IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

WanderAI: An AI Agent for Intelligent Tourism Guidance and Personalized Trip Planning

Viraj Rampure[1], Arpan Dabhade[1], Sudhanshu Pawar[1], Prof. Aman Kamble[2]
[1]Department of Computer Science and Engineering, Guide[2]
MIT School of Computing, MIT Art, Design and Technology University, Pune, India

Abstract : *WanderAI is proposed as an AI-powered tourism guide and trip-planning agent that consolidates destination discovery, itinerary preparation, route awareness, and weather-informed recommendations within a conversational interface. Travellers commonly assemble plans by switching among travel blogs, map applications, review platforms, and weather services. This fragmented process increases planning effort and can produce generic schedules that do not reflect an individual's interests, budget, available time, or travel pace. WanderAI addresses this problem through a Large Language Model (LLM)-based agent supported by Retrieval-Augmented Generation (RAG) and external tools. A traveller expresses a destination, dates or duration, budget, and interests in natural language. The agent interprets the request, retrieves relevant destination information from a curated vector database, obtains route and distance information from a mapping service, and checks forecasts through a weather API. It then generates a day-wise itinerary that can be refined through follow-up conversation. The paper presents the proposed system architecture, functional modules, agent workflow, data design, implementation methodology, and an evaluation plan. The first version focuses on trip planning and destination guidance rather than end-to-end booking. Since the supplied project material describes a proposed system and does not report experimental deployment, this paper defines expected behaviour and measurable evaluation criteria without claiming achieved performance.*

Keywords: Artificial Intelligence, Tourism Recommendation, Large Language Models, Retrieval-Augmented Generation, AI Agents, Itinerary Planning, Route Optimization, Weather-Aware Planning.

I. INTRODUCTION

Tourism planning is a multi-stage decision process. A traveller must select attractions, estimate the time required at each location, arrange transport between places, consider food and local experiences, and account for practical constraints such as weather and opening hours. The information needed for these decisions is distributed across independent websites and applications. Travellers therefore spend time collecting and reconciling information before a practical itinerary can be created.

A conventional search engine or map application can answer individual questions, but it does not necessarily maintain the traveller's complete set of preferences across the planning process. Generic travel itineraries may overlook budget, trip duration, personal interests, and preferred pace. They may also become impractical when weather conditions change. The WanderAI presentation illustrates this issue through a family planning a three-day Pune trip: even after consulting several tabs, the family may fail to notice forecast rain on the day scheduled for a hill-fort visit.

Recent advances in LLMs enable conversational systems to interpret natural-language requests and generate structured responses. However, an LLM may produce outdated or unsupported information when asked about specific attractions, routes, prices, or operating hours. RAG addresses part of this limitation by retrieving relevant information from an external knowledge base and providing it as context to the model [1]. Agent-based tool use extends the interaction by allowing the model to call services such as maps and weather APIs when current contextual information is required [2].

DOI: 10.48175/IJARSCT-40851

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, November 2026 Impact Factor: 8.2

WanderAI combines these capabilities into a single proposed travel companion. Its core purpose is to understand a traveller's requirements, retrieve destination knowledge, use route and weather information, and generate an itinerary that can be revised through conversation. The initial scope is intentionally limited to trip planning and destination guidance; hotel or ticket booking, payment processing, and real-time crowd prediction are reserved for later development.

The main contributions of this paper are: (1) a modular architecture connecting a conversational interface, Python backend, LLM agent, RAG knowledge base, application database, mapping service, and weather service; (2) a workflow for converting natural-language travel requirements into a day-wise itinerary; (3) a proposed retrieval and itinerary-ranking approach that makes constraints and live context explicit; and (4) an evaluation plan covering retrieval quality, itinerary feasibility, weather adaptation, response quality, and usability.

II. RELATED WORK AND TECHNICAL BACKGROUND

A. Large Language Models and Conversational Interfaces

Transformer-based language models provide the foundation for current natural-language understanding and generation systems [3]. In a travel-planning context, an LLM can interpret a request such as a multi-day visit with a stated budget and particular interests, then convert it into structured trip attributes. The model can also maintain conversational context so that a traveller can request changes without repeating the entire trip description. Nevertheless, generated content must be checked against reliable destination information and external data sources.

B. Retrieval-Augmented Generation

RAG combines parametric language-model knowledge with retrieved, non-parametric information [1]. For WanderAI, destination records are collected from reliable and permitted sources, cleaned, divided into meaningful chunks, converted into embeddings, and stored in a vector database such as FAISS or ChromaDB. At query time, the system retrieves content relevant to the traveller's destination and question. The retrieved passages are supplied to the LLM so that descriptions of attractions, history, culture, food, and local tips are grounded in curated material. This design is intended to reduce unsupported statements; it does not guarantee that every retrieved record is current, so source maintenance remains necessary.

C. Tool-Using Agents and Live Context

Tool-using agent approaches connect language-model reasoning with external actions. ReAct, for example, describes a pattern in which reasoning and action are interleaved [2]. In WanderAI, the agent may call a mapping service to obtain location, distance, route, and travel-time information, or a weather service to retrieve forecasts for the planned travel dates. The agent can then use those results when sequencing activities or suggesting an indoor alternative. API responses are time-sensitive and may be incomplete; the system should preserve the retrieval timestamp and communicate uncertainty where appropriate.

D. Mapping and Vector Search Technologies

Google Maps Platform and OpenStreetMap are identified as candidate mapping sources in the project synopsis and presentation [4], [5]. FAISS provides efficient similarity search over dense vectors [6], while ChromaDB is another candidate for storing and retrieving embedded destination content [7]. The final choice should be based on deployment requirements, data volume, operational cost, licensing, and integration complexity rather than assumed superiority of one option.

III. PROBLEM DEFINITION AND OBJECTIVES

The problem addressed by WanderAI is the absence of a single conversational system that can help a traveller plan a complete trip and provide destination guidance while considering personal preferences and changing contextual information. Current planning practices require users to switch among travel blogs, maps, review sites, and weather applications, manually compare information, and construct schedules. This leads to unnecessary effort, generic

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

recommendations, missed local attractions, and plans that may not remain suitable when weather or other conditions change.

The proposed system has the following objectives:

- Develop an AI agent that generates personalized, day-wise itineraries from natural-language trip requests.
- Provide tourist-guide information about attractions, history, culture, food, and local tips.
- Ground destination responses in a curated knowledge base through RAG.
- Use Google Maps API or OpenStreetMap for location, distance, and route-aware planning.
- Use a Weather API to review forecasts and suggest alternatives when conditions are unfavourable.
- Support multi-turn conversation so travellers can refine and modify their plans.
- Reduce manual research effort and provide a modular foundation for additional destinations and capabilities.

IV. PROPOSED WANDERAI SYSTEM

WanderAI is designed as a conversational AI agent that performs the roles of trip planner and tourist guide. A traveller interacts through a web interface, with a mobile interface planned using Flutter. The frontend sends a request to a Python backend implemented using FastAPI or Flask. The backend coordinates the LLM agent, retrieval pipeline, application database, mapping service, and weather service. The response is returned as a structured itinerary and supporting destination guidance.

A. System Architecture

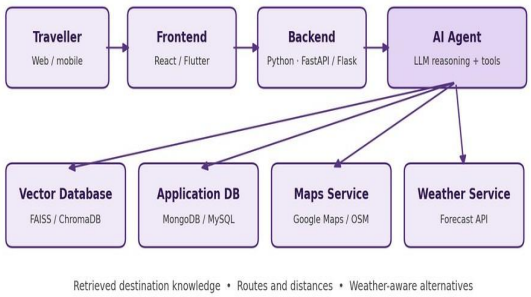


Fig. 1. Proposed modular architecture of WanderAI, showing the traveller interface, backend agent, retrieval store, application database, and external context services.

The architecture contains the following layers. The presentation layer accepts natural-language input and displays the itinerary, map information, and conversational responses. The backend layer validates requests, manages user sessions, and coordinates the agent. The AI layer interprets trip requirements, retrieves destination knowledge, and decides when external tools are required. The data layer stores destination embeddings and user trip records. The integration layer communicates with map and weather services. Authentication, secure API-key management, and role-based authorization are cross-cutting requirements identified in the project synopsis.

B. Functional Modules

Module	Primary responsibilities
Conversational Planner	Accepts natural-language trip requests; extracts destination, dates or duration, budget, and interests; supports multi-turn refinement.
Itinerary and Routing	Creates day-wise schedules; obtains map routes, distances, and travel-time estimates; sequences places to reduce unnecessary travel.
Tourist Guide (RAG)	Retrieves destination passages from FAISS/ChromaDB; answers questions about places, history, culture, food, and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

	local tips using retrieved context.
Module	Primary responsibilities
Live Context	Fetches weather forecasts; supports plan review and alternative or indoor suggestions; considers opening-hour information where available.
User and Data	Manages user accounts and saved trips using MongoDB or MySQL; supports retrieval of previously saved plans.

C. User Roles

The system defines three roles. Travellers submit requests, explore destination information, and refine itineraries. Content administrators curate and update destination records that feed the knowledge base. System administrators manage users, API configuration, and platform health. Separating these responsibilities helps maintain destination content while keeping platform-level operations controlled.

V. AGENT WORKFLOW AND PROPOSED PLANNING METHOD

The agent follows a five-stage workflow: understand, retrieve, use tools, generate, and refine. First, the LLM parses the traveller's request and identifies the destination, trip duration or dates, budget, interests, and other stated preferences. If essential information is missing, the conversational interface can request clarification. Second, the RAG component retrieves destination records relevant to the request. Third, the agent calls mapping and weather services when route or forecast information is needed. Fourth, it generates a day-wise itinerary. Finally, the traveller can provide feedback and the agent updates the plan while preserving unchanged requirements.

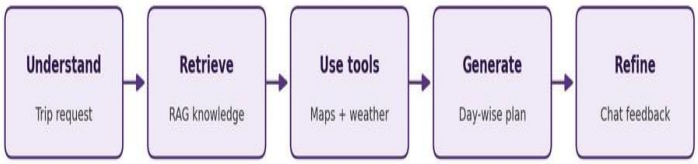


Fig. 2. Five-stage agent workflow proposed for trip planning and itinerary refinement.

A. Request Representation

A trip request can be represented as a structured record R =
{D, T, B, P, C}, where D denotes destination, T denotes travel dates or duration, B denotes budget constraints, P denotes interests and preferences, and C denotes additional constraints such as pace or accessibility when supplied by the traveller. The natural-language interface maps the request into this representation. The system should retain the original user wording alongside extracted fields so that uncertain interpretations can be clarified rather than silently assumed.

B. Retrieval and Grounding

Destination documents are preprocessed into chunks that preserve useful context, such as attraction name, location, description, category, historical information, food guidance, and source metadata. Each chunk is embedded and indexed in the selected vector database. For a query q, the retrieval component returns the top-k passages with the highest semantic similarity, subject to destination and category filters where available. The LLM receives the retrieved passages and is instructed to distinguish sourced facts from general suggestions. If the knowledge base does not contain an answer, the agent should state that the information is unavailable rather than inventing a fact.

C. Route- and Weather-Aware Itinerary Generation

Candidate attractions are selected according to the traveller's interests and trip constraints. The mapping service supplies route and travel-time estimates between candidate locations. The weather service supplies forecast information for the relevant dates. The itinerary generator then arranges activities into daily schedules while considering travel time, available hours, and weather suitability. When a forecast makes an outdoor activity unsuitable, the agent may propose

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

an indoor alternative or suggest moving the activity to another day, provided that the alternative is supported by retrieved destination information.

As a design abstraction, a candidate activity p may be ranked using a configurable utility function: $U(p) = \alpha \cdot Rel(p,P) + \beta \cdot Feas(p,T,B) + \gamma \cdot Weather(p) - \delta \cdot TravelCost(p)$. Here, Rel measures relevance to stated interests, Feas measures compatibility with time and budget constraints, Weather represents suitability under the forecast, and TravelCost represents the travel burden. The coefficients α, β, γ, and δ are design parameters to be tuned during evaluation; the project materials do not specify numerical weights. The function expresses the intended trade-off and is not presented as an implemented or experimentally validated algorithm.

D. Itinerary Refinement

The generated itinerary is treated as an editable plan rather than a fixed response. A traveller may request changes such as reducing travel, replacing an attraction, changing the pace, or prioritising food and cultural experiences. The agent updates the affected parts of the schedule while retaining the rest of the trip constraints. Saved plans are associated with the user through the application database, allowing the traveller to retrieve them later.

VI. IMPLEMENTATION METHODOLOGY

The project follows an iterative development approach in which the knowledge base, agent, backend, frontend, and integrations are developed and tested progressively. The technology choices below are the alternatives identified in the synopsis and presentation; the final implementation may select one option from each pair based on feasibility.

Layer	Proposed technologies / tools
AI	AI agent framework, Generative AI / LLM, Retrieval-Augmented Generation
Vector retrieval	FAISS or ChromaDB
Backend	Python with FastAPI or Flask; REST APIs
Frontend	React.js for web; Flutter for a planned mobile interface
Application database	MongoDB or MySQL
External services	Google Maps API or OpenStreetMap; Weather API
Layer	Proposed technologies / tools
Development and deployment	Visual Studio Code, Git, GitHub, cloud deployment as required

A. Development Phases

- Requirement analysis: identify traveller needs, user roles, and the first-version scope.
- Data and knowledge-base preparation: collect destination information from reliable and permitted sources, clean and chunk the content, generate embeddings, and index the records.
- Agent and tool development: design agent prompts and orchestration, implement RAG retrieval, integrate map and weather tools, and develop itinerary generation and refinement.
- Backend and data development: implement REST endpoints, user management, saved-trip operations, and database connectivity.
- Frontend development: build the conversational interface, itinerary views, and map presentation, then connect them to backend APIs.
- Testing and deployment: conduct unit, integration, retrieval, API-failure, and user-flow testing; deploy the application and perform final environment checks.
- Evaluation: test with intended users, collect feedback on itinerary quality and usability, and identify limitations for subsequent iterations.

VI-A. KNOWLEDGE-BASE PREPARATION AND MAINTENANCE

The destination knowledge base is a central dependency of WanderAI because the RAG component can only retrieve information that has been collected, processed, and indexed. The proposed ingestion pipeline begins with source

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, November 2026 Impact Factor: 8.2

selection. Destination descriptions, attraction details, cultural information, food guidance, and local tips should be obtained from reliable sources whose use is permitted. Each record should retain source attribution and, where available, a last-verified date. The content administrator reviews records before they are published to the retrieval index.

After collection, preprocessing removes duplicate passages, normalizes place names, and separates long documents into chunks that preserve semantic context. Each chunk should include metadata such as destination, attraction, category, language, source, and update date. Metadata filters can narrow retrieval to the requested city or topic before semantic similarity is applied. Embeddings are then generated and stored in FAISS or ChromaDB. When a source record changes, its associated chunks and embeddings should be updated so that obsolete information is not left active indefinitely.

A retrieval response should retain the identifiers of the passages used to generate an answer. This enables the application to display references or source notes and allows administrators to trace a questionable recommendation back to its source. Where a query concerns time-sensitive information such as operating hours, the knowledge base should be treated as supporting context rather than the final authority; the agent should prefer a current service response when such a service is available. If neither source provides sufficient evidence, the answer should explicitly communicate the gap.

VI-B. PROPOSED AGENT CONTROL LOGIC

The agent controller coordinates the sequence of retrieval and external tool calls. The following high-level procedure expresses the intended control logic. It is a design specification for implementation and does not claim that the complete pipeline has already been deployed:

Algorithm 1: WanderAI itinerary generation

Input: traveller message M; conversation context H; destination knowledge index K; map service G; weather service W

1. Parse M and H to extract destination, dates or duration, budget, interests, and explicit constraints.
2. If a required field is ambiguous, ask a focused clarification and update the request representation.
3. Retrieve relevant destination passages P from K using semantic similarity and metadata filters.
4. Assess whether route or forecast data is required for the requested itinerary.
5. If required, call G for location, route, distance, and travel-time estimates; call W for forecast data.
6. Generate candidate day-wise activities using traveller preferences and retrieved evidence.
7. Check schedule feasibility, travel sequence, available time, and weather suitability; revise conflicts where possible.
8. Return the itinerary with supporting destination information and note unavailable or uncertain facts.
9. Apply follow-up feedback to the affected itinerary elements while retaining unchanged constraints.
10. Save the approved trip when requested; log errors and API status for operational monitoring.

Output: structured itinerary, supporting information, and any stated limitations.

VI-C. APPLICATION DATA AND USER MANAGEMENT

The application database and vector database serve different purposes. The vector database stores destination text chunks and embeddings for semantic retrieval. MongoDB or MySQL stores application entities such as user profiles, saved trips, itinerary versions, and administrative metadata. A saved trip may contain a trip identifier, user identifier, destination, travel dates or duration, budget, preference fields, day-wise activities, route references, timestamps, and the current revision status. The design should preserve the distinction between user-provided preferences and agent-generated suggestions so that a traveller can review and revise the plan.

The content-administration workflow begins with an authenticated administrator adding or updating destination information. The system validates required fields, preprocesses the content, generates embeddings, and updates the vector index. The system-administration workflow manages user accounts, API configuration, and platform health.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, November 2026 Impact Factor: 8.2

Role-based authorization should prevent a traveller from modifying curated destination records and should restrict sensitive administrative operations to authorized accounts.

VI-D. SECURITY, PRIVACY, AND RELIABILITY CONSIDERATIONS

WanderAI may process personal travel preferences, dates, and saved itineraries. The application should collect only the information required for planning, protect stored records through appropriate access controls, and avoid exposing one traveller's saved trips to another user. Authentication and role-based authorization are included in the project requirements. API keys for LLM, mapping, and weather services should be stored on the backend in protected configuration rather than embedded in frontend code. Requests and errors should be logged without unnecessarily recording sensitive user content.

Reliability also depends on graceful handling of external-service failures. If a mapping or weather API is unavailable, the agent should not fabricate route distances or forecast conditions. It may return the parts of the itinerary supported by the knowledge base and clearly mark the missing live information. Similarly, a failed retrieval should not be silently replaced by unsupported model-generated facts. Timeouts, retries, API quotas, and cached responses may be considered during implementation, with cache age displayed or considered when the information is time-sensitive.

VI-E. ILLUSTRATIVE USER SCENARIO

Consider a family requesting a three-day visit to Pune with a moderate budget and an interest in heritage attractions and local food. The conversational planner extracts the destination, duration, budget, and interests from the request. The RAG module retrieves relevant attraction and food records. The mapping tool provides distances and estimated travel times between selected places, while the weather service supplies forecasts for the travel dates. The itinerary generator groups suitable activities into three days and avoids unnecessary cross-city movement where the route information supports a better sequence.

If the forecast indicates rain on the day initially assigned to a hill-fort visit, the live-context module flags the conflict. The agent may suggest an indoor attraction for that time slot and move the hill-fort visit to a more suitable day, subject to opening-hour and route constraints. The traveller can then request a slower pace or replace a suggested restaurant. The agent updates the relevant schedule elements and preserves the original destination and trip duration. This scenario illustrates the intended interaction; it is not a report of a completed user test.

VII. EVALUATION PLAN

The supplied synopsis and presentation describe the expected behaviour and development plan but do not provide completed experiments, a deployed prototype, user-study data, or measured performance results. Accordingly, the following evaluation plan defines how the proposed system should be assessed once an implementation is available. No numerical result is claimed in this paper.

Evaluation dimension	Suggested measurement	Purpose
Retrieval quality	Precision@k, Recall@k, and human relevance ratings	Check whether retrieved destination passages support the query.
Factual grounding	Fact-check rate and unsupported-claim count	Measure whether answers are consistent with curated content.
Itinerary feasibility	Constraint satisfaction, schedule conflicts, and route plausibility	Check time, budget, and sequencing constraints.
Route usefulness	Estimated travel distance/time compared with a baseline ordering	Assess whether route-aware sequencing reduces unnecessary travel.
Weather adaptation	Correct identification of affected activities and suitability of alternatives	Evaluate whether forecasts lead to sensible plan revisions.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Conversation al usability	Task completion, number of refinement turns, and user feedback	Assess ease of planning and modification.
System performance	Response latency, API failure rate, and retrieval latency	Measure responsiveness and operational reliability.

A suitable test set should contain representative requests with different trip lengths, budgets, interests, and destinations. Each request should be evaluated against manually reviewed destination facts and route constraints. Weather adaptation can be tested using recorded forecast scenarios or controlled API responses so that the same input conditions are reproducible. User evaluation should include participants who attempt to create and revise an itinerary, with feedback collected through a consistent questionnaire. Results should report sample size, test conditions, baselines, and limitations.

VIII. APPLICATIONS AND EXPECTED OUTCOMES

For travellers, WanderAI is expected to reduce the effort involved in assembling trip information, provide personalized day-wise itineraries, answer destination questions, and support plan changes when weather conditions are unfavourable. Tourism organisations and local guides may use curated destination content to improve the visibility of local attractions and provide visitors with structured digital guidance. In education and research, the project offers a practical application of AI agents, LLMs, RAG, and recommendation-oriented planning.

The expected outcome is a functional AI-based travel assistant that produces personalized itineraries, provides grounded destination information, uses map data for practical sequencing, adapts suggestions using weather forecasts, and allows iterative refinement through conversation. These are target outcomes to be verified through implementation and evaluation, not measured achievements reported by the present paper.

IX. LIMITATIONS AND FUTURE SCOPE

The quality of WanderAI depends on the coverage, accuracy, and freshness of its destination knowledge base and on the availability of external APIs. Incomplete or outdated attraction records can lead to weak recommendations even when retrieval works as designed. Mapping and weather services may impose usage limits, incur costs, or return incomplete information. Forecasts are uncertain, and opening hours may change without notice. LLM-generated plans may also contain infeasible schedules unless route, time, and other constraints are checked explicitly. The first version is not intended to be a complete booking platform and does not include ticket or hotel transactions, payments, or real-time crowd prediction.

After the first version is stable, the project may be extended with multilingual support, voice-based interaction, budget estimation and expense tracking, booking-service integration, offline access to saved itineraries and maps, recommendations informed by previous trips, real-time alerts for weather changes and closures, group trip collaboration, additional cities and countries, and analytics dashboards for tourism organisations.

X. CONCLUSION

This paper presented WanderAI, a proposed conversational AI agent for tourism guidance and personalized trip planning. The system combines an LLM-based conversational planner with RAG-based destination retrieval, mapping services, and weather information to generate day-wise itineraries that can be refined through follow-up interaction. Its modular architecture separates the user interface, Python backend, agent logic, vector database, application database, and external services. The paper also defined a staged implementation methodology and an evaluation plan covering retrieval quality, factual grounding, itinerary feasibility, weather adaptation, usability, and system performance. The immediate research and development priority is to implement and validate the core planning workflow using a curated destination knowledge base and reliable external data. Future work can then extend the platform with multilingual, voice, budget, booking, offline, and collaborative capabilities.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

REFERENCES

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020.
[2] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. International Conference on Learning Representations (ICLR), 2023.
[3] A. Vaswani et al., "Attention Is All You Need," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017.
[4] Google, "Google Maps Platform Documentation." [Online]. Available: https://developers.google.com/maps/documentation
[5] OpenStreetMap Foundation, "OpenStreetMap Documentation." [Online]. Available: https://wiki.openstreetmap.org/
[6] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.
[7] Chroma, "Chroma Documentation." [Online]. Available: https://docs.trychroma.com/
[8] FastAPI, "FastAPI Documentation." [Online]. Available: https://fastapi.tiangolo.com/
[9] MongoDB Inc., "MongoDB Documentation." [Online].
Available: https://www.mongodb.com/docs/
[10] Meta Open Source, "React Documentation." [Online]. Available: https://react.dev/
[11] Google, "Flutter Documentation." [Online]. Available: https://docs.flutter.dev/