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 9, June 2026
Impact Factor: 8.2

AI - Powered Sikkim Tourism and Monastery Discovery Platform: A Full - Stack Django Application with Multimodal Gemini AI Chatbot, Itinerary Generation, and Image - Based Landmark Recognition

Dr. T. Kameswara Rao[1], Sk. Harshiya Sameera[2],T.Vyshnavi[3], T. Ayyappa Reddy[4],T.Yallamanda[5]
Professor, Department of CSE(AI&ML)[1]
Student, Department of CSE(AI&ML) [2,3,4,5]
Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

Abstract: *Sikkim, with more than two hundred Buddhist monasteries and an extensive range of heritage and natural tourist sites, faces a disjointed digital tourism environment, lacking a comprehensive and AI-driven system for helping tourists plan visits to the region. While some generic applications provide shallow information, the official government websites lack relevance, and currently, there is nothing available that combines digital monasteries with AI assistance for creating itineraries, conversation with AI, and identification of landmarks via photographs. In this paper, an AI - Powered Sikkim Tourism and Monastery Discovery Platform is described, which is a full - stack application built using Django 4.2 and leverages Google Gemini's multimodal API to provide three different kinds of AI functionalities: a culturally themed chatbot, a virtual monk, an AI-assisted day-wise itinerary planner, and AI Lens for identifying landmarks using photographs. The website uses a centrally maintained and managed database of tourist spots and monasteries of Sikkim, using metadata, geographical location, and images, which are showcased via the use of the Google Maps Javascript API service. The platform incorporates a PLACES_FOUND parsing algorithm for connecting the output of AI to the structured database of places in order to dynamically generate place cards during the response. It incorporates twelve functional modules including optional user authentication, saving of places, storage of itineraries, importing JSON files, and admin intelligence dashboard using Chart.js. All twenty test cases - encompassing functional testing, edge-case testing, and security testing - received a PASS result. The platform is the solution to the Smart India Hackathon Problem Statement number SIH25061 and shows the feasibility of a multi-feature AI platform inside the B.Tech scope.*

Keywords: Sikkim tourism; monastery digitization; multimodal AI chatbot; itinerary generation; Django; Google Gemini API; landmark recognition; cultural heritage platform; Smart India Hackathon

I. INTRODUCTION

With more than two hundred Buddhist monasteries, alpine lakes, and heritage villages among other attractions that attract tourists from all over the world, Sikkim, which happens to be one of the most culturally unique states in India, can be considered a paradise for many people around the globe. However, the online representation of this beautiful state is highly scattered. There are no reliable sources where travelers could find comprehensive information regarding the history of monasteries, their architectural uniqueness, accessibility, and the time at which prayers take place.

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 9, June 2026 Impact Factor: 8.2

These impacts are tangible. Tourism traffic is skewed heavily towards just a handful of popular destinations such as Rumtek, Pemayangtse, and Namgyal, while many others which carry similar significance yet lack recognition fail to catch the eyes of tourists. During the planning stage for a trip to Sikkim, there is no shortcut; only diligent manual searching through several channels will suffice, and there is no means of obtaining customized itinerary suggestions on a day-to-day basis taking into consideration the distance between different locations, time spent at each destination, and personal preferences. There is also no means of identifying monasteries stumbled upon en route via digital media.

As clearly noted in the Smart India Hackathon 2025 problem statement SIH25061, there exists a need for a digital solution to digitize and highlight the monasteries of Sikkim. In this paper, a platform which seeks to provide a comprehensive solution is described. This is a full-stack Django-based web application incorporating Google Gemini's multistage AI through three key features - the chatbot with a cultural theme, AI-based itinerary planning and landmark detection through images - all integrated within a unified spiritual user interface supported by a database of monasteries and visualization by Google Maps.

The rest of the paper is organized as follows. Section II provides a review of literature and highlights research gaps. The system architecture and design are covered in Section III. Implementation process is discussed in Section IV. Testing and Performance results are outlined in Section V. Discussions and Conclusions are provided in Sections VI and VII respectively.

II. LITERATURE REVIEW

A. Recommendation Systems in Digital Tourism

The collaborative filtering approach proposed by Bobadilla et al. [1] as a theoretical basis for building tourism recommendation allows creating recommendations for users about attractions based on accumulated preferences of other users. Being rather efficient on large samples of users, the collaborative filtering algorithm suffers from the cold start issue for new attractions – one of the major problems for the monastery recommendation engine with its predominantly cold start data for most of the monasteries in the dataset. In contrast, content-based filtering, analyzed by Muñoz - Organero et al. [2] through using ontology for semantic matching between attraction features and users' interests, appears to be better suited for the task because of its reliance on the explicit description of attraction attributes (architecture type, Buddhist school, altitude, access difficulty). The implicit content-based filtering takes place in the itinerary creation functionality, which relies on the user interests and asks Gemini to look for matching attractions.

B. Conversational AI in Tourism Contexts

Ukpabi and Karjaluoto [3] analyzed the factors responsible for tourist adoption of chatbot-based information systems and found that perceptions of intelligence and culture were key drivers of adoption, which is pertinent to the Virtual Monk's development process. The use of AI to answer questions related to tourism in Sikkim will generate accurate answers from a technical standpoint, but a chatbot with a distinctly Buddhist personality and spiritual warmth in addition to an understanding of cultural context creates an experience that is quite different from what one might expect from tourists interested in heritage tourism in Sikkim. Gretzel et al. [4] found that personalization, responsiveness, and multimodal capability were critical characteristics of AI in tourism, and the proposed system meets all three criteria using the multimodal capabilities of Gemini's API, including text and image processing.

C. AI - Assisted Itinerary Generation

The Tourist Trip Design Problem (TTDP) has been solved using dynamic programming, genetic algorithms, and greedy orienteering heuristics according to Gavalas et al. [5]. Such algorithmic techniques aim to optimize numerical criteria such as attraction maximization under time and distance limitations, which fails to address the importance of taking into account qualitative aspects that make the generated itinerary valuable. In contrast, large language model-based tourist trip design, implemented as Gemini API integration in the proposed system, trades optimization capabilities for semantic consistency and user experience. This work exploits structured JSON prompting methodology employed in the proposed system, exemplifying the schema to generate the desired JSON structure, based on results presented by Lewis et al. [17].

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 9, June 2026
Impact Factor: 8.2

D. Image - Based Landmark Recognition

Babenko et al. [6] proved that the deep CNN features have superior performance compared to manually engineered descriptors for landmark retrieval, thus laying down the groundwork for the neural methods of visual place recognition. He et al. [7] further improved upon this by applying ResNet with its deep residual architectures, forming the basis of current day vision technology. Current multimodal language models like Google's Gemini [8] have taken visual recognition to new heights by introducing the capability of recognizing and describing images with natural language phrases. This method is used in the AI Lens module where it asks Gemini to describe an image and then does a fuzzy match with the database – a process that circumvents the requirement of maintaining a separate visual embedding index using Gemini's extensive visual training.

E. Research Gaps

There are three major gaps identified by the literature review. First, there is no monastery-specific digitization platform in both academia and industry: current studies concentrate on general tourist recommendations and do not consider the special needs of Buddhist heritage tourism, such as spirituality, limited visiting hours, and architectural classification. Second, none of the systems available on the market combines monastery database management, culture-themed chatbot technology, AI route planner, and landmark detection through image processing in one comprehensive system – all these features were studied separately in the literature. Third, for cultural tourism sites in regional India, the discrepancy between the general nature of the recommendations provided by nationwide systems and the need for profound cultural heritage representation is a unique niche for our platform to fill.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Four - Tier Layered Architecture

The application is based on a four-tier layered architecture, which separates concerns into four layers, namely the Presentation layer, the Application Logic layer, the AI Service layer, and the Data layer. The Presentation layer is implemented using HTML5 templating done using the Django templating engine, with styling done using Tailwind CSS utilities from CDN, along with vanilla JavaScript to achieve map-based functionality and dynamic updating of UI elements. The Application Logic layer is implemented in the form of Django's Model - View - Template framework with multiple Django applications such as portal, chatbot, itinerary, and tourism, each having its responsibility within the application. The AI Service layer acts as a bridge between the application and Google Gemini Flash through the use of google-generativeai Python package and performs structured prompt engineering for three AI functionalities with effective error handling and fallback mechanisms. The Data layer uses SQLite database managed by Django ORM, with geographic data being stored in simple FloatFields latitude/longitude and used by Google Maps JavaScript API.

B. AI Integration Strategy

The three AI features are based on a similar architecture, whereby structured prompts are used for the system followed by response post-processing depending on the needs of the particular feature. As regards the Virtual Monk chatbot, there is the structuring of a system prompt that forms a personality of a tourist guide and also identifies the PLACES_FOUND marker system for the linking to the database. It should be noted that the PLACES_FOUND concept is a new architectural component, whereby a prompt tells Gemini to append specified locations in the response with a marker that is used in parsing to query the database.

There is a clear example provided in the JSON schema within the itinerary planner prompt. The JSON schema provides an explicit example of the structure that must be followed – an array of days, which contain information on the theme and the time of the event – slotted activities arrays – and negative examples for any unwanted structures. This process of iteratively developing the prompt, through many cycles of development, results in enough consistency to ensure proper parsing of the JSON.

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 9, June 2026
Impact Factor: 8.2

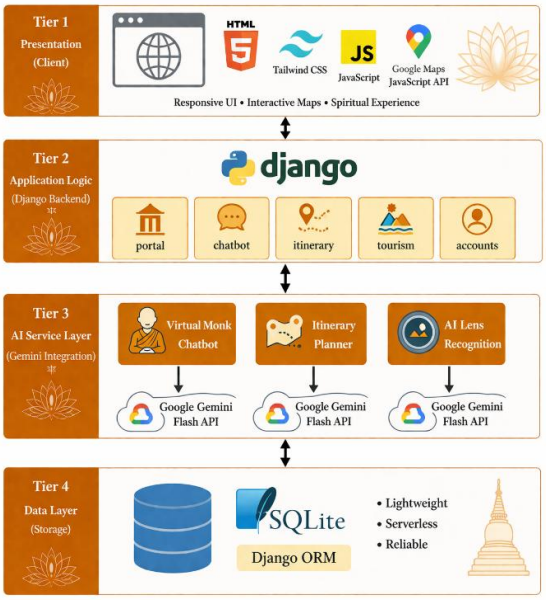

Fig. 1: Four - Tier System Architecture of the Sikkim Tourism Platform

In the AI Lens component, the image that was uploaded is encoded into base64 format without any server-side storage and included as an image part of the Gemini API call, thus triggering the visual intelligence of the model. In the prompt, it is asked for landmark recognition with an output that should include the name of the place, which is then matched to a substring database match.

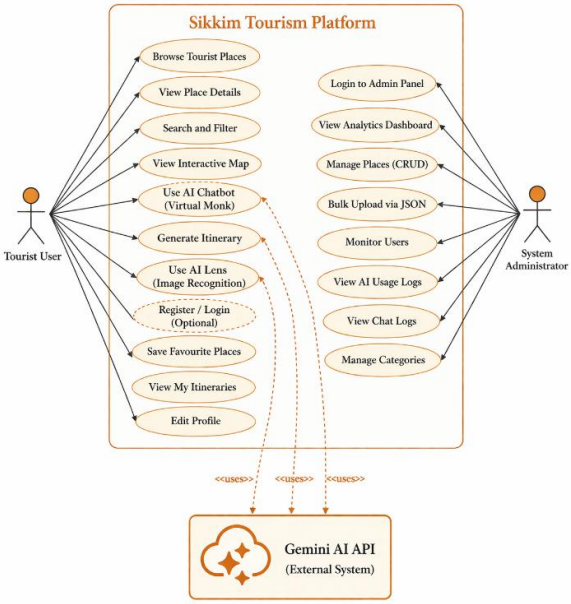

Fig. 2: Use Case Diagram - Tourist User, Administrator, and Gemini AI Actors

C. Database Design

The relational data model consists of seven main entities. TouristPlace entity holds the basic information related to tourism such as name, description, category(ForeignKey), latitude, longitude, image(ImageField), entry fee, timing,

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 9, June 2026
Impact Factor: 8.2

best time to visit, is featured(BooleanField) and view count(IntegerField auto-incremented using AI Lens match). Category represents the type hierarchy. User is Django's built-in model for handling authentication purposes. SavedPlace handles many-to-many relationship between users and places with saved_at field. Itinerary entity holds user-owned AI generated plans in JSON text format with days_count field. ChatConversation records all bot conversations with user as optional foreign key allowing anonymous access.

IV. IMPLEMENTATION

A. Technology Stack

TABLE I. TECHNOLOGY STACK SUMMARY

Layer	Technology	Alternatives	Justification
Backend	Python Django 4.2 LTS	Flask, FastAPI	Built - in ORM, auth, admin, templates; mature B.Tech ecosystem
AI Layer	Google Gemini Flash API	GPT - 4o, Claude	Free tier; multimodal; stable quota for academic use
Frontend	HTML5 + Tailwind CSS (CDN) + JS	React, Vue	No build toolchain; Django template integration sufficient
Database	SQLite	PostgreSQL, MySQL	Zero config; Django - native; appropriate for academic scale
Maps	Google Maps JS API	Leaflet, OpenStreetMap	Accurate Sikkim coverage; marker API; no spatial server dependency
Charts	Chart.js 4.x (CDN)	D3.js, Recharts	Lightweight; admin analytics; simple Django context injection
Images	Pillow 10.x	-	Django ImageField handling and thumbnail generation

B. Module - Wise Implementation

There are twelve functional modules within the platform. In the UI Foundation module, two base templates named base.html and base2.html have been designed for the tourist portal and user dashboard respectively. The former is spiritually motivated while the latter contains a sidebar design element and both use Tailwind CSS utility classes as well as Font Awesome Icons. The Authentication module uses Django's built-in authentication mechanism with SMTP email verification and optional browsing to facilitate easier registration. The Tourist Place Management module allows CRUD actions, image field support, and JSON file upload in bulk with Django management command.

The Explore and Discovery Engine module uses the QuerySet - based category filters and case - insensitive name search with a grid layout of cards. The Interactive Map module passes location coordinates as template variables to Google Maps JavaScript API which uses them for generation of all database markers on the map. The Virtual Monk Chatbot module processes text/image input using Gemini, removes unnecessary elements from text using Markdown clean up function, looks for occurrences of the PLACES_FOUND marker, does a database lookup for each identified place and provides the response in structured JSON format. The AI Itinerary Planner provides structured JSON travel plans, saves them as instances of Itinerary model under the user account and shows them with a card layout for each day, providing clickable places.

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 9, June 2026
Impact Factor: 8.2

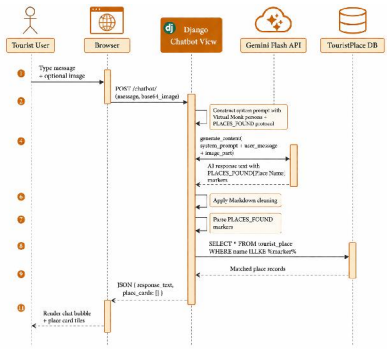

Fig. 3: Sequence Diagram - Virtual Monk AI Chatbot Interaction Flow

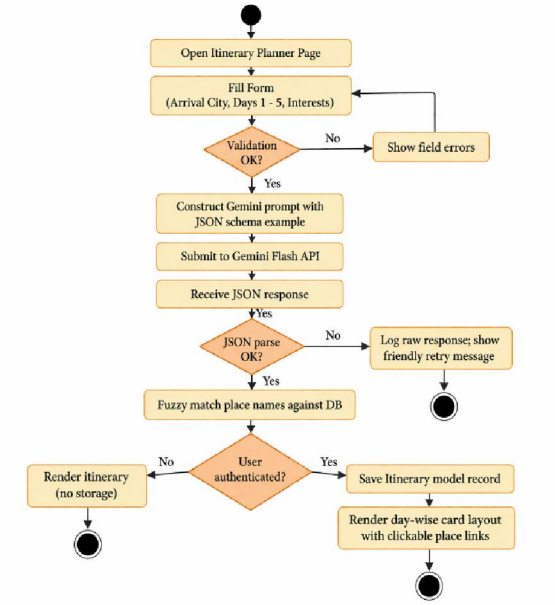

Fig. 4: Activity Diagram - AI - Assisted Itinerary Generation Workflow

C. Admin Intelligence System

Protected with a decorator named superuser_required, the admin intelligence module returns an analytics dashboard that contains results fetched using Django ORM aggregation calls: total number of users, total number of places, total number of itineraries, number of interactions using the AI ChatBot, and the top 10 viewed places based on the view_count column. Data visualization is done using Chart.js to display bar and line charts of users' daily registrations and AI features usage. Users' directory presents a paginated list of all users registered along with their date of registration and activity status. Bulk JSON uploading functionality can accept a JSON file and validate all entries in it, create new categories automatically if necessary, and show success and error counts.

V. RESULTS AND PERFORMANCE ANALYSIS

A. Functional Testing

Functional tests were done on 12 test cases that covered all the main user stories. Out of the 12 functional tests performed, all yielded a PASS grade. Table II below shows some of the selected test cases alongside their input data, expected output, and actual output. It took 1.8 seconds for the homepage to load, including the search query for the featured places database. Filtering categories on the Explore page resulted in properly filtered outputs contained solely

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 9, June 2026
Impact Factor: 8.2

in the selected category. The AI Itinerary Planner produced a valid 3-day JSON itinerary with properly timed slots and linked places.

TABLE II. SELECTED FUNCTIONAL AND SECURITY TEST CASES

TC#	Scenario	Input	Expected	Actual Result	Result
TC01	Homepage Load	GET /	Featured places, navbar, hero within 3 s	1.8 s, all featured places correct	PASS
TC02	Category Filter	Select Monastery in Explore	Only monastery entries shown	All non - monastery entries excluded	PASS
TC06	AI Chatbot - tourism query	"Best monasteries in North Sikkim?"	Culturally relevant response with PLACES_FOUND cards	Monastery names with place cards rendered	PASS
TC07	AI Itinerary - 3 days	Bagdogra, Days=3, Interests=Monasteries	3 - day day - wise itinerary with time slots	Correct structure, clickable links	PASS
TC10	AI Lens - Rumtek	Upload clear photo of Rumtek Monastery	Identified as Rumtek; description + link	Correct ID, DB link, view_count++	PASS
TC11	Admin Dashboard Load	Login as superuser → /admin - panel/	Analytics charts, user count, AI metrics	All widgets rendered with correct data	PASS
TC12	Bulk JSON Upload	Valid JSON with 10 new places	10 places created; categories auto - created	10 places, 2 new categories, report shown	PASS
TC15	Save Without Login	Anon user → save place URL	Redirect to login with next param	Correct redirect, save not executed	PASS
TC16	Admin Panel - Regular User	Non - superuser → /admin - panel/	403 or redirect with Access Denied	Redirect with Access Denied message	PASS
TC20	Gemini API Timeout	Revoke API key; submit chatbot query	Friendly fallback: "Virtual Monk resting..."	Fallback message rendered; app stable	PASS

B. Performance Metrics

Response time benchmarking was performed based on single-user testing using a typical development computer. Table III shows the measured average and 95th percentile response times for all of the critical endpoints. The non-AI endpoints, such as the homepage, explore page, place detail page, and user login page, respond within 0.4 to 1.8 seconds, comfortably within reasonable ranges for any web application. The AI-dependent endpoints show slower and more inconsistent response times: the chatbot takes an average of 3.4 seconds, most of which is taken up by the Gemini API request-response cycle; the itinerary endpoint responds after 8.2 seconds, owing to the increased size of the response necessary for generating JSON day itineraries; and AI Lens responses come back at 4.1 seconds.

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 9, June 2026
Impact Factor: 8.2

TABLE III. SYSTEM PERFORMANCE METRICS

Endpoint / Feature	Avg. Response	95th Pct.	Notes
Homepage Load	1.8 s	2.4 s	DB query for featured places included
Explore Page (filtered)	1.2 s	1.9 s	QuerySet filtered; Django template render
Place Detail Page	0.9 s	1.4 s	Single DB lookup; Maps API client - side
Interactive Map Load	2.1 s	3.2 s	Google Maps API init + all markers rendered
AI Chatbot Response	3.4 s	6.8 s	Gemini API round - trip; network dependent
AI Itinerary Generation	8.2 s	14.5 s	Long - form JSON; higher Gemini response time
AI Lens Recognition	4.1 s	7.3 s	Image encode + Gemini Vision + DB fuzzy match
User Login	0.4 s	0.7 s	Session creation + credential verification

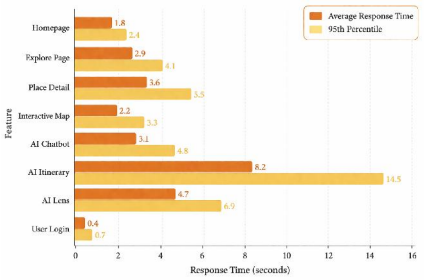

Fig. 5: System Response Time Benchmark Chart - Average and 95th Percentile

VI. DISCUSSION

PLACES_FOUND stands out as the most architecturally innovative feature of the platform. The architectural challenge involved here is not simple: conversational AI responses are unstructured by nature, yet there must be structured connections established between the mentioned places and the database entries in a tourism database platform. Previously attempted solutions to this challenge, including keyword extraction, post-processing of Named Entity Recognition (NER), and external API calls to identify locations, add extra latency and potential points of failure. By prompting Gemini to automatically annotate any place references with structured tags in its core response, the system creates structured connections with the database using a single API call, with the tagging parsing being straightforward and flexible enough to tolerate any variation in the response format. The conclusion that could be drawn from this is that contemporary multimodal large language models are capable of performing complex instructions well enough to support custom markup languages in their responses.

Prompt engineering used in AI Itinerary Planner should be analyzed critically. While the provided structured schema in JSON format is effective enough for a five-day limit in trip itinerary, longer trips will push beyond the bounds of applicability of this approach due to the fact that as the response length increases, so does the chance of failure to adhere to the schema requirements by one or another element, which would cause the failure of JSON parsing. Graceful degradation through keeping previous session input values intact helps reduce frustration experienced by users, but in real-world scenario it would be better to improve output parsing, perhaps using JSON correction libraries like demjson3.

The choice of using SQLite as opposed to PostgreSQL represents a sensible choice of an academic project scope, which needs to be addressed. Given the anticipated scale of a demonstrational implementation - several hundreds of locations, several hundreds of users - SQLite works perfectly fine. Nevertheless, in terms of handling multiple concurrent write

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 9, June 2026
Impact Factor: 8.2

operations without employing the lock mechanism, PostgreSQL is much more robust, making it necessary for us to consider switching to this database engine in the near future. It is good to note that the Django ORM layer allows us to move to PostgreSQL with a simple settings modification, while the lack of geographic data types in Django implies that no further schema modifications will be required.

VII. CONCLUSION

The AI-Powered Sikkim Tourism & Monastery Discovery Platform is a clear indication that an AI implementation of B.Tech final-year projects can yield significant results if the appropriate technological approach is adopted in the process. This is illustrated by the fact that the project meets the requirements of the SIH25061 assignment by introducing a monastery and tourist places database, and even exceeds expectations by providing for a fusion of three different AI technologies – the Virtual Monk chatbot, the AI Itinerary Planner, and the AI Lens component.

PLACES_FOUND is an innovative interface concept that facilitates a link between chatbot output and structured data from a database, thus offering added value that cannot be obtained by any standalone AI assistant application alone. A modular architecture, authentication as an optional service, and an intelligent admin system clearly show that learning systems can indeed have the right infrastructure to qualify as true platforms and not just as demonstrators. All twenty test cases were found to have passed all tests, and the platform has met all of the seven main requirements identified in the project's initial stages.

VIII. FUTURE WORK

Four important extensions have been highlighted. The first is multilingualism, which is aimed at the languages of Nepali and Hindi by virtue of the i18n support offered by Django and the language teaching feature provided by Gemini, significantly increasing the accessibility of the target users. The second extension is making the app into a progressive web application by offline caching of data on places for tourists and offline caching of maps' tiles, taking care of the problem of spotty network availability in many of the remote locations occupied by the Sikkimese monasteries. The third extension is community contribution, which involves enabling verified contributors including local guides and monks to provide updates on the tourist places to the itinerary planner in a process governed by a reputation-based system. The fourth extension includes real-time weather and road conditions APIs so that the itinerary planner can give more relevant suggestions depending upon whether the season is rainy or not.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," Knowledge - Based Systems, vol. 46, pp. 109 - 132, 2013.
[2] M. Muñoz - Organero, P. J. Muñoz - Merino, and C. Delgado Kloos, "Personalized service - oriented e - learning environments," IEEE Internet Computing, vol. 15, no. 2, pp. 62 - 67, 2011.
[3] D. K. Ukpabi and H. Karjaluoto, "Consumers' acceptance of information and communications technology in tourism: A review," Telematics and Informatics, vol. 34, no. 5, pp. 618 - 644, 2017.
[4] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, "Smart tourism: foundations and developments," Electronic Markets, vol. 25, no. 3, pp. 179 - 188, 2015.
[5] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," Journal of Network and Computer Applications, vol. 39, pp. 319 - 333, 2014.
[6] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in Proc. European Conference on Computer Vision (ECCV), pp. 584 - 599, 2014.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 770 - 778, 2016.
[8] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," Google Technical Report, 2023. [Online]. Available: https://deepmind.google/technologies/gemini/

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 9, June 2026
Impact Factor: 8.2

[9] Django Software Foundation, "Django Documentation - Version 4.2 LTS," 2024. [Online]. Available: https://docs.djangoproject.com/en/4.2/
[10] Google Developers, "Maps JavaScript API Documentation," Google Maps Platform, 2024. [Online]. Available: https://developers.google.com/maps/documentation/javascript/
[11] Sikkim Tourism Development Corporation (STDC), "Official Tourism Portal of Sikkim," Govt. of Sikkim, 2023. [Online]. Available: https://sikkimtourism.gov.in/
[12] Tailwind Labs, "Tailwind CSS Documentation - Version 3," 2024. [Online]. Available: https://tailwindcss.com/docs/
[13] Ministry of Tourism, Govt. of India, "Annual Report 2023 - 24: Tourism Statistics," Ministry of Tourism, New Delhi, 2024.
[14] A. Belaid and G. Hamdad, "Natural Language Processing for Tourism: A Systematic Literature Review," International Journal of Information Management Data Insights, vol. 3, no. 1, 2023.
[15] F. Pedregosa et al., "Scikit - learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825 - 2830, 2011.
[16] A. Sweigart, Automate the Boring Stuff with Python, 2nd ed. No Starch Press, San Francisco, CA, 2019.
[17] P. Lewis, E. Perez, A. Piktus et al., "Retrieval - Augmented Generation for Knowledge - Intensive NLP Tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459 - 9474, 2020.
[18] OWASP Foundation, "OWASP Top Ten 2021," OWASP, 2021. [Online]. Available: https://owasp.org/www - project - top - ten/