory, recommendation, planning, and tool use. It handled genuinely complex constraints — overlapping budgets, group preferences, timing conflicts — better than anything before it. Real-world dynamic conditions exposed coordination overhead the architecture wasn't built to absorb. TravelSage (2024) pursued similar ground with tighter API integration; evaluation metrics were too sparse to assess it fairly.

F. What Benchmarks Reveal

The TravelPlanner benchmark (2024) tested leading LLM agents including GPT-4 on real multi-constraint scenarios. Success rate: 0.6%. Not a narrow task definition. A genuine measure of how often these systems produced a valid, executable plan when complexity was realistic. The gap between demo performance and real-world delivery is larger than most papers in this space acknowledge.

IV. PROPOSED METHODOLOGY

The architecture is built around a single principle: the user should never hit a dead end. Every step in the flow has a clear next step, and failures at the API or data layer should degrade gracefully rather than crash the experience.

A. User Interaction and Input Validation

Users access the system through a browser, log in via Firebase Authentication, and enter their trip parameters — destination, travel dates, and budget.

Before anything is sent to the backend, the frontend validates the input: required fields, date logic, budget range. Most errors get caught here, before they waste a server round-trip.

B. Backend Processing and Planning Logic

The Node.js backend receives validated input and coordinates the rest. Planning logic runs as a combination of rule-based filters — eliminating options that don't fit the budget or schedule — and a ranking layer that orders remaining options by likely relevance. The ranking draws on dataset patterns rather than per-user history, which sidesteps the cold-start problem at the cost of some personalization depth.

C. Data Retrieval and Itinerary Assembly

Travel data comes from two sources: local JSON datasets for city and route information, and external APIs for live flight and hotel availability. The local datasets handle the bulk of structured queries quickly; the API calls run in parallel to minimize wait time. Once both data streams are back, the backend assembles a structured itinerary with destinations, options, timing, and pricing organized into a readable format.

D. Storage and Display

User profiles and booking records are stored in Firebase Firestore. The frontend renders the final itinerary and presents payment options when the user is ready to book.

E. System Architecture

The system architecture of TravelEase represents how different components such as the frontend, backend, database, and external APIs interact with each other. The user interacts with the frontend interface, which sends requests to the backend server. The backend processes the data, applies travel planning logic, and communicates with JSON datasets and external APIs to fetch relevant travel information.

Firebase is used for authentication and data storage. The processed results are then sent back to the frontend and displayed to the user.

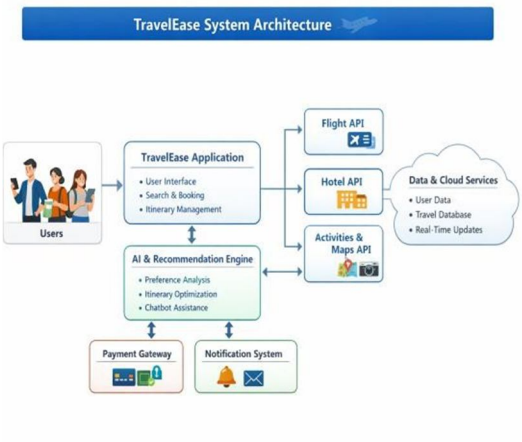


Fig:4.1 System Architecture

V. EVALUATION METRCS

The system was evaluated across six dimensions, chosen to reflect both technical performance and real user experience:

1) Response Time: Measures how long it takes from submitting trip details to seeing a complete itinerary. We tracked this across different destination types and API load conditions. The target was under three seconds for common routes; edge cases with slow API responses were logged separately.

2) Recommendation Accuracy: Asks whether the suggestions actually fit what the user asked for. We cross-checked generated itineraries against the input parameters — budget adherence, date fit, destination relevance — and flagged cases where any of the three were violated.

3) System Efficiency: Covers backend throughput and API coordination. This is mostly invisible to users until it isn't — slow retrieval or failed API calls surface immediately in response time and recommendation quality.

4) Usability (User Experience): Was assessed through task-based testing: could a new user navigate from the homepage to a completed booking without assistance? Where did people hesitate or get confused?

5) Data Retrieval Performance: Measures how accurately and quickly the system pulls flight, hotel, and route data. Accuracy matters here as much as speed — a fast but wrong result is worse than a slightly slower correct one.

6) Reliability: Tracks uptime and error rates under normal and stressed conditions. A planning system that crashes mid-booking is worse than no system at all.

VI. RESULTS

A. Home Page

The landing page gives users two paths immediately: Plan with AI or Book Manually. The navigation bar across the top— Home, Flights, Hotels, My Bookings, Smart Trip, Analytics — keeps the full system accessible at all times. The central headline, "Discover India Your Way," frames the application's scope without overcomplicating the entry point.

- Plan with AI – This option helps users create travel plans automatically based on their preferences.
- Book Manually – This option helps users search and book travel manually.

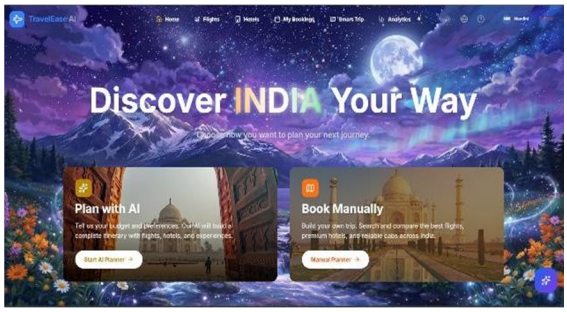


Fig 6.1 Home Page Mode

B. Flights Search and Booking page

Users enter origin, destination, date, and traveler count. Results appear in the central panel with airline, departure and arrival times, duration, and price. Left-side filters let users narrow by stops or carrier. The "Book Now" button on each result takes users directly into the booking flow without losing their search context.

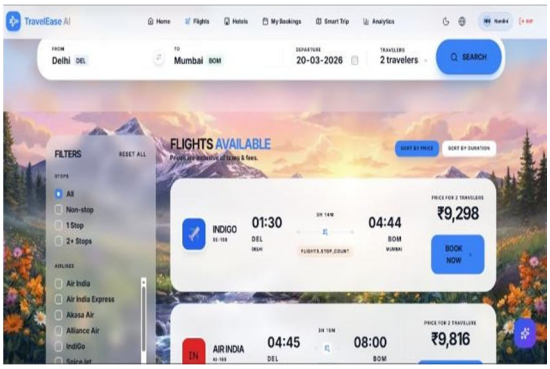


Fig 6.2 Flight Search Booking Mode

C. Payment Interface

Three payment methods are available: Card, UPI, and Net Banking. The UPI flow lets users type their ID manually or tap into GPay, PhonePe, Paytm, or BHIM. A save-for-later toggle reduces friction on repeat bookings. The booking summary on the right — route, passengers, itemized fare, taxes, total — stays visible throughout the payment process so users aren't confirming a number they can't verify.

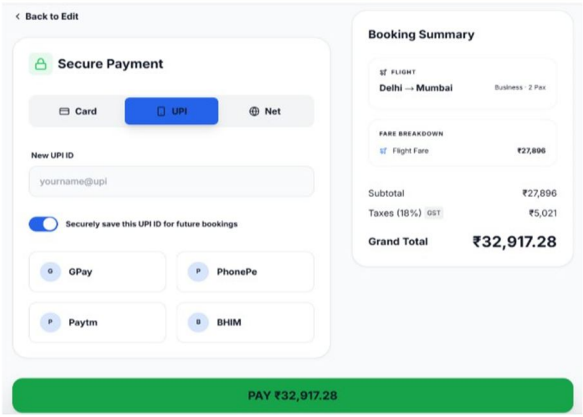


Fig 6.3 Payment Interface

VII. CONCLUSION

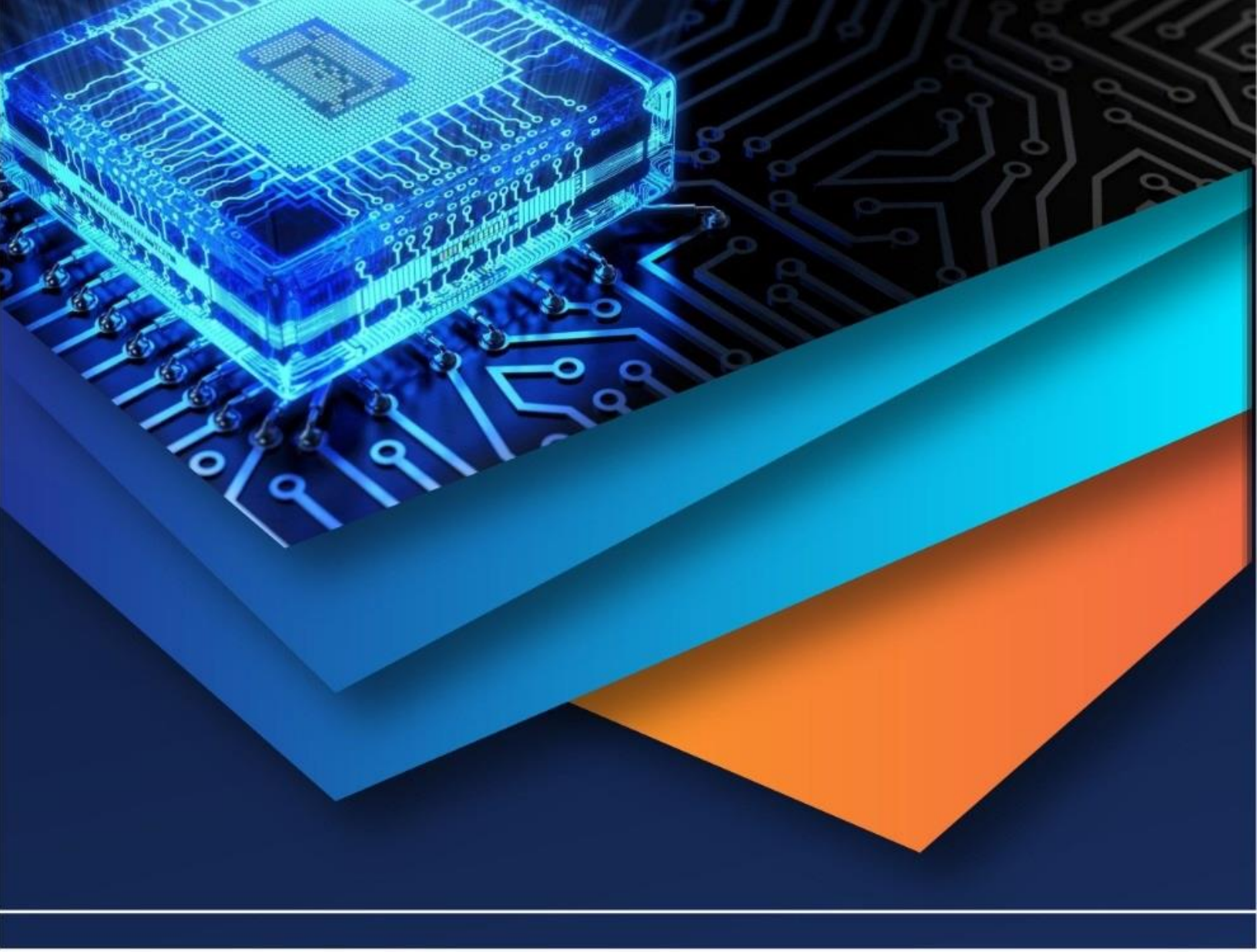
The results obtained from the Travel Ease system have shown that the system has successfully accomplished its purpose of providing a smart and efficient system for travel planning. The interfaces developed in the system, including the home page interface, flight search interface, and payment interface, have shown that the system can successfully process the user's queries and provide relevant results for the user. The system has also been able to provide a user-friendly and interactive interface that helps users navigate through the features and facilities of the system. The system has also been able to incorporate backend processing and API-based information retrieval to ensure that the user receives relevant and organized information for their travel plans.

REFERENCES

[1] Pushpakumara, T. D. C., & Ahsan, F. J. (2025). The evolution of AI chatbots in sustainable tourism: A systematic literature review. International Journal on Cybernetics & Informatics (IJCI), 14(3).

[2] Udandarao, V., Tiju, N. A., Vairamuthu, M., Mistry, H., & Kumar, D. (2025). Ramify: Designing and Evaluating an LLM Based Chrome Extension for Personalised Itinerary Planning. arrive preprint.

[3] Shih, Y.-K., & Kang, Y.-K. (2025). Design and Implementation of a Secure RAG-Enhanced AI Chatbot for Smart Tourism Customer Service: Defending Against Prompt Injection Attacks – A Case Study of Hsinchu, Taiwan. arrive preprint.
[4] K. T. Mohite, M. Borse, G. Gadekar, S. Hon, & S. Jagtap. (2024). Travel and Tourism Management System Using Chatbot Recommender. IJRASET
[5] Benaddi, L., Ouida, C., Souha, A., Jaki mi, A., Ragout, M., Aledhari, M., Oliveira, D., & Ochoa, B. (2024). Seq2Seq Model-Based Chatbot with LSTM and Attention Mechanism for Enhanced User Interaction. arrive preprint.
[6] Cassani, A., Ruberg, M., Salis, A., Goanese, G., & Benelli, G. (2024). Zia: a GenAI-powered local auntie assists tourists in Italy. arrive preprint.
[7] Exploring agent-based chatbots: A systematic literature review. Journal of Ambient Intelligence and Humanized Computing, 14, 11207–11226 (2023).
[8] Travel and Tourism Management System Using Chatbot Recommender. IJSREM, Version Feb 11, 2023.
[9] Fajar, A. N., & Baiza, A. (2023). Chatbot-based Culinary Tourism Recommender System Using Named Entity Recognition. JIPI, 7(4).
[10] K. T. Mohite, M. Borse, G. Gadekar, S. Hon, & S. Jagtap. (2024). Travel and Tourism Management System Using Chatbot Recommender.
[11] Kulkarni, N. K., & Marathe, N. (2022). Tour Planning Chatbot for Tourism and Travel Industry. International Journal of Engineering Research & Technology (IJERT), 11(05).
[12] Jannah, D., Manzoor, A., Cai, W., & Chen, L. (2020). A Survey on Conversational Recommender Systems. Deepa.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY