Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602

TRAVEL RECOMMENDATION SYSTEM

D. V. Kartheek[1], Mrs. P. Nancy Anurag[2]
[1]Student, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India
Email id: kartheeknaani007@gmail.com, nancy.anurag@aliet.ac.in

Abstract: Travel planning today is exciting but time-consuming. Searching through multiple websites, comparing hotels, estimating budgets, and building a day-by-day schedule for an unfamiliar destination takes hours of effort — and the result often feels generic and impersonal. The Travel Recommendation System addresses this by providing a full-stack, AI-driven web application that automates and personalizes the entire travel planning process. The system accepts user inputs such as origin city, destination city, travel dates, budget, number of travelers, and personal interests. Using these inputs it calls the Groq AI API powered by the LLaMA 3.1 8B Instant model to generate detailed day-by-day travel itineraries customized for each individual user. Every itinerary includes morning, afternoon, and evening activity suggestions with precise GPS coordinates, restaurant recommendations for lunch and dinner, hotel suggestions with star ratings and pricing, a complete budget breakdown across accommodation, food, activities, transport, and miscellaneous expenses, transport options including flights, trains, and buses, and practical travel tips specific to the destination. The frontend is built with Next.js 15 and Tailwind CSS, delivering a responsive multi-step trip planning wizard. Interactive maps are powered by Leaflet.js with OpenStreetMap tiles and Nominatim geocoding. Destination photographs are sourced from the Unsplash API. The backend is built with Node.js, Express.js, and MongoDB with JWT-based authentication ensuring secure, user-scoped access to all saved trips.

Keywords:
Travel Recommendation System, Artificial Intelligence, Large Language Model, LLaMA 3.1, Groq API, Itinerary Generation, Personalized Travel Planning, Next.js, React, Tailwind CSS, Node.js, Express.js, MongoDB, JWT Authentication, Leaflet.js, OpenStreetMap, Unsplash API, REST API, Full-Stack Web Application, Natural Language Processing.

1. INTRODUCTION

Travel planning in the modern world involves navigating a massive volume of information from flight aggregators, hotel booking platforms, travel blogs, review sites, and social media. A typical traveler must manually research attractions, cross-reference budget estimates, check operating hours, plan transport logistics, and design a day-by-day schedule — a process that can take hours or even days.

Existing travel tools such as Google Travel, TripAdvisor, and Sygic Travel are primarily database-driven systems that present static, pre-catalogued information. They lack the ability to dynamically synthesize a coherent, personalized itinerary that simultaneously accounts for the traveler's interests, duration of stay, budget, and travel style.

The Travel Recommendation System was developed to solve this problem. It uses the LLaMA 3.1 8B Instant large language model, accessed through the Groq API, to instantly generate a complete, structured, GPS-accurate travel plan for any destination in the world. The system integrates AI-generated itineraries, interactive maps, real destination photographs, hotel and restaurant recommendations, and budget breakdowns into a single, secure, user-friendly web platform.

https://doi.org/10.5281/zenodo.19605100

The backend is built with Node.js and Express.js following REST API principles, with MongoDB for persistent data storage. The frontend is developed with Next.js 15 and styled with Tailwind CSS. JWT-based authentication ensures each user's trips are securely stored and privately accessible.

2. Literature Survey

Recent A review of existing literature and systems in the domain of AI-assisted travel planning reveals the following key trends and gaps.

Early recommendation systems used collaborative filtering and content-based filtering to suggest destinations based on past user behavior. However, these approaches depend heavily on historical data and fail to adapt to novel user requirements or new destinations.

More recent research has explored the use of Natural Language Processing and transformer-based models such as BERT and GPT for generating travel narratives. Studies have shown that large language models can produce contextually appropriate travel plans when given structured prompts containing destination, duration, and preference information.

Research on map integration has demonstrated that embedding interactive maps within itinerary displays significantly improves user orientation and engagement. Open-source solutions such as Leaflet.js combined with OpenStreetMap provide a cost-effective alternative to commercial mapping APIs.

Based on the literature review the following gaps exist in current systems:

- Most itinerary tools are database-driven, not AI-generative.
- No widely available free system combines AI generation, map integration, and image fetching in a single platform.
- Existing AI travel tools do not consistently produce structured data such as GPS coordinates and budget breakdowns suitable for rendering in a rich UI.
- No single tool generates a complete plan covering transport, hotels, restaurants, and activities in one step.

The Travel Recommendation System addresses all of these identified gaps.

3. Proposed System

The Travel Recommendation System is a unified AI-driven web platform that automatically generates complete, personalized travel itineraries for any destination worldwide. Unlike existing tools it combines all aspects of travel planning into a single application, eliminating the need to visit multiple websites.

https://doi.org/10.5281/zenodo.19605100

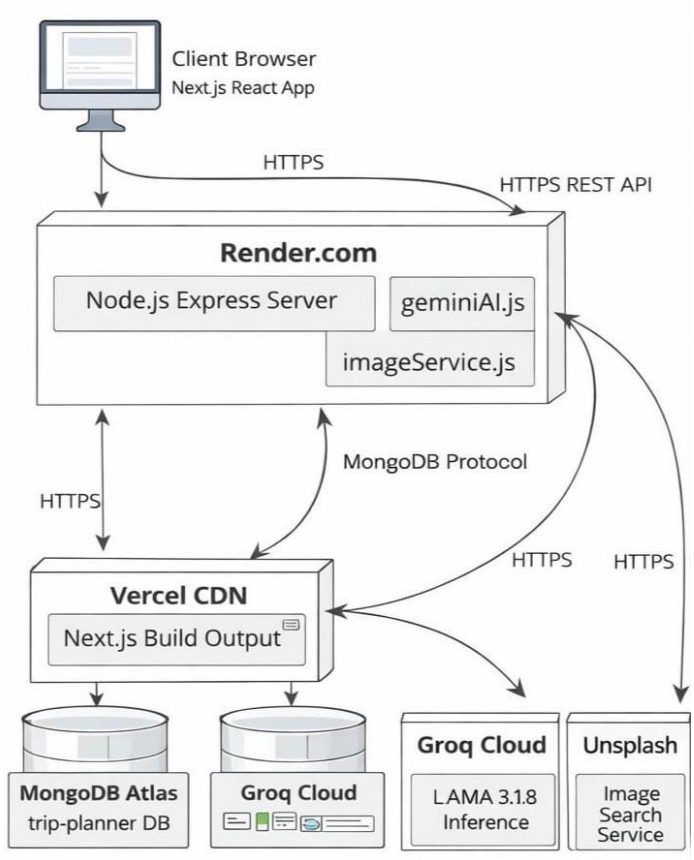


Fig 1: Proposed System Architecture

The key features of the proposed system are:

- AI Generation: Uses LLaMA 3.1 via Groq API to generate novel, personalized itineraries on demand.
- Structured Output: Returns valid JSON with GPS coordinates, cost breakdowns, restaurant names, and hotel details.
- Interactive Maps: Leaflet.js renders all attraction coordinates on OpenStreetMap.
- Destination Photos: Unsplash API provides real photographs for each attraction and hotel.
- User Accounts: JWT-secured MongoDB accounts allow saving and managing multiple trips.
- Budget Transparency: Complete cost breakdown across all five expense categories.
- Free Stack: Built entirely on free-tier services making it accessible without subscription costs.
- Fast Generation: Complete itineraries produced within 10 to 20 seconds.

4. Methodology

The methodology of the Travel Recommendation System is organized into the following steps:

https://doi.org/10.5281/zenodo.19605100

1. User Registration and Login: The user creates a free account by entering their name, email, and password along with optional travel interest preferences. JWT authentication is used to secure all subsequent interactions.

2. Trip Parameter Input: The user completes a three-step planning wizard entering origin city, destination city, travel dates, number of days, personal interests, total budget, currency, and number of travelers.

3. AI Prompt Construction: The system builds a detailed structured prompt containing all trip parameters and sends it to the Groq API. The prompt instructs the LLaMA 3.1 model to return only valid JSON conforming to a precise schema covering itinerary, hotels, budget, route, and travel tips.

4. Itinerary Generation: The Groq API processes the prompt and returns a complete structured JSON itinerary within 10 to 20 seconds. The response is cleaned and parsed before further processing.

5. Image Fetching: The imageService.js module fetches relevant destination photographs from the Unsplash API for each attraction, hotel, and trip cover image. An in-memory cache prevents duplicate API calls.

6. Data Storage: The complete trip document including the nested itinerary, hotel recommendations, route details, and image URLs is saved to MongoDB Atlas through Mongoose ODM under the authenticated user's account.

7. Result Display: The user is redirected to the trip detail page where the full itinerary is displayed. All attraction GPS coordinates are plotted on an interactive Leaflet.js map. Hotel recommendations, budget breakdown, and transport options are presented in organized sections

5. Proposed System Software Implementation & Results

The system implementation covers both frontend and backend components developed using modern web technologies.

Frontend Implementation (Next.js)

1. Next.js App Router provides file-based routing where each folder inside /src/app represents a route. Pages include Home, Login, Register, Plan Trip, Dashboard, My Trips, and Trip Detail.
2. React useState and useEffect Hooks manage local component state such as form inputs, loading status, error messages, and fetched data.
3. AuthContext using React Context API manages global authentication state including the JWT token and logged-in user object, making them available across all pages.
4. Axios HTTP Client with a request interceptor automatically attaches the JWT Bearer token to every outgoing API request.
5. Three-Step Planning Wizard collects trip parameters across three sequential steps with client-side validation before submission.
6. Leaflet.js Map Component renders all attraction GPS coordinates as interactive markers on OpenStreetMap. Clicking a marker shows a popup with attraction details.
7. Tailwind CSS utility classes provide responsive, mobile-first styling across all pages without writing separate CSS files.

https://doi.org/10.5281/zenodo.19605100

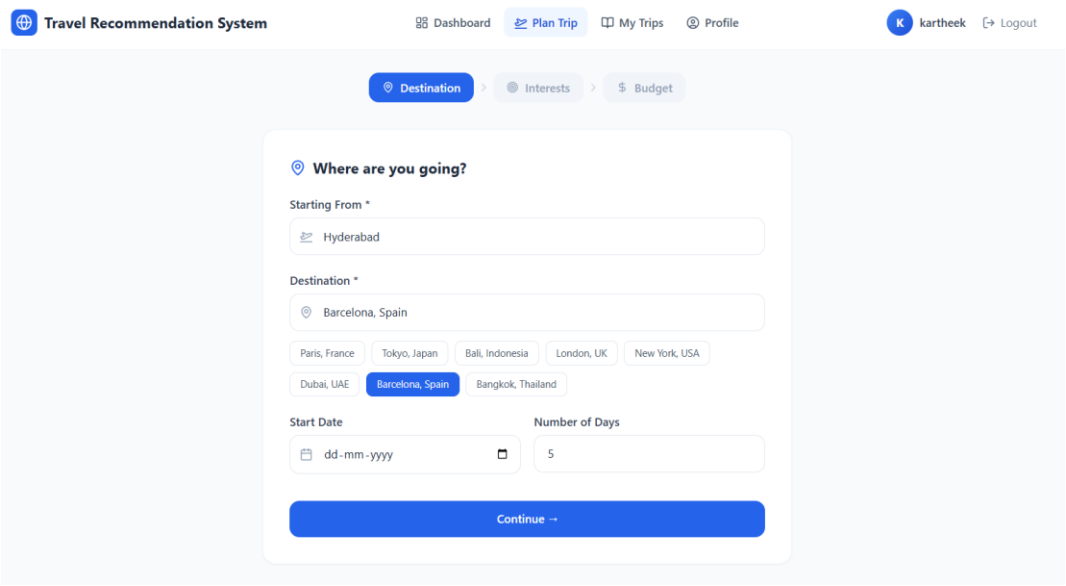
8. Protected Routes redirect unauthenticated users to the login page by checking the AuthContext user object inside useEffect on page load.
9. Loading States use skeleton cards and spinner components to prevent layout shift during data fetching operations.
10. Unsplash Images are displayed as cover images on trip cards and detail pages with object-cover styling and gradient overlays.

Backend Implementation (Node.js / Express)

11. Express.js Server initializes CORS middleware, JSON body parsing, and rate limiting before mounting route handlers for auth, trips, users, and images.
12. JWT Authentication middleware verifies the Bearer token on every protected route, extracts the user ID, and attaches the full user object to the request.
13. bcryptjs Password Hashing secures user passwords with a salt factor of 10 before storing them in MongoDB.
14. Groq SDK Integration in geminiAI.js sends a structured prompt to LLaMA 3.1 8B Instant and parses the returned JSON itinerary.
15. Rate Limiting applies a general limit of 100 requests per 15 minutes on all API routes and a stricter limit of 5 requests per minute on the AI generation endpoint.
16. Mongoose ODM defines schemas for User and Trip models and handles all database operations including creation, retrieval, update, and deletion.
17. Image Caching in imageService.js uses a JavaScript Map to avoid duplicate Unsplash API calls for the same destination query within a session.
18. CORS Middleware allows the frontend hosted on Vercel to communicate securely with the backend hosted on Render.com.

Results

The system was tested across multiple destinations and trip durations. The following results were observed:



https://doi.org/10.5281/zenodo.19605100

Fig 2. Trip Planning Wizard — Step 1 (Destination Input).

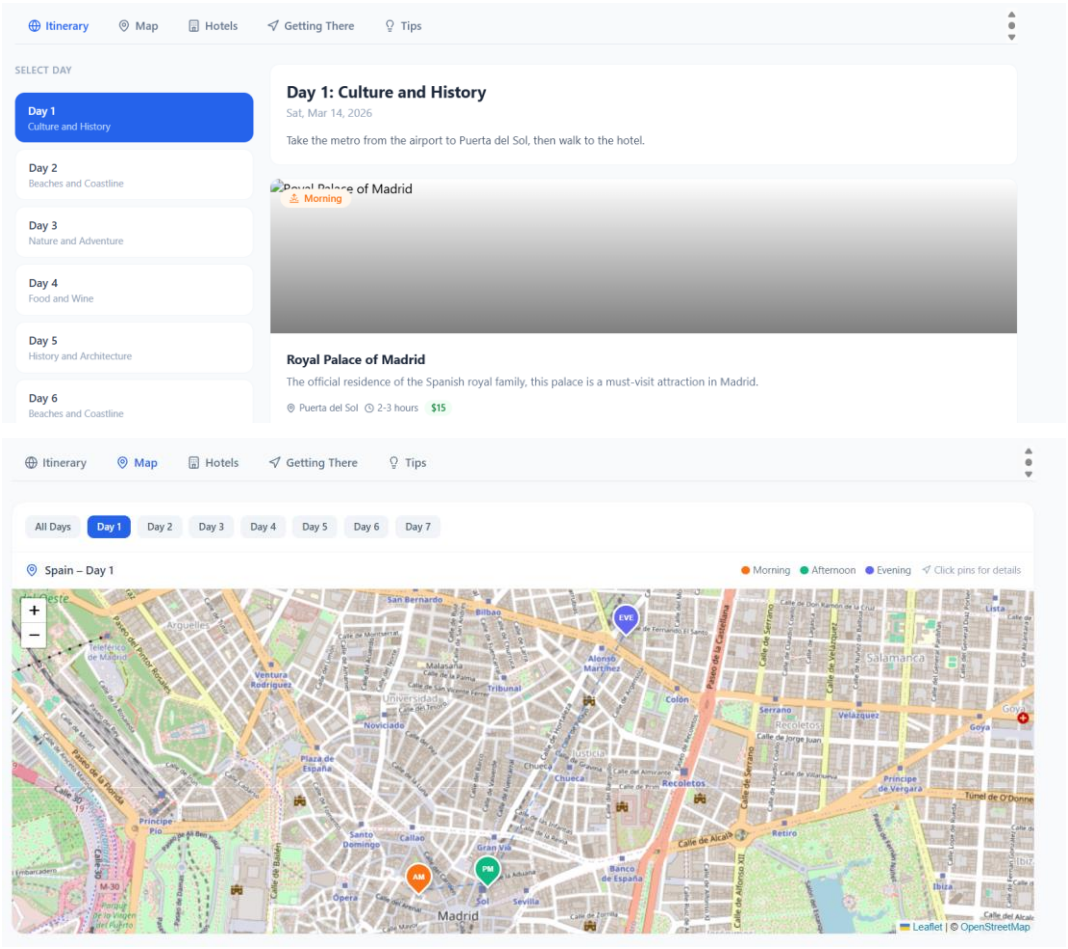


Fig 3 : AI-Generated Itinerary with Interactive Map

Overall the system demonstrates reliable performance in generating personalized itineraries, displaying interactive maps, and presenting structured budget and hotel information within an average response time of 15 seconds.

6.CONCLUSION

In this project we have successfully developed a Travel Recommendation System using artificial intelligence and modern full-stack web technologies. The main goal of the system is to automatically generate a complete, personalized, GPS-accurate travel itinerary for any destination worldwide — eliminating hours of manual research for the traveler.

The system processes user requirements through the following pipeline:

- Parameter collection via a multi-step planning wizard
- AI itinerary generation using LLaMA 3.1 via Groq API
- Destination image fetching from Unsplash API
- Persistent storage in MongoDB Atlas

https://doi.org/10.5281/zenodo.19605100

- Interactive display with Leaflet.js maps

The core of the Travel Recommendation System uses the LLaMA 3.1 8B Instant large language model accessed through the Groq API. This model is prompted with a structured schema that ensures the returned itinerary contains valid GPS coordinates, realistic cost estimates, curated restaurant and hotel recommendations, and practical travel tips. The JSON response is parsed and enriched with images before being saved and displayed.

From an implementation standpoint the system uses a three-tier architecture. The frontend is built with Next.js 15 and deployed on Vercel. The backend is built with Node.js and Express.js and deployed on Render.com. MongoDB Atlas serves as the cloud database. JWT authentication ensures that every user's saved trips are private and secure.

Overall this project demonstrates the practical application of large language models in solving a real-world problem. The Travel Recommendation System helps users plan better trips faster, reduces the burden of cross-platform research, and delivers a personalized travel experience through intelligent automation. In the future the system can be improved by adding real-time booking integration, collaborative trip planning, mobile application support, multi-language itineraries, and parallel image fetching to further reduce generation time.

7.REFERENCES

1. Next.js Documentation. (2024). App Router. Vercel Inc. https://nextjs.org/docs
2. Express.js Documentation. (2024). Express 4.x API Reference. OpenJS Foundation. https://expressjs.com
3. MongoDB Documentation. (2024). MongoDB Manual. MongoDB Inc. https://www.mongodb.com/docs
4. Groq Documentation. (2024). Groq API Reference. Groq Inc. https://console.groq.com/docs
5. Meta AI. (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.
6. Unsplash API Documentation. (2024). Unsplash Developer Reference. https://unsplash.com/documentation
7. Leaflet.js Documentation. (2024). Leaflet 1.9 Reference. https://leafletjs.com/reference.html
8. Vaswani, A., et al. (2017). Attention Is All You Need. NeurIPS 30.
9. JWT.io. (2024). JSON Web Tokens Introduction. Auth0. https://jwt.io/introduction
10. Tailwind CSS Documentation. (2024). Tailwind CSS v3. https://tailwindcss.com/docs.

https://doi.org/10.5281/zenodo.19605100