Archives available at journals.mriindia.com

International Journal on Advanced Computer Theory and Engineering

ISSN: 2319-2526

Volume 15 Issue 01, 2026

TripMind: An Agentic AI-Based System for End-to-End Travel Planning

[1]Nigar Sayyed, [2]Sanika Rane, [3]Bindu Yadav, [4]Suhani Tiwari, [5]Shraddha Sharma
[1,2,3,4] Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai
[5] Assistant Professor, Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai
Email: [1]nigar.f.sayyed@slrtce.in, [2]sanika.s.rane@slrtce.in, [3]bindu.p.yadav@slrtce.in, [4]suhani.d.tiwari@slrtce.in, [5]shraddha.sharma@slrtce.in

Peer Review Information

Submission: 19 March 2026

Revision: 08 April 2026

Acceptance: 24 April 2026

Keywords

Agentic AI, End-To-End Travel Planning, Multi-Agent Systems, Itinerary Generation, Conversational AI, Budget-Aware Planning, Decision Support.

Abstract

A trip usually means moving back and forth between multiple platforms for destination research, cost estimation, flight comparison, hotel search, weather checking, and itinerary preparation. While conversational assistants can suggest places or activities, they often stop short of turning user constraints into a usable trip plan. TripMind addresses this gap as an agentic AI-based system for end-to-end travel planning. It accepts free-form travel requests and turns them into structured outputs through a coordinated multi- agent pipeline. Separate agents handle budget feasibility, flight estimation, hotel planning, weather interpretation, itinerary generation, and final travel advice, while a Spring Boot backend manages session state, orchestration, persistence, and provider integration. The system also exposes planning-intelligence signals such as booking readiness, source reliability, and live-data coverage so that users can better judge the quality of the generated plan. The final response includes a feasibility verdict, cost breakdown, contextual recommendations, a day-by-day itinerary, and booking links populated with trip parameters. The work illustrates how an agentic architecture can support more grounded and practically useful travel planning than a generic conversational travel assistant.

Introduction

Travel planning is not a single decision but a chain of related decisions. A user typically has to compare destinations, estimate the total cost, check transport options, evaluate accommodation, consider weather or season, and finally organize activities into a workable schedule. Even with digital booking platforms and conversational tools, this process often remains fragmented. Information is available, but users still have to piece it together on their own before deciding whether a trip is realistic.

Recent work on travel-oriented language agents shows why this problem is difficult. Real-world travel planning requires constraint tracking, long-horizon reasoning, external information use, and plan consistency across multiple user conditions such as dates, budget, origin, and traveler count [1], [2]. A travel assistant that produces fluent text is not necessarily useful if it cannot preserve those constraints in a reliable way. In practice, users need more than suggestions; they need a plan they can evaluate and act on.

Research in reasoning-and-action frameworks and retrieval-grounded generation has also shown that language-model systems become more effective when generation is supported by structured steps, tools, or external context [4], [5]. That observation is especially relevant in travel planning, where one large response isoften less useful than a workflow composed of smaller planning decisions. Budget feasibility, flight estimation, hotel strategy, weather context, and

itinerary generation are related tasks, but they are not identical tasks.

This paper presents TripMind, an agentic AI-based system for end-to-end travel planning. The system converts free-form user requests into structured trip plans through a coordinated multi-agent workflow. Instead of producing only descriptive travel text, TripMind generates a feasibility verdict, cost breakdown, booking-oriented guidance, and a day-by- day itinerary while preserving key trip constraints. The design also includes ranking and fallback mechanisms so that weak or repetitive provider outputs do not directly degrade the final response [6]–[10].

The main contribution of this work is a practical planning framework that combines conversational input with structured backend reasoning. TripMind is intended to help users move from vague travel intent to a more grounded decision. To support that goal, the system also surfaces planning-intelligence metadata such as booking readiness, source reliability, and live- data coverage, allowing users to interpret the output more realistically.

Related Work

Research related to TripMind can be broadly grouped into four directions: travel planning with language agents, multi-agent approaches for trip generation, reasoning-and-tool-use frameworks for intelligent agents, and retrieval-grounded response generation.

Recent work has shown that travel planning is an especially challenging real-world task for language- based systems. The TravelPlanner benchmark formalized this difficulty by demonstrating that travel planning requires language agents to satisfy multiple simultaneous constraints while gathering and organizing information over long planning horizons [1]. The benchmark showed that current agents often struggle with constraint tracking, tool usage, and overall plan validity, even when strong large language models are used. This finding is directly relevant to TripMind, since one of the primary design goals of the proposed system is to avoid treating travel planning as unrestricted text generation and instead structure the workflow through specialized planning modules.

In parallel, domain-specific AI systems for personalized travel planning have also begun to emerge. TravelAgent proposed a travel-oriented intelligent assistant that combines planning, recommendation, memory, and tool-usage modules to improve personalization and comprehensiveness [2]. This line of work highlights the value of modularity in travel-oriented conversational systems. However, such systems are primarily framed as assistant-style planners, whereas TripMind places stronger emphasis on budget feasibility, booking-path generation, source transparency, and provider fallback behavior as first-class parts of the planning pipeline.

Earlier work on multi-agent travel planning also supports the decomposition of trip planning into interacting specialist components. Camacho et al. proposed an intelligent travel planning system in which multiple autonomous agents cooperatively search, filter, and reconstruct distributed travel information into candidate solutions [3]. Although this work predates modern large language models, it remains important because it established travel planning as a coordination problem suited to multi-agent design. TripMind extends this broader idea into a contemporary agentic AI setting by combining specialized planning agents with conversational interaction, backend orchestration, and provider integration.

Beyond the travel domain, recent AI research has emphasized the importance of combining reasoning with external actions and knowledge access. ReAct demonstrated that coupling reasoning traces with tool- based actions improves both interpretability and performance in decision-oriented tasks [4]. Likewise, retrieval-augmented generation has been shown to improve factual grounding and reduce unsupported responses in knowledge-intensive tasks by combining parametric language models with retrieved context [5]. These ideas are reflected in TripMind's use of coordinated planning modules, external provider integration, and grounded response assembly.

System Architecture

The proposed system, referred to as TripMind, is an agentic AI-based system for end-to-end travel planning. As shown in Fig. 1, the system follows a modular client-server architecture composed of four major layers:

(1) a user interaction layer, (2) a session and orchestration layer, (3) a specialized multi-agent planning layer, and (4) a provider and persistence layer. This design enables TripMind to move beyond conventional chatbot-style interaction by combining conversational input handling with structured backend planning logic and provider-assisted travel intelligence.

1. User Interaction Layer

The user interacts with the system through a web-based interface that supports conversational trip specification and plan inspection. The interface accepts natural-language inputs such as destination, origin, number of travelers, budget, and travel dates.

Unlike fixed-form travel search systems, the front end is designed to support incremental and noisy user input, allowing the trip state to be built progressively across multiple chat turns. Once the system has captured sufficient planning information, the same interface presents the generated plan, including the feasibility verdict, cost breakdown, planning intelligence, booking paths, and itinerary.

2. Session and Orchestration Layer

The backend is implemented using Spring Boot, which exposes REST endpoints for chat-based planning, plan generation, replanning, trace retrieval, and feedback persistence [6]. A session management component maintains the evolving trip state across user interactions, including destination, origin, travel dates, traveler count, budget, and preference signals. This layer is responsible for normalizing user input, extracting structured constraints from conversational text, and determining whether the trip request is complete enough to trigger the planning workflow.

Once the session reaches a planning-ready state, the orchestration component creates a structured trip request and dispatches it through the planning pipeline. The orchestration layer also aggregates outputs from downstream agents, combines them into a unified trip response, and exposes planning intelligence such as booking readiness, source reliability, and live-data coverage.

3. Multi-Agent Planning Layer

TripMind adopts a specialized multi-agent architecture in which each agent is responsible for one bounded planning function. The budget agent performs initial feasibility analysis and allocates category-level spend lanes for flights, accommodation, activities, and contingency. The flight agent produces route-aware airfare estimates and generates booking links that preserve user trip parameters. The hotel agent derives stay strategy and lodging estimates from trip duration, destination, and budget allocation. The weather agent adds contextual interpretation of the travel window using either provider data or season-aware fallback reasoning. The trip advisor agent synthesizes the outputs of earlier agents into a higher-level verdict and recommendation layer. Finally, the itinerary agent generates a day-by-day travel plan by combining destination context, budget limits, weather signals, and discovered places.

4. Provider Integration and Fallback Layer

TripMind integrates multiple external providers to enrich planning quality. Geocoding and place discovery are supported through Geoapify and related points-of- interest sources [8], while weather context is obtained through OpenWeather when available [9]. Booking support is implemented through generated handoff links for external travel platforms, allowing users to continue from planning to live search without manually re- entering trip parameters. For flight and hotel planning, the current system uses a centralized pricing engine for market-level estimation and then redirects users to external booking providers for final live availability and fare confirmation [10], [11].

Because provider quality may vary across destinations, a ranking and fallback layer is used before results are exposed to the user. Low-quality or repetitive place outputs are filtered, re-ranked, or replaced with area- based itinerary anchors to prevent overly narrow or misleading plans. Similarly, if live weather coverage is unavailable, the system falls back to a season-aware heuristic interpretation rather than producing empty or unreliable output.

5. Persistence and Traceability

TripMind persists session and plan information to support replanning, traceability, and feedback collection. Session-level state is stored to preserve planning context across interactions, while generated plan summaries and user feedback can be retained for later analysis. In addition, the architecture supports event-style workflow tracing through Redis Streams to record agent progress and stage-level completion [7]. This makes the system easier to debug, more transparent during demonstrations, and better aligned with multi-stage planning workflows.

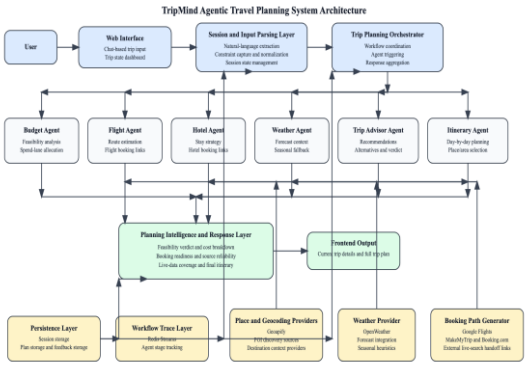

Fig 1: System architecture of the proposed TripMind end-to-end travel planning system.

As shown in Fig. 1, TripMind follows a modular client- server architecture in which user trip intent is progressively transformed into a

structured planning response. The workflow begins at the Web Interface, where the user provides free-form travel input such as destination, origin, traveler count, budget, and travel dates. This input is forwarded to the Session and Input Parsing Layer, which extracts structured constraints from conversational text, normalizes noisy user input, and maintains an evolving trip-state snapshot across multiple interactions.

Once the required constraints are available, the structured trip request is passed to the Trip Planning Orchestrator. This component acts as the central coordination layer of the system. Its role is to trigger the specialized planning agents, manage the sequence of execution, and aggregate their intermediate outputs into a unified final response. Instead of relying on a single model to produce the entire travel plan, TripMind distributes the planning problem across multiple bounded agents.

The Budget Agent serves as the feasibility anchor of the workflow by determining whether the trip is financially practical and by allocating category-level spend lanes for flights, accommodation, activities, and contingency. The Flight Agent uses the structured trip request to generate a route-aware flight estimate and booking-link context. The Hotel Agent derives stay strategy and accommodation estimates from the destination, trip duration, and allocated hotel budget. The Weather Agent contributes either forecast-based or season-aware travel context, while the Trip Advisor Agent synthesizes recommendations, alternatives, and the overall verdict. Finally, the Itinerary Agent generates a day-by-day plan using destination context, budget signals, weather interpretation, and retrieved place information.

These agent outputs are collected by the Planning Intelligence and Response Layer, which combines them into a structured user-facing plan. This layer produces the final feasibility verdict, cost breakdown, itinerary, and planning intelligence indicators such as booking readiness, source reliability, and live-data coverage. The aggregated response is then returned to the frontend, where it is displayed as the user's current trip plan.

The lower support layer of Fig. 1 represents the system's external integrations and infrastructure dependencies. The Persistence Layer stores session information, plan data, and feedback records. The Workflow Trace Layer, implemented through Redis Streams, supports agent-stage traceability and event- style monitoring. The Place and Geocoding Providers enrich the hotel and itinerary modules with destination- aware location data, while the Weather Provider supplies environmental context to the weather module. The Booking Path Generator creates outbound travel- provider links so that users can move from planning to live search using their trip parameters without re-entering the same details manually.

Overall, Fig. 1 illustrates that TripMind is not implemented as a standalone chatbot, but as an agentic planning system in which conversational input, structured backend reasoning, provider-informed enrichment, and response aggregation work together to generate a practical travel-planning output.

Methodology

The methodology of TripMind is designed to convert free-form travel intent into a structured planning output through a multi-stage backend workflow. The complete process includes conversational input acquisition, session-state construction, planning orchestration, specialized agent execution, provider-informed enrichment, and response aggregation. The methodology emphasizes modularity, transparency, and robustness so that the system can support real-world travel planning despite noisy user input and varying provider quality.

1. Conversational Input Processing

TripMind begins by accepting travel requests through a chat-oriented web interface. Instead of requiring the user to fill out a rigid form, the system allows travel details to be provided incrementally through natural language. The input processing layer extracts core planning fields such as destination, origin, number of travelers, travel dates, budget, and optional preference signals including trip style or special requests. Because real users often provide incomplete or noisy input, the parser is designed to normalize spelling variations, resolve partial constraints, and maintain a continuously updated session snapshot.

2. Session State and Constraint Formation

After extraction, the captured trip information is stored as a structured session object. This object acts as the central constraint container for the system and includes the user's route, travel window, traveler count, budget, and preference signals. The session state is continuously updated during interaction and is also used to determine whether the system should remain in a clarification phase or trigger plan generation. In this methodology, budget, dates, and traveler count are treated as hard planning constraints, while vibe and special-request information are treated as soft guidance signals.

3. Multi-Agent Planning Workflow

Once the session reaches a planning-ready state, the orchestrator dispatches the request to a set of specialized agents. The budget agent performs initial feasibility analysis and allocates category-level spend lanes. The flight agent generates planning-level airfare estimates and booking-path context. The hotel agent derives stay strategy and accommodation estimates. The weather agent contributes either live or heuristic seasonal context. The trip advisor agent synthesizes the outputs of earlier agents into a practical verdict and recommendation layer. The itinerary agent then constructs a day-by-day travel plan by combining destination context, budget constraints, weather signals, and place-discovery results.

4. Provider Integration and Data Grounding

To improve realism, the methodology integrates external providers for places, weather, and booking- path generation. Place discovery is supported through location and POI services [8], while weather interpretation is enriched through forecast APIs when available [9]. Booking functionality is implemented through provider-specific search URLs that embed the route, dates, and traveler count into external booking pages [10], [11].

However, third-party outputs are not accepted blindly. Provider data is first passed through a filtering and ranking stage. Weak, repetitive, or destination- inappropriate place results are penalized, and when the retrieved place set is too noisy, the itinerary generator switches to an area-based planning mode. This prevents the system from presenting overly narrow or misleading itineraries.

5. Response Aggregation and Planning Intelligence

After agent execution, the orchestration layer aggregates all intermediate outputs into a unified response object. The final response contains the trip feasibility verdict, cost breakdown, route summary, hotel strategy, weather context, recommendations, and a day-by-day itinerary. In addition, TripMind generates a planning-intelligence layer containing metadata such as booking readiness, source reliability, live-data coverage, and trip archetype. This information helps users interpret the generated plan more realistically, especially in cases where estimates are being used in place of live supplier inventory.

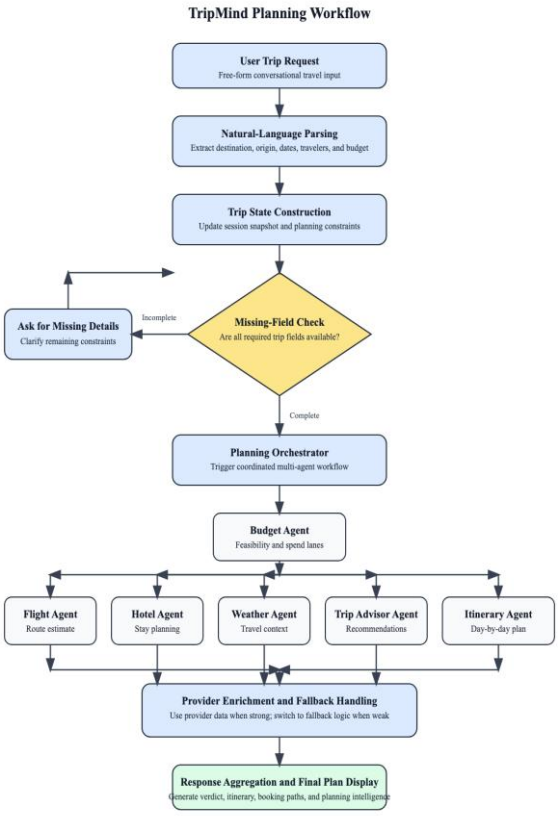

Fig 2: Workflow of the proposed TripMind end-to-end travel planning pipeline.

As shown in Fig. 2, the TripMind methodology begins when the user submits a travel request through the conversational interface. The first stage of the workflow is natural-language parsing, where the system extracts structured trip attributes such as destination, origin, travel dates, traveler count, budget, and optional preference signals. These extracted elements are then stored in the trip state construction stage, where the session snapshot is updated and maintained across multiple conversational turns. Once the trip state is updated, the workflow enters a constraint completeness check. At this point, the system determines whether sufficient information has been collected to begin travel planning. If one or more mandatory fields are missing, TripMind does not trigger the planning workflow; instead, it returns to the user with a clarification request asking only for the remaining required details. This step ensures that the system does not generate an under-specified or invalid travel plan.

When all required trip constraints are available, the request is passed to the planning orchestrator, which activates the multi-agent workflow. The budget agent executes first because financial feasibility acts as the anchor for downstream planning. The output of the budget agent is then used by the other planning agents,

including the flight agent, hotel agent, weather agent, trip advisor agent, and itinerary agent. This sequencing ensures that later planning decisions remain consistent with the trip's financial constraints.

During agent execution, TripMind also performs provider enrichment and fallback handling. External provider data such as places, weather information, and booking-path parameters are incorporated wherever available. If provider outputs are weak, repetitive, or incomplete, fallback logic is applied before the final plan is assembled. This may include season-aware weather interpretation or area-based itinerary generation in place of low-quality place-specific planning.

After all agent outputs are produced, the system enters the response aggregation stage. Here, intermediate results are combined into a unified travel response containing a feasibility verdict, cost breakdown, route and stay planning, contextual recommendations, and a day-by-day itinerary. In parallel, the system generates a planning intelligence layer, which summarizes metadata such as booking readiness, source reliability, and live- data coverage. The final response is then returned to the frontend and displayed to the user as a structured trip plan.

The final step of the workflow supports user-driven replanning and refinement. If the user changes the destination, budget, dates, or any other trip constraint, the workflow can be re-triggered using the updated session state. Thus, Fig. 2 represents TripMind not as a one-shot query system, but as an iterative planning pipeline capable of handling progressive refinement under real-world travel constraints.

Table 1: Agent Roles In TripMind

Agent	Primary Input	Primary Output	Responsibility
Budget Agent	Budget, dates, travelers, destination	Feasibility and spend lanes	Determines whether the trip is practical within financial constraints.
Flight Agent	Route, dates, travelers, budget lane	Flight estimate and booking links	Generates route-aware flight planning and cost estimation.
Hotel Agent	Destination, nights, hotel budget lane	Stay estimate and hotel links	Suggests stay strategy based on location and budget.
Weather Agent	Destination and travel dates	Weather/season context	Adds environmental planning context and packing advice.
Trip Advisor Agent	Outputs of prior agents	Recommendations and alternatives	Produces a final planning verdict and quality assessment.
Itinerary Agent	Destination context, budget, weather	Day-by-day itinerary	Builds the final trip structure and chronological activity list.

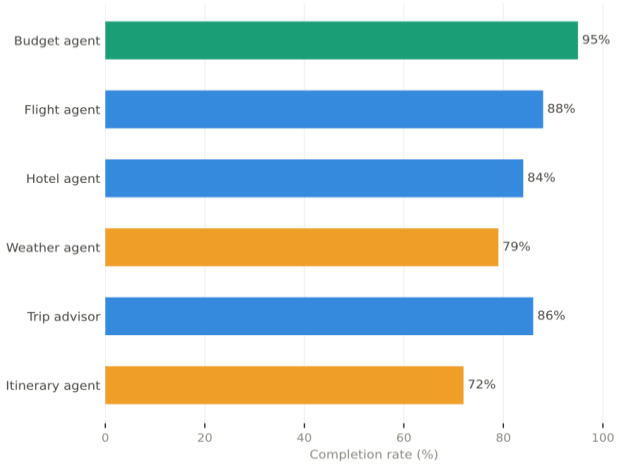

Fig 3: Agent Completion rate

Table 2: Technology Stack Used in TripMind

Component	Technology Used	Purpose
User Interface	Web-based frontend	Conversational trip input, trip-state display, and generated plan presentation.
Backend Framework	Spring Boot	REST API development, orchestration, session management, and service modularization.

Persistence Layer	JPA-backed storage	Session persistence, generated plan storage, and feedback retention.
Workflow Trace Layer	Redis Streams	Agent-stage tracking and event-style workflow tracing.
Place and Geocoding Layer	Geoapify and connected POI sources	Destination geocoding, place discovery, and itinerary enrichment.
Weather Layer	OpenWeather with seasonal fallback logic	Forecast-based or heuristic travel-context generation.
AI Layer	Configurable LLM integration	Conversational response polishing and selected agent-level enrichment.
Booking Handoff Layer	External booking-link generation	Redirecting users to live flight and hotel search pages with populated trip context.

Experimental Setup

The experimental setup for TripMind was designed to validate the system as an end-to-end travel-planning platform rather than as an isolated language-generation model. The focus of the setup was to verify that the implemented architecture could accept natural-language user trip requests, maintain session state, trigger the multi-agent planning workflow, integrate provider- backed travel context, and generate structured planning outputs.

1. Development and Execution Environment

TripMind was implemented as a web-based full-stack application. The backend services were developed and executed using Spring Boot, while the frontend was rendered through a browser-accessible planning interface. Session persistence and plan storage were handled through the backend persistence layer, and workflow-stage tracing was supported through Redis Streams where enabled. Testing was conducted in a local development environment with browser-based interaction and direct API verification for backend endpoints.

2. Software Components

The primary software components used in the system include the Spring Boot backend framework, persistence support through JPA-based storage, Redis Streams for workflow tracing, Geoapify for destination and place discovery, OpenWeather for weather enrichment, and configurable large-language-model integration for conversational and agent-level enrichment. External booking continuity was implemented through generated booking links for travel-provider handoff rather than internal transaction execution.

3. Input Conditions

The system was evaluated using natural-language trip requests of varying structure and completeness. These included: partially specified trip requests, complete trip requests submitted in a single message, multi-turn conversational refinement, budget-sensitive planning cases, destination changes and replanning cases, and cases involving weak or repetitive provider outputs. The purpose of these inputs was to observe whether TripMind could preserve correctness of trip-state accumulation, avoid unnecessary repetition in user interaction, and maintain stable planning behavior under realistic conversational conditions.

4. External Data and Provider Usage

TripMind does not depend on a single static dataset in the same way as a supervised classification project. Instead, it operates over dynamic user constraints and provider-enriched travel context. Geocoding and points-of-interest data were obtained through external place APIs, while weather information was obtained from forecast APIs when available or replaced with season- aware fallback logic when live coverage was incomplete. Flight and hotel planning used heuristic pricing logic combined with external booking-path generation for real search-page continuation.

5. Scope of Validation

The scope of validation in this work was system-level and scenario-driven. The evaluation was intended to verify whether the implemented architecture produced coherent travel plans consistent with user constraints and whether fallback mechanisms behaved safely when provider quality was weak. Since no controlled benchmark dataset or formal user-study dataset was used in the current implementation, the experimental setup should be interpreted as practical system validation rather than statistical model evaluation.

Evaluation Metrics

TripMind was evaluated using functional and qualitative metrics that reflect how a planning system behaves in practice. The goal here is not to report predictive accuracy against a benchmark dataset, but to assess whether the workflow captures constraints correctly,

completes the planning stages reliably, and produces outputs that remain useful under realistic travel-planning conditions.

1. Constraint Capture Accuracy
This metric checks whether the system correctly captures and preserves the core details provided by the user. These include destination, origin, travel dates, number of travelers, budget, and optional preference signals. It is considered satisfactory when the stored session state matches the intended trip request without repeated or unnecessary clarification.

2. Planning Completion Behavior
This metric focuses on whether the system starts planning at the right stage and completes the expected workflow once the required information is available. A successful case is one in which the system moves from information gathering to plan generation only after the mandatory fields have been collected, and then returns the outputs expected from the relevant planning agents.

3. Booking Continuity
This metric examines whether outbound booking links preserve the user's trip context correctly. The check is based on whether route, travel dates, and traveler count are carried into booking paths so that users can continue from planning to live search without entering the same details again.

4. Itinerary Relevance and Robustness
This metric looks at whether the itinerary remains plausible and destination-appropriate. It is also used to observe how the system behaves when provider outputs are weak, repetitive, or too narrow. The metric is treated as satisfactory when the system still produces a usable travel plan by activating filtering, ranking, or fallback behavior where needed.

5. Transparency of Output
This metric evaluates whether the result clearly communicates how grounded the plan is. In TripMind, that includes indicators such as booking readiness, source reliability, and live-data coverage. A transparent output helps the user distinguish between planning- level estimates and details that are supported by live or provider-backed context.

6. Workflow Stability
This metric is used to assess whether the system stays coherent across incomplete requests, multi-turn corrections, and replanning cases. The focus is on whether session handling, orchestration, and response aggregation remain stable during realistic conversational use.

Results And Discussion
The implemented TripMind system demonstrates that an agentic AI architecture can be used to produce structured, budget-aware, and booking-oriented travel plans from free-form user requests. The system supports multi-turn conversational input, session-based constraint accumulation, specialized planning-agent coordination, and provider-assisted itinerary generation in a unified planning workflow. Unlike generic travel chat interactions, the output of TripMind is organized as a decision-oriented travel artifact consisting of a feasibility verdict, cost breakdown, contextual advice, and a day-by-day itinerary.
Fig. 3 shows an example TripMind output in which the system presents a feasibility verdict, budget breakdown, planning-intelligence metadata, and a structured planning response in a unified result view.
To validate the implemented workflow under realistic usage conditions, scenario-based system verification was performed, as summarized in Table 3.

Table 3: Scenario-Based Validation Of TripMind

Scenario	Expected Behavior	Observed Behavior	Status
Incomplete trip request	Ask only for missing details	Missing fields were requested without triggering premature planning	Passed
Complete trip request	Trigger full planning workflow	Budget, flight, hotel, weather, advisor, and itinerary outputs were generated	Passed
Multi-turn trip refinement	Update trip state without restarting the session	Revised budget, destination, or dates were reflected in the updated plan	Passed
Weak place-provider output	Use fallback itinerary strategy	Area-based planning was used when place-specific results were weak or repetitive	Passed
Booking continuity	Preserve trip parameters in outbound links	Route, dates, and traveler count were carried into booking links	Passed
Workflow stability	Maintain planning consistency across conversational turns	Session continuity and planning aggregation remained stable	Passed

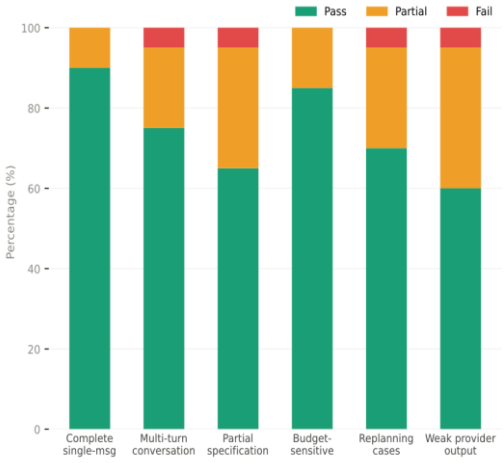

Fig 4: Scenario based validation pass rate

One of the main strengths of TripMind is its modular design. Since each planning task is handled by a dedicated agent, the system is easier to debug, refine, and explain. This also improves fault isolation, because weak outputs from one provider or one planning component do not necessarily invalidate the entire plan. Another important strength is the inclusion of planning intelligence metadata such as booking readiness, source reliability, and live-data coverage. These signals help make the final output more transparent and defensible, especially in scenarios where the system must rely partly on heuristics rather than fully live supplier data.

The fallback-aware design is another significant strength. In travel planning, external APIs may return repetitive or low-value place results, especially for some destinations. By introducing ranking, filtering, and area-based itinerary fallback, the system avoids visible failure modes of naive provider-driven itinerary generation. This makes the system more stable under real-world API variability.

Despite these strengths, the current implementation has clear limitations. First, flight and hotel estimates are planning-level estimates rather than end-to-end live supplier confirmations. Although booking links direct the user to real search pages, final inventory and prices are verified outside TripMind. Second, itinerary quality can still depend on the richness and consistency of third-party provider results. Third, the current work focuses primarily on system design and implementation rather than large-scale benchmark evaluation or controlled user study results.

Overall, the current implementation shows that travel planning benefits from being treated as a structured coordination problem rather than a free-form language generation problem. The results support the argument that budget-aware modular orchestration is more suitable for this domain than generic chatbot output alone.

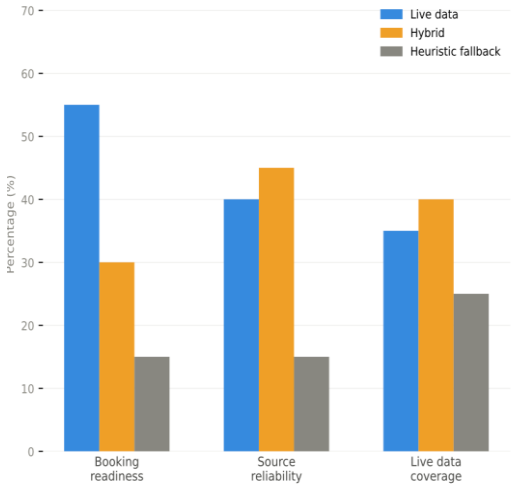

Fig 5: Planning intelligence signal distribution

Threats To Validity

Several threats to validity should be considered when interpreting the current work.

1. Provider Dependency

TripMind relies on external travel and weather providers for part of its planning context. As a result, the quality of itinerary enrichment and contextual relevance may vary depending on provider coverage, destination support, and API response quality. Although fallback mechanisms reduce this risk, provider dependence remains a source of variability.

2. Limited Quantitative Evaluation

The present study focuses on system design, implementation, and functional validation rather than large-scale quantitative benchmarking. Since no controlled benchmark dataset or formal user-study dataset was used in the current version, the paper does not claim statistical superiority over alternative travel- planning systems.

3. Planning-Level Estimation

Flight and hotel outputs in TripMind are currently planning-level estimates rather than full live supplier confirmations. While the booking links provide continuity into real booking environments, the final booking prices and inventory are validated outside the platform. Therefore, the generated cost breakdown should be interpreted as a planning aid rather than a guaranteed quotation.

4. Destination Variability
Travel planning quality may differ across destinations because some destinations have stronger provider support and richer place information than others. Even with ranking and area-based fallback logic, itinerary quality may still vary depending on how well the destination is represented by external providers.

5. Interaction Variability
Because TripMind accepts noisy and free-form conversational input, user phrasing can influence parsing behavior and planning flow. Although the system includes normalization and multi-turn state management, conversational systems remain sensitive to ambiguous phrasing, spelling variation, and incomplete specification.

6. Generalizability
The current implementation is oriented toward demonstrating a modular agentic travel-planning architecture. The conclusions should therefore be understood as evidence of architectural viability rather than universal claims about all travel-planning environments, all user populations, or all booking ecosystems.

Conclusion And Future Work
This paper presented TripMind, an agentic AI-based system that turns free-form travel requests into structured, budget-aware trip plans. Rather than relying on one broad conversational response, the system organizes the task through specialized agents for budget feasibility, flight estimation, hotel planning, weather interpretation, itinerary generation, and final advisory synthesis. That structure makes the output more useful for planning decisions than a generic travel-chat response.
The current implementation shows that travel planning benefits from structured coordination and constraint- aware reasoning. TripMind produces a feasibility verdict, cost breakdown, booking-oriented guidance, and a day-by-day itinerary, while also exposing planning-intelligence signals such as booking readiness, source reliability, and live-data coverage. Its ranking and fallback mechanisms also help the system remain usable when external provider outputs are incomplete or repetitive.
There is still room to strengthen the system. Live supplier integration for flights and hotels would improve booking realism. Better destination grounding and preference memory could make itineraries more personalized. Explicit constraint-validation layers could also help verify that final plans satisfy captured user requirements. Future work can further strengthen the evaluation through user studies and benchmark-based comparisons with baseline conversational systems.
Overall, TripMind demonstrates that combining multi- agent orchestration, conversational AI, and structured planning logic can produce a more practical travel- planning workflow than a conventional one-shot assistant.

References
J. Xie, K. Zhang, J. Chen, T. Zhu, R. Lou, Y. Tian, Y. Xiao, and Y. Su, “TravelPlanner: A Benchmark for Real-World Planning with Language Agents,” in Proc. 41st Int. Conf. Machine Learning (ICML), 2024, pp. 54590–54613. [Online].
Available:
https://proceedings.mlr.press/v235/xie24j.html

A. Chen, X. Ge, Z. Fu, Y. Xiao, and J. Chen, “TravelAgent: An AI Assistant for Personalized Travel Planning,” arXiv preprint arXiv:2409.08069, 2024. [Online]. Available: https://arxiv.org/abs/2409.08069

D. Camacho, D. Borrajo, and J. M. Molina, “Intelligent Travel Planning: A MultiAgent Planning System to Solve Web Problems in the e-Tourism Domain,” Autonomous Agents and Multi-Agent Systems, vol. 4, no. 4, pp. 387–392, 2001. doi: 10.1023/A:1012767210241

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” in Proc. 11th Int. Conf. Learning Representations (ICLR), 2023. [Online].
Available:
https://iclr.cc/virtual/2023/oral/12647

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih,
T. Rocktaschel, S. Riedel, and D. Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” arXiv preprint arXiv:2005.11401, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

Spring, “Spring Boot Reference Documentation,” 2026. [Online]. Available: https://docs.spring.io/spring-boot/documentation.html. Accessed: Apr. 24, 2026.

Redis, “Redis Streams,” 2026. [Online]. Available: https://redis.io/docs/latest/develop/data-types/streams/. Accessed: Apr. 24, 2026.

Geoapify, “Places API Developer

Documentation," 2026. [Online]. Available: https://apidocs.geoapify.com/docs/places/. Accessed: Apr. 24, 2026.

OpenWeather, "Weather API Documentation," 2026. [Online]. Available: https://openweathermap.org/api. Accessed: Apr. 24, 2026.

Amadeus for Developers, "Flight APIs Tutorial," 2026. [Online]. Available: https://developers.amadeus.com/self-service/apis-docs/guides/developer-guides/resources/flights/. Accessed: Apr. 24, 2026.

Booking.com, "Booking.com Demand API," 2026. [Online]. Available: https://developers.booking.com/demand/docs/open-api/3.2/demand-api. Accessed: Apr. 24, 2026.