International Journal of Scientific Research in Engineering & Technology
Volume6, Issue6 (November- December2026), PP: 189-195.
https://www.doi.org/10.59256/ijsreat.20260606026
www.ijsreat.com
ISSN No: 2583-1240

Smart Trip: Smart Travel Ecosystem for Planning and Bookings

Sushma C[1], Basavakumar[2], Chetan[3], Rafiya Anjum[4], Sandesh Rajesh Avaradi[5]
[1]Assistant Professor, Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Vis vesvaraya Technological University, Karnataka, India.
[2,3,4,5] Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Visvesvaraya Technological University, Karnataka, India.

To Cite this Article: *Sushma C[1], Basavakumar[2], Chetan[3], Rafiya Anjum[4], Sandesh Rajesh Avaradi[5], "Smart Trip: Smart Travel Ecosystem for Planning and Bookings", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 06, November-December2026, PP:189-195.*

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 moving between booking portals, itinerary tools, messaging groups and shared-expense ledgers. This paper presents SmartTrip, a cross-platform mobile application that brings trip planning, transport and hotel booking, travel packages, group coordination and expense settlement into one system. A traveller enters the trip details once and receives a day-wise itinerary written by Google Gemini 2.0 Flash, can browse hotel, bus and train options and six themed packages, can form a travel group through a six-character invitation code, and can record expenses that the language model sorts into eight categories before a settlement routine works out who should pay whom. The system is built in layers: a React Native (Expo) client with Redux Toolkit state management, a service layer of JavaScript modules, Firebase Authentication for identity, a Supabase PostgreSQL database, and Gemini 2.0 Flash as an external AI service reached through one module. Fourteen functional test cases are documented: eleven passed, and three negative-path cases have no recorded outcome. The hotel, bus and train listings are sample data, no payment gateway is connected, and the evaluation is functional only. Itinerary quality, response time, scalability and user-study results are not reported, and the paper closes with its limitations and the evaluations that should follow.*

Keywords: *Travel planning; AI itinerary generation; Expense categorisation; Group bill settlement; React Native; Firebase; Supabase.*

I. INTRODUCTION

A journey is 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 to share the cost. Information technology has changed every one of these steps [1], and research on smart tourism now treats the traveller's phone as the main point of contact between people, places and service providers [2]. Tourism recommender systems have been surveyed extensively [3], [4], and mobile tourist guides add context such as location and time [5], building on the content-based, collaborative and hybrid families of recommendation methods [6].

Most of this work is about what to visit. The practical chores around a trip receive much less attention: reserving transport and rooms, keeping a group in step, and squaring accounts at the end. Travellers in India typically use one platform for booking, another for planning and a third for shared expenses, so the same destination, dates and budget are entered several times. Large language models can now write plans from free-text requests [7], yet a benchmark of travel queries with many interacting constraints found that even strong language agents seldom produced a fully valid plan [8], and generated text may contain statements that nothing supports [9]. A planner built on such a model should therefore offer its output as guidance that the traveller can check. Our reading of the literature was selective rather than systematic, so we make no claim about what other tools may offer.

This paper describes SmartTrip, a mobile application that keeps planning, booking, group coordination and expense settlement on one data model, so that the traveller enters trip data once. Our contributions are:

1. A layered architecture that connects a React Native client, a state-management layer, a service layer, managed authentication, a cloud relational database and an external AI service.
2. Eight functional modules covering authentication, trip management with AI itineraries, booking, travel packages, group management, expense tracking with AI categorisation, bill settlement and profile management, together with the processing logic behind them.
3. A functional check of the implemented features through fourteen documented test cases, supported by screenshots of the working interface.

Objectives:

- To build a unified cross-platform mobile application for trip planning and management using React Native and Expo.
- To use Google Gemini 2.0 Flash to write personalised day-wise itineraries, to propose trips on the home screen and to

assign a category to each expense from its description.

- To store users' trips, bookings, groups and expenses in a Supabase PostgreSQL backend, with sign-in handled by Firebase Authentication.
- To support group travel through invitation-code membership, shared expense records and an automatic calculation of who owes whom.

The rest of the paper is organised as follows. Section II explains the architecture, workflow, modules 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

SmartTrip keeps the user interface, application state, service logic, identity management and data storage apart (Fig. 1). The presentation layer consists of React Native [10] screens built with Expo [11] and reached through a bottom-tab navigator with six destinations: Home, Trips, Bookings, Packages, Groups and Profile. Redux Toolkit [12] holds shared state in five slices: authentication, trips, bookings, expenses and groups. A service layer of JavaScript modules communicates with outside systems; one module handles the AI service, and others handle trips, bookings, packages, groups and settlement. Firebase Authentication [13] verifies identity, and a Supabase [14] PostgreSQL [15] database persists trips, bookings, groups, members and expenses. Gemini 2.0 Flash [7] is reached only through the AI service module, so prompt construction stays in one place. As drawn in Fig. 1, the application modules connect directly to the Supabase database; the project does not use a separate application server of its own.

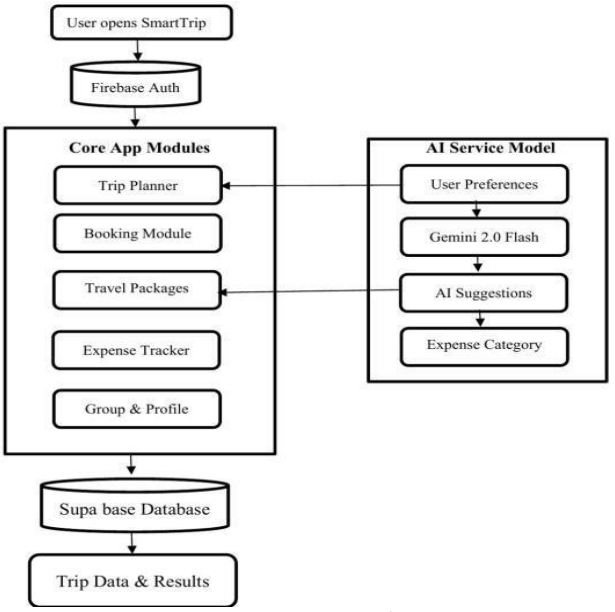


Fig. 1. System architecture of SmartTrip.

Workflow

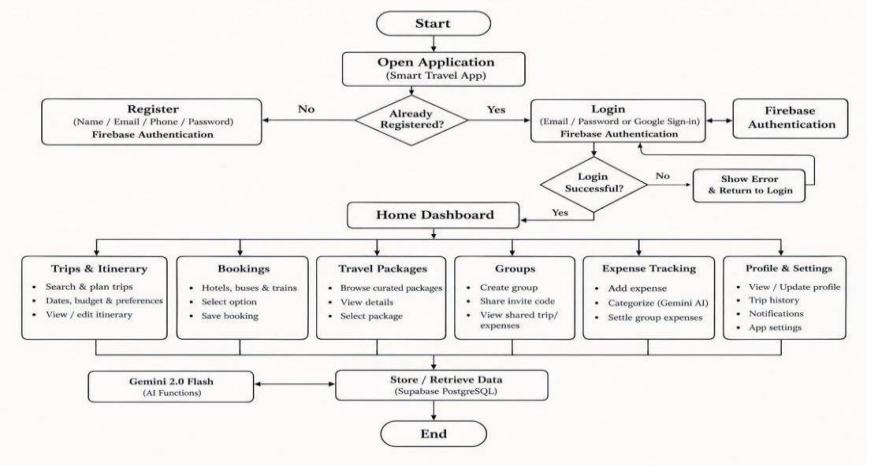


Fig. 2. Activity diagram of the SmartTrip workflow.

The overall workflow is shown in Fig. 2. When the application is opened, a new user registers and a returning user logs in; in both cases Firebase Authentication checks the credentials. Invalid details produce an error message and return the user to the login screen. After successful authentication the Home dashboard gives access to six modules: trips and itinerary, bookings, travel packages, groups, expense tracking, and profile and settings. Gemini 2.0 Flash is called for itinerary generation, trip suggestions and expense categorisation, and Supabase PostgreSQL stores and returns the records behind every module.

Fig. 3 gives the data-flow view. Five processes (user authentication, trip planning, booking and payments, group management and expense management) exchange data with the traveller and with five data stores: users (D1), trips (D2), bookings (D3), groups (D4) and expenses (D5). The booking process is labelled "Booking and Payments" in the project report, but the current implementation stores booking records only and does not process payments.

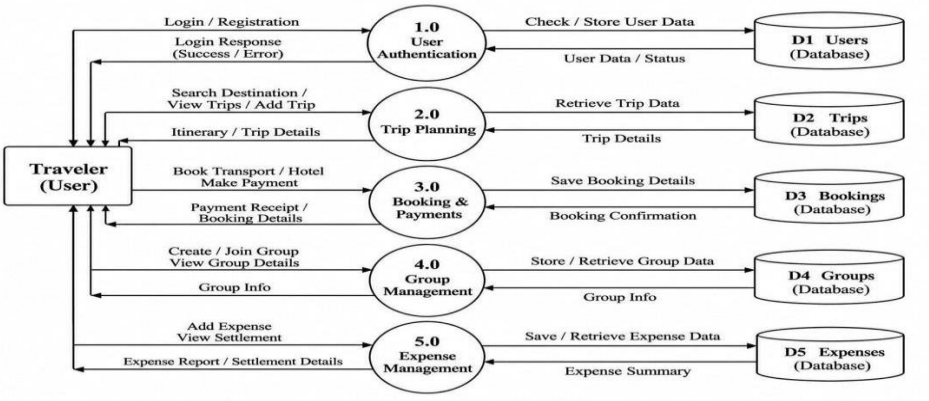


Fig. 3. Data flow diagram of Smart Trip.

Functional Modules

The functionality is divided into the eight modules summarised in Table 1. Each module takes its input from a screen and hands its output to the screen or to the database.

Module	Main inputs	Main outputs
Authentication	Name, email, phone and password, or a Google account	Verified session and Home dashboard
Trip management and AI itinerary	Destination, dates, budget, number of travellers, trip type, transport mode	Stored trip and a day-wise itinerary with morning, afternoon and evening activities
Booking	Source, destination and date; chosen option; for hotels, check-in and check-out dates, rooms and guests	Confirmed booking record; booking list with a cancel action
Travel packages	Package category and selected package	Package details for six categories
Group management	Group name; invitation code	Group with a generated code; membership records
Expense tracking and AI categorisation	Title, amount, date, optional category, payer, members in the split	Stored expense labelled with one of eight categories
Bill settlement	Expense records of a group	Net balance per member and suggested payments
Profile management	Profile edits; sign-out request	Updated profile; ended session

Table 1: Functional modules of Smart Trip

Technology Stack

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

Component	Technology	Role
Client	React Native 0.81.5 with Expo SDK 54	Cross-platform screens, forms and navigation
State management	Redux Toolkit	Five slices shared across screens
Authentication	Firebase Authentication	Email-and-password and Google sign-in, persistent sessions
Secure storage	Expo Secure Store	Supabase session tokens kept on the device
Database	Supabase (PostgreSQL)	Users, trips, bookings, groups and expenses
External AI service	Google Gemini 2.0 Flash	Itineraries, trip suggestions and expense categories
Tooling	Node.js, Visual Studio Code	Development and package management

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

Processing Logic

Authentication. A new user enters a name, email, phone number and password, and the form checks password strength as the user types. Credentials are verified by Firebase, or the user may choose Continue with Google. On success Firebase returns a session and an ID token, the app stores the user in the Redux store and opens the Home screen, and Firebase keeps the session across restarts.

Trip planning and itinerary. The planning form collects destination, dates, budget, number of travellers, trip type and transport mode. The app builds a prompt from these values and sends it to Gemini 2.0 Flash, which returns a plan with morning, afternoon and evening activities for each day. The itinerary screen requests this plan automatically when it is opened, and the trip is saved in the Supabase trips table. The Home screen also requests trip suggestions from the model on every launch.

Booking and packages. The user searches hotels, buses or trains, selects an option and confirms. Hotel bookings carry check-in and check-out dates, rooms and guests; bus and train options show operator, seat availability and class. Confirmed bookings are saved in the bookings table and can be cancelled, which changes the stored status to cancelled. Six curated packages (Beach, Mountain, Nature, Heritage, Adventure and Pilgrimage) can be filtered by category and opened for full details. All hotel, bus, train and package data in the current version are built-in samples.

Group management. Creating a group generates a unique six-character invitation code. Another user joins by entering the code, which the app verifies; an invalid code is rejected and no membership is created.

Expense recording and AI categorisation. Each expense carries a title, amount, date, payer and, for shared expenses, the members included in the split. When no category is chosen, the description is sent to the model, which labels the expense as Food, Transport, Stay, Activities, Shopping, Medical, Fuel or Others. The labelled record is stored in the expenses table.

Shared-expense settlement. Let ae be the amount of expense e, Pe the set of members who share it, and ci the total paid by member i. The share of member i, summed over the expenses in which i takes part, and the resulting net balance are

$$si = \Sigma\, ae / |Pe| \quad (1)$$

$$bi = ci - si \quad (2)$$

A positive b_i mark a member who should receive money and a negative value one who owes money. Because every expense is paid once and shared in full, Σci and Σsi both equal the total of all expenses, so the balances add up to zero; this serves as a quick check that the recorded payers and amounts are consistent. The routine then separates debtors from creditors and repeatedly pairs a debtor with a creditor, recording a payment equal to the smaller of the two outstanding amounts. Each payment clears at least one member, so at most n − 1 payments are needed for n members with non-zero balances [16]. This reduces the number of transactions but does not guarantee the absolute minimum, because finding the true minimum is a harder problem [16]. A settlement only moves balances between members; it does not create a new expense.

III. RESULT

Snapshots

Welcome, registration and home screens. The welcome screen (Fig. 4a) introduces the application and offers Get Started for new users and "I already have an account" for existing ones. The registration screen (Fig. 4b) collects name, email, phone number and password and offers Continue with Google as an alternative. The Home screen (Fig. 4c) provides a search box, shortcuts to hotels, buses, trains and packages, quick actions for group trips, expenses and AI planning, and a scrolling row of trending destinations with starting prices.

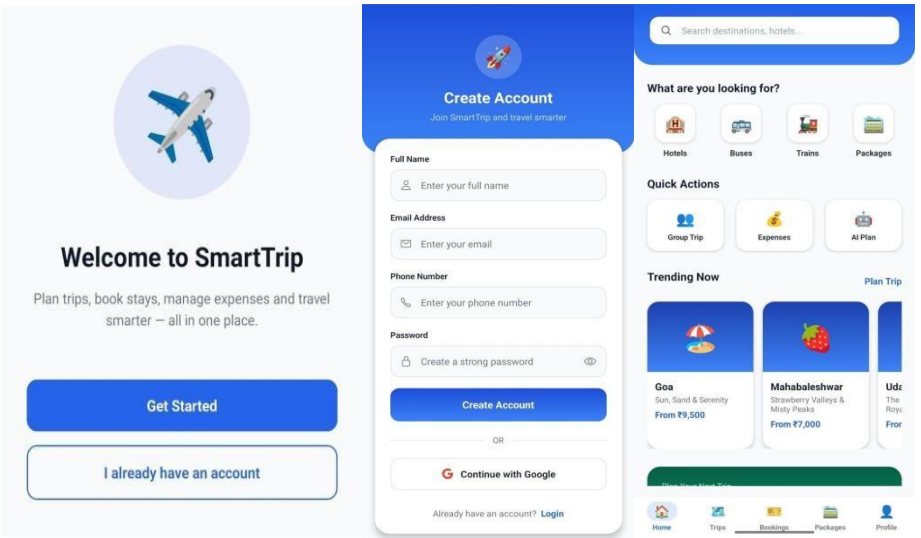


Fig. 4. (a) Welcome screen. (b) Registration with Google option. (c) Home dashboard.

Travel packages. The package screen (Fig. 5) reports the number of packages found and offers sorting by trending, top rated and price. A row of AI picks suggests packages for the user's interests, and each offer shows destination, duration, review count, inclusions, a discount badge and a discounted price per person.

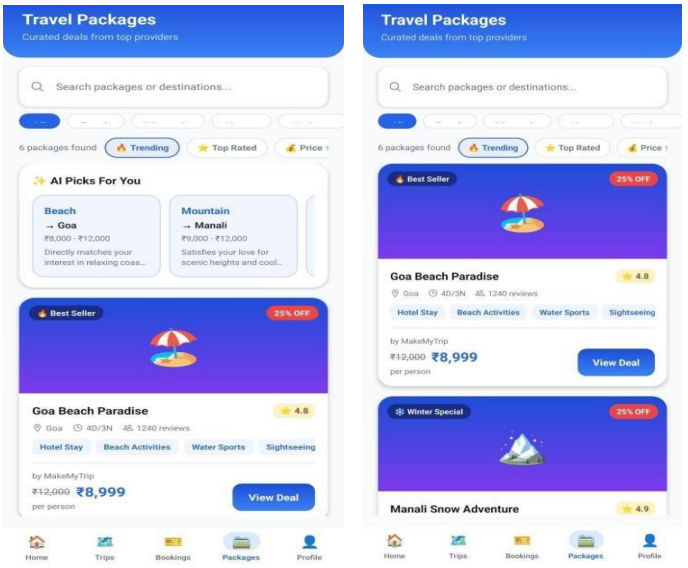


Fig. 5. Travel package selection with AI picks and package details.

Train, bus and hotel booking. The train screen (Fig. 6a) lists train names and numbers, stations, departure and arrival times, duration, running days and fares by class. The bus screen (Fig. 6b) shows operator, bus category, route, timings, seats left and price per seat. The hotel screen (Fig. 6c) shows ratings, amenities, discounts and nightly rates. These screens are filled from sample data, so they demonstrate the interface and data flow and not live inventory.

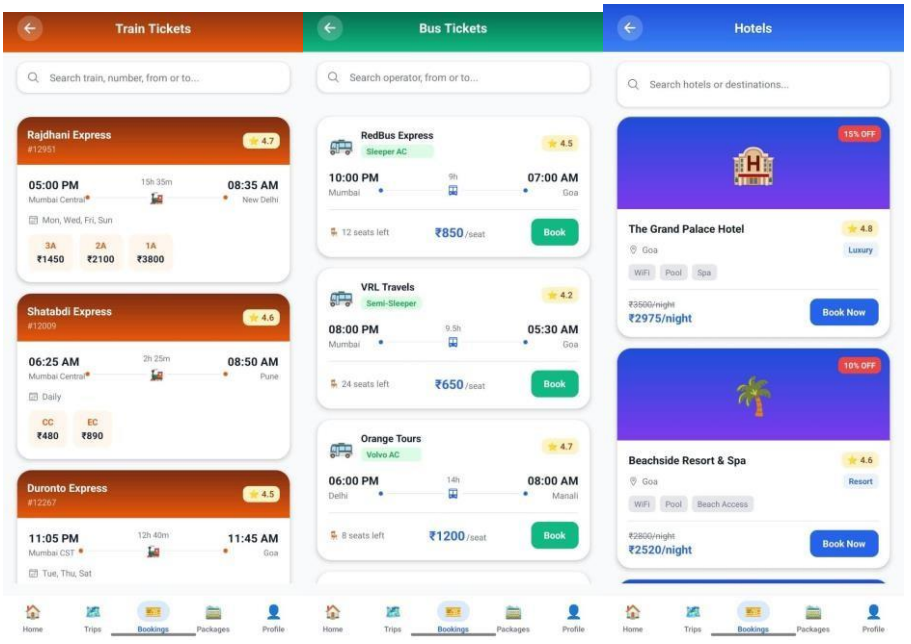


Fig. 6. (a) Train tickets. (b) Bus tickets. (c) Hotel booking.

Functional Testing

Fourteen test cases are documented for the main features and for three error situations (Table 3). The first eleven cases were run with the input shown and passed; they confirm that authentication, itinerary generation, booking search, group joining, AI categorisation and settlement work on the inputs tried. The last three cases, TC-12 to TC-14, define the expected behaviour for an unregistered email, an unavailable AI service and a booking search with missing details, but the actual application output was never recorded. They are therefore reported as not recorded and are not counted as passes. 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 and do not measure robustness.

ID	Feature	Input	Observed / expected result	Status
TC-1	Login, empty fields	Blank email and password	Error message shown	Pass
TC-2	Login, valid user	Registered email and password	User signed in, Home opens	Pass
TC-3	Login, wrong password	Valid email, wrong password	Error shown, login denied	Pass
TC-4	Registration	Name, email and password	Account created, Home shown	Pass
TC-5	Continue with Google	Tap Google sign-in	Signed in, Home displayed	Pass
TC-6	AI itinerary	Goa, 3 days, ₹15,000	Day-wise itinerary displayed	Pass
TC-7	Booking search	Source, destination and date	Options listed from sample data	Pass
TC-8	Join group, valid code	Correct invitation code	Member added to group	Pass
TC-9	Join group, invalid code	Incorrect invitation code	Error shown, no membership created	Pass
TC-10	AI expense category	Dinner at restaurant, ₹1,200	Categorised as Food	Pass
TC-11	Group bill settlement	A pays ₹900; B and C owe ₹300 each	Two payments of ₹300 shown	Pass
TC-12	Login, unregistered email	Unregistered email and password	Login denied with an error (expected)	Not recorded
TC-13	Itinerary, AI unavailable	Valid trip details, AI service down	Appropriate error shown (expected)	Not recorded
TC-14	Booking search, missing details	Blank source or destination	Missing details requested (expected)	Not recorded

Table 3: Functional test cases and outcomes

Observed Outputs and Consistency Checks

The screenshots and test cases allow a few arithmetic checks. In TC-11 member A pays ₹900 for an expense shared equally by A, B and C. By equation (1) each share is ₹300, and by equation (2) the balances are +₹600 for A and −₹300 for each of B and C. They add up to zero, and two payments of ₹300, from B to A and from C to A, settle the account, which agrees with the two payments recorded in the test. The package and hotel screens show a second consistency check on the displayed discounts: the Goa Beach Paradise package is listed at ₹12,000 with 25% off and shown at ₹8,999, against $0.75 \times 12{,}000 = ₹9{,}000$; the Grand Palace Hotel falls from ₹3,500 to ₹2,975 per night at 15% off; and Beachside Resort falls from ₹2,800 to ₹2,520 at 10% off. All three agree with the stated percentages

One item was not reconciled. The report cites seven relational tables in one place and five logical data stores (D1 to D5) in its data-flow diagram, and we have not traced how the two descriptions map onto each other. These checks confirm arithmetic consistency on demonstration data and are not performance measurements.

IV. DISCUSSION

The main strength of SmartTrip is that planning, booking, group coordination and expense handling work on the same set of records. A user can generate an itinerary, reserve transport and rooms, form a group and settle the costs without leaving the application, and the modular, feature-based code structure makes it easier to replace one part, for example the source of booking data. Firebase keeps users signed in across restarts, and cloud storage lets them return to saved trips and bookings.

The system has several limitations. Hotel, bus, train and package options are sample data, there is no payment processing, and live availability and real-time updates are not provided. Most features need an internet connection, and the AI features depend on the availability, delays and usage limits of the Gemini service. Generated itineraries and categories may be inaccurate or out of date and should be checked before anyone relies on them [9]. The documentation does not say what happens when the model returns something other than one of the eight category labels, how the AI service key is protected on the device, or what access rules protect each user's data in Supabase, so privacy and security cannot be assessed from it. The settlement routine reduces the number of payments but is not guaranteed to find the minimum. Finally, the evaluation is limited to fourteen functional checks, three of which have no recorded outcome, so behaviour when the AI service fails, when an account does not exist or when a form is incomplete remains unverified. Itinerary quality, response time, scalability and usability were not measured, and we make no such claim.

V. CONCLUSION

This paper presented SmartTrip, a mobile application that brings trip planning, AI-written itineraries, transport and hotel search, travel packages, group formation, expense tracking with automatic categorisation and bill settlement together in one system built with React Native, Firebase Authentication and Supabase PostgreSQL. A traveller enters the trip details once, receives a day-wise itinerary, and can then form a group, record expenses and see who should pay whom. Eleven of the fourteen documented functional tests passed, which supports the correct operation of the main flows. Because the itineraries and expense labels come from an external AI service, they are offered as guidance that the traveller should verify.

The next steps are to record the outcomes of the three pending negative tests, add boundary and automated regression tests, measure response times and run a user study comparing planning effort with and without the application. On the functional side, live booking interfaces, digital payment gateways, live route tracking, location sharing and real-time group communication are natural extensions, and grounding the model's suggestions in verified travel data and reviewing the access rules on stored user data would improve reliability and privacy.

REFERENCES

1. D. Buhalis and R. Law, "Progress in information technology and tourism management: 20 years on and 10 years after the Internet—The state of eTourism research," Tourism Management, vol. 29, no. 4, pp. 609–623, 2008, doi: 10.1016/j.tourman.2008.01.005.
2. U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, "Smart tourism: Foundations and developments," Electronic Markets, vol. 25, no. 3, pp. 179–188, 2015, doi: 10.1007/s12525-015-0196-8.
3. 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.
4. 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.
5. 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.
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. Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023.
8. 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.
9. 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.
10. Meta Platforms, "React Native documentation." [Online]. Available: https://reactnative.dev/docs/getting-started (accessed Nov. 14, 2026).
11. Expo, "Expo documentation." [Online]. Available: https://docs.expo.dev (accessed Nov. 14, 2026).
12. Redux Toolkit Team, "Redux Toolkit documentation." [Online]. Available: https://redux-toolkit.js.org (accessed Nov. 14, 2026).
13. Google, "Firebase Authentication documentation." [Online]. Available: https://firebase.google.com/docs/auth (accessed Nov. 14, 2026).
14. Supabase, "Supabase documentation." [Online]. Available: https://supabase.com/docs (accessed Nov. 14, 2026).
15. PostgreSQL Global Development Group, "PostgreSQL documentation." [Online]. Available: https://www.postgresql.org/docs/ (accessed Nov. 14, 2026).
16. T. Verhoeff, "Settling multiple debts efficiently: An invitation to computing science," Informatics in Education, vol. 3, no. 1,pp. 105–126, 2004.