

# Customer Conversational Intelligence Platform Powered by an LLM Agent

**SkyWings Airline Assistant**

For the completion of

**PG-level Advanced Certification Programme in  
Computational Data Science**



**Indian Institute of Science**

**Cohort 8**

## **Group 1**

- Ranjith Menon
- Pankaj Goyal
- Satgur Prasad Rao
- Sree Lakshmi
- Aditi Ravishankar
- Amit Dutta
- Harshal R Chikhale
- Vikas Yadav
- Sajeesh K K
- Anusha Harlapur
- Raghavendra

**Mentor: Pratyusha M**

# Table of Contents

<b>1 Problem Statement.....</b>	<b>2</b>
<b>2 Background.....</b>	<b>2</b>
<b>3 Motivation.....</b>	<b>3</b>
<b>4 Data Challenge and Approach.....</b>	<b>3</b>
4.1 Dataset Exploration.....	4
4.1.1 RSiCS Dataset (Relational Strategies in Customer Support).....	4
4.1.2 3K Conversations Dataset.....	4
4.1.3 Twitter Customer Support (TWCS) Dataset.....	4
4.2 Data Synthesis and Manual Labeling.....	5
4.2.1 Intent Classes for Manual Labeling.....	5
4.3 Model Evaluation.....	5
4.3.1 Models.....	5
4.3.2 Evaluation Results.....	6
<b>5 Transition to RASA NLU.....</b>	<b>6</b>
<b>6 Architecture Overview (End-to-end pipeline).....</b>	<b>8</b>
<b>7 Key Components.....</b>	<b>9</b>
7.1 User Interface.....	9
7.1.1 Streamlit.....	9
7.1.2 WhatsApp.....	10
7.2 Integration Layer.....	11
7.2.1 Twilio Server.....	11
7.2.2 FastAPI Integration.....	11
7.3 Conversational AI Layer.....	11
7.3.1 RASA Core.....	11
7.3.2 RASA Action.....	11
7.4 Data Management Layer.....	12
7.4.1 DB Server.....	12
7.5 Policy Knowledge Base.....	12
7.6 Retrieval-Augmented Generation.....	12
7.6.1 Ollama Server.....	12
<b>8 LLM Performance Assessment.....</b>	<b>13</b>
<b>9 Analytics Dashboard.....</b>	<b>13</b>
<b>10 Challenges and Learnings.....</b>	<b>15</b>
10.1 Key Challenges.....	15
10.2 Learnings.....	16
<b>11 Future Roadmap.....</b>	<b>16</b>
<b>12 Conclusion.....</b>	<b>17</b>
<b>13 References.....</b>	<b>17</b>
<b>14 Codebase.....</b>	<b>17</b>

# 1 Problem Statement

Existing customer support systems in the airline industry are often slow, inconsistent, and unable to handle transactional requests at scale. Generic chatbots lack the domain understanding, contextual memory, and backend integration needed to provide accurate, real-time assistance for tasks like bookings, cancellations, and policy queries. There is a clear need for a domain-specific AI assistant that can combine natural language understanding with actionable system access to improve customer experience and operational efficiency.

## 2 Background

In the digital era, customer expectations have evolved dramatically across all service-oriented industries, and the airline sector is no exception. Travelers today expect instantaneous, personalized, and reliable support, whether they're booking a flight, checking baggage policies, requesting special assistance, or dealing with last-minute changes. Traditional customer service models—based on phone calls, emails, or even generic live chat—are increasingly viewed as inefficient, slow, and fragmented, particularly during peak travel seasons, disruptions, or emergencies.

Airlines typically handle millions of customer interactions annually, many of which involve repetitive, rule-based queries. These include questions about flight availability, fare rules, cancellation policies, baggage allowances, and COVID-19 regulations. Responding to such queries with human agents alone leads to long response times, high labor costs, and limited scalability. Moreover, these conventional systems often fail to deliver consistent information, especially when customer service agents are distributed across regions and time zones.

- **Complex and Structured Domain:** Generic bots trained on open-domain data often misinterpret or mishandle this complexity, leading to incorrect responses or broken user experiences.
- **Need for Transactional Capabilities:** Most airline customer queries are not just informational—they are actionable. Customers frequently want to book flights, cancel reservations, and check PNR status. Out-of-the-box chatbots generally lack these transactional capabilities.
- **Multi-Turn, Contextual Interactions:** Air travel planning often involves multi-turn conversations with interdependent context—e.g., “I want to fly from Delhi to Bangalore on Friday... what’s the cheapest fare?” followed by “What about in the morning?” Such interactions demand memory, disambiguation, and intent tracking, which most generic NLP models are not equipped to manage effectively without domain-specific tuning.

- **Policy & Document-Driven Information:** Airlines must provide accurate answers based on ever-changing internal policies, regulations, and procedures. This includes travel restrictions, refund eligibility, special assistance, and baggage rules. A domain-specific assistant must be capable of retrieving and summarizing relevant policy content in response to natural language questions, something generic LLMs often cannot do without hallucinating or missing nuance.
- **Omnichannel Expectations:** Customers now expect support across multiple channels - web and messaging platforms like WhatsApp. A robust airline assistant must be available 24/7 across all these platforms, offering consistent functionality, tone, and branding.

### 3 Motivation

Traditional customer service models, reliant on human agents, often struggle with high volumes of repetitive queries, leading to long wait times and inconsistent service. Additionally, nuanced or complex queries related to airline policies, special requests, or urgent issues often require human intervention, but customers do not want to wait long for answers. AI-driven chatbots can address routine queries instantly, automating tasks like bookings, cancellations, and flight status updates while redirecting more complex questions to human agents or providing relevant links and FAQs. This ensures that customers get quick responses for simple tasks, while still having access to expert support for detailed or policy-specific issues.

With customers increasingly expecting 24/7 support across various platforms, AI chatbots help airlines scale their operations while improving efficiency and customer satisfaction. By automating repetitive interactions and redirecting nuanced queries to appropriate resources, airlines can offer faster, more reliable service, reduce operational costs, and stay competitive in a technology-driven landscape.

### 4 Data Challenge and Approach

Developing a robust and domain-aware conversational AI system for the airline industry presents a unique set of challenges, particularly around the availability, structure, and specificity of training data.

Early in our development, we identified several limitations with data availability. We followed a structured three-step approach towards addressing our data availability challenges:

## 4.1 Dataset Exploration

Our project began with the analysis of publicly available conversational datasets to assess their suitability for the airline sector. Three primary datasets were explored.

### 4.1.1 RSiCS Dataset (Relational Strategies in Customer Support)

- Focus: Multi-intent detection, utterance reduction, relational emotion tagging (e.g., rant, gratitude).
- Limitation: Lacks concrete airline-specific intents (e.g., "change booking", "check baggage allowance").
- Value: Excellent for pre-processing and LLM prompt optimization.
- Insight: 15.5% of messages are multi-intent, with longer, verbose text ideal for simplification.

### 4.1.2 3K Conversations Dataset

- Focus: Natural, casual dialogue modeling.
- Limitation: Generic and not task-oriented; no airline-relevant vocabulary.
- Value: Useful for response style modeling and tone consistency.

### 4.1.3 Twitter Customer Support (TWCS) Dataset

- Focus: Customer-agent tweet interactions, especially around service issues.
- Strength: Airline-specific tweets were successfully isolated through keyword filtering.
- Limitations: Skewed toward complaints and customer support, lacking diversity in service categories like infant travel or check-in assistance.
- Techniques Used: Sentence-BERT + KMeans clustering, PCA visualization for dominant intent groupings.

Capability	RSiCS	3K Conversations	TWCS (Twitter)
Intent Classification	✗ Structural only	✗ Not suitable	✓ Airline-rich
Multi-Intent Detection	✓ 15.5%	✗	⚠ Implicit
Tone/Emotion Tags	✓ Rant, Gratitude	⚠ Basic	✓ 79% Negative
Airline Relevance	⚠ Sparse	✗ None	✓ Strong
Dialogue Style	⚠ Limited	✓ Coherent	✓ Realistic

 *Visual: Comparative Matrix of Dataset Capabilities*

## 4.2 Data Synthesis and Manual Labeling

Despite deriving valuable insights from the three datasets, we faced key limitations:

- Lack of task-oriented, airline-specific intents.
- No pre-labeled classes such as “Modify Ticket”, “Travel Documents”, “Infant Policy”, etc.
- Inadequate coverage of diverse customer queries encountered in real-world airline operations.

To bridge this gap, we turned to data synthesis.

We manually curated and labeled ~1600 synthetic queries across 12 business-critical intents, including:

### 4.2.1 Intent Classes for Manual Labeling:

Baggage, Check-in & Boarding, Booking Modifications and Cancellations, Travel Documents, Fares & Payments, Refunds, Flight Operations, Passenger Services, Loyalty and Rewards, Customer Support, Other, and Irrelevant.

Each query was crafted to resemble realistic customer inquiries, drawing from known airline FAQs and operational scenarios.

## 4.3 Model Evaluation

Following the generation of high-quality synthetic queries and their manual labeling across 12 foundational intent categories, we conducted a series of modeling experiments to evaluate which classification pipeline would best serve our airline-specific chatbot system. We adopted BERT embeddings as the common input representation strategy for all models, ensuring consistent semantic encoding of queries.


### 4.3.1 Models

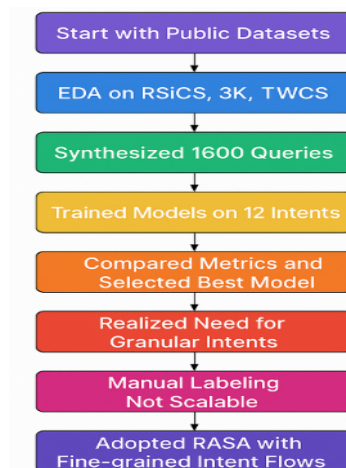
- **XGBoost with BERT Embeddings:** A gradient boosting framework optimized for speed and performance, especially on structured tabular features derived from embeddings.
- **BERT Classifier with BERT Embeddings:** A deep learning classifier using the [CLS] token representation, fine-tuned on our synthetic intent dataset.
- **DistilBERT Classifier with BERT Embeddings:** A distilled version of BERT, offering a lighter model with fewer parameters and faster inference while retaining much of BERT’s representational power.
- **CatBoost with BERT Embeddings:** A gradient boosting decision tree model known for handling categorical data effectively and preventing overfitting, evaluated with dense vector inputs from BERT.


### 4.3.2 Evaluation Results

Each model was assessed on standard metrics—accuracy, precision, recall, and F1 score, using a hold-out validation set. We observed that while all models performed reasonably well, the BERT Classifier consistently outperformed others across most metrics, particularly in handling complex, multi-word airline-specific intents.

Model	Accuracy	Precision	Recall	F1-Score
<b>BERT + BERT Embeddings</b>	0.94	0.95	0.94	0.94
<b>CatBoost + BERT Embeddings</b>	0.92	0.92	0.92	0.92
<b>DistilBERT + BERT Embeddings</b>	0.86	0.87	0.86	0.86
<b>XGBoost + BERT Embeddings</b>	0.70	0.70	0.69	0.69

 *Visual: Comparison of Models*



 *Visual: Iterative Evolution of Data-Driven Approach for Airline Chatbot Development*

## 5 Transition to RASA NLU

During the testing phase of our chatbot, we encountered several challenges that hindered the effectiveness of our initial approach. One of the main issues was that some intent groups (such as "Booking Modifications and Cancellations") were too broad, making it difficult for the chatbot to handle specific tasks efficiently. As we dug deeper, it became clear that we needed a finer granularity of intents to accurately address user needs. For instance, rather than having a single intent for all modifications, we required distinct intents like Date Change, Cancel Ticket, and Flight Upgrade. This change would enable more precise, targeted responses and improve overall user satisfaction.

However, as we sought to define and implement these more granular intents, the process of data generation and manual labeling became overwhelming and unsustainable. The sheer volume of required data for training the models, combined with the complexity of accurately labeling each example, introduced significant overhead.

To overcome this, we explored RASA NLU, an open-source conversational AI framework that specializes in modular intent definitions and dialogue management. RASA provided the flexibility we needed to define granular intents declaratively without requiring an extensive amount of labeled data. The framework supports the use of synonyms and rules, allowing us to train models effectively with fewer examples. This made the development process more agile, reducing the time and resources needed for training new intents.

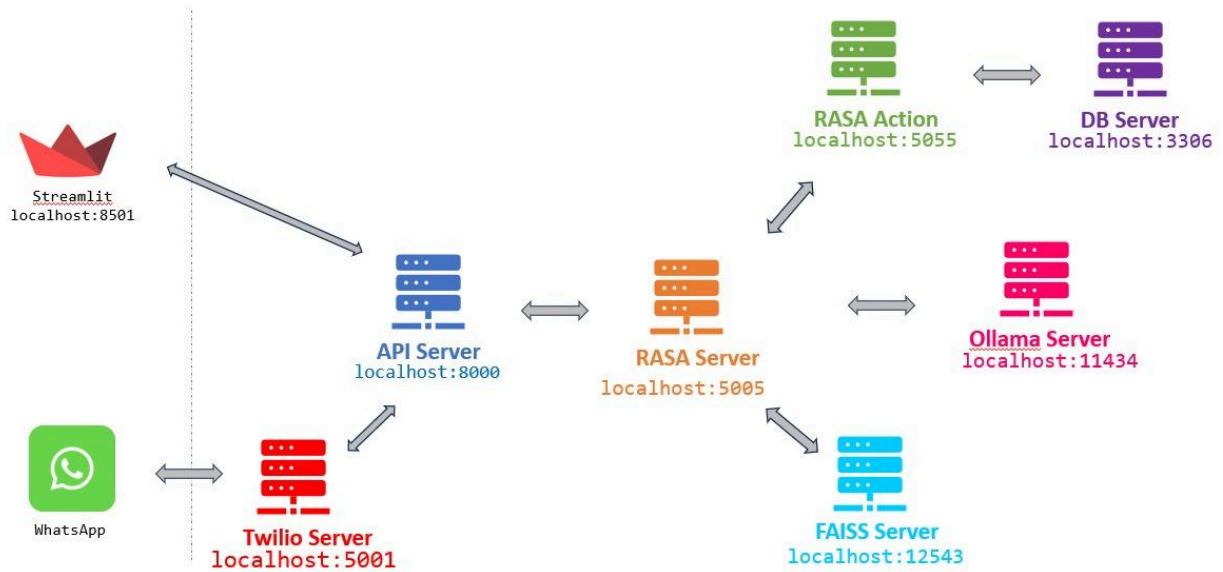
Key features of RASA NLU that make it the optimal solution include:

- **Granular Intent Definition:** RASA allows for precise differentiation between specific user requests (e.g., flight upgrades vs. date changes).
- **Data Efficiency:** It supports training with minimal labeled data by using **synonyms and rules**, enhancing intent recognition.
- **Custom Conversation Flows:** RASA enables us to define structured dialogues, creating a seamless user experience.
- **Extensibility and Modularity:** RASA's modular design makes it easy to add new intents or integrate additional APIs as services evolve.

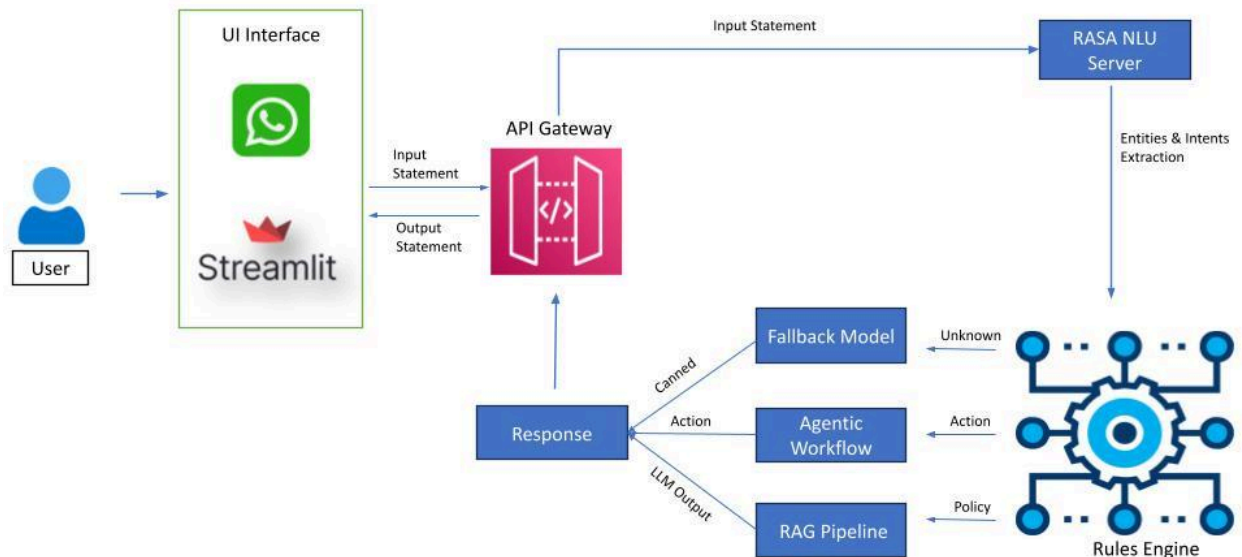
By integrating RASA NLU, we were able to reduce the complexity of training new intents, increase the accuracy of intent recognition, and streamline the development cycle, allowing for rapid iterations and more dynamic interaction handling. With RASA, we could handle granular tasks more effectively. This transition to RASA NLU ultimately allowed us to create a more flexible, scalable, and intelligent chatbot capable of handling a wide variety of customer queries with higher efficiency and minimal human intervention.



## 6 Architecture Overview (End-to-end pipeline)



Visual: Service Landscape



Visual: Logical Flow

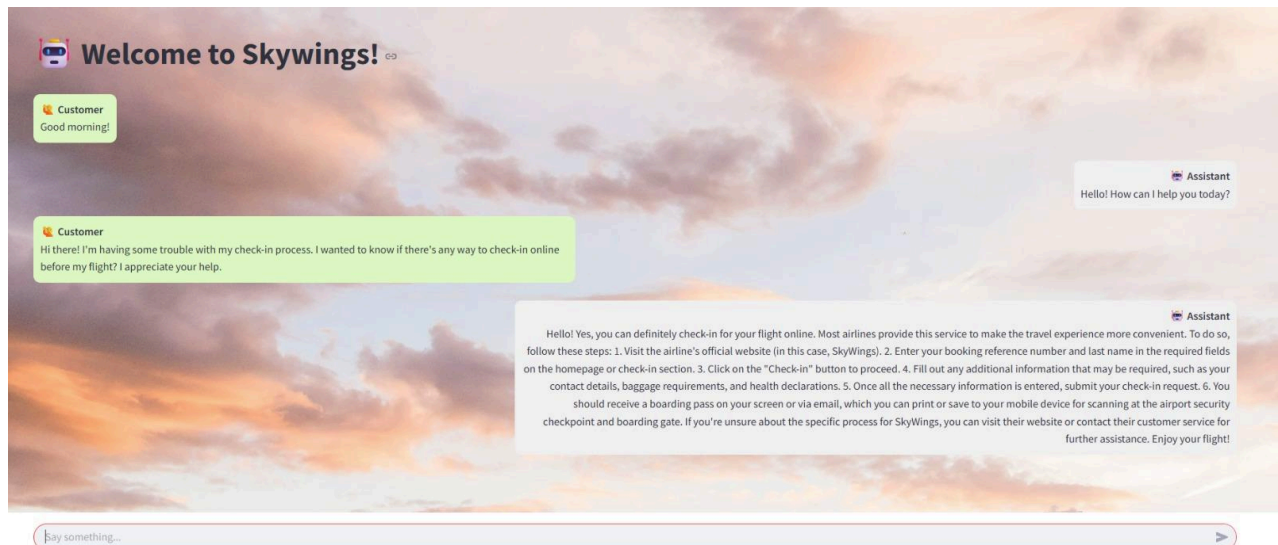
## 7 Key Components

The architecture of the SkyWings chatbot is designed with a range of integrated components that work together to deliver efficient and seamless interactions. Below is a detailed description of each key component:

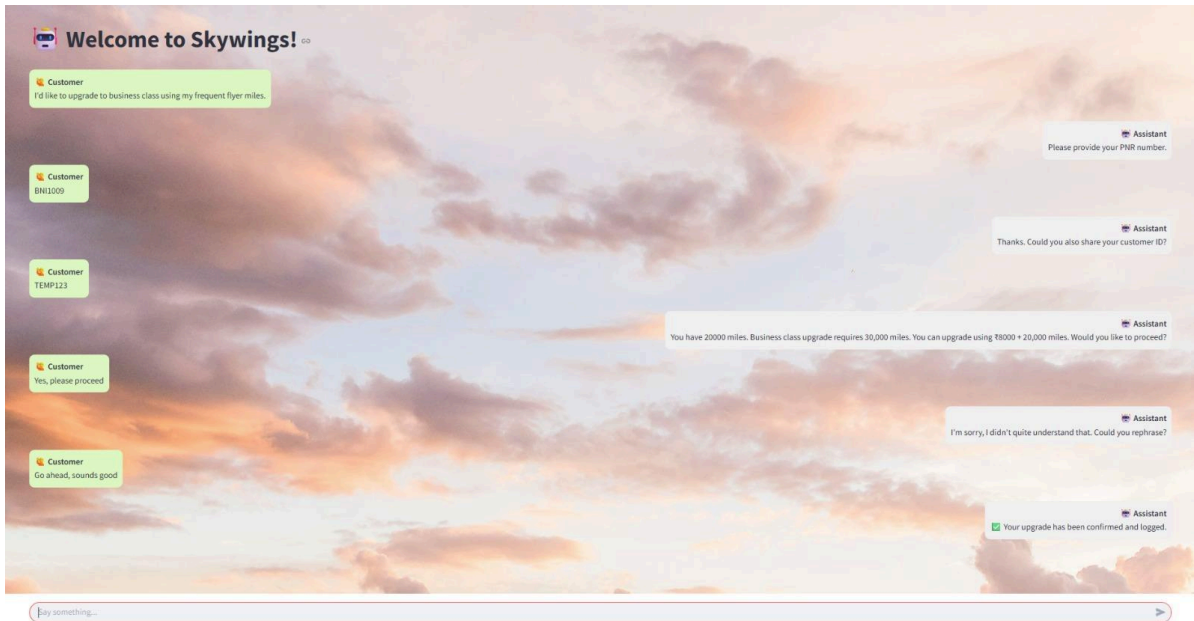
### 7.1 User Interface

#### 7.1.1 Streamlit

**Role:** Streamlit serves as a web application interface for the chatbot, running at localhost:8501. It allows users to interact with the system via a graphical interface, offering a visual environment that could display additional details, visualizations, or reports. This interface extends the chatbot's utility beyond messaging platforms, offering broad accessibility for visualization or non-urgent queries.



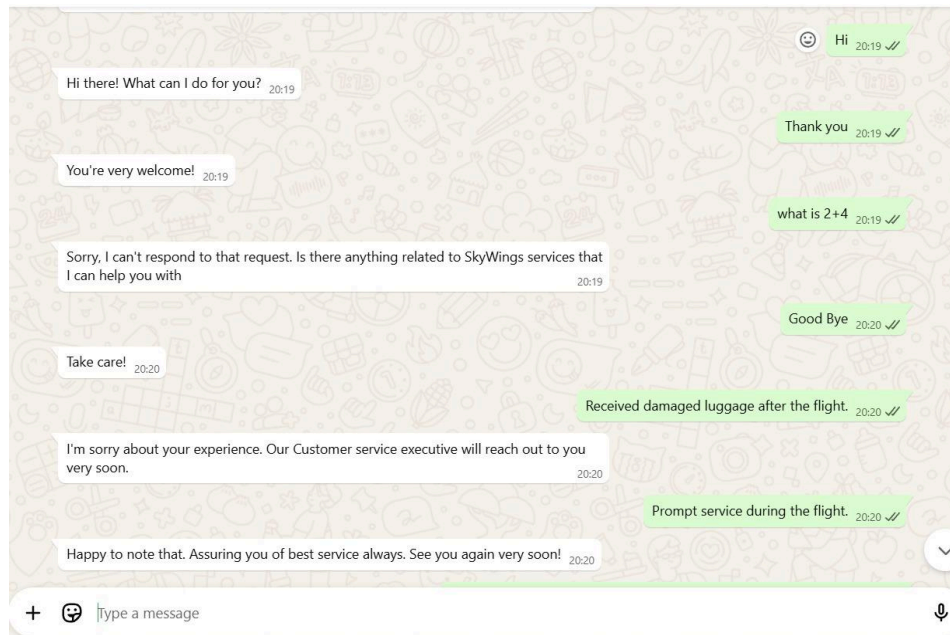
 *Visual: Streamlit UI (1)*



 Visual: Streamlit UI (2)

### 7.1.2 WhatsApp

Role: Acts as the user interface where customers initiate interactions with the chatbot. Using WhatsApp, customers can inquire about flight bookings, modifications, cancellations, and general airline policies. The chatbot responds to queries directly through WhatsApp, providing an accessible, real-time communication platform for users.



 Visual: WhatsApp Interface

## 7.2 Integration Layer

### 7.2.1 Twilio Server

Role: Twilio acts as the communication gateway between WhatsApp and the rest of the system. It handles the messaging API from WhatsApp, receiving incoming messages and relaying them to the central API server. Additionally, it is responsible for sending the chatbot's responses back to the user via WhatsApp.

### 7.2.2 FastAPI Integration

Role: The FastAPI framework powers the entire system's communication layer, facilitating quick and scalable API routing. It acts as the backbone for routing requests, managing interaction flow between the user interface, conversational engine, and backend services, while also ensuring high performance and ease of expansion.

## 7.3 Conversational AI Layer

### 7.3.1 RASA Core

Role: The core conversational AI engine is responsible for understanding natural language input, managing conversation context, and generating responses. It uses Natural Language Understanding (NLU) and dialogue management techniques to ensure accurate intent detection and smooth conversation flow. The RASA server is essential for handling user queries and maintaining engagement.

### 7.3.2 RASA Action

Role: The RASA Action Server is responsible for executing custom business logic and handling tasks that go beyond simple text-based responses. When the RASA Server determines that an action requires dynamic processing—such as retrieving available flights, modifying bookings, or canceling a reservation—it delegates this task to the Action Server. This server then interacts with the backend systems, including databases and external APIs, to perform the required operations.

This component embodies the principles of Agentic AI, where the chatbot is not just a passive responder but an active problem-solver capable of taking meaningful actions on behalf of the user. By leveraging structured action flows, the RASA Action Server enables the assistant to function as an intelligent agent—performing transactions, managing stateful conversations, and seamlessly integrating with operational systems in real time.

## 7.4 Data Management Layer

### 7.4.1 DB Server

**Role:** The database server is used for storing persistent data such as flight details, booking information, and other relevant data. It ensures that the chatbot can access up-to-date information to respond accurately to user queries, facilitating dynamic responses.

## 7.5 Policy Knowledge Base

To enable the assistant to handle nuanced airline policy queries, we compiled a structured policy knowledge base by sourcing publicly available documents from leading airline websites. These included terms and conditions, baggage policies, cancellation rules, and refund procedures. The documents were preprocessed and embedded into vector representations, allowing the system to semantically retrieve relevant sections in real time.

This corpus serves as the foundation for the chatbot's Retrieval-Augmented Generation (RAG) capability. When the RASA server detects a policy-related intent, the query is routed to the Ollama-hosted LLM, which consults this embedded corpus, via FAISS, to generate accurate and contextually relevant answers. This significantly reduces user frustration caused by redirection to static FAQ pages or long wait times for human agents.

## 7.6 Retrieval-Augmented Generation

### 7.6.1 Ollama Server

**Role:** The server for managing large language models (LLMs) enhances the chatbot's language understanding and generation capabilities. It assists in answering more complex queries, particularly those related to airline policies, by processing questions with advanced natural language models. The LLM integrates seamlessly with the chatbot's other components, allowing for rich and informed responses.

To handle these queries, the system employs a Retrieval-Augmented Generation (RAG) architecture. When the RASA Server identifies that a user's question pertains to airline policies, it redirects the query to the LLM via the Ollama Server. Rather than relying solely on the LLM's internal knowledge, the system first retrieves the most relevant snippets from the curated airline policy documents embedded in a vector database. The LLM then generates a response grounded in this retrieved information, ensuring factual accuracy and domain relevance.


This hybrid approach allows the chatbot to offer detailed, policy-aware answers without hardcoding every possible rule or relying on generic replies. It also improves response trustworthiness by anchoring answers in real documentation.

## 8 LLM Performance Assessment

To identify the most effective language model for response generation, we conducted an LLM evaluation using an ensemble-based evaluation strategy. Each candidate model was presented with a common set of prompts, and in a novel peer-review setup, the models evaluated and rated the quality of responses generated by the others.

These cross-model ratings were aggregated, and average scores across all prompts were computed to derive a final performance score for each LLM. This comparative evaluation enabled us to objectively select the best-performing model for integration into our chatbot pipeline.

Llama3	Phi	Mistral	Gemma:2b	Zephyr
7.31	7.35	8.27	6.70	7.90

 *Visual: Evaluation Scores*

## 9 Analytics Dashboard

To derive business insights, we developed a comprehensive analytics dashboard using Superset. It provides actionable insights into user behavior, interaction patterns, and service effectiveness, enabling data-driven decisions for continuous improvement.

From a business perspective, this dashboard helps identify high-demand routes, optimize resource allocation, and uncover conversion bottlenecks, driving both operational efficiency and customer satisfaction.

- **User Interaction Overview:** Visualizes insights on usage of the chatbot across platforms (WhatsApp vs. Streamlit), helping identify preferred touchpoints.
- **Conversion Rate Tracking:** Measures the percentage of users who progress from enquiry to booking, giving insight into the assistant's effectiveness.
- **Intent and Sentiment Analysis:** Displays user intents (e.g., booking, cancellation) with sentiment overlays, allowing better understanding of user emotions behind queries.
- **City-Based Enquiry Distribution:** Breaks down arrival and departure city requests, highlighting the most queried routes and travel patterns.
- **Enquiry Intent by Volume:** Shows percentage distribution of user intents (e.g., booking, flight changes), helping in service prioritization.
- **Peak Usage Insights:** Identifies time-based spikes in user traffic, assisting with resource planning and chatbot scaling.
- **Arrival Enquiry Table:** Tabular view listing arrival cities alongside dominant intent and percentage share, enabling deeper demand mapping.



## Business Insights

**Conversion Rate** : Number of users booked flight / Total users

**Average Requests per User** : Average number of enquiries by users

Conversion Rate

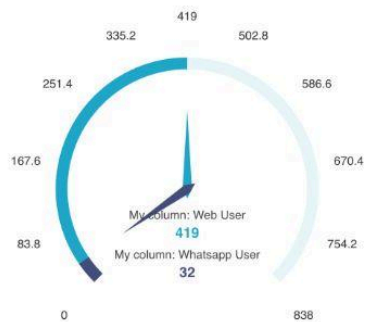
1

2.22

Average Enquiries per User

**User Interaction Hub** : Number of users using web and whatsapp for enquiries

User Interaction Hub



**Booking-Arrival City** : Percentage distribution of users based on booked arrival cities

**Booking-Departure City** : Percentage distribution of users based on booked departure cities.

Booking- Arrival City

Arrival City	Percentage of Users
Bangalore	100
Delhi	14.29
Goa	14.29
Hyderabad	14.29
Pune	14.29

Booking - Departure City

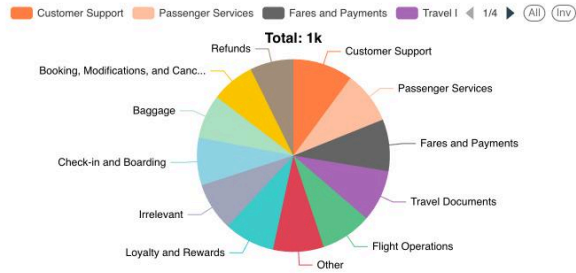
Departure City	Percentage of Users
Delhi	28.57
Kochi	71.43
Kolkata	14.29
Mumbai	14.29
Pune	100



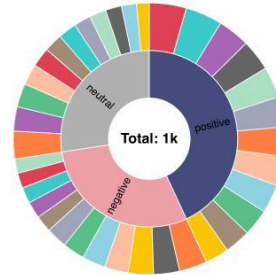
Visual: Superset Dashboard (1)

Enquiry Intent of Users : Distribution of enquiry intent of users  
 Enquiry Intent with Sentiment : Classification of enquiry intents on sentiments

Enquiry Intent of Users



Enquiry Intent with Sentiment



Peak Hours : Peak hours of enquiries


Arrival Enquiries : User enquiry intents categorized by arrival city

Peak Hours

Hour	Percentage of Users
17	77.1
16	2.1
18	1.6
10	1.9
12	5.2
13	8.3
14	3.8

Arrival Enquiries

Arrival City	Intent	Enquiry Percentage
Bangalore	Baggage	23.73
Bangalore	Booking, Modifications, and Cancellations	25.42
Bangalore	Check-in and Boarding	11.86
Bangalore	Customer Support	16.95
Bangalore	Fares and Payments	11.86
Bangalore	Flight Operations	11.02
Bangalore	Irrelevant	16.95
Bangalore	Loyalty and Rewards	18.64
Bangalore	Other	16.95
Bangalore	Passenger Services	15.25
Bangalore	Refunds	5.93

 Visual: Superset Dashboard (2)

## 10 Challenges and Learnings

### 10.1 Key Challenges

- **Granular Intent Definition vs. Maintenance Overhead**  
 Defining highly specific intents improved precision but introduced scalability issues. RASA's extensibility helped mitigate this with rules and synonym support.
- **Lack of Labeled Data**  
 Domain-specific queries lacked sufficient training examples. We addressed this using zero-shot classification and Retrieval-Augmented Generation (RAG) to reduce data dependence.



- **Human Escalation Bottlenecks**  
Static FAQ links or delayed agent handovers degraded the UX. Integrating LLMs via Ollama allowed richer policy responses without immediate escalation.
- **System Integration Complexity**  
Orchestrating Twilio, RASA, Action Servers, LLMs, and Streamlit required careful API design. FastAPI enabled reliable, modular routing between services.
- **Multi-turn Context Handling**  
Booking modifications and other multi-step workflows required enhanced memory management. Slot tracking and dialogue policies in RASA handled these effectively.
- **Infrastructure Load and Latency**  
Running LLMs and multiple server processes (RASA, DB, API, Ollama) concurrently stressed local resources. Performance optimization and caching strategies were necessary to maintain responsiveness.
- **Limited Compute Resources**  
LLM inference and vector similarity queries (especially in retrieval-heavy workflows) were computationally intensive. With limited GPU/CPU availability, tasks occasionally stalled, necessitating careful orchestration and fallback planning.

## 10.2 Learnings

- **Modularity Enables Flexibility**  
A component-based architecture allowed us to iterate and scale rapidly without disrupting the whole system.
- **Agentic AI is Powerful When Guided**  
LLMs are most effective when combined with retrieval and context control, improving response quality without compromising accuracy.
- **Fallbacks Must Be Thoughtful**  
A confidence-driven fallback strategy enhanced trust by clearly signaling uncertainty or routing to human agents only when needed.
- **Hybrid NLP Is Future-Ready**  
Combining rule-based NLU with generative models gave us the best of both worlds—control and creativity.

## 11 Future Roadmap

- **Production Deployment**  
Move from localhost testing to cloud-based or containerized deployment for real-world availability and scale.
- **Multi-Language Support**  
Expand NLU and LLM capabilities to support multilingual users, starting with high-volume languages like Hindi.

- **Voice and IVR Integration**  
Extend beyond WhatsApp and web to support voice-based queries via IVR systems or speech-to-text APIs.
- **Payment Gateway Integration:** Enable seamless in-chat payments by integrating secure payment gateways into the booking flow, allowing users to complete flight purchases directly through WhatsApp or the web interface.

## 12 Conclusion

This project demonstrates the potential of a domain-specific conversational AI assistant tailored to the airline industry. By combining RASA's modular intent handling with the power of Retrieval-Augmented Generation (RAG) via LLMs, we successfully addressed the limitations of traditional bots in handling complex, policy-driven queries. The integration across multiple platforms—WhatsApp, Streamlit, and backend APIs—ensured a seamless experience for both users and developers.

Key learnings around data efficiency, intent granularity, fallback strategies, and agentic task execution have laid a strong foundation for scalable, intelligent automation. With an extensible architecture and a clear roadmap ahead, this assistant is well-positioned to evolve into a full-service digital travel companion, delivering both operational efficiency and user satisfaction.

## 13 References

[Relational Strategies in Customer Service\(RSiCS\)](#)

[3K Conversations Dataset for ChatBot](#)

[Customer Support on Twitter](#)

[Huggingface Airline Customer Service Dataset](#)

[Huggingface AirDialogue Dataset](#)

[Chatbots in customer service: Their relevance and impact on service quality](#)

[DIET Classifier with sparse input features only](#)

## 14 Codebase

[Github Repository](#)