International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

FLIGHTER-AI – AI Based Flight Booking Assistant

P. Vinesh Sri Sai
UG Student, Dept of CSE (Data Science)
Vidya Jyothi Institute of Technology
Hyderabad, Telangana, India
2005vinesh@gmail.com

MD. Saud Baig
UG Student, Dept of CSE (Data Science)
Vidya Jyothi Institute of Technology
Hyderabad, Telangana, India
saudmansurbaig@gmail.com

S. Akhil Rajesh
UG Student, Dept of CSE (Data Science)
Vidya Jyothi Institute of Technology
Hyderabad, Telangana, India
junnuakhil56@gmail.com

R. Sathvik Reddy
UG Student, Dept of CSE (Data Science)
Vidya Jyothi Institute of Technology
Hyderabad, Telangana, India
sathikabhi@gmail.com

Dr. K. S. R. K. Sarma
Assistant Professor,
Dept of CSE (Data Science)
Vidya Jyothi Institute of Technology
Hyderabad, Telangana, India
kaipasarma@gmail.com

How to Cite this Article:
Sai, P. V. S., Baig, M. S., Rajesh, S. A. & Reddy, R. S. (2026). FLIGHTER-AI – AI Based Flight Booking Assistant. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.938

https://doi.org/10.55041/ijcope.v2i4.938

ABSTRACT— Flight booking applications help users search, compare, and reserve airline tickets efficiently. It is important for such systems to provide accurate, real-time information and a smooth booking experience to meet user expectations. This project focuses on developing an AI-based flight booking application that allows users to search for available flights, compare fares, and complete bookings through a single platform. By using live flight data such as departure time, arrival time, travel duration, and ticket price, the system presents clear and standardized information to users for better decision-making. The application integrates real-time airline data services to fetch up-to-date flight information and uses secure online payment processing to complete reservations. A relational database is used to store booking, passenger, and transaction details reliably. Automated confirmation and ticket delivery improve efficiency and reduce manual effort. The system also supports future enhancements such as personalized recommendations and intelligent fare analysis. The goal of this project is to provide a reliable, scalable, and user-friendly flight booking solution that enhances the overall travel booking experience.

INTRODUCTION

This document presents a comprehensive system design and implementation report for an AI-Powered Flight Booking System — a production-grade, full-stack web application that leverages large language model (LLM) technology to deliver a fully conversational flight booking experience. Unlike traditional booking platforms that rely on rigid form-based interfaces, this system enables users to interact naturally through a chat interface, expressing their travel intent in plain language and receiving intelligent, structured responses in real time.

The system integrates a React-based chat frontend with a Python FastAPI backend, a Groq-hosted LLaMA Instant language model for natural language understanding, the Amadeus GDS API for live flight data, the Tavily API for hotel search via web intelligence, and Stripe for secure payment processing. Data persistence is managed through a PostgreSQL relational database accessed via the SQLAlchemy ORM.

The architecture follows a clean separation of concerns: the LLM is solely responsible for intent classification, entity extraction, and response formatting, while all business logic — including API calls, booking creation, payment orchestration, and passenger management — is handled exclusively on the backend. This design ensures reliability, auditability, and security.

Key highlights of the system include a stateful conversational pipeline, real-time frontend synchronization via a polling mechanism, webhook-based idempotent payment confirmation, a rich database schema supporting multi-passenger bookings and reusable passenger profiles, and robust error handling at every layer. The system is designed for extensibility, with a clear pathway toward microservices, caching layers, and additional travel verticals.

This report covers the complete technical specification of the system, including architectural design, component breakdown, database schema, API design, data flow diagrams, security model, performance considerations, and identified limitations with proposed future enhancements.

I. PROBLEM DEFINITION

The rapid maturation of large language models (LLMs) presents a compelling opportunity to reimagine the travel booking workflow. An AI-powered chat interface can understand natural language queries such as 'Find me a direct flight from London to New York next Friday, economy class for two adults,' extract all required structured fields, execute the appropriate backend operations, and return a formatted, human-readable response — all within a single, fluid conversational turn.

However, naively delegating business logic to an LLM introduces significant risks: LLMs are prone to hallucination, non-determinism, and unreliable structured output. A robust system must therefore use the LLM narrowly and precisely — for language understanding — while keeping all execution logic firmly in deterministic backend code.

Conventional flight booking platforms — such as those built on legacy GDS (Global Distribution System) interfaces — present users with complex, multi-step form-based workflows. Users must independently navigate origin and destination fields, date selectors, passenger type dropdowns, fare class filters, and seat maps. This paradigm, while functional, creates significant friction, particularly for non-technical users, travellers with complex itineraries, or those unfamiliar with airline jargon such as IATA codes, fare classes, or ticketing rules.

The cognitive load imposed by such interfaces often leads to booking abandonment, user errors, and a poor overall customer experience. Furthermore, traditional systems rarely maintain conversational context — if a user changes their mind mid-flow, they must restart the process entirely rather than simply expressing a new intent.

1.2 PROJECT FEATURES

The chat interface is the primary user touchpoint of the system. Built in React, it renders a real-time message stream resembling popular messaging applications such as WhatsApp or Slack. Users type natural language queries — for example, 'I want to fly from Dubai to London on the 15th of next month, two adults, business class' — and receive structured, formatted responses from the AI agent. The interface maintains a visible message history, displays loading indicators during processing, and supports multi-turn dialogue, allowing users to refine their requests, ask follow-up questions, and navigate the complete booking flow without leaving the chat window.

Related Work

Traditional flight booking systems such as Expedia, Skyscanner, and MakeMyTrip rely heavily on structured, form-based user interfaces where users must manually input parameters such as origin, destination, travel dates, and passenger details. While these platforms provide comprehensive search and comparison capabilities, they often lack flexibility in handling ambiguous or conversational user queries. The rigid interaction model increases cognitive load and may lead to user friction, especially for non-technical users or those unfamiliar with airline-specific terminology such as IATA codes or fare classes.

Recent advancements in Artificial Intelligence, particularly in Natural Language Processing (NLP), have enabled the development of conversational agents and chatbots for travel planning. Research in this area has explored the use of large language models (LLMs) to interpret user intent and extract structured information from unstructured text. Systems leveraging transformer-based architectures have demonstrated strong capabilities in intent classification, entity recognition, and dialogue management. However, many existing implementations either rely entirely on rule-based pipelines or over-delegate critical business logic to AI models, leading to issues such as inconsistency, hallucination, and lack of reliability.

Several studies have proposed hybrid architectures that combine AI-driven language understanding with deterministic backend systems. In such approaches, the LLM is used strictly for interpreting user input and generating human-readable responses, while all core operations such as API calls, database transactions, and payment processing are handled by controlled backend logic. This separation ensures system reliability, auditability, and security, especially in domains involving financial transactions and real-time data integration.

In the context of travel systems, integrations with Global Distribution Systems (GDS) such as Amadeus have been widely adopted to retrieve real-time flight data. Similarly, payment processing platforms like Stripe provide secure and scalable solutions for handling online transactions through webhook-based confirmation mechanisms. While these technologies are well-established individually, their integration into a unified conversational workflow remains an area of ongoing development.

The proposed Flighter AI system builds upon these existing approaches by combining a conversational interface with a modular backend architecture. Unlike traditional systems, it enables users to interact using natural language while maintaining strict backend control over execution logic. By integrating LLM-based intent detection with real-time APIs and a robust database design, the system aims to deliver a more intuitive, reliable, and scalable flight booking experience.

II. METHODOLOGY

1. User Input
The user enters a flight query through a chat-based interface.

2. Intent Detection & Entity Extraction
A large language model (LLM) processes the input to identify user intent (e.g., search, booking) and extract key details such as origin, destination, and travel date.

3.Validation
The extracted data is validated, and missing information is requested from the user if necessary.

4. Backend Processing
Based on the intent, the system calls appropriate services

5. Database Management:
User, booking, and passenger details are stored in a PostgreSQL database.

III. PROPOSED SYSTEM

The proposed system, Flighter AI, is an AI-powered conversational flight booking application that enables users to search, compare, and book flights using natural language interactions. Unlike traditional form-based systems, it integrates a chat interface with a large language model (LLM) for intent detection and entity extraction, while all core operations such as flight search, booking, and payment processing are handled by a secure backend. The system utilizes real-time flight data APIs, a relational database for managing users and bookings, and a webhook-based payment workflow to ensure reliability and scalability. This approach provides a more intuitive, efficient, and user-friendly travel booking experience.

IV. IMPLEMENTATION DETAILS

The system is implemented using a React-based frontend for chat interaction and a FastAPI backend for handling business logic. The backend integrates with external APIs such as Amadeus for flight data, Tavily for hotel search, and Stripe for payment processing. PostgreSQL is used for database management, and JWT is used for authentication. The system processes user queries, retrieves relevant data, and returns structured responses through the chat interface.

4.1 ALGORITHMS USED

The system primarily utilizes Natural Language Processing (NLP) algorithms based on transformer architecture to perform intent classification and entity extraction from user queries. A large language model (LLaMA) is used to interpret conversational input and convert it into structured data. Additionally, rule-based validation algorithms are applied to ensure completeness and correctness of extracted parameters. The system also employs standard search and filtering logic for processing flight data, along with polling algorithms to periodically check booking and payment status. Together, these approaches enable efficient, accurate, and real-time execution of the flight booking workflow.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The system was tested across various scenarios including flight search, booking, and payment workflows. The results demonstrate accurate intent detection and smooth conversational interaction. Real-time API integration ensured efficient data retrieval, and payment processing worked reliably with webhook confirmation. The polling mechanism successfully updated booking status. Overall, the system provides a user-friendly and efficient flight booking experience.

System Interface – Home Page:

The above figure shows the main interface of the system where users can perform all features

Fig. 1. Home UserInterface

In this figure, the user interacts with UI

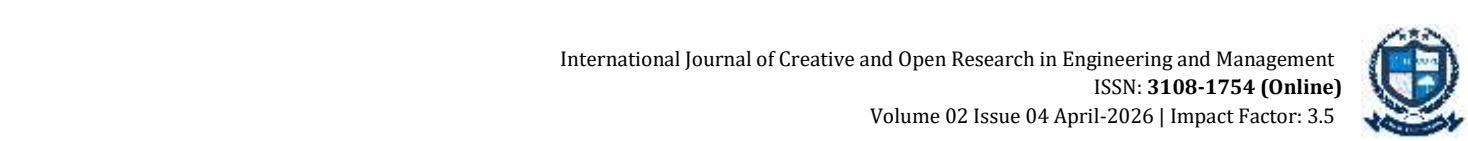

Fig. 2. Flight selection UI

Fig. 3. Booking Page

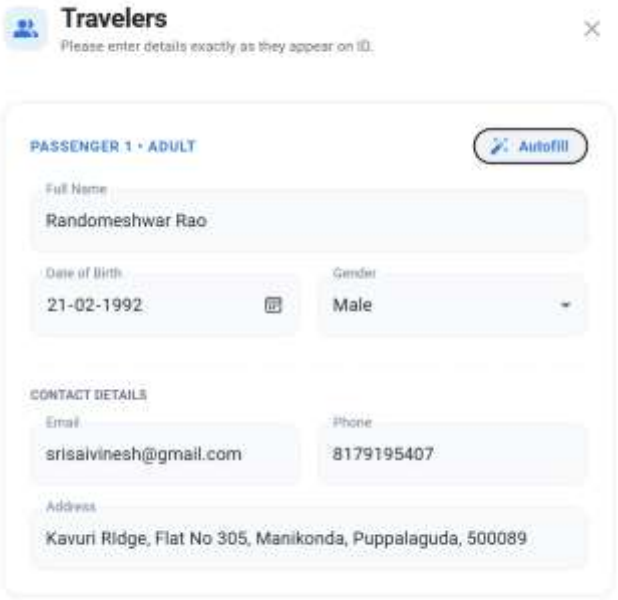

Fig 3: gives the UI of final booking of flights for the passengers

VI. CONCLUSION

The Flighter AI system successfully demonstrates the integration of conversational AI with a structured backend for flight booking. It improves user experience by enabling natural language interaction while ensuring reliability through backend-controlled execution. The system is scalable, efficient, and suitable for modern travel applications.

VII. FUTURE SCOPE

The proposed Flighter AI system can be further enhanced by integrating advanced AI capabilities such as personalized recommendations based on user preferences and travel history. Future improvements may include support for multi-language interaction to cater to a global audience and the development of mobile applications for better accessibility. The system can also be extended to include additional travel services such as cab booking, travel insurance, and itinerary planning. Implementing real-time price prediction and fare optimization using machine learning models can further improve user decision-making. Additionally, migrating the system to a microservices architecture with caching and scalability enhancements will make it more efficient and suitable for large-scale real-world deployment.

VIII.ACKNOWLEDGMENT

We would like to express our sincere gratitude to our project guide, **Dr. K. S. R. K. Sarma**, Associate Professor, Department of Computer Science and Engineering (Data Science), Vidya Jyothi Institute of Technology, Hyderabad, for his valuable guidance, continuous support, and encouragement throughout the development of this project. His insightful suggestions and motivation greatly contributed to the successful completion of this work.

We would also like to thank the **Head of the Department and faculty members of the CSE (Data Science) department** for providing the necessary support and resources required for carrying out this project. We extend our sincere thanks to the **Principal and management of Vidya Jyothi Institute of Technology** for providing the infrastructure and academic environment that helped us complete this project successfully.

Finally, we express our heartfelt gratitude to our **parents, friends, and well-wishers** for their constant encouragement and support during the course of this work.

IX. REFERENCES

[1] T. M. Mitchell, Machine Learning. New York, USA: McGraw-Hill, 1997.

[2] Vaswani et al., “Attention Is All You Need,” Advances in Neural Information Processing Systems (NeurIPS), 2017.

[3] OpenAI, “GPT Models and Applications,” Available: https://platform.openai.com/docs/

[4] Groq Inc., “Groq LPU and LLaMA Model Documentation,” Available: https://groq.com/

[5] Amadeus for Developers, “Flight Offers Search API,” Available: https://developers.amadeus.com/

[6] Tavily, “AI Search API Documentation,” Available: https://tavily.com/

[7] Stripe, “Stripe Checkout and Webhooks Documentation,” Available: https://stripe.com/docs

[8] FastAPI Documentation, Available: https://fastapi.tiangolo.com/

[9] PostgreSQL Global Development Group, "PostgreSQL Documentation," Available: https://www.postgresql.org/docs/

[10] React Documentation, Available: https://react.dev/

[11] SQLAlchemy Documentation, Available: https://docs.sqlalchemy.org/

[12] PyJWT Documentation, Available: https://pyjwt.readthedocs.io/

[13] Hugging Face, "Transformers Library Documentation," Available: https://huggingface.co/docs/transformers

[14] Jurafsky, D. and Martin, J. H., Speech and Language Processing, 3rd ed., Pearson, 2023.

[15] Amatriain, X., "Building Machine Learning Powered Applications," O'Reilly Media, 2020.