International Journal of Scientific Research in Engineering & Technology
Volume6, Issue5 (September-October 2026), PP: 218-225.
https://www.doi.org/10.59256/ijsreat.20260605026
www.ijsreat.com
ISSN No: 2583-1240

TravelIQ: An AI-Powered Smart Travel Recommendation and Budget Planning System

Nagashri J A[1], Pruthvi D Dyamenahalli[2], Rohith K P[3], Sandeep Shrinivas Yallaraddi[4], Shifa H[5]

[1]*Assistant Professor, Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology / Visvesvaraya Technological University, India.*
[2,3,4,5] *Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology / Visvesvaraya Technological University, India.*

To Cite this Article: *Nagashri J A[1], Pruthvi D Dyamenahalli[2], Rohith K P[3], Sandeep Shrinivas Yallaraddi[4], Shifa H[5],"TravelIQ: An AI-Powered Smart Travel Recommendation and Budget Planning System", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 05, September-October 2026, PP:218-225.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *Planning a trip usually means hopping between search engines, booking sites, notes and spreadsheets, and on a group trip the money is often sorted out by hand at the end. This paper presents TravelIQ, a web application that keeps trip planning, a conversational travel assistant and shared-expense tracking in one place. A traveller fills in a single form and receives a day-wise itinerary, can put follow-up questions to the assistant, and can then divide a budget across spending categories, record expenses, split them among companions and compare planned and actual spending on a dashboard. The system is built in layers: a React and Vite frontend, a Python FastAPI backend that exposes REST endpoints, a MongoDB database and an external AI service that is reached only through the backend. The implementation was checked with twelve functional test cases, of which ten passed and two exposed known limitations (weather forecast availability and incomplete form input), and the arithmetic of the budget and settlement screens was verified on a six-person demonstration trip. The evaluation is functional only. Recommendation accuracy, itinerary feasibility, response time and user-study results are not reported, and the paper closes with its limitations and the evaluations that should follow.*

Keywords: *Travel planning; AI travel assistant; Budget management; Expense settlement; Web application; React; FastAPI.*

I. INTRODUCTION

A trip is really a bundle of linked decisions: where to go, when to travel, how to get there, where to stay, what to do each day and how much money to set aside. Most travellers make these decisions in several unconnected places. Tourism recommender systems have been studied for a long time, and the available surveys describe the techniques, interfaces and artificial intelligence methods used to help tourists choose [1], [2], [9], [10], building on the content-based, collaborative and hybrid families of recommendation methods [6]. Mobile guides add the traveller's situation, such as location and time, to the picture [3], and ontology-based systems such as SigTur/E-Destination match leisure activities to individual tastes [7].

Most of this work is about what to visit and in what order, and route design has become an optimisation problem in its own right [8]. The financial side of a trip receives much less attention: splitting a budget across categories, noting what was actually spent, and squaring accounts when several people travel together. These are exactly the chores that cause friction during and after a group trip. Language-model assistants can now answer travel questions in everyday language [11], yet their answers are not guaranteed to be right. A benchmark built around travel queries with many interacting constraints found that even strong language agents rarely produced a fully valid plan [4], and neural text generators are known to produce statements that nothing supports [5]. A planner built on such an assistant should therefore offer its output as guidance that the traveller can check. Among the sources we consulted, shared-expense settlement and budget analytics were not treated as part of a travel-planning system; our reading was deliberately limited rather than systematic, so we make no claim about what other tools may offer.

This paper describes TravelIQ, a web application that brings trip planning, an AI travel assistant and trip-level financial management together, so that the traveller enters the trip data once. Our contributions are:

1. A layered architecture that connects a browser interface, a REST backend, a database and an external AI service for travel planning and expense management.
2. Seven functional modules covering trip information, conversational assistance, budget allocation, expense recording, group settlement, financial analytics and data management, together with the processing logic behind them.

3. A functional check of the implemented features through twelve test cases, supported by screenshots of the working interface.

Objectives:

- To collect trip details and preferences, present a personalised day-wise travel plan, and offer a conversational assistant for follow-up travel questions such as accommodation suggestions.
- To let users create trip budgets with category-wise allocations and record expenses against them.
- To support shared trips by splitting expenses among travellers and showing balances and suggested settlements.
- To summarise planned and actual spending on a dashboard with charts and filters, and to verify the implemented functions through test cases.

The rest of the paper is organised as follows. Section II explains the design, workflow and processing logic. Section III presents the interface, the test cases and the observed outputs. Section IV discusses strengths and limitations, and Section V concludes the paper.

II. MATERIAL AND METHODS

System Architecture

TravelIQ keeps the user interface, application logic, data storage and external services apart (Fig. 1). The user layer covers registration and login, entry of trip details, management of budgets and expenses, and conversation with the AI assistant. The frontend is written with React and Vite; it holds the pages and forms, uses React Router for navigation and sends requests through Axios. The backend is written in Python with FastAPI. It exposes the REST endpoints [14], applies the business logic for travel plans, budgets and expenses, and connects to the AI service. Users, trips, budgets, expenses and participants are stored in MongoDB, and an output layer returns the plan, the assistant's replies, the budget dashboard and the trip summaries. The AI service is drawn as optional in the architecture, with OpenAI as an example, and it is reached only through the backend.

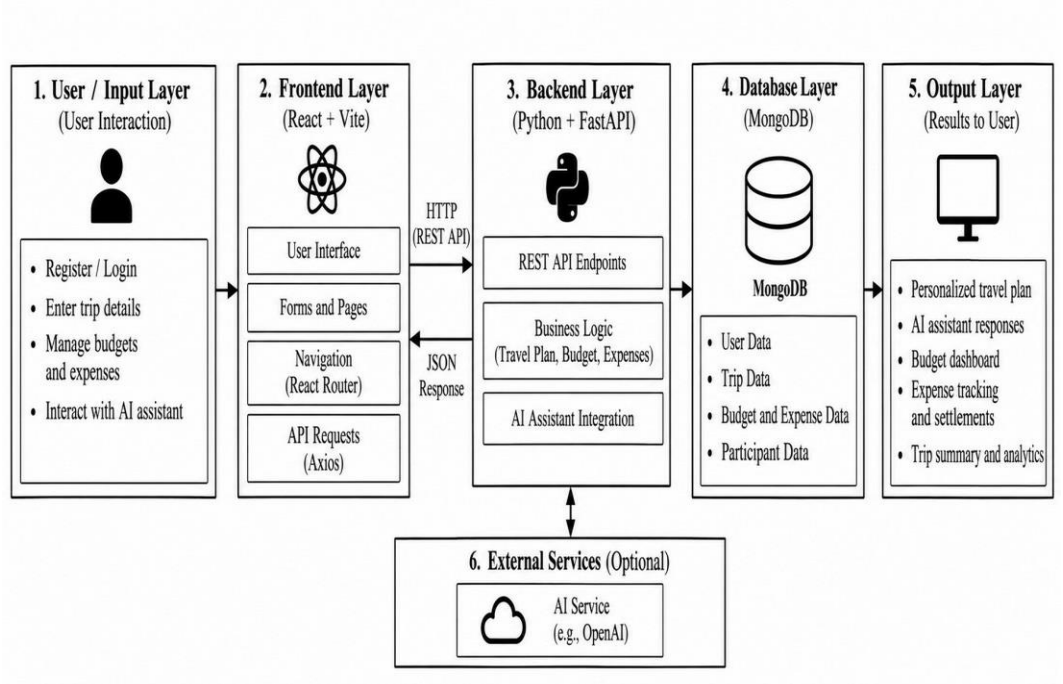

Fig. 1. System architecture of TravelIQ.

Frontend and backend exchange JSON over HTTP. A request from the interface is handled by the backend, which reads or writes the database where needed and returns a JSON response for the frontend to display. A question for the assistant follows the same route: the backend forwards it to the AI service and passes the generated reply back.

Workflow

The overall workflow is shown in Fig. 2. After the user logs in and opens the application, trip details and preferences are entered and validated. Invalid input produces an error message and returns the user to the form; valid input leads to the generated plan. From the plan the user may consult the AI assistant as often as needed and come back, or move on to the optional management of budgets and expenses.

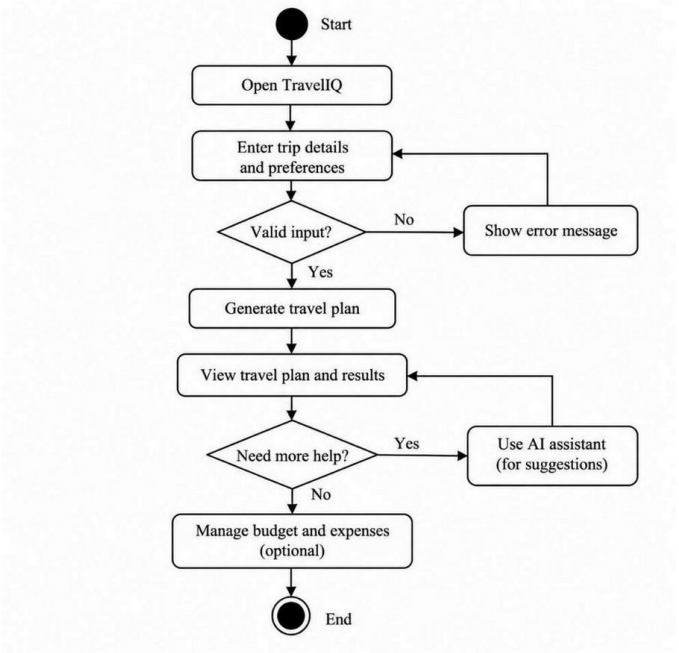

Fig. 2. Activity diagram of the TravelIQ planning workflow.

Functional Modules

The functionality is divided into the seven modules summarised in Table 1. Each module takes its input from the interface and hands its output to the dashboard or the chat panel.

Module	Main inputs	Main outputs
Trip information	Origin, destination, start and end dates, budget, transport mode, travel type, number of people, preferences	Personalised day-wise travel plan
AI travel assistant	Free-text travel query	Generated reply in a chat panel; follow-up questions allowed
Trip budget	Trip name, total budget, currency, dates, category allocations	Saved trip budget with category-wise allocations
Expense recording	Description, category, amount, date, payer, split method, notes	Expense list, updated totals, edit and delete actions
Group settlement	Traveller names, expenses, contributions, split method	Contribution, share and net balance per traveller; suggested settlements
Financial dashboard and analytics	Trip, budget, category, currency and date filters	Summary cards and charts for category-wise spending, budget versus actual and spending over time
Data management	Trip, budget, expense and participant records	Stored and retrieved records for dashboards and analytics

Table 1: Functional modules of TravelIQ

Technology Stack

Table 2 lists the technologies and the role each one plays.

Component	Technology	Role
Frontend	React with Vite	User interface, forms, pages and visual summaries
Navigation	React Router	Movement between planning, budget, expense and analytics views
API requests	Axios	HTTP requests from the frontend to the backend
Backend	Python with FastAPI	REST endpoints, business logic and AI service integration
Database	MongoDB	Users, trips, budgets, expenses and participants
External service	AI service (OpenAI given as example)	Responses to travel queries

Table 2: Technologies used in TravelIQ

Processing Logic

Trip information. The form collects origin, destination, dates, budget, transport mode, travel type, number of people and free-text preferences. If a required detail is missing or invalid the user is asked to correct it; otherwise the request is submitted and the result is displayed. The number of people is not typed in but is set from the chosen travel type, and the budget on this form is entered in US dollars.

AI assistant. An empty query is rejected. A valid query is prepared and sent to the AI service through the backend. If a reply comes back it is shown in the chat and the user may ask a follow-up; if not, the user is told that no response could be obtained. Whether the conversation context is retained depends on what the service supports. In the accommodation example of Fig. 4b the reply refers to the budget, the number of travellers and the length of stay, so the trip details entered on the form are passed to the assistant together with the question.

Budget management. The trip name, total budget, currency, dates and allocations are validated before a budget is saved, retrieved and displayed. The budget form offers six categories: transportation, accommodation, food and beverages, activities and sightseeing, shopping, and insurance and miscellaneous expenses. The allocations are optional, and the form shows their running total next to the heading so that the traveller can compare it with the total budget before saving.

Expense recording. Each expense carries a trip, description, category, amount, date, payer, split method and optional notes. A valid record is saved and the expense summary is refreshed. Each entry in the expense list shows its description, category, date and payer together with the amount, and it can be edited or deleted; the totals are recalculated after every such change.

Shared-expense settlement. For a trip with n travellers, let a_e be the amount of expense e and c_i the total paid by traveller i. Under equal splitting, the share of traveller i and the resulting net balance are

$$s_i = (1 / n) \Sigma a_e \quad (1)$$

$$b_i = c_i - s_i \quad (2)$$

where the sum in (1) runs over the shared expenses of the trip. A positive b_i marks a traveller who should receive money and a negative value one who owes money. When every expense is paid by one traveller and shared by all, the balances add up to zero. The application then separates those who owe from those who should receive and works out suggested payments between them. This zero-sum property follows directly from the two equations, since the total paid, Σc_i, equals the total of the shared expenses, Σa_e, and the shares add up to the same amount; it therefore serves as a quick check that the recorded payers and amounts are consistent. Because the balances sum to zero, any set of payments that brings every balance to zero settles the trip, and repeatedly pairing the traveller who owes the most with the one who is owed the most needs at most $n - 1$ payments [13]. A recorded settlement only moves balances between travellers; it is not an expense, so it does not change the actual spending used in equation (3).

Financial analytics. For the chosen filters the application retrieves the matching budget and expense records and computes the planned budget B, the actual expenses A (the sum of the recorded amounts), the remaining balance R and the average spending per trip over T tracked trips:

$$R = B - A \quad (3)$$

$$\text{Average per trip} = A / T \quad (4)$$

It also identifies the top expense category, which is the category with the largest total of recorded amounts, and draws charts of category-wise spending, budget against actual spending, and spending over time. A positive remaining balance means that spending is still within the planned budget, and a negative one means that the budget has been exceeded. The saved-budget card expresses the same comparison as the share of the budget already spent, which drives its progress bar. Because all of these quantities are computed from the records that match the chosen filters, changing a filter changes the planned budget, the actual expenses and the number of tracked trips together, so the summary cards, the charts and the average per trip always describe the same selection. Presenting the key totals as summary cards beside the charts follows common dashboard design practice [15].

III. RESULT

Snapshots

Home page. The home page (Fig. 3) is the entry point. Its navigation bar links to the home page, the budget dashboard, the about and how-it-works sections, trip sharing and contact details, and it includes a light and dark theme switch. A “Start Planning Trip” button leads to the planning form, and three cards highlight AI-powered recommendations, custom itineraries and hassle-free planning. The user name is hidden in the figure.

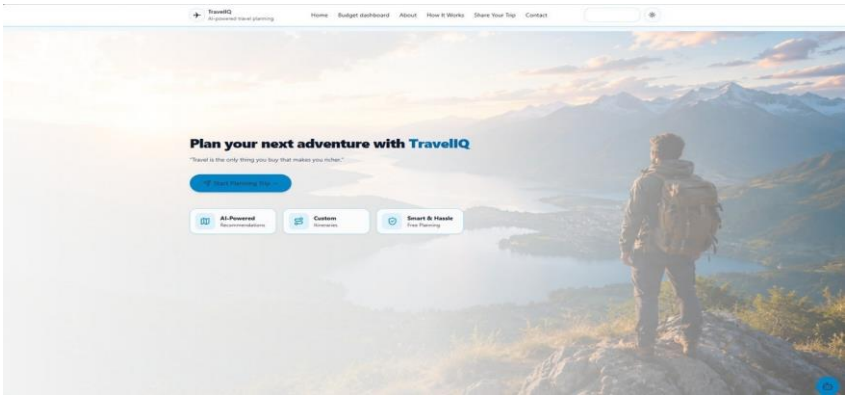

Fig. 3. TravelIQ home page (user name hidden).

Planning form and AI travel assistant. The planning form (Fig. 4a) asks for the traveller's name and email, origin, destination, start and end dates, budget, transport mode, travel type (solo, couple, family or group/friends), number of people and free-text preferences. The number of people is filled in automatically from the travel type, as the form itself notes; name and email are hidden in the figure. The assistant (Fig. 4b) is a chat panel that greets the user as an AI travel copilot and accepts questions in natural language. In the captured exchange the user asks for accommodation in Odisha, and the reply mentions the stated budget, the number of travellers, the length of stay and an accommodation allocation of $120 (about $30 per night) before listing budget-friendly options. This suggests that the trip details are available to the assistant. The names of the establishments are generated suggestions that we have not verified, so they are not discussed here

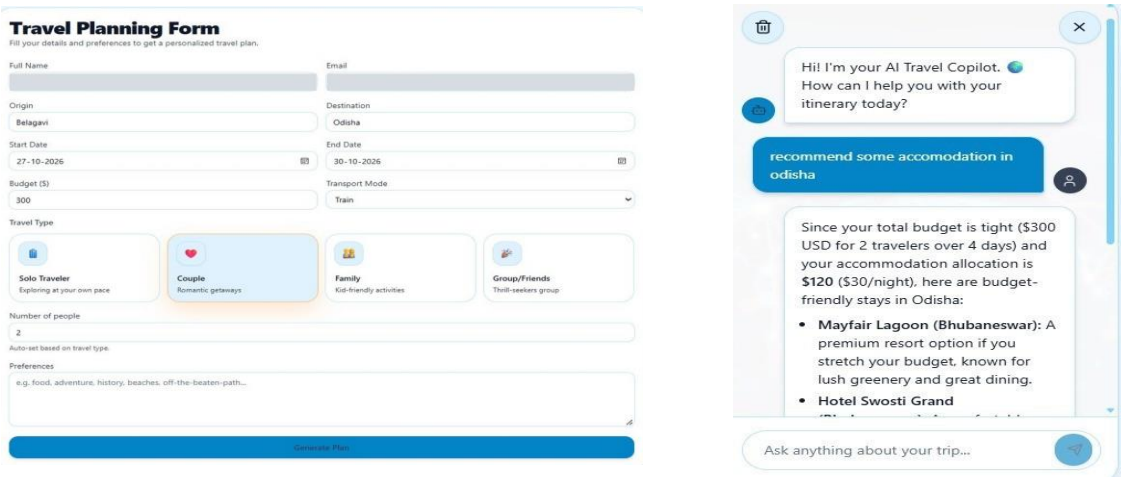

Fig. 4. (a) Trip planning and preference form. (b) AI travel assistant answering an accommodation query.

Day-wise itinerary. Once the form is submitted, the application presents an itinerary organised by day (Fig. 5). In the visible part each day has a heading, a short summary and a list of time-slotted activities: arrival, transfer and check-in on the first day, and a morning-to-evening exploration plan on the second. The transfer entry gives an approximate duration and carries a note that the fare is not verified and should be confirmed locally. That note shows the plan separating what the system can support from what the traveller must check.

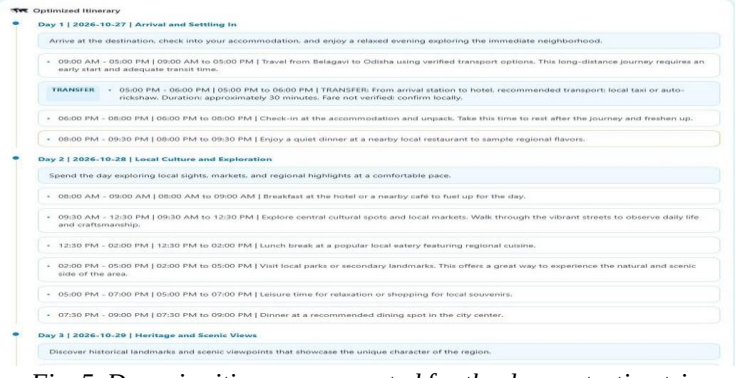

Fig. 5. Day-wise itinerary generated for the demonstration trip.

Budget dashboard and finance analytics. The budget dashboard (Fig. 6) shows summary cards for the planned budget, actual expenses, remaining balance, average per trip, top category and number of trips tracked. Below the cards is a form for creating a trip budget with a currency, dates and the six category allocations. A budget is created for a trip that already exists in the user's account, so the planned amount, currency and dates are attached to that trip and are not typed again elsewhere. After each change the dashboard shows a short confirmation message that can be dismissed, and a refresh button reloads the saved data. The saved-budgets panel (Fig. 7) lists each budget with a progress bar and edit and delete buttons, and the analytics panel beneath it offers filters for trip, budget, category, currency and date range and shows category-wise spending, budget against actual, and spending by date.

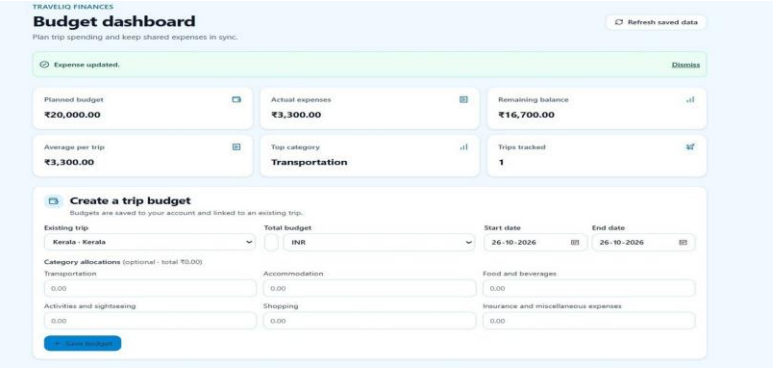

Fig. 6. Budget dashboard with summary cards and budget creation form.

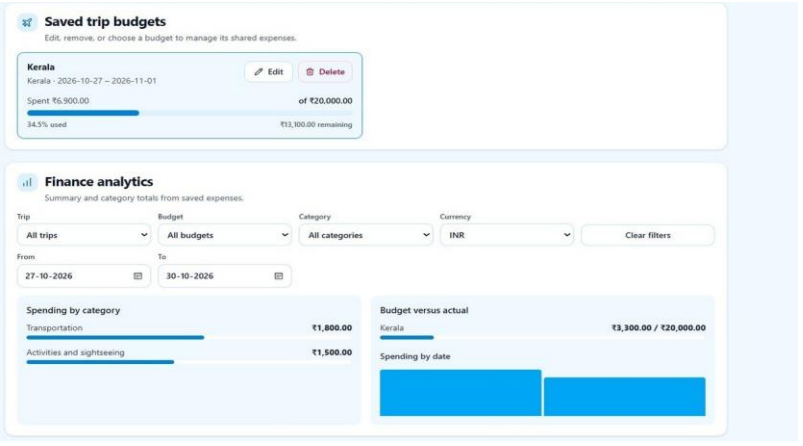

Fig. 7. Saved trip budgets and finance analytics.

Expense tracking and group settlement. The expense interface (Fig. 8) lets the user pick a trip budget and enter a description, amount, category, date, payer, split method and notes. Recorded expenses appear in a list with edit and delete icons. The panel beside the form manages the travellers of the selected trip and tabulates, for each of them, the amount contributed, the share and the net balance. A list of balances and settlements follows, with a note in the interface that settlements adjust balances and are not counted as expenses.

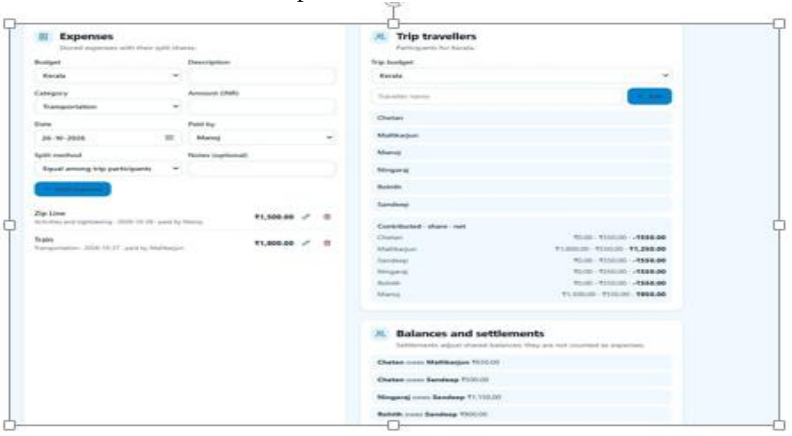

Fig. 8. Expense tracking, traveller management, and balances and settlements.

Functional Testing

Twelve test cases were run against the main features and two known limitations (Table 3). For each case the input, the expected result and the observed result were recorded. Ten cases, TC-1 to TC-10, passed and confirm that registration, trip planning, the assistant, budgeting, expense recording, settlement and analytics work on the inputs tried. The last two cases, TC-11 and TC-12, were included on purpose to expose how the system behaves in two situations that are known to need attention: when the weather forecast service has no data to return, and when the planning form is submitted with a field left empty. Reporting them gives a clearer picture of where improvement is needed. The test environment, the person who ran the tests and the number of repetitions are not documented, so the results show that the main functions work on the inputs tried rather than measuring robustness.

ID	Feature	Input	Observed result	Status
TC-1	Registration	Name, email, password	User registered successfully	Pass
TC-2	Login	Valid email and password	Login successful	Pass
TC-3	Travel information	Origin, destination, dates, budget	Details entered and processed	Pass
TC-4	AI travel assistant	Travel-related query	AI response shown in the chat	Pass
TC-5	Create trip budget	Name, budget, currency, dates	Budget created and displayed	Pass
TC-6	Budget allocation	Categories and amounts	Allocations shown under the chosen categories	Pass
TC-7	Expense records	Description, category, amount, date	Expense entered and listed	Pass
TC-8	Traveller management	Traveller details	Traveller information shown in the trip section	Pass
TC-9	Shared expense settlement	Contributions and shares	Balances and suggested settlements shown	Pass
TC-10	Finance analytics	Trip, category and date filters	Charts and summaries displayed	Pass
TC-11	Weather forecast	Destination and travel dates	Forecast is not always available; the service may return no data	Fail
TC-12	Incomplete trip form	Planning form with one field left empty	Itinerary is not generated until every field is filled in	Fail

Table 3: Functional test cases and outcomes

Observed Outputs and Consistency Checks

The screenshots document two demonstration scenarios. In the first, a couple travelling by train from Belagavi to Odisha for four days on a budget of 300 USD submit the planning form, receive a day-wise itinerary and ask the assistant for accommodation (Figs. 4 and 5). In the second, a budget of ₹20,000 is created for a trip named Kerala and two expenses of ₹1,500 and ₹1,800 are recorded (Figs. 6–8). The dashboard then shows actual expenses of ₹3,300 and a remaining balance of ₹16,700, which agree with the sum of the two expenses and with equation (3). The average per trip equals the actual expenses because one trip is tracked, as equation (4) predicts.

The settlement panel gives a second consistency check. With six travellers and ₹3,300 of shared expenses, the equal share is ₹550 each, in line with equation (1). The traveller who paid ₹1,800 has a net balance of ₹1,250, the one who paid ₹1,500 has ₹950, and the four who paid nothing each show −₹550. These values follow equation (2) and add up to zero. Two items were not reconciled. The suggested-settlement entries listed below the balance table in Fig. 8 have not been checked against those balances, and the "Spent" figure on the saved-budget card in Fig. 7 (₹6,900) differs from the ₹3,300 reported on the dashboard and in the category totals; we have not traced the cause. These checks confirm arithmetic consistency on demonstration data and are not performance measurements.

IV. DISCUSSION

The main strength of TravelIQ is that planning, assistance and money management work on the same trip record. A user can see the budget set for a trip next to the money actually spent, and a group can see each member's balance without a separate calculation. The interface also signals uncertainty where it exists, for instance in the unverified-fare note of the itinerary, which fits the concern about generated content raised in Section I.

The system has several limitations. It needs an internet connection and relies on an external AI service, so the quality and currency of its guidance depend on that service and on the preferences the user supplies. Estimated costs may differ from real prices, and AI-generated answers may be inaccurate or out of date and should be verified [5]; grounding the generated text in retrieved documents is one proposed way to reduce such errors [12]. The weather forecast is not always available because it depends on an external service, and the itinerary is generated only when every field of the planning form is filled in; showing

a fallback message and allowing optional fields would make both cases smoother. The planning form takes its budget in US dollars while the budget module works in a selectable currency, and the two are not yet harmonised. Because the application stores accounts and travel records it needs proper protection of user data; the security measures in place are not described here. Finally, the evaluation is limited to twelve functional checks on demonstration data, and the two unreconciled items noted in Section III still need to be resolved. Recommendation accuracy, itinerary feasibility, response time and usability were not measured, and we make no such claim.

V. CONCLUSION

This paper presented TravelIQ, a web application that brings trip planning, a conversational AI assistant and trip-level financial management together in one system built with React, FastAPI and MongoDB. A traveller enters the trip details once, receives a day-wise itinerary, can put follow-up questions to the assistant, and can then allocate a budget, record and split expenses, view balances with suggested settlements, and compare planned and actual spending on a dashboard.

By keeping these tasks on the same trip record, TravelIQ reduces the need to move between separate search tools, notes and spreadsheets, and it removes the manual calculation that a group trip usually needs at the end. Because the itinerary and the assistant's answers come from an external AI service, they are offered as guidance that the traveller should verify. TravelIQ therefore serves as a practical, integrated aid for planning a trip and managing its cost, particularly for travellers who journey in groups.

REFERENCES

1. F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*, 3rd ed. New York, NY, USA: Springer, 2022, doi: 10.1007/978-1-0716-2197-4.
2. J. Borràs, A. Moreno, and A. Valls, "Intelligent tourism recommender systems: A survey," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7370–7389, 2014, doi: 10.1016/j.eswa.2014.06.007.
3. D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," *Journal of Network and Computer Applications*, vol. 39, pp. 319–333, 2014.
4. J. Xie, K. Zhang, J. Chen, T. Zhu, R. Lou, Y. Tian, et al., "TravelPlanner: A benchmark for real-world planning with language agents," in *Proc. 41st Int. Conf. Machine Learning (ICML)*, PMLR 235, 2024, pp. 54590–54613.
5. Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, et al., "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, Art. 248, 2023.
6. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
7. Moreno, A. Valls, D. Isern, L. Marin, and J. Borràs, "SigTur/E-Destination: Ontology-based personalized recommendation of tourism and leisure activities," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 633–651, 2013, doi: 10.1016/j.engappai.2012.02.014.
8. D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014, doi: 10.1007/s10732-014-9242-5.
9. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.
10. F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.
11. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020, pp. 1877–1901.
12. P. Lewis, E. Perez, A. Piktus, F. Petroni, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020, pp. 9459–9474.
13. T. Verhoeff, "Settling multiple debts efficiently: An invitation to computing science," *Informatics in Education*, vol. 3, no. 1, pp. 105–126, 2004.
14. R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Information and Computer Science, Univ. California, Irvine, CA, USA, 2000.
15. .S. Few, *Information Dashboard Design: The Effective Visual Communication of Data*. Sebastopol, CA, USA: O'Reilly Media, 2006.