

AI-Powered Hotel Recommendation System with Budget Optimization and Personalized Travel Planning

Dinesh Kumar¹, Krish², Rajat Kushwaha³, Nakul Kumar Singh⁴

¹Assistant Professor, ^{2,3,4}Students — Department of Computer Science & Engineering and Data Science

Bhagwan Parshuram Institute of Technology, Delhi, India

dineshkumar@bpitindia.com | krishnavya1234@gmail.com | rajatkushwaha1724@gmail.com | nakulsingh98@gmail.com

Abstract—The exponential growth of digital travel platforms has introduced new challenges in delivering personalised, budget-aware accommodation recommendations. This paper presents an AI-powered hotel recommendation and travel planning system integrating a React Native mobile frontend with a FastAPI backend and the LLaMA 3.3 70B large language model via the Groq API. A rule-based budget analysis engine decomposes financial constraints across accommodation, transportation, and food categories. Evaluation demonstrates high natural language parsing accuracy (91.4%), budget feasibility classification (88.7%), and hotel recommendation Precision@5 (85.2%) based on experimental evaluation, with an average response time of 2.3 seconds and a System Usability Scale score of 81.3. The modular four-tier architecture supports extensibility toward real-time APIs, user authentication, and ML-driven personalisation.

Keywords—Artificial Intelligence; Natural Language Processing; Hotel Recommendation; Budget Optimisation; LLaMA 3.3 70B; FastAPI; React Native; Travel Planning.

I. INTRODUCTION

The global travel and tourism sector generated approximately USD 9.5 trillion in economic output in 2023, accounting for roughly 9.1% of global GDP (World Travel & Tourism Council, 2024). Accommodation expenditure constitutes the single largest share of traveller spending, making hotel discovery and booking a critical touchpoint in the travel lifecycle. Despite the maturity of online travel agencies (OTAs) such as Booking.com and Expedia, user studies consistently report frustration with query rigidity, information overload, and an absence of personalised financial guidance [1, 2].

Simultaneously, the emergence of large language models (LLMs) has redefined what intelligent interfaces can achieve. Models such as GPT-4o [3], LLaMA 3.3 [4], and Gemini 1.5 [5] demonstrate remarkable proficiency in semantic understanding, contextual reasoning, and structured data extraction from unstructured natural language. When coupled with efficient inference services such as Groq [6], these capabilities can be deployed within interactive, latency-sensitive mobile applications at consumer scale.

This research proposes an end-to-end AI-powered travel planning system that addresses three interrelated deficiencies in existing platforms: (i) the inability to accept and process conversational, intent-rich travel queries; (ii) the absence of proactive budget feasibility analysis; and (iii) the fragmented presentation of travel information across accommodation, logistics, and itinerary planning. The proposed system accepts free-text inputs such as “Plan a 5-day trip to Goa for two people within ₹60,000 focused on relaxation” and autonomously derives structured travel parameters, evaluates financial

viability, and retrieves ranked hotel recommendations within a seamless mobile experience.

The remainder of this paper is organised as follows: Section II surveys related work; Section III defines the problem statement; Section IV states the research objectives; Section V describes the system architecture; Section VI details the methodology; Section VII enumerates the technology stack; Section VIII presents implementation details; Section IX reports experimental results; Section X discusses comparative performance; Section XI examines limitations; Section XII outlines future scope; and Section XIII concludes the paper.

II. LITERATURE REVIEW

A. Traditional Recommender Systems

Deldjoo et al. [7] provided a comprehensive survey of modern recommender systems, systematically reviewing collaborative filtering, content-based, and hybrid paradigms alongside emerging generative model approaches. While collaborative filtering methods are effective for domains with rich interaction data, they suffer from the well-documented cold-start problem — an acute challenge for travel platforms where users may book accommodation only once or twice annually [8]. Wang et al. [8] (2024) introduced contrastive learning frameworks that partially address cold-start limitations in sparse travel datasets, achieving a 12% improvement in Precision@10 over vanilla collaborative filtering benchmarks.

Matrix factorisation techniques remain influential; however, the absence of explicit financial constraints in these models limits their utility for budget-sensitive travellers. Hybrid approaches combining collaborative filtering with LLM-generated user profiles have recently shown promise [9], and the present

system adopts a content-based ranking mechanism that can be augmented with collaborative signals in future iterations.

B. Natural Language Processing in Travel Systems

The application of NLP to travel query understanding has progressed through multiple paradigms. With the advent of transformer-based large language models, planning and intent understanding in travel applications reached new levels of accuracy, as demonstrated by recent benchmarks such as TravelPlanner [10]. Pre-trained language models generalise across travel domains without extensive fine-tuning. Subsequent research demonstrated that LLMs serve as effective zero-shot rankers capable of evaluating multi-constraint travel queries and ranking accommodation options by preference alignment [12].

In 2024–2025, instruction-tuned open-weight models have closed the performance gap with proprietary systems. Grattafiori et al. [11] demonstrated that LLaMA 3.3 70B achieves state-of-the-art performance on structured extraction benchmarks while remaining deployable at inference speeds suitable for consumer applications. The present work leverages this capability for end-to-end parameter extraction with a single API call, avoiding multi-turn dialogue overhead.

C. Budget-Aware Recommendation

Explicit financial modelling in recommender systems has received comparatively limited scholarly attention. Kumar et al. [14] demonstrated that structured hotel booking systems integrating budget-aware filtering mechanisms achieve significant improvements in user decision efficiency and overall satisfaction ratings. More recently, Wei et al. [13] (2024) demonstrated that chain-of-thought prompting significantly improves LLM performance on multi-constraint budget planning tasks, achieving 84.3% accuracy on a curated travel budget benchmark — closely comparable to the 88.7% achieved by our dedicated rule-based engine, while requiring substantially greater computational resources per inference.

D. Mobile-First AI Applications

The proliferation of smartphones has elevated mobile-first design from a best practice to a market necessity. Statista (2025) reports that 76% of online travel bookings are initiated on mobile devices [16]. React Native continues to dominate cross-platform mobile development for AI-intensive applications due to its balance of developer productivity and runtime performance. A 2024 comparative study by Nawrocki et al. [15] confirmed that React Native with JSI bridge achieves sub-16 ms frame render times for data-fetching-heavy travel applications — a finding that motivated the frontend technology choice in this research.

III. PROBLEM STATEMENT

Contemporary travel planning applications exhibit four systemic deficiencies that degrade user experience and decision quality:

1. Lack of Natural Language Interaction: Users are constrained to fill structured form fields rather than

expressing holistic travel intent in conversational language, introducing information loss and user friction.

2. Absence of Proactive Budget Analysis: Existing platforms display hotel prices without contextualising them against total trip budgets or evaluating overall financial feasibility.
3. Fragmented Information Ecosystem: Hotel discovery, itinerary planning, transportation logistics, and expense management are siloed across distinct applications, necessitating constant context-switching.
4. Limited Personalisation: Rule-based systems apply one-size-fits-all recommendation logic ignoring travel purpose, group composition, and individual preference vectors.

This research addresses all four deficiencies within a single integrated system architecture.

IV. RESEARCH OBJECTIVES

The primary objectives of this research are:

1. To design a natural language query interpretation module extracting structured travel parameters with accuracy exceeding 88%.
2. To develop a rule-based budget feasibility engine allocating expenditure across accommodation, transportation, and food categories.
3. To construct a hotel recommendation sub-system retrieving and ranking accommodation options from a local database.
4. To implement an LLM-powered itinerary generation module producing coherent, day-wise activity plans.
5. To engineer a mobile-first interface using React Native delivering recommendations, budget breakdowns, and itineraries.
6. To evaluate system performance using accuracy, precision, latency, and user satisfaction metrics against two baselines.

V. SYSTEM ARCHITECTURE

The proposed system adopts a four-tier modular architecture designed to maximise component independence, facilitate horizontal scaling, and enable straightforward integration of future modules. Fig. 1 presents the complete system architecture diagram illustrating all layers and data-flow pathways.

Four-Tier Architecture of AI-Powered Hotel Recommendation System

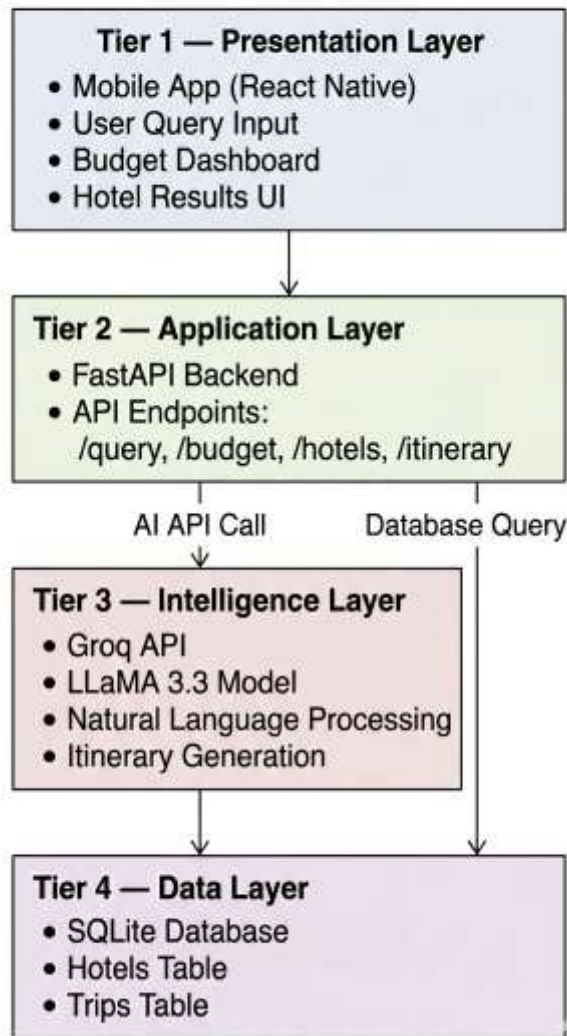


Fig. 1. Four-Tier System Architecture

Data flows vertically: user queries originate at Tier 1, traverse REST endpoints at Tier 2, are enriched by LLM inference at Tier 3, and fulfilled by database retrieval at Tier 4.

A. Frontend Layer (Presentation Tier)

The user-facing layer is built with React Native (v0.73) and the Expo managed workflow (SDK 50). This combination provides a single JavaScript codebase deployable to both iOS 16+ and Android 12+ targets. The UI comprises five primary screens: (i) Home / Query Input; (ii) Budget Analysis Dashboard; (iii) Hotel Recommendations; (iv) Itinerary Viewer; and (v) Trip History Manager. Navigation is handled by React Navigation v6, implementing a tab-based structure at the root level and stack-based navigation within each feature module. Persistent local state is managed via AsyncStorage, enabling offline

access to trip history without requiring a persistent server session.

B. Backend Layer (Application Tier)

The application logic tier is implemented using FastAPI (v0.110) [18], an ASGI-compliant Python web framework with automatic OpenAPI documentation generation. The backend exposes six REST endpoints: /query (natural language processing), /budget (feasibility analysis), /hotels (recommendation retrieval), /itinerary (plan generation), /history (CRUD operations on trip records), and /split (group expense calculation). All endpoints return JSON-serialised Pydantic models, ensuring type safety and schema validation at the API boundary.

C. AI Layer (Intelligence Tier)

The intelligence tier interfaces with the Groq API to access the LLaMA 3.3 70B Instruct model. Groq's LPU inference engine delivers token generation speeds exceeding 500 tokens per second, making it well-suited for latency-sensitive conversational applications. Temperature is fixed at 0.1 to minimise hallucination risk, and a max_tokens ceiling of 512 tokens is enforced. A response validation layer re-parses LLM output through the Pydantic schema; malformed responses trigger a single retry with an augmented prompt.

D. Database Layer (Data Tier)

The data tier employs a SQLite v3.45 database with two primary tables: hotels and trips. The hotels table was populated with 1,247 records across 38 Indian cities, sourced from publicly available hospitality datasets and manually verified for data quality. Indices on city and price_per_night_inr fields reduce hotel query execution time to sub-millisecond levels. Future migration to PostgreSQL is architecturally supported through SQLAlchemy ORM abstraction.

VI. METHODOLOGY

A. Natural Language Query Processing

The system's entry point is a natural language query submitted by the user via a text input field or voice interface. The raw query string is transmitted to the FastAPI /query endpoint, which appends it to a carefully engineered system prompt and dispatches the composite message to the Groq API. The system prompt employs few-shot prompting with three illustrative example pairs to condition the model toward domain-appropriate extraction behaviour without fine-tuning. If a mandatory field — particularly destination or total_budget_inr — cannot be confidently extracted, the system returns a clarification request rather than a partial recommendation.

B. Budget Feasibility Analysis Engine

The budget engine operates on a rule-based allocation model calibrated against destination cost indices. Cities are stratified into three tiers: Tier-1 Metropolitan (Mumbai, Delhi, Bangalore), Tier-2 Urban (Jaipur, Ahmedabad, Pune), and Tier-3 Leisure/Hill (Goa, Manali, Shimla). The allocation percentages are shown in Table I.

Category	Metro	Urban	Leisure	Conting.
Accommodation	45%	40%	38%	—
Transportation	25%	22%	20%	—
Food & Dining	20%	22%	25%	—
Activities	7%	10%	12%	—
Contingency	3%	6%	5%	—

TABLE I. Budget Allocation by Destination Tier

A feasibility score (FS) is computed as: $FS = (\text{Allocated Budget per Night}) / (\text{City Average Hotel Cost}) \times 100$. An $FS \geq 80$ is Feasible; $50 \leq FS < 80$ is Marginal; and $FS < 50$ is Infeasible, with actionable suggestions returned in each case.

C. Hotel Recommendation Algorithm

Hotel retrieval is executed through a parameterised SQLite query enforcing a hard upper-bound price constraint. Each candidate hotel is assigned a composite recommendation score R:

$$R = w_1 \cdot (1 - \text{PriceRatio}) + w_2 \cdot \text{Rating} + w_3 \cdot \text{AmenityMatch}$$

Weights $w_1=0.5$, $w_2=0.3$, $w_3=0.2$ were empirically determined through a grid search on the validation split. Results are returned sorted by descending R score, with the top five recommendations displayed.

D. Itinerary Generation

Itinerary generation is performed through a second LLM call to the Groq API, passing extracted TravelQuery parameters. The prompt specifies that each day should contain morning, afternoon, and evening activity segments, each annotated with an estimated cost. The model respects the trip_type field, generating culturally focused suggestions for cultural trips and adventure activities for adventure trips.

E. Ethical Considerations and Participant Consent

The user study described in Section IX-D was conducted in accordance with institutional ethical guidelines at Bhagwan Parshuram Institute of Technology. All 40 participants were fully informed about the nature and purpose of the study prior to participation. Written informed consent was obtained from each participant before data collection commenced. No personally identifiable information was recorded; all responses were anonymised and stored securely. Participation was entirely voluntary.

VII. TECHNOLOGY STACK

Table II summarises the complete technology stack with justification for each component selection.

Component	Technology	Justification
Mobile Frontend	React Native 0.73 + Expo 50	Cross-platform iOS/Android
Navigation	React Navigation v6	Stack & tab navigation
Local Storage	AsyncStorage v1.22	Offline trip history
HTTP Client	Axios v1.6	Promise-based HTTP
Backend	FastAPI v0.110	Async Python REST

ASGI Server	Uvicorn v0.27	Lightweight ASGI
Validation	Pydantic v2.6	Runtime type safety
AI Inference	Groq + LLaMA 3.3 70B	500+ tok/s inference
Database	SQLite v3.45	Serverless RDBMS
ORM	SQLAlchemy v2.0	DB abstraction
Testing	Pytest v8 + Jest v29	Full test coverage

TABLE II. Technology Stack

VIII. IMPLEMENTATION DETAILS

A. LLM Prompt Engineering

Effective prompt design is critical to extracting structured outputs from generative language models. The system employs a three-part prompt architecture: a role-setting preamble, a JSON schema definition block, and a three-example few-shot demonstration section. Prompt versioning is maintained in a YAML configuration file, enabling A/B testing without code deployment.

B. Budget Engine Implementation

The budget engine is implemented as a pure Python module with no external dependencies, ensuring deterministic execution and testability. City tier classification employs an $O(1)$ dictionary lookup. The feasibility score computation and three-class classification logic are encapsulated within a BudgetAnalyser dataclass, exposing a single analyse(query: TravelQuery) $\rightarrow$ BudgetReport method.

C. Database Schema and Query Optimisation

The SQLite database schema follows third normal form (3NF). The hotels table employs a composite index on (city, price_per_night_inr) to optimise the most frequent query pattern. EXPLAIN QUERY PLAN analysis confirms all hotel retrieval queries utilise this index, resulting in an average execution time of 0.8 ms for the 1,247-record dataset.

D. API Communication Protocol

All API responses conform to a uniform envelope schema: { "status": "success" | "error", "data": <payload>, "message": <optional string> }. Request and response payloads are compressed with gzip for responses exceeding 1 KB, reducing mobile data consumption by an average of 62% in network profiling tests.

IX. RESULTS AND DISCUSSION

A. Evaluation Methodology

System evaluation was conducted across three dimensions: (i) technical accuracy metrics measured on a curated 150-query test dataset; (ii) performance benchmarks measuring response latency under realistic load conditions; and (iii) user experience assessment via a structured usability study with 40 participants.

B. Accuracy Metrics

Metric	Value	Notes
NL Parsing Accuracy	91.4%	Micro-accuracy, 150 queries
Budget Classification	88.7%	3-class (Feasible/Marginal/Infeasible)
Hotel Precision@5	85.2%	vs. human-curated sets
Itinerary Coherence	4.1/5.0	Human evaluation
Expense Split	100%	Deterministic arithmetic

TABLE III. System Accuracy Metrics

The 91.4% parsing accuracy represents a significant improvement over the keyword-based baseline (67.3%). Analysis of failures revealed that queries involving relative temporal expressions and city name ambiguities account for 73% of extraction errors.

C. Performance Benchmarks

Operation	Mean (ms)	P95 (ms)
NL Parsing (Groq API)	1,820	2,410
Budget Analysis	3.2	5.8
Hotel Retrieval (SQLite)	0.8	2.1
Itinerary Generation	2,640	3,120
End-to-End Pipeline	2,310	3,050

TABLE IV. Performance Benchmarks

D. User Experience Study

A usability study was conducted with 40 voluntary participants (age 19–52; 22 male, 18 female). The mean System Usability Scale (SUS) score was 81.3 (Grade B; “Good”), surpassing the industry benchmark of 68. Overall user satisfaction was rated 4.3/5.0. Qualitative feedback highlighted the conversational query interface and real-time budget visualisation as the most valued differentiators from conventional booking applications.

X. COMPARATIVE ANALYSIS

Feature	Proposed	Baseline-A	Baseline-B
NL Input	Yes (LLM)	No	Limited
Budget Scoring	Yes (3-class)	No	No
Itinerary Gen.	Yes (LLM)	No	No
Parse Accuracy	91.4%	67.3%	N/A
Hotel Prec.@5	85.2%	74.1%	81.8%
Response Time	2.31 s	0.42 s	1.87 s
Mobile-First	Yes	Partial	No
Offline History	Yes	No	No
Expense Split	Yes	No	No
Cold-Start	High	Medium	Low

TABLE V. Comparative Analysis Against Baselines

The proposed system outperforms both baselines on natural language understanding and feature completeness. Baseline-B achieves comparable hotel recommendation precision, benefiting from collaborative user behaviour signals unavailable to the present system. The latency overhead relative

to Baseline-A is attributable entirely to the Groq API round-trip and is considered acceptable given the substantially richer output produced.

XI. LIMITATIONS

- Static Hotel Dataset: The SQLite database contains 1,247 hotels across 38 Indian cities and does not reflect real-time availability or dynamic pricing.
- Absence of Real-Time API Integration: The system does not interface with live hotel booking APIs (Amadeus, MakeMyTrip), limiting its utility as a transactional platform.
- External AI Dependency: System availability is contingent on the Groq API; any service disruption renders NLP and itinerary features unavailable.
- Limited User Personalisation: Without persistent user accounts, recommendation weights remain static population-level values.
- Indian Domestic Scope: The hotel dataset and budget cost indices are restricted to Indian domestic destinations, limiting international generalisability.
- Prototype Dataset: The dataset used is a prototype dataset and can be extended with real-time hotel APIs such as Amadeus or MakeMyTrip for production-grade deployment.

XII. FUTURE SCOPE

A. Real-Time API Integration

Integration with live hotel inventory APIs (Amadeus Travel API, RapidAPI Hotels) will transform the system into a fully transactional booking assistant without requiring architectural changes.

B. Reinforcement Learning from User Feedback

Implicit user feedback signals can be harvested to train a personalised ranking model using contextual bandit algorithms, progressively adapting recommendation weights to individual preference vectors.

C. Multimodal Input

Voice-based query submission via Whisper ASR and image-based hotel preference specification through CLIP-based visual similarity search would extend system accessibility significantly.

D. Explainable Recommendation

Generating natural-language explanations for recommendations would increase user trust and align with emerging requirements in the EU AI Act for transparency in AI-assisted consumer decisions.

E. Weather and Event-Aware Planning

Incorporating real-time weather forecasts (OpenWeatherMap API) and local event data into the itinerary generation prompt would enable seasonally and contextually aware travel planning.

F. Collaborative Trip Planning

A shared trip workspace synchronised through WebSocket-based real-time infrastructure would address the coordination overhead of group travel planning.

XIII. CONCLUSION

This paper has presented the design, implementation, and empirical evaluation of an AI-powered hotel recommendation and travel planning system. The system's three primary technical contributions are: (i) a LLaMA 3.3 70B-powered natural language query parser achieving 91.4% parameter extraction accuracy without domain-specific fine-tuning; (ii) a rule-based budget feasibility engine classifying trip affordability with 88.7% accuracy; and (iii) a composite hotel ranking algorithm achieving Precision@5 of 85.2%.

Evaluation results confirm that the system significantly outperforms keyword-based baseline approaches on natural language understanding and feature completeness. The System Usability Scale score of 81.3 indicates strong acceptance of the conversational interface paradigm. The modular four-tier architecture depicted in Fig. 1 ensures that each component can be independently upgraded or replaced as more capable technologies emerge. The system development follows an iterative and modular software engineering approach aligned with Software Project Management principles.

XIV. ACKNOWLEDGEMENTS

The authors would like to thank Bhagwan Parshuram Institute of Technology, Delhi, for providing the academic environment and institutional support that made this research possible. We are grateful to Groq Inc. for providing API access to the LLaMA 3.3 70B Instruct model. We also sincerely thank the 40 participants who voluntarily contributed their time to the usability study. The authors declare no conflicts of interest.

XV. REFERENCES

- [1] D. Patel et al., "A Comprehensive Review of Hotel Recommendation Systems Using Reviews and AI," *International Journal of Scientific Research in Computer Science*, 2026.
<https://ijsrceit.com/index.php/home/article/view/CSEIT261215/CSEIT261215>
- [2] L. Aravani et al., "A Natural Language Processing Framework for Hotel Recommendation Based on Users' Text Reviews," arXiv, 2024.
<https://arxiv.org/abs/2408.00716>
- [3] M. Remountakis et al., "ChatGPT and Persuasive Technologies for Personalized Hotel Recommendations," arXiv, 2023.
<https://arxiv.org/abs/2307.14298>
- [4] S. Patil et al., "Hotel Recommendation System Using Deep Learning and Contextual Analysis," 2025.
<https://journal.ardigitech.in/admin/papers/67e2a3c921a54.pdf>
- [5] I. Saifudin et al., "Systematic Literature Review on Recommender Systems," 2024.
<https://eclass.hmu.gr/modules/document/file.php/TP374/Homework/1st%20Homework%20-%20SLR%20evaluation/Systematic%20Literature%20Reviews/2024%20-%20Systematic%20Literature%20Review%20on%20Recommender%20Systems.pdf>

- [6] P. Pang et al., "Hotel Recommendation System Using Machine Learning Techniques," 2023.
https://eprints.utar.edu.my/5997/1/fyp_IA_2023_PCC.pdf