Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

# AI Based Context Aware Nearby Places Recommendation System by Haversine Algorithm

**Muskan Sahu[1], Shraddha Dhote[2], Priyanka Ughade[3], Prof. Satish Chadokar[4]**

Students[1,2,3], Assistant Professor[4]

[1,2,3,4]Department of Computer Science & Engineering, Shri Balaji Institute of Technology and Management, Betul (M.P.), India

**Abstract:**

Finding suitable nearby places to visit is increasingly difficult because information about location, weather, ratings, budget and personal taste is scattered across many platforms. Artificial intelligence (AI) and context-aware recommender systems (CARS) address this problem by combining user preferences with situational signals such as location, time, weather, emotion and cost. This review paper surveys five recent studies that represent the main directions of the field: Ask Nearby, a retrieval-augmented, LLM-based neighborhood information system (2025); the Deep Adaptive Interest Network (DAIN), a deep-learning model for dynamic interest modeling (2024); Doorob, an emotion-aware hybrid tourism recommender that uses facial emotion recognition (2026); an offline, ontology-enhanced deep-learning tourism recommender for the Santurbán páramo (2025); and TourMate, a machine-learning mobile application that predicts tourist-place suitability (2024). Each work is summarized, and the techniques are compared on data sources, context factors, algorithms, evaluation metrics and deployment. The comparison shows that hybrid and context-aware models generally outperform single-method baselines, but that most systems ignore live weather, depend on simulated or small datasets, offer little explanation of their output, or raise privacy concerns. We identify these research gaps and show how they motivate Smart Explore, an AI-based, context-aware Android application that ranks nearby places using distance, weather suitability, ratings, opening status, user preferences and available visiting time. Challenges and future directions are also discussed.

**Keywords:** Context-aware recommendation; Tourism recommender system; nearby places discovery; Hybrid recommendation; Deep learning; Location-based services

## 1. Introduction

Tourism and local exploration are an important part of modern life. People travel to discover new destinations, visit historical places, enjoy natural attractions and experience different cultures. Tourism also supports national economies and sustainable development [3]: for Bangladesh, for example, the tourism industry contributed about 4.4 percent of GDP in 2018–2019 and created jobs, revenue and

infrastructure [5]. In Saudi Arabia and Colombia, tourism is likewise treated as a strategic pillar for economic diversification and regional development [3], [4].

At the same time, the growth of digital platforms has created an information-overload problem. Visitors and residents face a very large number of places, reviews, ratings, photos and social-media posts, and these sources often conflict with each other [3]. A person who is new to a city may not even know which attractions are nearby, which are open, which suit the current weather or which fit the available budget and time [5]. Choosing a place therefore becomes a time-consuming manual task that requires comparing several applications.

Smartphone's, GPS, mobile internet and Artificial Intelligence (AI) now make it possible to solve this problem automatically. A **recommender system** filters a large set of items and presents the ones most likely to match a user's needs. Classical recommenders fall into four groups: collaborative filtering (CF), which assumes that users with similar past behavior will like similar items; content-based filtering (CB), which recommends items with attributes similar to those the user liked before; context-aware methods, which add situational information such as location, time or weather; and hybrid methods, which combine two or more of these approaches [3]. Collaborative filtering suffers from the cold-start problem and data sparsity, while content-based filtering can reduce diversity; hybrid and context-aware designs were proposed to overcome these limits [3], [4].

### 1.2 Importance of Context-Aware Recommendation

User interests are not static. They change with time, environment and need [2]. A tourist who prefers outdoor parks on a sunny morning may need an indoor museum or restaurant when it rains in the evening. Context-aware recommender systems (CARS) therefore extend conventional models by using spatial, temporal, environmental and social variables when ranking items [4]. Context can be explicit or implicit, directly observed or inferred, and static or dynamic [4]. In tourism, the most influential context factors reported in the literature are location and distance, weather conditions, time of day, seasonality, budget, travel companions, popularity and even the user's emotional state [3], [4].

Context-awareness matters most for **nearby-place discovery**. When a user asks "which cafe near me is open late?", the answer depends on the user's current position, the time and the place's opening hours. Niu et al. formalized this need as the Local Life Information Accessibility (LLIA) problem and argued that proximity-based planning, such as the "15-minute city", requires efficient access to information about nearby places, not only physical closeness [1]. Recent advances in deep learning, large language models (LLMs), ontology's and affective computing make it possible to build such systems, but each approach has strengths and weaknesses that need to be understood before a new system is designed.

### 1.3 Motivation: The Smart Explore Project

This review is written to support our major project, **Smart Explore – an AI-Based Context-Aware Nearby Places Recommendation System**. Smart Explore is an Android application that detects the user's current location, retrieves nearby tourist attractions, temples, parks, restaurants and other useful places through Maps/Places and Geolocation APIs, collects real-time weather through a Weather API [5], and ranks places with a context-based weighted scoring algorithm. The score uses distance, weather suitability, ratings, opening status, user preferences and available visiting time. Every recommendation is accompanied by place details, a best-time-to-visit suggestion, an explanation of why it was suggested

and map-based navigation. The application is built with React Native, Node.js, Express.js and MongoDB . Reviewing existing research helps us position Smart Explore, borrow proven techniques and avoid known weaknesses.

### 1.4 Objectives of the Review

The purpose of this review is to:

- Summarize five recent and representative research papers on context-aware, personalised and location-based recommendation;
- Explain the techniques used in each paper, including retrieval-augmented generation, deep neural networks, hybrid collaborative–context models, facial emotion recognition, ontology's and classical machine-learning classifiers;
- Compare these techniques on data, context factors, algorithms, evaluation metrics, deployment and limitations;
- Identify the research gaps that remain open, especially for weather-aware and explainable nearby-place recommendation;
- Discuss challenges and future directions, and relate them to the design of Smart Explore.

## 2. Literature Review

This section summarizes the five selected studies in the order of their technical approach: LLM-based retrieval, deep interest modeling, emotion-aware hybrid recommendation, ontology-based deep learning and machine-learning classification. For each paper we describe the problem, the method, the data, the main results and the stated limitations.

### 2.1 Doorob: Emotion and Context-Aware Recommendation for Urban Tourism (Aldayel et al., 2026)

**Problem.** Aldayel et al. [3] address information overload and the cold-start problem in tourism platforms in Saudi Arabia. Many users never give explicit ratings, so CF has little data. The authors propose that a user's facial expression while viewing a place can serve as *implicit feedback*.

**Method.** The Doorob web system implements three recommenders adapted from an open-source Recommenders repository: Smart Adaptive Recommendations (SAR) for collaborative filtering, Vowpal Wabbit (VW) for context-aware recommendation and LightFM for hybrid recommendation. The only context used in VW is the distance between the user and the place. Facial emotion recognition (FER) is added with DeepFace and OpenCV: after the user opens a place page and waits ten seconds, ten seconds of video are analyzed frame by frame into seven emotions. Happy and surprise scores are treated as positive, angry, sad, fear and disgust as negative, and neutral is ignored. The difference between positive and negative scores is mapped to a 1–5 rating (for example, a score of 60 or more gives a rating of 5). If the camera is refused, the user can rate manually.

**Data and results.** Places in Riyadh were collected with the Google Places API (October 2024). Because no real tourism ratings were available, MovieLens ratings were remapped to place identifiers, giving 1,048,575 interactions from 6,781 users on 8,836 places with 98.25% sparsity. In evaluation, SAR achieved the lowest rating error (RMSE 0.89, MAE 0.72), VW achieved the best ranking precision (85.80%) and recall (63.41%), and LightFM achieved the highest AUC (95.88%). All three had higher

precision and NDCG than the SVD and TF-IDF baselines, although SAR's AUC (54.94%) was lower than SVD's (73.56%).

**Limitations.** The authors stress that remapped movie ratings are a simulation and may not represent real tourism behavior, that emotion-based ratings behave as a complementary signal and not a replacement for explicit ratings, and that camera-based emotion recognition raises privacy, bias and consent concerns.

### 2.2 AskNearby: LLM-Based Neighborhood Information Retrieval (Niu et al., 2025)

**Problem.** Niu et al. [1] define the Local Life Information Accessibility (LLIA) problem: enabling residents to obtain timely, relevant and context-aware information about their neighborhood through two complementary modes. *Active retrieval* answers explicit questions such as "Which supermarket near me is open late?", while *passive recommendation* proactively surfaces useful local content. The authors argue that existing systems work at city scale, rely on keyword matching and fixed search radii, and treat space and time as isolated filters.

**Method.** AskNearby combines a three-layer Retrieval-Augmented Generation (RAG) pipeline with a cognitive-map model. The user query is first analyzed by an LLM to extract spatial entities and semantic intent. *GeoRAG* uses a spatial database (PostGIS) to keep only candidates within a distance threshold of the query or user location. *GraphRAG* uses a graph database (NebulaGraph) to expand the semantic intent through related categories, tags and events. *VectorRAG* embeds the enriched query with the bge-large-zh-v1.5 model and retrieves similar posts with pgvector using Euclidean distance. For passive recommendation, the cognitive-map model scores an item as the product of three terms: semantic relevance (based on a TF-IWF representation of the functional meaning of locations at different times), spatial proximity (an exponential distance-decay function) and public familiarity (how often the place is visited by the community). The three exponents are initialised to 1 and were not tuned.

**Data and results.** About 20,000 geotagged posts from the RedNote platform covering Shenzhen were used. AskNearby was compared with GPT-4o, DeepSeek-R1, Qwen3-Turbo, RedNote search, Gaode Maps and Baidu Maps. The full system reached Precision@4 of 75.6%, NDCG@4 of 0.96, a hallucination rate of 2.5% and spatial–temporal relevance of 83.8%, whereas GPT-4o obtained 58.3%, 0.86, 15.8% and 41.7% respectively. RedNote's own search had slightly higher precision (78.3%) but weaker spatial–temporal relevance (56.9%). An ablation study showed that removing GeoRAG reduced spatial–temporal relevance from 83.8% to 58.7%, confirming the value of explicit geographic constraints. For recommendation, combining semantic, spatial and popularity signals gave Hit@10 of 0.708 and MRR of 0.421, better than any single factor. Field deployments reported 46% uptake among new students on a university campus within 30 days and 4.1 queries per resident per month in a dense urban community.

**Limitations.** The authors note that the cognitive-map model needs richer individual preference modeling, that the data cover only one city and one platform, and that weights should be tuned by grid search in future work.

### 2.3 Context-Aware Tourism Recommender with Deep Learning and Ontology (Flórez et al., 2025)

**Problem.** Flórez et al. [4] design a recommender for sustainable tourism in the Santurbán páramo, a fragile high-mountain ecosystem in Colombia with limited internet connectivity. The system must work offline, protect privacy and promote responsible behavior.

**Method.** A domain ontology was built in Protégé using OWL 2 and GeoSPARQL. It models points of interest (hotels, viewpoints, natural reserves, safe routes, restaurants), tourist services, and environmental context, with geometries stored as POINT and LINESTRING in Well-Known Text. Ontology data were converted to a feature vector of normalized distance, rating and cost; distance has the highest weight, then rating, then cost. A Keras deep neural network (dense layers of 128 and 64 neurons with ReLU, 30% dropout, and one linear output) predicts a relevance score and is converted to TensorFlow Lite for on-device inference. The final score combines the network (60%) with a local user preference profile (40%). Distance is computed with the Haversine formula. The app is written in Flutter, stores data in SQLite, uses geofencing to push conservation messages near sensitive zones, and works offline.

**Results.** On the validation split (80/20), the model reached RMSE 0.1955, MAE 0.0508, MSE 0.0039, $R^2$ 0.9959 and an AUC of 0.99. In a case study near the town of Berlin, the top recommendations were a nature reserve and two lagoon trails, matching the user's stated interest in hiking and natural sites, while institutional places ranked last.

**Limitations.** The scoring uses only three numeric features, no weather or time context, and a single region; the residual plot shows a very small validation sample, so the near-perfect $R^2$ and AUC should be interpreted with care.

### 2.4 DAIN: Deep Adaptive Interest Network (Huang et al., 2024)

**Problem.** Huang et al. [2] observe that traditional recommenders use static interest models and therefore fail when user interest changes with time, environment and need. They propose the Deep Adaptive Interest Network (DAIN) to model changing interest and to use contextual information in the same network.

**Method.** DAIN has an input layer of sparse user and item identifiers, an embedding layer that converts them into dense latent vectors, several fully connected neural collaborative-filtering layers with ReLU activation, and an output layer that predicts a preference score. Context (time, location, device type) is added as an extra input, so that the prediction becomes $\hat{y} = \sigma(f(p, q, c))$, where p and q are the user and item vectors and c is the context vector. The model is trained by minimizing mean squared error. The reported settings are an embedding size of 64, three layers of 128, 64 and 32 neurons, a learning rate of 0.001 and a batch size of 256.

**Data and results.** Experiments used MovieLens-1M (6,040 users, 3,706 items, about 1.0 million interactions), Amazon Electronics (192,403 users, 63,001 items, about 1.69 million interactions) and Yelp (45,481 users, 11,537 items, about 1.57 million interactions). DAIN was compared with matrix factorization (MF), neural collaborative filtering (NCF) and DeepFM using MAP@10, NDCG@10 and HR@10. On MovieLens-1M, DAIN reached 0.096, 0.158 and 0.318, improving on DeepFM by 7.9%, 8.9% and 6.0%. On Yelp, the dataset closest to local-place recommendation, DAIN obtained MAP@10 of 0.050, NDCG@10 of 0.085 and HR@10 of 0.165, again the best among the compared models.

**Limitations.** The authors acknowledge high computational cost for large datasets, weak handling of noisy or outlier behavior, the absence of multimodal data such as text and images, and the need for privacy mechanisms such as federated learning or differential privacy. Note also that the datasets are general (movies, electronics, business reviews) rather than tourism-specific, so the contribution of context was not tested with live weather or distance.

### 2.5 TourMate: Multi-Factor Tourist Place Recommendation Using Machine Learning (Khan and Chowdhury, 2024)

**Problem.** Khan and Chowdhury [5] note that tourists in Bangladesh lack a tool that suggests suitable destinations according to budget, time and interest. They build the TourMate Android app to predict whether a destination is suitable for a traveller and to recommend local places, hotels, transport and travel agencies.

**Method.** A dataset of 868 responses with 17 features was collected from travellers and travel agencies through Google Forms. Features include sea, mountain, history and entertainment preference, hotel and transport need, trip duration, budget, travel guide, safety, food interest, tourist-friendliness and starting point. Categorical values were label-encoded, important features were identified with an Extra Trees classifier (budget was the strongest), and the data were split 70/30. Five classifiers were trained and tuned with grid search: K-Nearest Neighbors, Logistic Regression, Random Forest, Decision Tree and Naive Bayes. The best model was saved as a pickle file and served through a Flask API hosted on the Railway cloud; the Android app (Java, Firebase for login) calls this API.

**Results.** Random Forest was best, with accuracy about 98%, precision 98%, recall 99%, specificity 97%, F1-score 98% and AUC 1.00. Logistic Regression (91%), Decision Tree (about 96%) and Naive Bayes (91%) followed, while KNN reached only 63%. A survey of 100 reviewers found that more than 60% rated the app features as good.

**Limitations.** The system returns a binary “suitable / not suitable” answer rather than a ranked list, does not use live location or weather, depends on a modest self-reported dataset from one country, and the near-perfect scores suggest that the classifier should be validated on independent data. The authors list personalized itineraries, social features and wider geographic coverage as future work.

### 2.6 Summary of the Review

Taken together, the five studies show a clear progression. TourMate treats recommendation as classification on a questionnaire dataset; DAIN and Doorob use latent-factor and hybrid models trained on interaction data; Flórez et al. add semantic knowledge and on-device inference; and AskNearby adds LLM retrieval, geographic reasoning and explicit modeling of spatial–temporal context. Context enters each system differently: through distance (Doorob, Flórez et al.), through user-stated conditions (TourMate), through location and time semantics (AskNearby) or through a generic context vector (DAIN). None of the five integrates real-time weather as a ranking factor, which is the central context signal of Smart Explore.

## 3. METHODOLOGY

### A. Overview of the Proposed System

SmartExplore is a context-aware nearby place recommendation system that suggests suitable places based on the user's location, weather conditions, preferred category, and available time. It uses React Native with Expo for the frontend and Python FastAPI for the backend. OpenStreetMap provides nearby place data, while Open-Meteo provides real-time weather information. The system follows a client-server architecture to process user context and generate recommendations.

### B. User Location Detection

The system uses the Expo Location API to obtain the user's current latitude and longitude through GPS. Users can also manually search for a location when GPS is unavailable. These coordinates are used to identify nearby places.

### C. Nearby Place and Weather Data Collection

Nearby Points of Interest (POIs), such as parks, cafes, museums, and nature spots, are retrieved using OpenStreetMap through the Overpass API. Weather information, including temperature and current weather conditions, is collected from Open-Meteo. This information helps determine the suitability of different places under current environmental conditions.

### D. Context Analysis and Distance Calculation

SmartExplore considers five contextual parameters: user location, weather conditions, preferred category, available time, and distance. The Haversine formula is used to calculate the geographical distance between the user's location and nearby places.

[
d = 2R \times \arctan2(\sqrt{a},\sqrt{1-a})
]

Where (R) represents Earth's radius (6371 km), and (a) is calculated using the latitude and longitude differences.

### E. Weighted Rule-Based Recommendation Algorithm

The system uses a weighted rule-based scoring approach to rank nearby places. Each place is evaluated using four parameters:

- Category Score: Maximum 30 points.
- Distance Score: Maximum 30 points.
- Weather Suitability Score: Maximum 30 points.
- Rating Score: Maximum 10 points.

The final recommendation score is:

[
S(P) = C(P) + D(P) + W(P) + R(P)
]

The maximum score is 100. Places matching the user's preferences and current conditions receive higher scores. Available time is also considered while prioritizing destinations.

### F. Recommendation Ranking

After calculating the scores, places are sorted in descending order. The highest-scoring places are displayed with their names, categories, distances, available ratings, recommendation reasons, and Google Maps navigation links.

**G. System Architecture and Implementation**

The system consists of three main components:

1. **Frontend:** React Native and Expo provide the user interface.
2. **Backend:** Python FastAPI handles data processing, distance calculation, and recommendation scoring.
3. **External APIs:** OpenStreetMap and Overpass API provide nearby place information, Open-Meteo provides weather data, and Expo Location provides GPS coordinates.

The frontend communicates with the backend through REST APIs to retrieve and display recommendations.

**H. Algorithm: SmartExplore Recommendation Process**

**Input:** User location, preferred category, weather conditions, and available time.

**Output:** Ranked list of recommended places.

**Steps:**

1. Detect the user's current location.
2. Retrieve nearby places using OpenStreetMap.
3. Fetch current weather information.
4. Identify user preferences and available time.
5. Calculate distances using the Haversine formula.
6. Calculate category, distance, weather, and rating scores.
7. Generate the final recommendation score.
8. Rank places in descending order.
9. Display suitable recommendations to the user.

### 3.1 Comparative Analysis

The five studies differ in goal, data and technique, so they are compared along four dimensions: approach, context and data, evaluation results, and system capabilities. Smart Explore is added to the last table to show where the proposed project fits.

#### 3.1.1 Comparison of Approaches

| Study | Year | Region / domain | Core technique | Recommendation type |
|---|---|---|---|---|
| Doorob | 2026 | Riyadh, Saudi Arabia; urban tourism | SAR (CF), Vowpal Wabbit (context), LightFM (hybrid) + facial emotion recognition | Top-K and rating prediction |
| AskNearby | 2025 | Shenzhen, China; neighborhood information | LLM + GeoRAG, GraphRAG, Vector RAG; cognitive-map scoring | Active retrieval and passive recommendation |
| Flórez et al. | 2025 | Santurbán páramo, Colombia; ecotourism | Ontology (OWL, GeoSPARQL) + DNN on TensorFlow Lite + preference fusion | Ranked list of nearby points of interest |
| DAIN | 2024 | General (movies, electronics, local | Deep neural collaborative filtering with context vector | Rating / top-K prediction |

| Study | Year | Region / domain | Core technique | Recommendation type |
|---|---|---|---|---|
| | | businesses) | | |
| TourMate | 2024 | Bangladesh; tourist places | KNN, Logistic Regression, Random Forest, Decision Tree, Naive Bayes | Suitability classification and local-place list |

### 3.2 Strengths and Weaknesses of the Techniques

| Technique | Strengths | Weaknesses |
|---|---|---|
| Collaborative filtering (SAR, SVD, NCF) | Learns from many users; no item features needed | Cold start; needs many ratings; ignores context |
| Deep neural networks (DAIN, Flórez et al.) | Capture non-linear user–item–context relations; scalable | Training cost; need large data; low interpretability |
| Hybrid (LightFM, Doorob) | Mitigates cold start by using item metadata | Rating accuracy can be poor (LightFM RMSE 5.98); more complex |
| Context-aware (Vowpal Wabbit) | Fast online learning; uses live context such as distance | Only as good as the context features supplied |
| Ontology + GeoSPARQL | Explicit semantics; supports spatial reasoning and offline use | Manual construction; small coverage; maintenance effort |
| RAG with LLM | Current local facts; natural-language queries; low hallucination when grounded | Infrastructure heavy (graph, vector, spatial databases); depends on data quality |
| Facial emotion recognition | Implicit feedback without explicit ratings | Privacy and consent issues; bias; modest predictive power |
| Classical ML classifiers (Random Forest, etc.) | Simple, fast, easy to deploy on a mobile backend | Needs labeled data; binary output; no ranking by default |

## 4. Research Gaps

The reviewed studies make real progress, but several limitations recur. Table 4.1 lists the main gaps, the evidence for each in the reviewed papers and the response planned in Smart Explore.

| Gap | Evidence in the reviewed work | Why it matters | Planned response in Smart Explore |
|---|---|---|---|
| G1. No real-time weather in ranking | None of [1]–[5] uses weather as a ranking factor | Outdoor places are unsuitable in rain or extreme heat; indoor options should be preferred | Weather API with weather-suitability score for indoor/outdoor places |
| G2. Weak | Only AskNearby handles | A closed or | Opening status, |

| Gap | Evidence in the reviewed work | Why it matters | Planned response in Smart Explore |
|---|---|---|---|
| visiting-time reasoning | temporal constraints; others ignore opening hours and best time | crowded place is a poor recommendation | available time and best-time-to-visit suggestion |
| G3. Simulated, small or indirect data | Doorob remaps MovieLens ratings; TourMate uses 868 survey rows; Flórez et al. use a small regional set | Reported accuracy may not generalize to real tourist behavior | Live data from Maps/Places and Weather APIs; user feedback stored in MongoDB |
| G4. Little explanation of results | Only AskNearby grounds answers in retrieved text; the others return scores or lists | Users trust and act on recommendations they understand | Recommendation explanation shown with every suggestion |
| G5. Inconsistent and incomplete evaluation | No baseline in [4]; NDCG above 1 in [3]; near-perfect scores without independent test sets in [4], [5] | Results cannot be compared or trusted | Planned tests of location accuracy, API function, relevance and usability |
| G6. Privacy and ethical risk | Camera-based emotion capture in [3]; continuous GPS in [4]; behavioural logs in [2] | Sensitive data need consent and minimal collection | Permission-based location, minimal stored profile |
| G7. Cold start and static preferences | TourMate and [4] rely on one-time questionnaires; emotion in [3] is a modest signal | New users receive generic results | Preference setup plus favourites and history updates |
| G8. Heavy infrastructure or compute | Three-database stack in [1]; deep networks in [2] | Hard to run on student projects and low-end phones | Lightweight weighted scoring first; ML added later |
| G9. Narrow geography and language | China (Chinese data) [1], Saudi Arabia [3], Colombia [4], Bangladesh [5] | Results may not transfer to Indian cities, temples | Focus on nearby places in India; Hindi and English in future |
| G10. Discovery, planning and navigation are separate | [5] gives suitability only; [1] gives answers only; maps are external | Users still switch between several apps | Single app: discovery, weather, recommendations, maps and navigation |

## 5. Challenges and Future Scope

### 5.1 Challenges

The development of an AI-based context-aware nearby places recommendation system faces several challenges. **Data availability and quality** are major concerns because real tourist-behavior data are scarce, ratings are missing for many places (Doorob recorded N/A ratings), and available surveys are small [3], [5]. This issue can be addressed by combining Places API data with in-app feedback and collecting consent-based usage logs. Another challenge is **fusing heterogeneous context**, as location, weather, time, cost, and user preferences have different units and update rates [4]. To overcome this problem, all factors should be normalized to a common scale, and their weights should be adjusted based on user feedback.

The system may also face the **cold start problem**, where new users and newly added places have no historical data [3]. This can be addressed by conducting a short preference survey and using content-based features and popularity information. **Dependence on external APIs** is another important challenge because Maps, Places, and Weather APIs have usage limits, costs, and possible downtime. This issue can be reduced by implementing caching techniques, alternative data sources, and graceful degradation mechanisms. Furthermore, **connectivity** can be a concern in remote areas where internet access may be limited or unavailable [4]. To overcome this limitation, favorite places and recent search results can be stored locally to provide offline access.

Finally, **privacy, consent, and bias** are important considerations because facial data, GPS traces, and user behavioral logs are sensitive, while emotion-recognition models may also introduce bias [3]. These concerns can be addressed through opt-in permissions, on-device processing, and data minimization techniques. By implementing these possible solutions, the proposed system can become more reliable, efficient, secure, and user-friendly.

### 5.2 Future Scope

The reviewed research papers and the proposed SmartExplore system highlight several important future research directions. One major direction is **retrieval-augmented conversational search**, which allows users to ask questions in natural language, such as "Which indoor place nearby is open now?" and receive answers based on verified place data. This approach can follow the GeoRAG concept of filtering places by distance before generating recommendations [1]. Another direction is **richer context integration**, which involves considering additional factors such as temperature, rainfall, humidity, wind speed, weather alerts, crowd levels, festivals, events, travel companions, and user mood, as these factors can influence tourist decisions [3]–[5].

Furthermore, **explainable recommendations** can improve user trust by displaying factor-wise reasons, such as distance, weather suitability, ratings, and opening status, for each suggested place. Users can also be allowed to adjust the weights of different recommendation factors according to their preferences. **Community features** can also be introduced to allow users to share reviews, photographs, and ratings, which will gradually help reduce the problem of data scarcity. Another important direction is **smart itinerary planning**, where the system generates personalized daily travel plans based on weather conditions, available time, distance, and user preferences. Reinforcement-learning approaches for itinerary generation, as discussed in [4], can be explored for this purpose.

Finally, **platform expansion and evaluation** are essential for future development. The application can be deployed on both Android and iOS platforms and evaluated through field studies involving real users. Its performance can be measured using accuracy, relevance, usability, and response-time metrics. These future directions indicate that the field is moving beyond single-factor, accuracy-focused recommendation models towards systems that are context-rich, explainable, privacy-aware, and practically deployed on real devices. SmartExplore aims to contribute to this development by providing a practical and locally relevant mobile application for intelligent nearby-place recommendations.

## 6. Conclusion

This paper reviewed five recent studies on personalized, context-aware recommendation and compared their techniques for nearby-place and tourism recommendation. AskNearby shows that grounding an LLM in geographic, graph and vector retrieval cuts hallucination to 2.5% and sharply improves spatial–temporal relevance [1]. DAIN shows that adding context and dynamic interest modeling to a neural collaborative-filtering network improves MAP, NDCG and hit rate over MF, NCF and DeepFM [2]. Doorob shows that collaborative, context-aware and hybrid models each excel in a different metric and that facial emotion can serve as a complementary implicit signal, with serious privacy caveats [3]. Flórez et al. show that ontology with GeoSPARQL, a compact neural network and a preference profile can deliver private, offline recommendations on a phone [4]. TourMate shows that a Random Forest classifier on multi-factor survey data can predict place suitability with about 98% accuracy and can be deployed through a Flask API and an Android app [5].

The comparison leads to three main findings. First, hybrid and context-aware designs consistently give better results than single-method baselines, and distance and user preference are the most widely used context factors. Second, real-time weather, visiting-time reasoning and explanation of recommendations are missing from all five systems, and several reported results rest on simulated, small or survey-only data. Third, privacy, evaluation consistency and deployment cost remain open challenges.

These findings directly shape the proposed **Smart Explore** application, which integrates location, real-time weather, ratings, opening status, user preferences and available time into a weighted scoring model, explains each recommendation, and offers navigation in one mobile application. Future work will tune the weights from real user feedback, add machine-learning ranking, offline access, multilingual and voice support, and evaluate the system with a field study. In this way the project aims to turn the lessons of the reviewed research into a practical, locally relevant tool for exploring nearby places.

## REFERENCES:

### Research papers reviewed

1. M. Aldayel, A. Al-Nafjan, R. Alwadiee, S. Altammami, A. Alnafaei and L. Alzahrani, “Emotion and Context-Aware Artificial Intelligence Recommendation for Urban Tourism,” *J. Theor. Appl. Electron. Commer. Res.*, vol. 21, no. 3, Art. 95, 2026, doi: 10.3390/jtaer21030095. Research Paper
2. L. Niu, Z. Deng, B. Li, N. Huang, R. Liu and W. Zhang, “AskNearby: An LLM-Based Application for Neighborhood Information Retrieval and Personalized Cognitive-Map Recommendations,” in *Proc. 8th ACM SIGSPATIAL Int. Workshop on AI for Geographic Knowledge Discovery (GeoAI ’25)*, Minneapolis, MN, USA, Nov. 2025, arXiv:2512.02502, doi: 10.1145/3764912.3770813. Research Paper

3. M. Flórez, E. Carrillo, F. Mendes and J. Carreño, "A Context-Aware Tourism Recommender System Using a Hybrid Method Combining Deep Learning and Ontology-Based Knowledge," *J. Theor. Appl. Electron. Commer. Res.*, vol. 20, no. 3, Art. 194, 2025, doi: 10.3390/jtaer20030194. Research Paper
4. S. Huang, H. Yang, Y. Yao, X. Lin and Y. Tu, "Deep Adaptive Interest Network: Personalized Recommendation with Context-Aware Learning," arXiv preprint arXiv:2409.02425, 2024. Research Paper
5. A. Khan and M. Chowdhury, "TourMate: A Personalized Multi-factor Based Tourist Place Recommendation System Using Machine Learning," *Int. J. Intelligent Systems and Applications (IJISA)*, vol. 16, no. 4, pp. 55–71, 2024, doi: 10.5815/ijisa.2024.04.04. Research Paper

**Technical and web sources used in the Smart Explore project**

6. Google Maps Platform "Maps JavaScript API." developers.google.com/maps/documentation/javascript
7. OpenWeather, "Current Weather API." openweathermap.org/api
8. Google Developers, "Geolocation API." developers.google.com/maps/documentation/javascript/geolocation
9. MDN Web Docs, "HTML, CSS, and JavaScript Documentation." developer.mozilla.org
10. React, "React Documentation." react.dev
11. Node.js, "Node.js Documentation." nodejs.org/docs
12. MongoDB, "MongoDB Documentation." mongodb.com/docs
13. World Tourism Organization (UN Tourism), "Tourism and Sustainable Development." unwto.org