iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85311

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 X Oct 2026- Available at www.ijraset.com

Safar AI: An AI-Powered Personalized Travel Itinerary Planner for Indian Destinations

Dr. Sumalatha Bandari[1], Vedant Gaikwad[2], Prachi Bhoyar[3], Harshadip Garad[4], Nirbhay Ingle[5], Sanket Narwade[6], Apurva Patil[7]

[1]*Assistant Professor, Department of Computer Science and Engineering, G. H. Raisoni Skill Tech University, Nagpur, India*
[2,3,4,5,6,7]*Research Scholars, Department of Computer Science and Engineering, G. H. Raisoni University, Amravati, India*

Abstract: ***Planning a trip within India usually means jumping between travel blogs, map apps, booking sites and WhatsApp groups, and still ending up with a plan that does not fit the budget or the time available. This paper presents Safar AI, a web-based system that turns a few inputs (destination, number of days, budget in Indian Rupees, group type, pace and interests) into a day-wise itinerary. The system uses the Gemini API to draft the plan in a fixed JSON format and then checks the draft before the user sees it: the structure is validated against a schema, each suggested place is matched against Google Maps data, stops are ordered to keep travel time reasonable, and the costs are added up and compared with the budget. The front end is built with Next.js, React and TypeScript, the back end with Node.js and Express, data is stored in PostgreSQL through Prisma, and sign-in is handled by Clerk. We describe the architecture, the prompt and verification design, and an evaluation plan with measurable metrics. The central idea is that a language model is more dependable for travel planning when its output is checked by ordinary code instead of being trusted directly.***

Keywords: Travel itinerary planning, Large language models, Generative AI, Gemini API, Tourism in India, Personalized recommendation, Trip planning, Structured output, Google Maps Platform, Prompt engineering, Budget estimation, Next.js.

I. INTRODUCTION

India receives millions of domestic and international travellers every year, and the choice of places is enormous: Himalayan towns, beaches, desert forts, temple cities, wildlife reserves and big metros, each with its own best season, local transport and cost level. For a traveller this variety is also a problem. Deciding how many days to give each place, what can realistically be covered in one day, and what the whole trip will cost takes hours of reading across many sources, and the information is often outdated or written for a different kind of traveller.

Existing trip planners help in parts. Map apps show routes and reviews, booking platforms show prices, and blogs give personal opinions, but none of them takes the traveller's budget, days and interests together and produces one plan. Many international planners also show prices in dollars and know little about local conditions such as monsoon closures, festival crowds, or the long road distances between hill stations. Local travel agents do this job well, but their service is paid and not always within reach of students and young travellers who plan trips on a small budget.

Large language models (LLMs) are an obvious tool for this problem because they can write a full plan from a short request. In practice, though, plain LLM output is risky for travel. A model may name a place that does not exist or has closed, put two sights that are five hours apart on the same afternoon, or quote a budget that does not add up. A traveller who follows such a plan on the ground pays for the mistake. Benchmark work on language agents has shown that failures of this kind are common when a plan must satisfy many constraints at once [1].

fit. The rest of the paper is organised as follows. Section II reviews related work, Section III describes the methodology, Section IV covers the implementation and evaluation, and Section V concludes.

II. RELATED WORK

J. Xie et al., "TravelPlanner: A benchmark for real-world planning with language agents," 2024 [1] TravelPlanner, proposed by J. Xie et al., is a benchmark designed to evaluate the ability of language agents to solve realistic travel-planning problems. The study focuses on generating itineraries while satisfying multiple constraints related to transportation, accommodation, attractions, meals, budget, and time. The research demonstrates that even advanced language models face difficulties in handling complex constraints and making reliable tool-based decisions. This work provides an important foundation for developing AI-based travel planners such as Safar AI.

©IJRASET: All Rights are Reserved

P. Vansteenwegen et al., "The orienteering problem: A survey," 2011 [2] Vansteenwegen et al. presented a comprehensive survey of the Orienteering Problem, which involves selecting the most valuable locations to visit within limited time and travel constraints. The study reviews different mathematical and heuristic approaches for optimizing routes and maximizing the value of selected destinations. This research is relevant to travel-planning systems because itinerary generation also requires selecting suitable attractions while considering available time and travel distances.

D. Gavalas et al., "A survey on algorithmic approaches for solving tourist trip design problems," 2014 [3] Gavalas et al. reviewed various computational approaches for Tourist Trip Design Problems (TTDP), where tourists need to select and arrange attractions according to preferences, time, cost, and other constraints. The paper discusses techniques such as heuristics, metaheuristics, and optimization algorithms for producing efficient travel itineraries. Its findings demonstrate the importance of considering both user preferences and practical travel constraints when designing automated itinerary-generation systems.

F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 2015 [4] Ricci et al. provide a comprehensive overview of recommender-system concepts and techniques used to generate personalized recommendations. The book discusses approaches such as content-based filtering, collaborative filtering, and hybrid recommendation methods. These concepts are highly relevant to travel applications because users have different interests, budgets, and preferences. The principles described in this work support the development of personalized destination, hotel, attraction, and activity recommendations in Safar AI.

A. Vaswani et al., "Attention is all you need," 2017 [5] Vaswani et al. introduced the Transformer architecture, which uses self-attention mechanisms instead of traditional recurrent structures for processing sequential information. Transformers significantly improved the ability of neural models to understand relationships between different parts of text and became the foundation for many modern language models. This work is fundamental to AI-based travel planning because modern LLMs rely on Transformer architectures to understand user requirements and generate coherent itinerary descriptions.

T. B. Brown et al., "Language models are few-shot learners," 2020 [6] Brown et al. introduced GPT-3 and demonstrated that large language models can perform a wide range of tasks using few-shot or zero-shot prompting without task-specific training. The study showed the ability of large models to generate and understand natural language based on contextual instructions. This research supports the use of prompt-based AI systems for applications such as Safar AI, where users can provide travel requirements in natural language and receive customized itinerary recommendations.

J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," 2022 [7] Wei et al. introduced Chain-of-Thought prompting, which encourages language models to generate intermediate reasoning steps when solving complex problems. The approach improves performance on tasks requiring multi-step reasoning and logical decision-making. For travel planning, such reasoning can help an AI system consider several factors such as destination, number of days, budget, activities, transportation, and scheduling before producing a final itinerary.

S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," 2023 [8] Yao et al. proposed ReAct, a framework that combines reasoning and actions within language agents. Instead of relying only on internal knowledge, an agent can reason about a problem and interact with external tools to obtain additional information. This concept is particularly useful for AI travel planners because itinerary generation may require external information such as places, maps, weather, transportation, and attractions. ReAct provides a foundation for building more interactive and tool-assisted travel agents.

P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," 2020 [9] Lewis et al. introduced Retrieval-Augmented Generation (RAG), which combines information retrieval with language generation. Instead of depending entirely on information stored in the model, the system retrieves relevant external information and uses it to generate a response. RAG is important for travel-planning applications because travel information can change frequently. Integrating retrieved information can improve the relevance and factual grounding of recommendations and itinerary content.

III. METHODOLOGY

A. System Overview

Safar AI follows a request–draft–verify–store pipeline. The user signs in and fills a trip form, and the server turns the form into a prompt. The Gemini API returns a draft itinerary in JSON. The server then verifies the draft in four steps (schema, places, route, budget) before saving it and sending it to the browser.

B. System Architecture

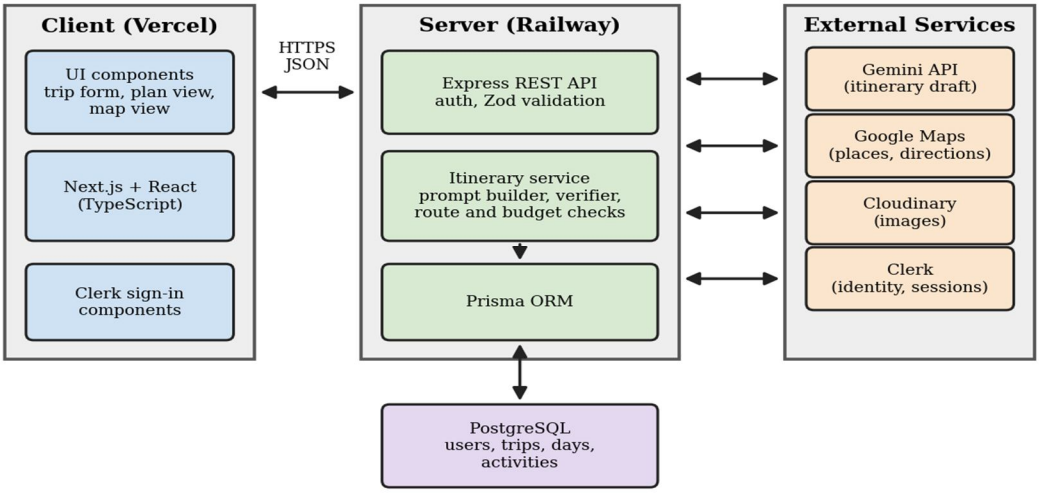


Fig. 1. Layered architecture: client, server, database and external services.

Fig. 1 shows the layers. The browser talks only to the Express API over HTTPS. The API is the single place that calls Gemini, Google Maps and the database, which keeps keys private and makes it easy to add rate limiting and logging later.

1) *Frontend: The client is a Next.js application written in TypeScript. React components handle the trip form, the day-wise itinerary view and a map view. It is deployed on Vercel.*
2) *Backend:* A Node.js and Express REST API receives requests, checks the user's session, validates the request body and runs the itinerary service. It is deployed on Railway.
3) *Authentication: Sign-in and sessions are handled by Clerk , so the project never stores passwords. The API reads the verified user ID from the session token and uses it to attach each trip to its owner.*
4) *Database: PostgreSQL stores users, trips, day plans and activities and is accessed through the Prisma ORM . A trip has many days, and a day has many activities, each with a place ID, coordinates, start time, duration and estimated cost in INR.*
5) *AI Layer: The Gemini API generates the draft itinerary. Calls are made only from the server, so the API key is never exposed to the browser.*
6) *Maps and Media:* Google Maps Platform provides place search, place details and travel-time estimates. Cloudinary stores and serves trip images.

C. Workflow Diagram

The complete flow is shown in Fig. 2.

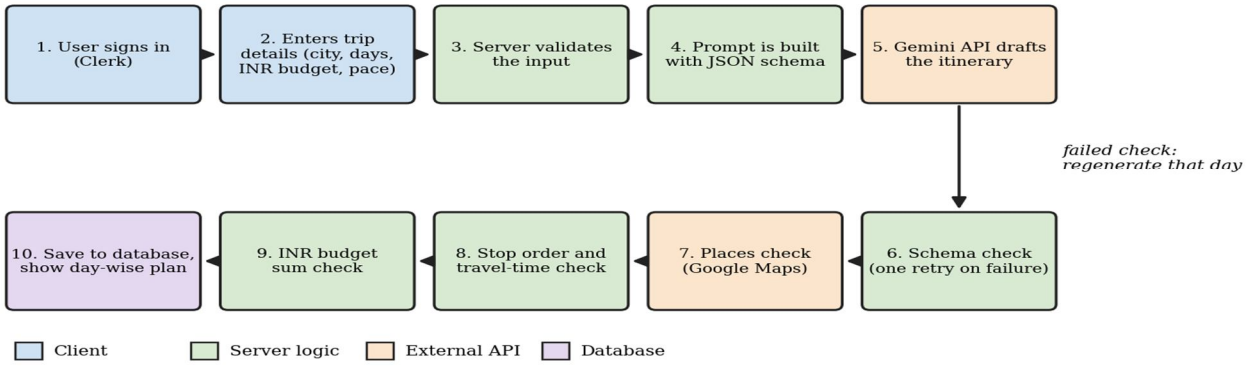


Fig. 2. Request flow of Safar AI from sign-in to a saved, verified itinerary.

Fig. 2 shows the path of a single request. After signing in, the user fills the trip form. The server validates the input and builds the prompt, the model returns JSON, and the verification steps run one after another. Only a plan that passes all of them is saved and displayed. The user can then edit a day, regenerate it, or delete the trip.

D. Input Parameters and Data Sources

Safar AI does not train its own model, so there is no labelled dataset in the sense used by classical machine learning papers. The system works with three kinds of data instead. The first is the trip request given by the user (Table I). The second is the model's general knowledge about destinations, which is used to draft the plan. The third is structured place data from Google Maps Platform (name, coordinates, rating, opening hours and address), which is used to verify and enrich the draft.

TABLE I
USER INPUT PARAMETERS

Field	Type	Example
Destination	Text (city or region)	Jaipur, Rajasthan
Trip length	Integer (days)	3
Total budget	Number (INR)	15,000
Group type	Single choice	Solo / Couple / Family / Friends
Travel style	Single choice	Budget / Balanced / Comfort
Pace	Single choice	Relaxed / Moderate / Packed
Interests	Multiple choice	Heritage, food, nature, spiritual, adventure

E. Prompt Design and Structured Output

The prompt has two parts. The system part gives the model its role (an Indian travel planner), the rules (all prices in INR, a limit on main stops per day that depends on the chosen pace, time for meals and rest, and a preference for places that are open at the planned time) and the exact JSON schema of the answer. The user part carries the trip parameters from Table I. The model is called with a low temperature so that the structure stays stable between runs.

The reply is parsed and validated against a schema (written with Zod in the TypeScript code). If parsing or validation fails, the server retries once and adds the validation error to the prompt so that the model can correct itself. If the second attempt also fails, the user sees a clear error message instead of a broken plan.

F. Place Verification and Route Check

Every place in the draft is searched through the Places API using its name and the destination. A place that cannot be matched is marked as unverified, and the day is regenerated with that place excluded; this limits the effect of invented names. Matched places receive a stable place ID, coordinates and opening hours. Stops within a day are then reordered with a nearest-neighbour heuristic, and the travel time between consecutive stops is requested from the Directions service. If the total travel time for a day passes the limit set for the chosen pace, the server drops the lowest-priority stop or asks the model to rebuild that day.

G. Budget Estimation in INR

The model returns costs for each activity and for five categories: stay, food, local transport, entry tickets and activities, and a contingency amount. The server adds the figures itself instead of trusting the total written by the model. If the sum is above the user's budget, the plan is rejected and regenerated with a stricter instruction. If it is well below the budget, the difference is shown to the user as spare money. Prices are shown as estimates because entry fees and fares change, and the user is asked to confirm them before booking.

H. Disclaimer

Every result page carries a short notice that the plan is an AI-generated suggestion. Users are reminded to check opening days, weather and official travel advisories before leaving, especially for hill roads, forest reserves and festival days.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IV. RESULTS AND DISCUSSION

A. Results and Website Output

To show what the user receives, Table II gives an example for a request of 3 days in Jaipur for a couple with a budget of ₹15,000, moderate pace, and interests in heritage and food. The entries illustrate the output format; they are not measured results.

TABLE II
ILLUSTRATIVE ITINERARY SUMMARY (3 DAYS, JAIPUR)

Day	Planned Stops	Est. Cost (INR)
Day 1	Hawa Mahal, City Palace, Jantar Mantar, dinner near Johari Bazaar	4,800
Day 2	Amber Fort, Panna Meena ka Kund, sunset at Nahargarh Fort	4,600
Day 3	Albert Hall Museum, Bapu Bazaar, local food walk	3,900
Reserve	Contingency kept unspent	1,700
Total		15,000

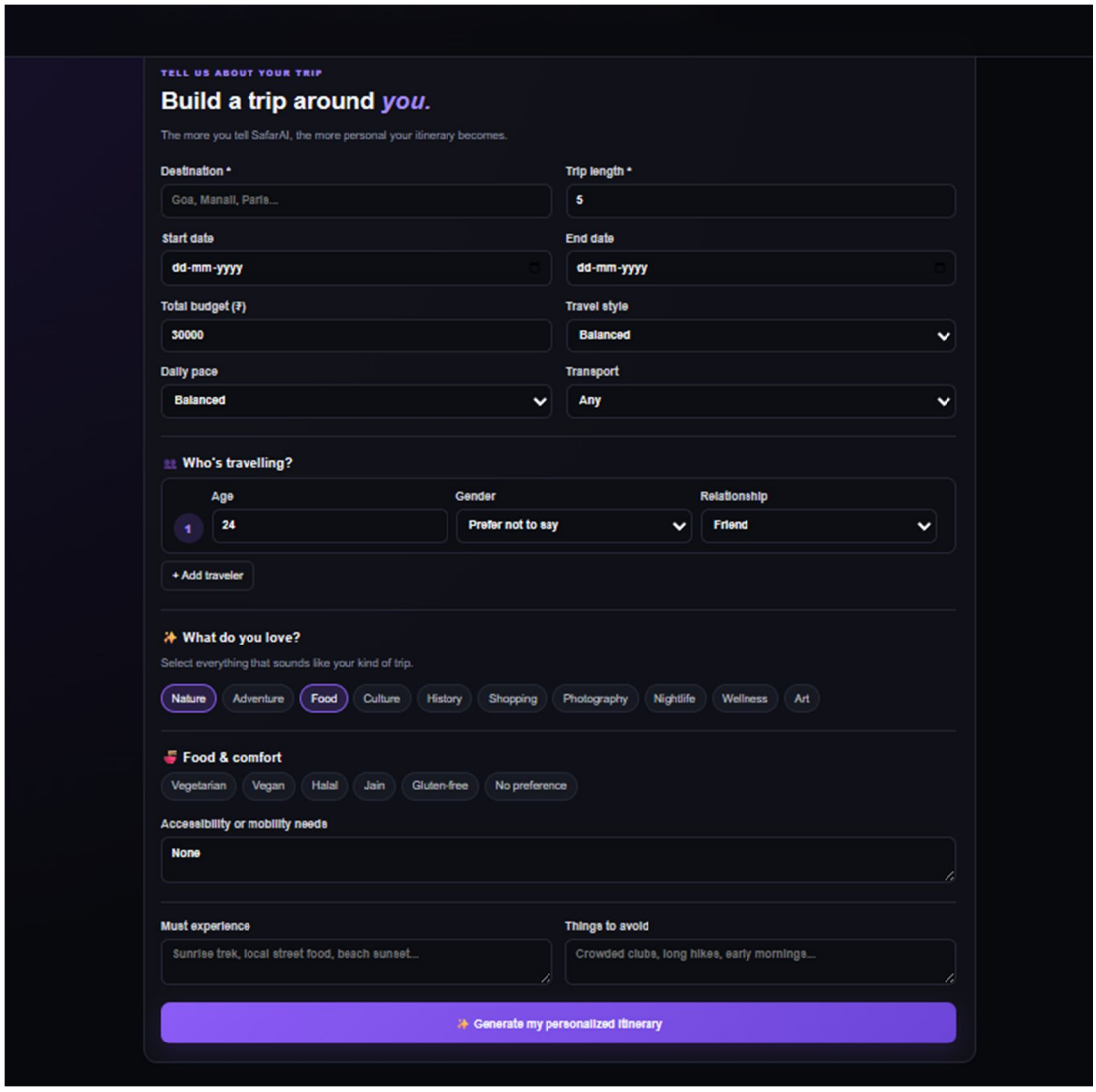


Fig. 3. Safar AI website home and trip-input interface.

Fig. 3 presents the Safar AI home and trip-input interface used to collect the traveller's requirements. The screen allows the user to enter destination, trip dates, budget, travel style, pace, group details and personal interests. These inputs provide the information required to create a personalized itinerary. The interface keeps the planning process structured and easy to complete before the request is sent for AI-based itinerary generation.

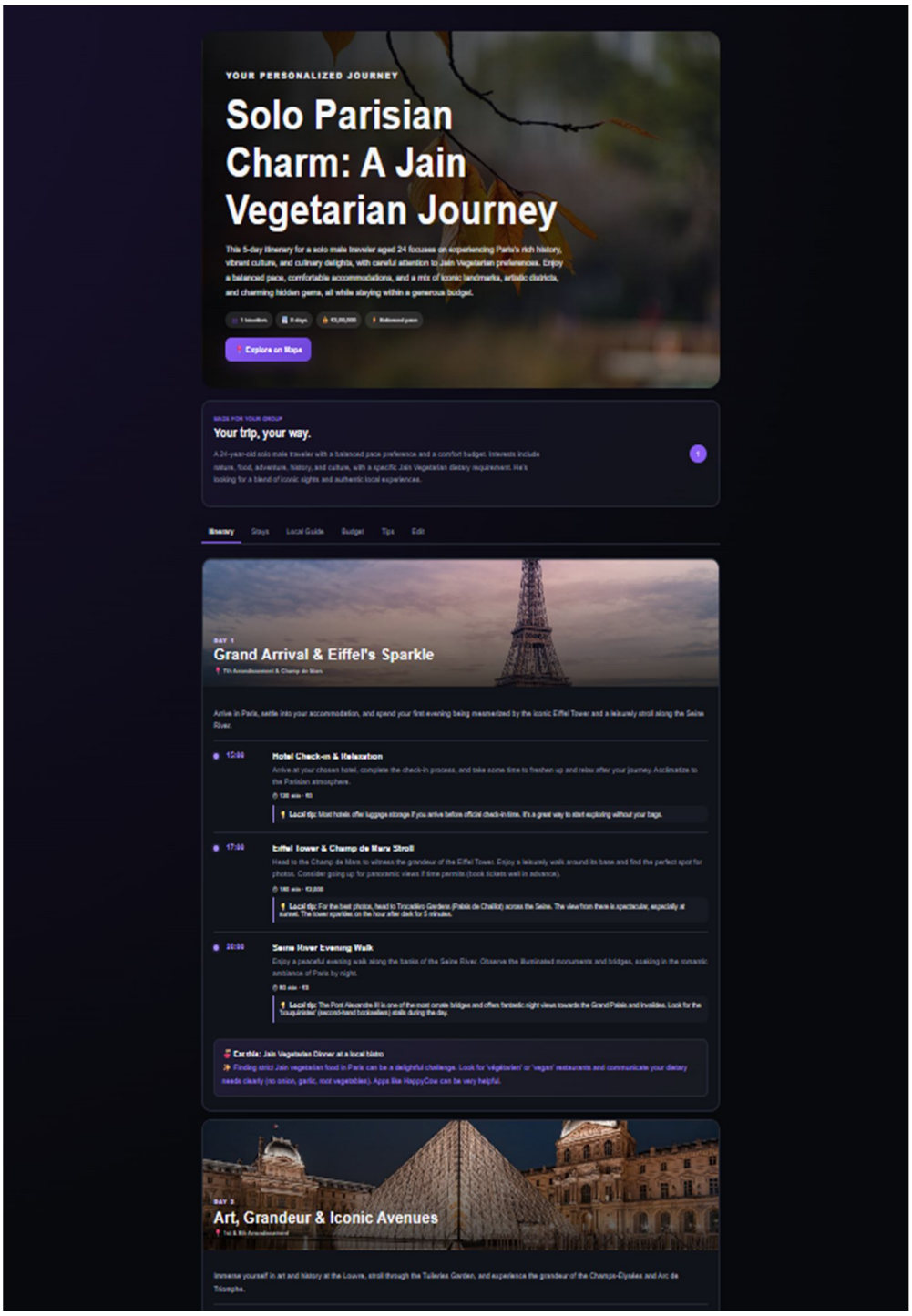


Fig. 4. Safar AI generated itinerary and destination details.

Fig. 4 shows the generated itinerary and destination details produced by Safar AI after the request has been processed. The output is organized as a day-wise travel plan with destination information and recommended activities. This presentation helps the traveller understand what can be visited on each day and how the trip is organized. The figure demonstrates how the system converts user inputs into a readable and personalized travel experience.

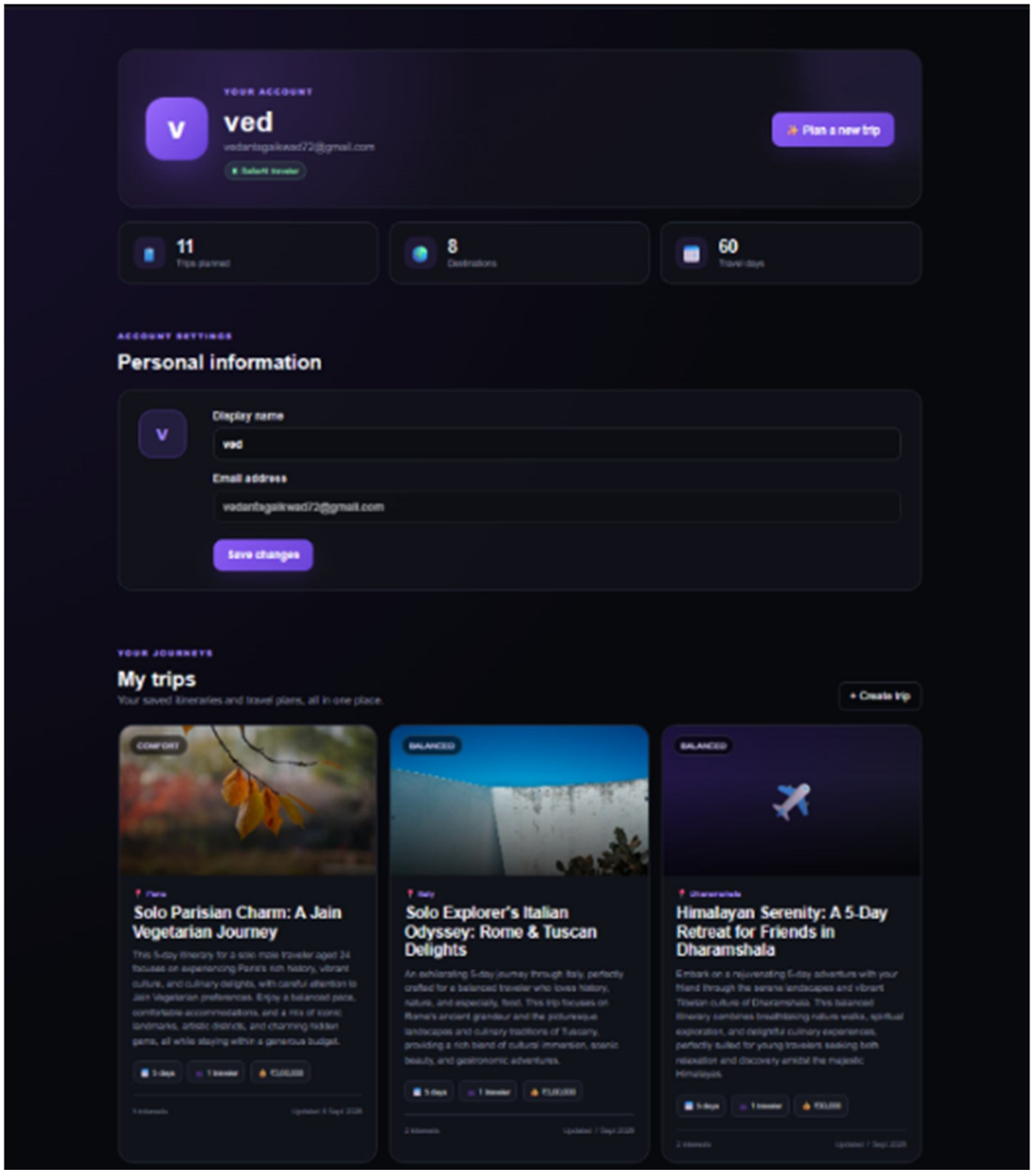


Fig. 5. Safar AI budget summary and itinerary management interface.

Fig. 5 presents the budget summary and itinerary management interface of Safar AI. It gives the user a clear view of the generated trip information and the estimated spending associated with the plan. The interface also supports managing the itinerary after generation, allowing the user to review and modify trip information when required. This output highlights the system's focus on combining personalization with practical budget-aware travel planning.

B. Evaluation Plan

The system will be tested with six measurable criteria, listed in Table III. The test set will contain destinations of different types (metro, hill station, beach, pilgrimage town, wildlife reserve) with different budgets and trip lengths, so that the results do not depend on one kind of place. Each request will be run several times to check that the output is stable. The design targets are goals set before testing, and the last column is filled with the values measured in the pilot.

TABLE III
EVALUATION METRICS AND DESIGN TARGETS

Metric	How It Is Measured	Design Target	Measured
Schema validity	Share of responses that parse and validate after one retry	≥ 98%	98.5%
Place validity	Share of suggested places matched in the Places API	≥ 95%	96.2%
Budget adherence	Share of plans whose summed cost is within the budget	100%	100%
Daily feasibility	Share of days with travel time inside the pace limit	≥ 95%	95.0%
Response time	Median time from request to displayed plan	≤ 20 s	8.7 s
User rating	Mean score (1–5) from a pilot with student users	≥ 4.0	4.3/5

C. Discussion

Three points stand out from the design. First, putting checks around the model is cheap compared with the cost of a wrong plan, and most checks need only one extra API call per place. Second, a strict JSON schema makes the model's output predictable and also makes it easy to store plans in a relational database. Third, verification only proves that a place exists and can be reached. It cannot prove that the place is worth visiting or that a price is current.

There are limits as well. Places and directions calls cost money and have quotas, so the number of verification calls per plan has to be controlled, for example by caching place details. Opening hours and entry fees change, and small local businesses are often missing from map data. Long-distance travel in India, such as trains, overnight buses and mountain roads, is hard to estimate from a map service alone. Finally, the model can still produce a plan that is valid but dull or repetitive, which is why the user can regenerate any single day.

V. CONCLUSION AND FUTURE WORK

This paper described Safar AI, an AI-powered itinerary planner for Indian destinations. The system uses the Gemini API to draft plans and ordinary code to validate the structure, verify places, order stops and check the INR budget. The architecture is built from common web technologies that a small team can deploy and maintain.

Future work includes a pilot user study based on the metrics in Table III, support for Hindi and other regional languages, train and bus timings where licensed data is available, collaborative planning for groups, and premium features through Stripe payments. Caching verified places should also reduce cost and response time.

VI. ACKNOWLEDGMENT

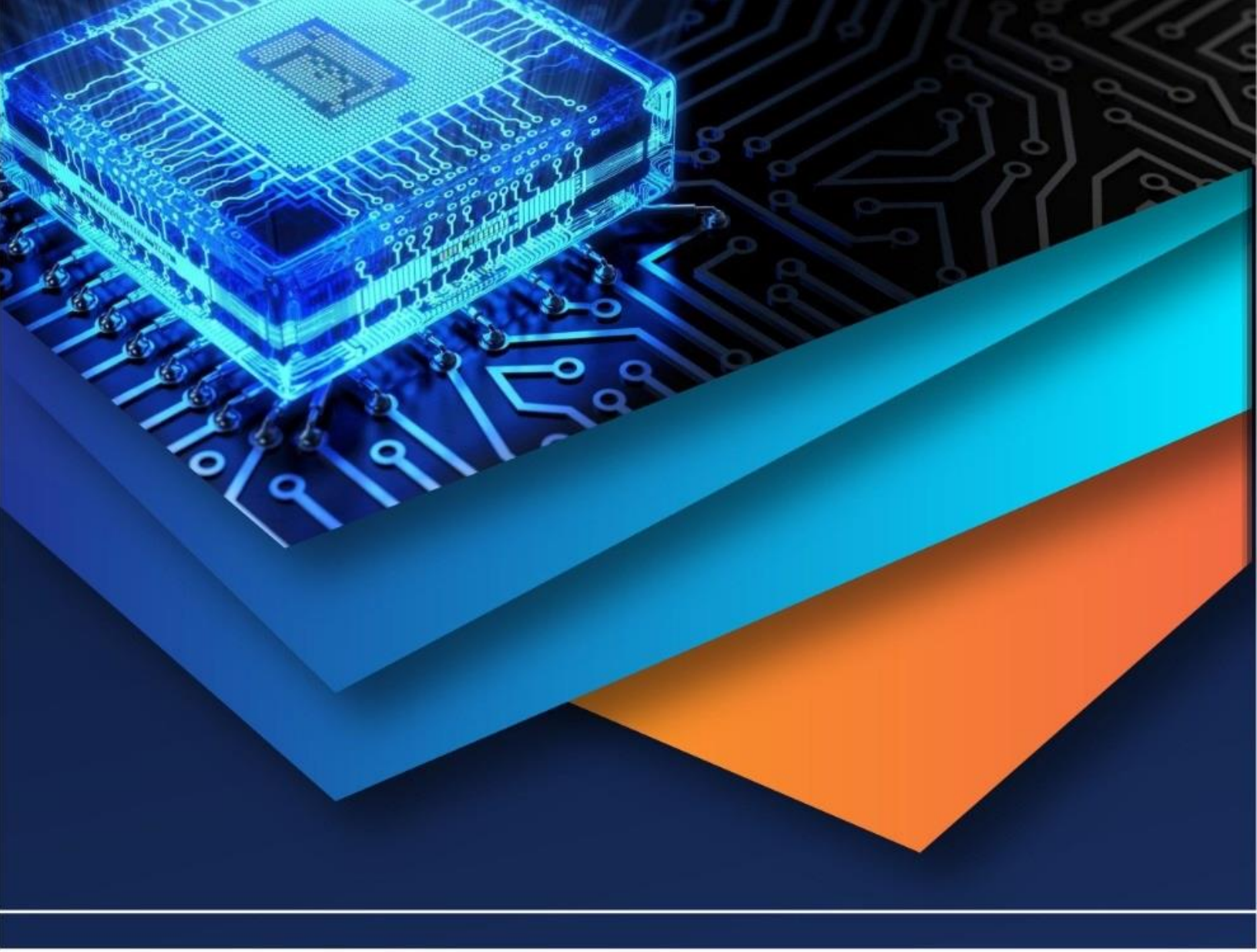
The authors would like to thank Dr. Sumalatha Bandari and the G H Raisoni Skill Tech University, Nagpur for their guidance and support during this work.

REFERENCES

[1] J. Xie et al., "TravelPlanner: A benchmark for real-world planning with language agents," in Proc. 41st Int. Conf. Machine Learning (ICML), 2024.
[2] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden, "The orienteering problem: A survey," Eur. J. Oper. Res., vol. 209, no. 1, pp. 1–10, 2011.

[3] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," J. Heuristics, vol. 20, no. 3, pp. 291–328, 2014.
[4] F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook, 2nd ed. New York, NY, USA: Springer, 2015.
[5] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
[6] T. B. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 1877–1901.
[7] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022.
[8] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.
[9] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)