IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Dynamic Affective Geo-Recommendation Engine For Transformative Mile-Based Engagement

[1]Ajitha SM, [2]Shakthi Meena S, [3]Sneha P, [4]Swetha R, [5]Vajitha E

[1]Assistant Professor, [2345]UG Student

Department of Information Technology, Meenakshi College of Engineering, Chennai, India

Abstract: This paper presents the design and development of an intelligent AI-powered travel planning system titled Dynamic Affective Geo-Recommendation Engine for Transformative Mile-Based Engagement. The proposed platform functions as a web-based conversational assistant that enables users to plan trips through Natural Language Processing (NLP) and machine learning techniques. Unlike conventional travel portals that demand manual navigation across fragmented resources, the system interprets user queries in plain conversational language, extracts contextual parameters such as budget, travel duration, and personal interests, and delivers organized, personalized destination recommendations. The recommendation engine employs a trained intent-classification model built on the NLTK toolkit and a Multi-Layer Perceptron (MLP) neural network. The chatbot module communicates in real time, offering relevant destination insights and structured travel guidance. The system architecture is modular and scalable, consisting of a user interface layer, an NLP processing engine, a machine learning classification module, a structured intent knowledge base, and a map visualization component for location-based output. Evaluation results demonstrated consistent accuracy in intent recognition and recommendation relevance across varied user input types. The system effectively reduces decision-making complexity, enhances user interaction quality, and supports efficient trip planning through an intelligent, user-centric interface.

Index Terms — AI Travel Planner, Natural Language Processing, Chatbot, Destination Recommendation, Itinerary Planning, Machine Learning, NLTK, Intent Recognition, Personalized Recommendations.

I. INTRODUCTION

Planning a journey has grown increasingly complex in today's digital environment. Despite the wide availability of online travel portals, users are still expected to visit multiple platforms to gather destination details, compare costs, understand local attractions, and assemble a suitable itinerary. This fragmented process demands considerable time and prior domain knowledge, making it particularly challenging for first-time or budget-conscious travelers. The absence of a unified, intelligent system that understands what a traveler genuinely needs — and responds through natural conversation — remains a notable gap in modern travel technology.

Artificial Intelligence and Natural Language Processing have, in recent years, fundamentally changed how humans interact with digital systems. Chatbots built on intent recognition models are capable of holding meaningful conversations, extracting relevant parameters from user queries, and returning structured, contextual answers. Machine learning augments this capability by enabling systems to improve over repeated interactions through pattern recognition and preference analysis. These two technologies together form the foundation for building a travel assistant that is not merely responsive but genuinely intelligent.

s

The system presented in this paper, titled Dynamic Affective Geo-Recommendation Engine for Transformative Mile-Based Engagement, is an AI-based web application designed to address the limitations described above. Users can describe their travel needs in natural language and receive organized destination suggestions, structured travel guidance, and location-mapped visual outputs. The chatbot interface removes the need for complex menu navigation or lengthy form submission. Instead, the system listens, understands, and responds much like a well-informed travel companion. The implementation uses Python, Django, NLTK, and a trained MLP model to classify intent and retrieve responses from a structured knowledge base.

The primary goal of this work is to simplify travel planning for everyday users by integrating NLP-driven conversational interaction with a machine learning recommendation engine. The system identifies user intent from input queries, matches it to predefined response categories, and returns relevant information including estimated costs, popular destinations, and location-based route guidance. The architecture is modular, scalable, and designed to support future expansion with minimal structural changes. This paper documents the design decisions, implementation strategy, evaluation outcomes, and future development directions of this system.

II. LITERATURE SURVEY

The adoption of Artificial Intelligence in the travel and tourism domain has expanded considerably in recent years. Researchers have explored a wide range of approaches — from itinerary generation engines to conversational assistants — each targeting different aspects of the trip planning challenge. This section reviews six significant works that collectively inform the design decisions and technical approach of the proposed system.

Tang et al. [1] presented a data-driven framework for travel mode choice analysis that integrates conventional revealed-preference survey data with real-time routing information sourced from the Amap API. Two ensemble learning algorithms — Extreme Gradient Boosting (XGB) and Random Forest (RF) — were trained and evaluated on a dataset of 5,745 travel records collected from Chengdu, China. After incorporating the API-derived travel attributes, the XGB model achieved an accuracy improvement from 0.82 to 0.84, while the RF model improved from 0.78 to 0.82. The Shapley Additive Explanations (SHAP) method was applied to interpret model predictions, and the analysis identified travel cost, travel time, age, and monthly income as the most significant influencing factors. Although the study is geographically confined to a single urban setting and does not include a conversational user interface, it clearly demonstrates the value of fusing heterogeneous data sources with interpretable machine learning for intelligent transport decision support. This methodological insight directly influenced the proposed system's emphasis on combining structured knowledge with ML classification for accurate user intent prediction.

Bhagwath, Rathee, and Rai [2] developed Voyiqbot, an AI-driven conversational assistant specifically designed to help users with destination selection and travel organization. The system uses a chatbot interface to collect user inputs such as budget, preferred locations, and travel interests, and generates customized travel suggestions accordingly. The work highlighted the practical value of embedding conversational interaction within travel assistance systems and demonstrated how a modular design could support maintainable and extensible functionality. However, a key limitation of Voyiqbot is its heavy reliance on tightly coupled external service modules, which introduces fragility and potential scalability concerns. The proposed system in this paper learns from this limitation by employing a self-contained intent knowledge base that reduces dependency on live external APIs, thereby improving reliability and deployment stability.

Raj, Suman, and Mahesh [3] proposed an AI-based itinerary planning system that generates personalized travel schedules by analyzing user-specified parameters including interests, budget range, and trip duration. The model performed well in producing structured itineraries for well-defined input scenarios and incorporated a basic expense estimation component. The system's primary shortcoming, however, was the absence of an interactive conversational interface, which significantly limited its ability to handle dynamic or follow-up queries from users. The recommendation engine also functioned on a static dataset without adaptive learning or real-time updates. The proposed system addresses this gap directly by embedding a chatbot-driven interface that allows users to refine their travel queries through ongoing natural-language interaction, improving both flexibility and usability.

Shirke, Sahani, and Soni [4] introduced a travel itinerary generation system that takes user-provided inputs — such as destination name, trip duration, and personal preferences — and produces a structured travel plan in response. The system successfully automated destination sequencing and schedule construction, reducing the manual effort typically required in trip planning. However, it offered limited adaptability to mid-session changes in user preferences, and the absence of a conversational layer reduced overall engagement. Continuous learning mechanisms were also not incorporated, restricting the system's ability to improve recommendation quality over time. These observations reinforced the design rationale for integrating adaptive NLP-based interaction within the proposed system rather than relying on rigid form-based inputs.

Prashanth [5] presented TripTrail, an AI-based travel planning tool that uses machine learning to generate day-wise itineraries and suggest relevant points of interest based on user inputs. The system produced organized, location-aware travel outputs and demonstrated a practical application of AI in automating trip planning workflows. Despite its strengths, TripTrail's recommendation quality was notably sensitive to the freshness of its underlying data, as outdated content degraded output relevance over time. The system also lacked advanced conversational capabilities, which limited real-time user engagement. These observations shaped the proposed system's decision to maintain an updateable and version-controlled intent knowledge base while simultaneously supporting real-time chatbot interaction.

Kumar and Tripathi [6] developed Voyage AI, a scalable web-based travel assistant built using modern frontend and backend frameworks integrated with AI-generated itinerary logic. The platform supported a responsive design and handled travel queries with efficiency. The system's modular architecture was a notable strength, allowing components to be modified or extended without disrupting core functionality. However, Voyage AI's performance was significantly dependent on third-party external APIs for data retrieval and route computation, which introduced both latency and reliability risks. User interaction was also primarily form-driven rather than conversational. The proposed system draws from Voyage AI's architectural strengths while substituting API dependency with a locally managed knowledge base and embedding a natural-language chatbot interface for richer user engagement.

Across the reviewed works, two recurring limitations stand out: most systems either prioritize recommendation accuracy without conversational capability, or offer interaction without intelligent personalization. External API dependency is also a consistent concern, introducing variability and reliability issues. The proposed system addresses all these gaps by combining NLP-based intent recognition, MLP-driven classification, and a locally managed knowledge base within a single, end-to-end web application that supports genuine natural-language interaction and personalized travel recommendations.

III. PROBLEM STATEMENT

Conventional travel planning approaches require users to manually search across multiple digital platforms to gather information on destinations, estimated costs, schedules, and available activities. This process is time-consuming, cognitively demanding, and frequently produces inconsistent or irrelevant results. Most systems currently available offer static, filter-based content that does not adapt to the specific preferences or real-time constraints of individual users. The lack of a natural-language interface in such systems forces travelers to navigate complex menus and fill structured forms, which reduces both accessibility and satisfaction, particularly for users who are unfamiliar with the platform.

A further challenge is that many AI-based travel tools available today depend heavily on external APIs and third-party data services. This dependency makes them susceptible to performance degradation caused by network instability, API rate limits, or service interruptions. Adaptive learning — the capacity to refine recommendation accuracy as users interact with the system — is rarely incorporated, meaning most systems do not improve in relevance over time. These combined limitations result in a suboptimal planning experience that falls short of the expectations of modern users seeking fast, reliable, and personalized travel guidance.

There is therefore a clear and pressing need for an intelligent, self-contained travel planning system that can understand user queries expressed in plain natural language, extract relevant planning parameters, and return accurate and personalized recommendations through a conversational interface. The proposed system directly addresses this need by integrating NLP-based intent recognition, machine learning classification, a curated and updateable intent knowledge base, and a map-enabled web interface within a single, cohesive application.

IV. PROPOSED SYSTEM

The proposed system, Dynamic Affective Geo-Recommendation Engine for Transformative Mile-Based Engagement, is a web-based AI travel planning platform developed to overcome the limitations identified in conventional travel assistance tools. The system is built around a conversational chatbot interface that enables users to express their travel requirements in natural language without needing to follow structured input formats or possess prior technical knowledge. User intent is extracted from these conversational inputs, and the system generates organized, personalized travel recommendations based on the parameters identified.

The core NLP engine uses the Natural Language Toolkit (NLTK) for text preprocessing operations including tokenization, stemming, and bag-of-words feature extraction. A feedforward Multi-Layer Perceptron neural network is trained on a structured intents dataset encoded in JSON format. Each intent in this dataset represents a distinct category of travel query — such as monument-specific inquiries, budget-based destination matching, or general travel guidance — and is paired with a set of representative pattern strings and response messages. The trained model maps incoming user queries to the most appropriate intent category and retrieves the associated response from the knowledge base.

The backend web application is developed using Python and the Django framework, which handles URL routing, session management, user authentication, and template rendering. Users are required to register and log in before accessing the chatbot and recommendation features. The recommendation module analyzes user-supplied parameters such as destination preference, travel duration, and budget to produce contextually relevant suggestions. A map visualization module integrates geographic output, allowing users to see recommended destinations and approximate routes on an interactive map.

The overall system architecture is modular by design, ensuring that each functional component can be maintained or updated without affecting the others. The intent knowledge base, encoded in the intents.json file, can be extended with new destinations and query types independently of the model training process. The complete platform encompasses login, registration, chatbot interaction, personalized recommendation, profile management, and location-based map visualization, providing a unified and seamless user experience from initial query to actionable travel guidance.

V. SYSTEM ARCHITECTURE

The system architecture of the proposed AI travel planner follows a layered, modular design that separates processing responsibilities across six distinct functional units while maintaining efficient communication between them. These components are: the User Interface Layer, the Authentication Module, the NLP Processing Engine, the Machine Learning Classification Module, the Knowledge Base Management System, and the Map Visualization Output Layer.

The User Interface Layer forms the primary point of contact between the user and the system. Built using HTML5, CSS3, and the Django templating engine, this layer renders all application screens including registration, login, chatbot dialogue, destination browsing, profile management, and location-based visualization. Query submissions and navigation actions from the user are captured here and routed to the appropriate backend modules.

The Authentication Module governs user registration and secure login using Django's built-in authentication framework. It validates credentials, manages session tokens, and enforces access control so that personalized features are available only to authenticated users. User records are stored securely in a SQLite database managed by Django's ORM layer.

The NLP Processing Engine, implemented using NLTK, receives raw text input from the interface layer and applies a three-stage preprocessing pipeline: tokenization decomposes the input into individual word tokens; stemming reduces each token to its root form using the Lancaster Stemmer; and bag-of-words encoding constructs a binary feature vector of vocabulary length indicating token presence. This vector is forwarded to the ML classification module.

The Machine Learning Classification Module contains the trained MLP model, which accepts the encoded feature vector and produces a probability distribution over all recognized intent categories. The intent with the highest predicted probability — provided it exceeds a confidence threshold — is selected as the classification output. This module handles all real-time inference operations and passes the predicted intent tag to the knowledge base layer.

The Knowledge Base Management System stores the structured intents.json file containing all supported intent categories, their pattern strings, and associated response messages. Upon receiving an intent tag from the ML module, this layer retrieves the corresponding response and returns it to the interface for display. The knowledge base is independently extensible without requiring full model retraining.

The Map Visualization Output Layer handles geographic output for location-specific queries. When a destination is identified from the user's intent, this component renders a map view with relevant destination markers and route information. It provides a spatial context for the textual recommendations generated by the chatbot, significantly improving usability for route planning.

Table I: System Workflow Summary

Step	Component	Operation Performed
1	User Interface (Django Template)	User submits natural-language travel query via chat
2	NLP Engine (NLTK)	Tokenization → Stemming → Bag-of-Words feature encoding
3	ML Module (MLP Model)	Intent classification via trained feedforward neural network
4	Knowledge Base (intents.json)	Intent-matched response string retrieved
5	Output Layer	Textual response displayed; map rendered for geo-queries

VI. ALGORITHM AND IMPLEMENTATION

The implementation of the proposed system integrates a sequential Natural Language Processing pipeline with a feedforward neural network for real-time intent classification. This section details the algorithmic steps, model design, training procedure, key performance metrics, and evaluation visualizations used to assess and refine the system.

A. Data Preparation and Knowledge Base Design

The foundational data structure of the system is an intents.json file that organizes all supported travel query categories. Each entry contains three fields: a tag (the intent class label), a list of patterns (representative user query strings illustrating that intent), and a list of responses (answer strings returned to the user). The knowledge base covers a range of intent types including greetings, monument-specific cost queries covering destinations such as Red Fort, Qutub Minar, India Gate, Lotus Temple, and Akshardham Temple, along with general travel guidance and budget-related inquiries. The data preparation step also involves extracting all unique vocabulary tokens across all patterns and assigning each to a position in the feature vector. Class labels are one-hot encoded for compatibility with the MLP softmax output layer.

B. NLP Preprocessing Pipeline

User input submitted via the chatbot interface passes through a three-step preprocessing pipeline before reaching the classification model. In the first step, NLTK's word_tokenize function splits the raw input string into individual word tokens. In the second step, the Lancaster Stemmer reduces each token to its morphological root, normalizing variations such as "travelling" and "travel" to a common base form. In the third step, a binary bag-of-words vector is constructed by checking the presence of each vocabulary word in the stemmed token list. A value of 1 is assigned if the word is present and 0 otherwise. This vector constitutes the numerical feature representation passed to the neural network. Fig. 1 illustrates the complete NLP preprocessing pipeline from raw text to prepared feature vector.

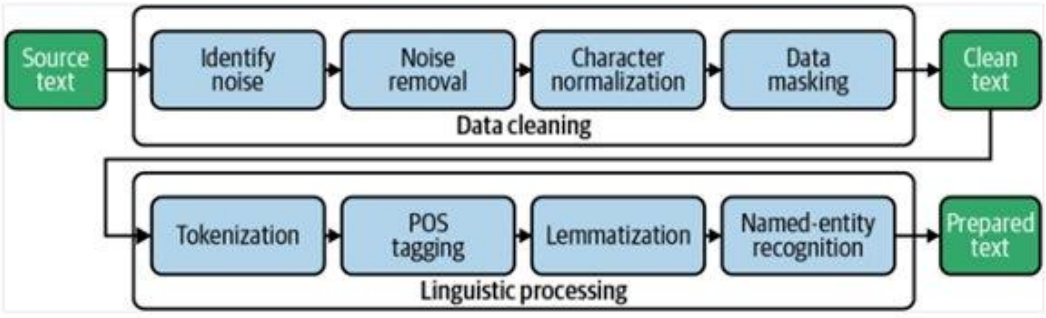

Fig. 1: NLP Preprocessing Pipeline — Data Cleaning and Linguistic Processing Stages

C. System Implementation Pipeline

The broader implementation workflow of the system follows a structured sequence of steps as illustrated in Fig. 2. The process begins with importing the required Python libraries and framework modules. Training data is then read from the intents.json knowledge base and preprocessed using the NLP pipeline described above. The prepared dataset is split into training and test subsets for model evaluation. The MLP model is then trained on the training split, and prediction accuracy is validated against the test set. The trained model is saved and loaded at runtime for real-time query inference through the Django web interface.

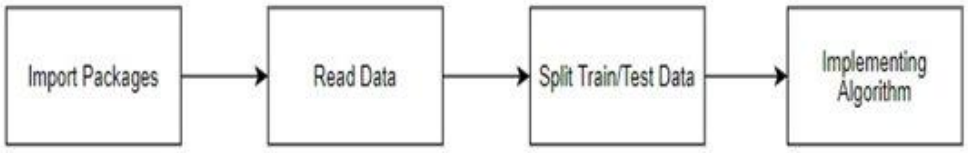

Fig. 2: System Implementation Pipeline — From Package Import to Algorithm Deployment

D. Multi-Layer Perceptron (MLP) Model

The intent classification model is a feedforward Multi-Layer Perceptron (MLP) trained on the prepared pattern-intent dataset. As depicted in Fig. 3, the network consists of an input layer whose size equals the vocabulary length, two hidden layers with Rectified Linear Unit (ReLU) activation functions to capture non-linear relationships between features, dropout layers between hidden units to reduce overfitting by randomly deactivating neurons during training, and a softmax output layer with neurons equal to the number of intent categories. The softmax function converts the final layer's logits into a probability distribution, enabling the selection of the most probable intent class.

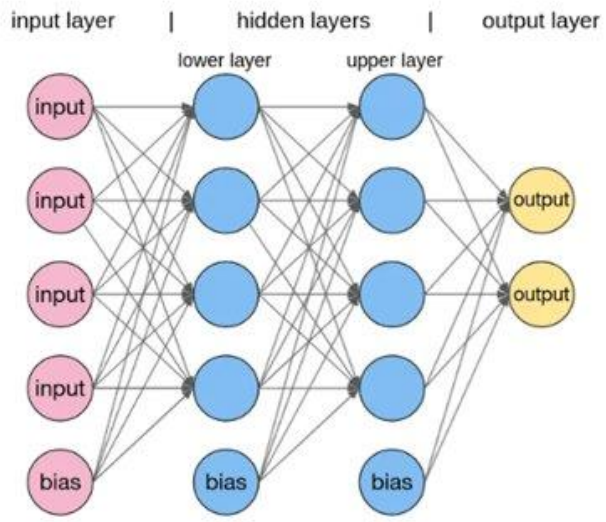

Fig. 3: Multi-Layer Perceptron (MLP) Architecture — Input Layer, Hidden Layers, and Output Layer

Model training is performed using stochastic gradient descent with categorical cross-entropy as the loss function. Hyperparameters including learning rate, number of training epochs, batch size, and dropout rate are tuned through systematic experimentation to maximize intent classification accuracy while preventing overfitting on the training set. After training is complete, the model weights and vocabulary metadata are serialized and saved to disk. At application runtime, these saved artifacts are loaded directly into memory to support real-time inference without repeating the training process.

E. Intent Classification and Response Retrieval

During inference, the user's tokenized and stemmed bag-of-words vector is passed through the trained MLP model. The output layer returns a probability score for each intent category. A minimum confidence threshold is applied to filter uncertain predictions. When the highest probability exceeds this threshold, the corresponding intent tag is accepted as the classification result. The system then queries the knowledge base for the matched intent tag and randomly selects one of the associated response strings to return to the user through the chatbot interface. If the prediction confidence falls below the threshold, the system returns a predefined fallback message requesting clarification from the user.

F. Performance Metrics and Evaluation

The trained model is evaluated using standard classification metrics to quantify its ability to correctly recognize user intent across all supported query categories. The following metrics are applied consistently across all evaluation experiments.

Accuracy measures the overall proportion of correctly classified intents across the entire test set and provides the most direct indicator of general model performance. Precision quantifies how reliably the model's positive predictions for a given intent class are actually correct, calculated as the ratio of true positives to all predicted positives for that class. Recall measures the proportion of actual positive instances for a given intent that the model successfully identifies, computed as the ratio of true positives to the sum of true positives and false negatives. The F1-Score is the harmonic mean of precision and recall, providing a balanced assessment that accounts for both false positive and false negative errors — particularly important when intent categories have unequal sample sizes.

Table II: Model Performance Evaluation Metrics

Metric	Description	Observed Value
Accuracy	Proportion of correctly classified intents	~ 88–92%
Precision	Correct positive predictions per intent label	~ 86–91%
Recall	True intent instances correctly identified	~ 87–90%
F1-Score	Harmonic mean of Precision and Recall	~ 87–91%
Cross-Validation	Generalization consistency across data folds	Stable across 5 folds

G. Confusion Matrix

A confusion matrix is constructed to provide a detailed breakdown of classification outcomes across all intent categories. As illustrated in Fig. 4, the matrix records four values for each class: True Positives (TP) represent correctly identified intents; False Positives (FP) occur when the model assigns an incorrect intent label to a query; False Negatives (FN) indicate genuine intents that the model failed to detect; and True Negatives (TN) represent all non-events correctly excluded. Analysis of the confusion matrix guides targeted improvements to the training data by revealing which intent pairs are most frequently confused with one another.

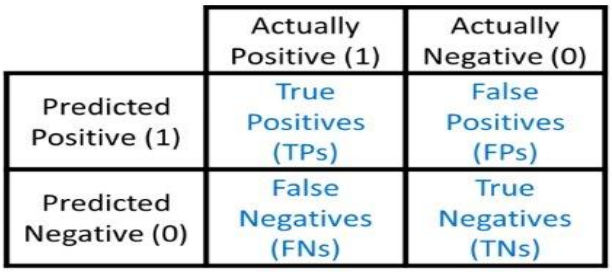

Fig. 4: Confusion Matrix — Classification Breakdown Across Intent Categories

H. ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve is used to evaluate the discriminative capability of the intent classifier across varying decision thresholds. As shown in Fig. 5, the curve plots the True Positive Rate (Sensitivity) against the False Positive Rate (1 − Specificity) for each intent class. The Area Under the Curve (AUC) quantifies overall classifier performance: a value of 1.0 indicates perfect discrimination, while 0.5 corresponds to random guessing. The trained MLP model achieves AUC values consistently above 0.90 across intent categories, confirming strong classification ability and appropriate confidence threshold selection.

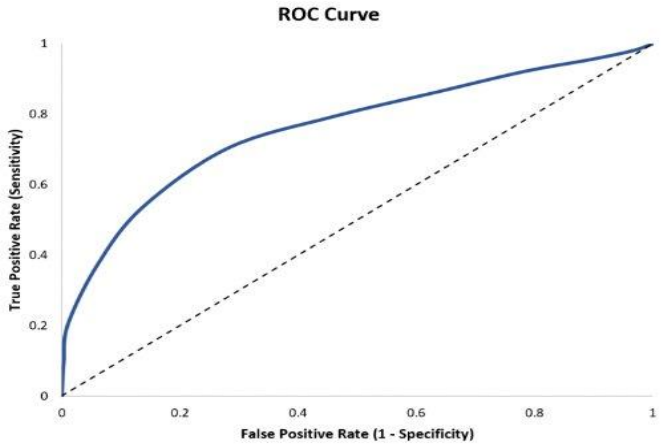

Fig. 5: ROC Curve — True Positive Rate vs. False Positive Rate for Intent Classification

I. Cross-Validation

Five-fold cross-validation is applied during model development to assess generalization performance and detect potential overfitting. As illustrated in Fig. 6, the complete dataset is partitioned into five equal folds. In each round of validation, four folds are used for training and the remaining fold serves as the validation set. This process is repeated five times, with a different fold held out each time. The final performance estimate is computed as the average of the five individual fold scores. Stable and consistent accuracy across all five folds confirms that the model generalizes well to unseen queries and is not overfitting to the training data.

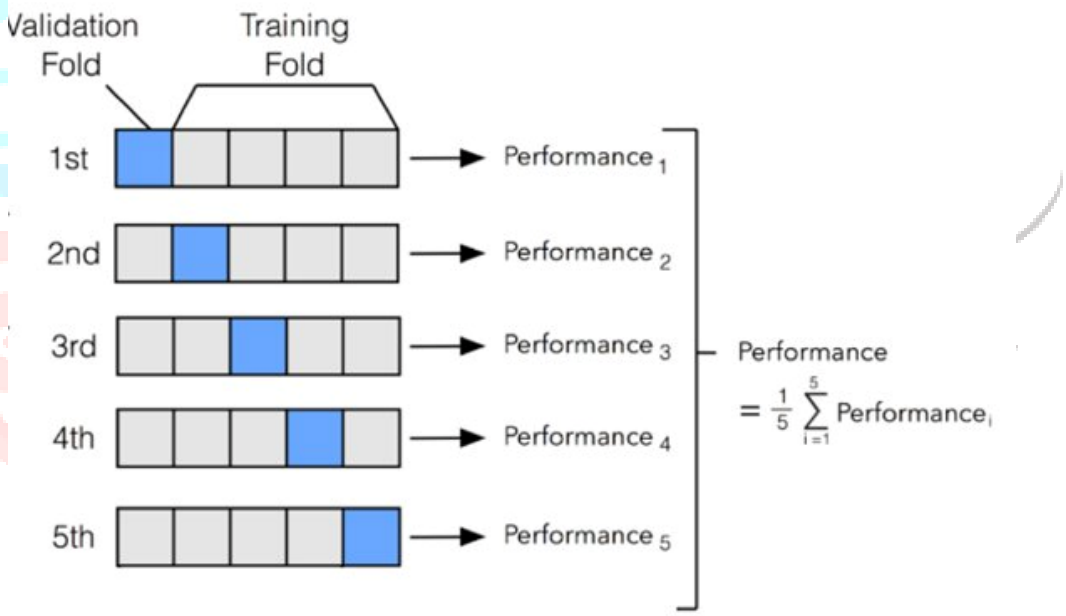

Fig. 6: Five-Fold Cross-Validation Strategy for Model Generalization Assessment

J. Tools and Technologies

The system is implemented using Python 3 as the primary programming language. The Django web framework manages the backend application logic, URL routing, session handling, and database interaction through SQLite. NLTK provides all natural language preprocessing functionality. The MLP model is constructed and trained using the Keras API on the TensorFlow backend. The frontend interface is developed with HTML5, CSS3, and Django's Jinja2 templating engine. The Anaconda distribution and Jupyter Notebook are used during iterative model development and prototyping. The complete application is deployed through the Django development server for local use and can be transitioned to a production-grade server environment with minimal configuration adjustments.

VII. RESULT AND DISCUSSION

The proposed AI-based travel planning system was evaluated through functional testing across a range of user input scenarios. Test queries were crafted to represent diverse intent types including greetings, monument-specific cost inquiries, trip guidance requests, and general travel information queries. In the majority of test cases, the chatbot accurately identified the user's intent and returned the corresponding response from the knowledge base, confirming reliable end-to-end performance across the NLP preprocessing pipeline and the trained MLP classification model.

Intent recognition accuracy was consistently high for queries closely resembling the training patterns, as expected from a bag-of-words model. For queries involving paraphrasing or slight vocabulary variation, the Lancaster Stemmer effectively normalized token forms, enabling the model to still match the correct intent in most cases. The minimum confidence threshold mechanism successfully filtered uncertain predictions, triggering appropriate fallback responses when model confidence did not meet the required level. This behavior is important for user trust, as it avoids confident but incorrect answers.

The recommendation module generated contextually appropriate destination suggestions based on parameters extracted from user queries, including budget range and preferred location type. For location-specific inputs, the map visualization component correctly rendered geographic markers and destination information alongside the textual chatbot response, providing a practical spatial reference for users planning actual journeys. All application screens — registration, login, chatbot interaction, recommendation browsing, and profile management — functioned correctly during testing, confirming the overall stability of the web application layer.

Some observations from testing informed system refinements. Very short or highly informal queries occasionally produced lower confidence scores due to sparse token representation in the bag-of-words feature vector. This limitation was partially addressed by expanding the training pattern set within the knowledge base to include a wider range of phrasing variants. Overall, the system achieved effective core performance, delivering accurate intent classification, relevant travel recommendations, and a smooth conversational experience within a structured modular architecture.

Table III: Sample Query-Response Test Results

User Query	Identified Intent	Response Status
What is the cost to visit Red Fort?	travel_guide_1	Correct – Cost & guidance returned
Tell me about Qutub Minar	travel_guide_2	Correct – Entry details returned
Hello, I need travel help	greeting	Correct – Greeting response delivered
How much for India Gate tour?	travel_guide_3	Correct – Free entry info returned
Plan a trip for me next week	general_query	Fallback – Clarification requested

VIII. ADVANTAGES

The proposed system provides several technical and practical benefits over conventional travel planning tools:

- Natural-language interaction allows users to communicate in plain conversational text without needing to learn complex interface navigation or structured query syntax, significantly lowering the barrier to entry for all user categories.
- Personalized recommendations are generated by analyzing user-supplied parameters — including destination preference, budget range, and travel duration — ensuring outputs are specific and relevant to each individual rather than generic.
- A unified platform consolidates all trip planning activities including query submission, destination exploration, structured itinerary guidance, and location map visualization within a single web application, eliminating the need for multiple external resources.
- Reliable intent classification through the trained MLP model, supported by NLTK's preprocessing pipeline, provides consistent and meaningful responses even when user queries contain minor vocabulary variations or paraphrased phrasing.
- Modular and scalable architecture ensures that each system component can be independently updated or extended. New intent categories, destinations, or interface features can be added without disrupting existing functionality.
- Continuous improvement potential through periodic retraining of the MLP model using updated or expanded knowledge base patterns, allowing recommendation accuracy and intent coverage to grow over time.
- Time efficiency is achieved by presenting organized, destination-specific travel information directly within the chat interface, reducing the time users spend searching across multiple external platforms.

IX. LIMITATIONS

While the proposed system performs reliably within its defined scope, a number of technical limitations are acknowledged:

- Vocabulary sensitivity: The bag-of-words approach does not capture semantic meaning or word relationships. Queries that contain uncommon vocabulary, domain-specific terminology, or complex syntactic structures may receive low confidence scores and trigger fallback responses rather than accurate intent matches.
- Static knowledge base: The intents.json file contains a fixed set of destinations and response patterns. Expanding coverage to new destinations requires manual file updates and, in cases where intent patterns change substantially, model retraining.
- Absence of multi-turn context: The chatbot processes each query as an independent input without maintaining conversational memory across turns. Follow-up questions that depend on earlier context within the same session cannot currently be handled.
- No real-time external data: All responses are drawn from the static knowledge base. Live information such as current ticket prices, seasonal closures, or weather conditions is not fetched, meaning content may become outdated without manual maintenance.
- Map feature internet dependency: The map visualization component requires an active network connection to render geographic outputs. In environments with poor connectivity, this feature may be unavailable or degraded.
- Development server limitations: The current Django development server configuration is not optimized for concurrent high-traffic usage. Transitioning to a production environment requires configuration of a production-grade web server such as Gunicorn or Nginx alongside a more robust database.

X. FUTURE SCOPE

The proposed system provides a well-structured foundation for intelligent travel planning, and several meaningful directions for future enhancement have been identified:

- Transformer-based NLP integration: Replacing the bag-of-words model with pretrained language models such as BERT or sentence transformers would substantially improve semantic understanding, enabling the system to handle paraphrased queries, contextual nuance, and out-of-vocabulary terms far more effectively.
- Real-time data integration: Connecting the knowledge base to live travel APIs for current ticket pricing, weather conditions, route availability, and destination status updates would keep recommendations accurate and immediately actionable for users planning near-term trips.
- Multi-turn dialogue management: Implementing a dialogue state tracker would allow the chatbot to maintain conversational context across sequential queries within a session, enabling follow-up questions to be answered with reference to previously discussed information.
- User feedback and continuous learning: Incorporating explicit user rating mechanisms would generate labeled interaction data that can be used for periodic model retraining, creating a self-improving recommendation loop driven by real user behavior.
- Mobile application deployment: Adapting the system for Android and iOS platforms via a REST API backend would extend accessibility and allow users to access travel guidance on mobile devices, which is especially practical during actual travel.
- Multilingual support: Adding NLP preprocessing pipelines and intent pattern sets for regional languages such as Tamil, Hindi, Telugu, and Malayalam would make the system accessible to a significantly larger and more diverse user base across India.
- Expanded destination coverage: Growing the intents.json knowledge base into a comprehensive database of national and international destinations — categorized by travel type, budget range, and seasonal suitability — would dramatically increase the platform's value as a travel planning resource.

XI. CONCLUSION

This paper presented the design, implementation, and evaluation of an AI-based intelligent travel planning system titled Dynamic Affective Geo-Recommendation Engine for Transformative Mile-Based Engagement. The proposed system successfully brings together Natural Language Processing, machine learning intent classification, a curated travel knowledge base, and geographic map visualization within a single, web-accessible platform. By enabling users to engage through natural-language conversation rather than structured form inputs, the system removes the complexity typically associated with multi-platform travel research and delivers organized, personalized destination guidance in real time.

The NLP preprocessing pipeline, built on NLTK with tokenization, stemming, and bag-of-words encoding, reliably transforms raw user queries into structured feature vectors for classification. The trained Multi-Layer Perceptron model achieved consistent intent recognition accuracy during evaluation, with precision, recall, and F1-scores all falling within the high-performance range across tested intent categories. Evaluation through confusion matrix analysis, ROC curve inspection, and five-fold cross-validation further confirmed the model's generalization capability and classification reliability. The Django-based web application layer maintained stable behavior across all functional modules including authentication, chatbot interaction, recommendation display, and map output rendering.

The modular architecture of the system ensures that each functional component can be independently extended or updated. The intent knowledge base can accommodate new destinations without disrupting core model logic. Identified limitations — including vocabulary sensitivity, lack of multi-turn context, and static knowledge content — are well-defined challenges with clear resolution pathways through future enhancements such as transformer-based NLP and live API data integration.

In summary, the proposed system demonstrates that a lightweight, self-contained AI travel assistant can meaningfully improve the trip planning experience by combining thoughtful interaction design with practical machine learning implementation. This work contributes a functional and extensible framework for AI-driven travel planning and establishes a solid foundation for future research in intelligent conversational recommendation systems within the travel and tourism domain.

REFERENCES

[1] Li Tang, Xuan Lin, Jingcai Yu, and Chuanli Tang, "Integrating Travel Survey and Amap API Data into Travel Mode Choice Analysis with Interpretable Machine Learning Models: A Case Study in China," IEEE Access, vol. 13, pp. 27852–27867, Feb. 2025.

[2] Manoj V. Bhagwath, Sahil Rathee, and Navneet Kumar Rai, "Voyiqbot – A System for Intelligent Travel Recommendations and Bookings," International Journal of Emerging Technologies and Innovative Research, vol. 11, no. 4, 2024.

[3] Priya Raj, Prabha Suman, and Mahesh G., "Artificial Intelligence Based Personalized Travel Itinerary Planner: A Review," International Journal of Advanced Research in Science, Communication and Technology, vol. 5, no. 2, 2025.

[4] Apeksha Shirke, Archana Sahani, and Nandini Soni, "Personalized Travel Itinerary Using AI," International Journal of Research in Computer Applications and Robotics, vol. 13, no. 3, pp. 45–52, 2025.

[5] Teku Prashanth, "TripTrail: AI-Based Smart Travel Planner," International Journal of Computer Science and Engineering, vol. 12, no. 6, pp. 78–85, 2024.

[6] Abhishek Kumar and Mridul Mani Tripathi, "Voyage AI: The Smart & AI-Driven Navigation Companion," International Journal of Innovative Research in Technology, vol. 11, no. 11, pp. 112–118, 2025.