Document ID	**ASAPP-BOT-01-RUNBOOK**
Document Title	Airline Support Chatbot - Operational Runbook
Service Owner	Team Botathon (L3 Support)
Service Tier	Tier 1 (Customer-Facing)
Document Version	1.0
Last Updated	23.10.2025
Project	ASAPP <> HACKATHON
Problem Statement	Problem 2

Team Details (L3 Support):

- **Team Name:** Botathon (01)
- **Members:**

  Koushik M (22i231),

  Vyas M (22i274),

  SaiKrishnan J (22i321),

  Sanjay A C (22i355).

1.0 Service Overview

1.1 Service Description

This document provides operational procedures for the **Airline Support Chatbot** (Service ID: ASAPP-BOT-01). This is a Tier-1, customer-facing service designed to automate the fulfillment of high-volume airline customer requests as defined by "Hackathon Problem 2."

The service is a stateful Python/Django application providing a conversational chat interface. It leverages a BERT-based NLU (Natural Language Understanding) engine for intent classification and a Django REST Framework (DRF) backend for task execution.

1.2 Business Impact

Service degradation or outage has a **direct, critical impact on customer experience**. Failure of this service will result in a surge of call/chat volume to human support agents, increasing operational costs and reducing customer satisfaction.

1.3 Service Level Objectives (SLOs)

Metric	Objective	Description
Availability (Uptime)	99.9% (3-nines)	Percentage of HTTP 2xx/3xx responses from the /health endpoint.
API Latency (p99)	< 800ms	99% of all POST /api/process requests must be served in under 800ms.
NLU Fallback Rate	< 1.0%	Percentage of requests that fail BERT classification and use the keyword fallback. A high rate indicates severe quality degradation.

2.0 System Architecture

2.1 Core Component Architecture

This diagram illustrates the static components of the application and their relationships.

Architecture diagram illustrates the end-to-end architecture of a Django-based airline chatbot system.

The User interacts through the Frontend UI (chat.html, chat.js), which sends HTTP API requests to the Django backend. Inside the backend, the Router directs requests to the API Controller (views.py), which serves as the core orchestrator.

The Controller first identifies the user's intent using the NLU module (intent_classifier.py) powered by a Sentence Transformer model (all-MiniLM-L6-v2). Based on the intent, it queries or updates Data Models (models.py) that communicate with the SQLite database. The Serializer then formats the processed data into a structured JSON response, which is returned to the Frontend for display to the user.

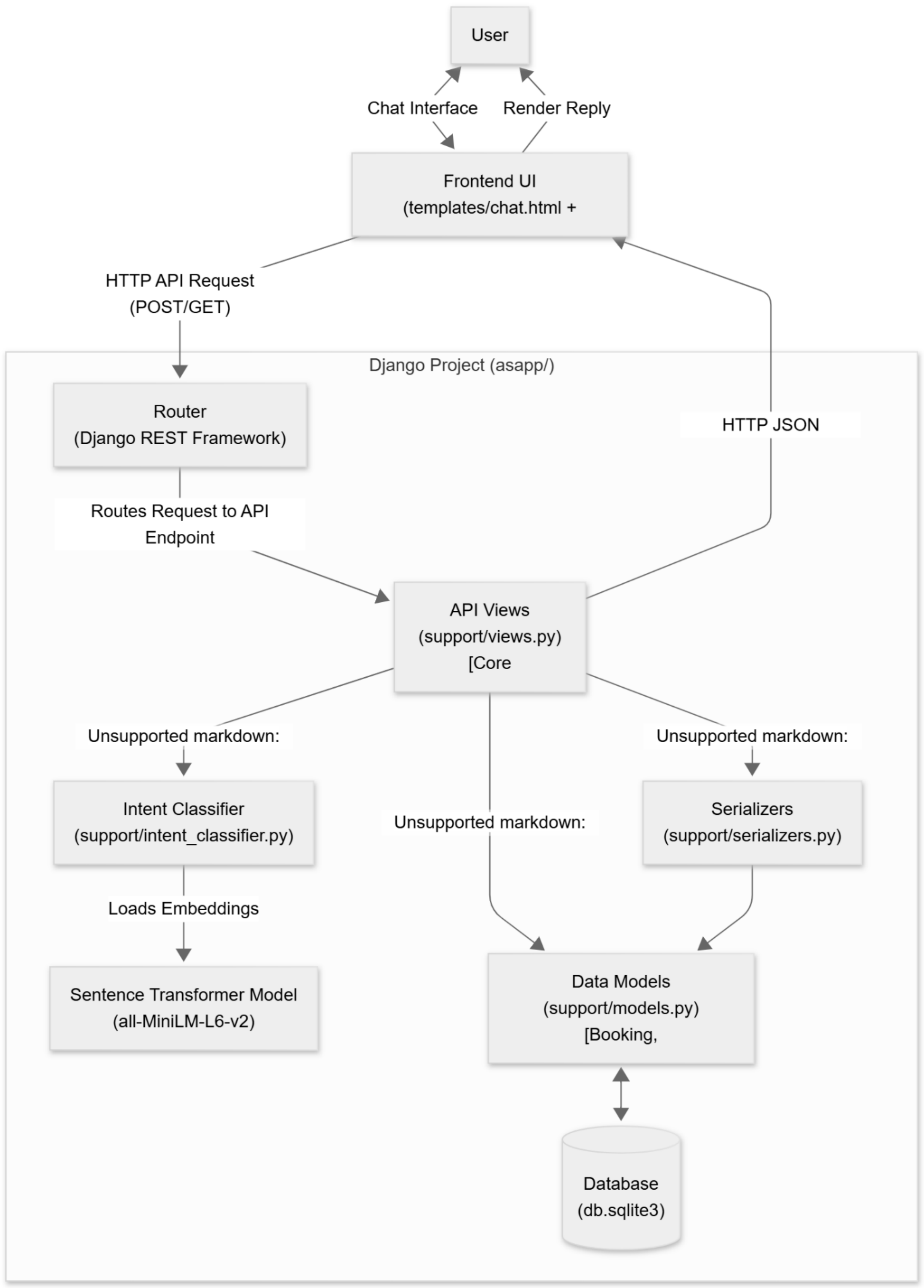

Fig 1 Architectural Diagram

2.2 Request Workflow

This sequence diagram illustrates the end-to-end flow for a single customer request.

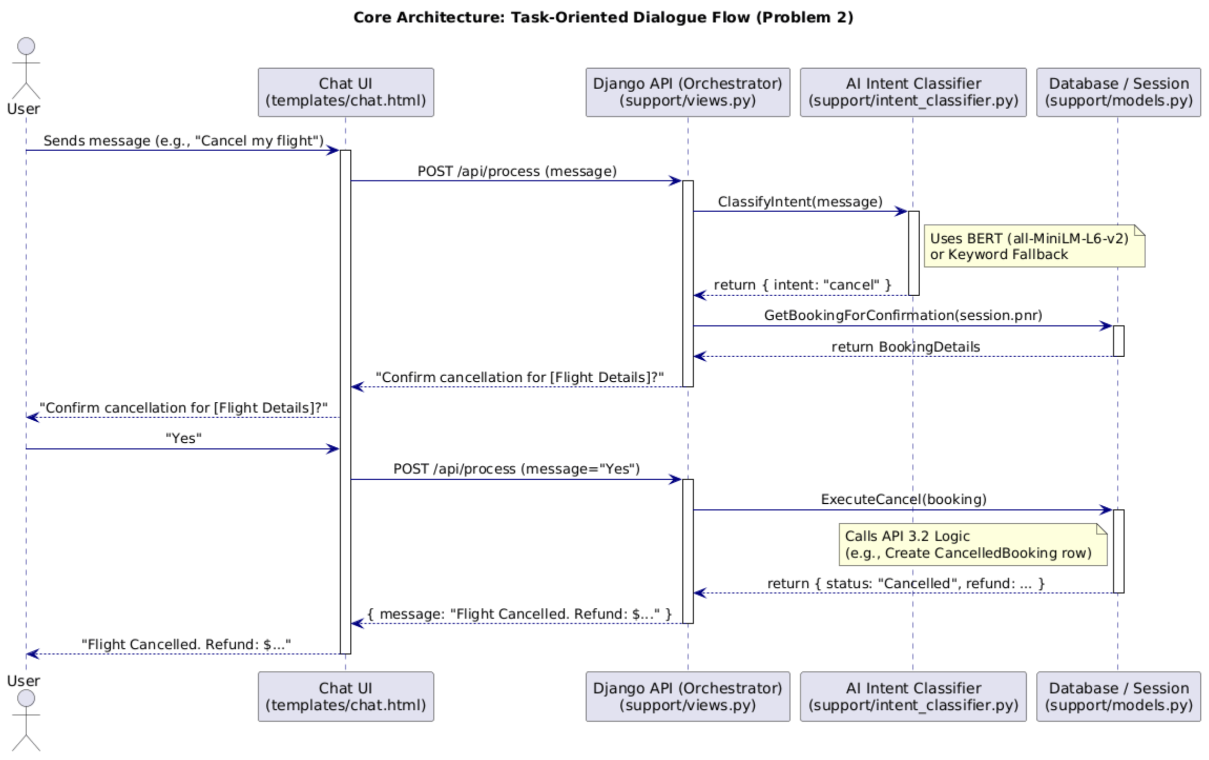
Core Architecture: Task-Oriented Dialogue Flow (Problem 2)
User
Chat UI (templates/chat.html)
Django API (Orchestrator) (support/views.py)
AI Intent Classifier (support/intent_classifier.py)
Database / Session (support/models.py)
Sends message (e.g., "Cancel my flight")
POST /api/process (message)
ClassifyIntent(message)
Uses BERT (all-MiniLM-L6-v2) or Keyword Fallback
return { intent: "cancel" }
GetBookingForConfirmation(session.pnr)
return BookingDetails
"Confirm cancellation for [Flight Details]?"
"Confirm cancellation for [Flight Details]?"
"Yes"
POST /api/process (message="Yes")
ExecuteCancel(booking)
Calls API 3.2 Logic (e.g., Create CancelledBooking row)
return { status: "Cancelled", refund: ... }
{ message: "Flight Cancelled. Refund: $..." }
"Flight Cancelled. Refund: $..."

Fig 2 Workflow Diagram

3.0 Critical Dependencies

Service availability is dependent on the health of the following external systems.

Dependency	**Owner**	**Failure Impact**	**Triage**
Production DB (PostgreSQL)	DBA Team	**CRITICAL (Hard Outage)**	Service will fail health checks. 5xx errors.
Airline Booking/Status API	API Partner Team	**HIGH (Service Degradation)**	Cancel and Status intents will fail. High latency.
NLU Model Artifact	CI/CD (Artifactory)	**CRITICAL (Degradation)**	All requests will use fallback. NLU Fallback Rate alert will fire.

4.0 Monitoring, Alerting & Health

4.1 Health Check Endpoint

- **Endpoint:** GET /health
- **Purpose:** Liveness/Readiness probe for Kubernetes and load balancers.
- **Success (HTTP 200):** Application server is running, and a database connection is successfully established.
- **Failure (HTTP 503):** Database connection failed or application server is down.

4.2 Key Monitoring Dashboards

- **Primary Dashboard:** [Link to Grafana/Datadog Dashboard]
- **Logging:** [Link to Splunk/ELK Dashboard]

4.3 Primary Alerts & Triage

Alert (Priority)	Threshold	Description & Triage Path
P0: High 5xx Error Rate	> 5% over 5 min	**Service is down.** Check logs for CRITICAL or OperationalError. **Execute SOP-01.**
P1: High API p99 Latency	> 800ms over 5 min	**Service is slow.** Users are impacted. Check dependency (External API) latency. **Execute SOP-02.**
P1: High NLU Fallback Rate	> 5% over 5 min	**Service quality is degraded.** BERT model is failing. **Execute SOP-03.**
P2: High Pod Memory (OOM)	> 90% usage	Pod is at risk of OOMKilled. Usually precedes a P0 alert. Investigate for memory leaks.

5.0 Standard Operating Procedures (SOPs)

SOP-01: Triage P0: High 5xx Error Rate

1. **Check Logs:** Immediately query logs for CRITICAL, ERROR, or OperationalError during the alert window.
2. **Triage Cause:**
    - **IF** logs show OperationalError: could not connect to server:
        - **Diagnosis:** Database is down or unreachable.
        - **Action: Escalate to DBA Team (P0).**
    - **IF** logs show OOMKilled or MemoryError:
        - **Diagnosis:** Pod/container is crashing due to memory exhaustion (likely NLU model load).

        - **Action:** Pod should self-heal (restart). If it enters a CrashLoopBackOff state, **Escalate to L3 (Botathon Team).**
    - **IF** the alert began *immediately* after a new deployment:
        - **Diagnosis:** Bad code/artifact deployment.
        - **Action: Execute Recovery Procedure 6.2 (Rollback).**

SOP-02: Triage P1: High API p99 Latency

1. **Check Dashboards:** Open the Primary Dashboard and view the "External API Latency" panel.
2. **Triage Cause:**
    - **IF** External API Latency is high (e.g., > 2000ms) and matches the spike in our API p99 Latency:
        - **Diagnosis:** The external Airline API (a dependency) is slow. Our service is blocked.
        - **Action: Escalate to API Partner Team (P1).** No restart required.
    - **IF** External API Latency is normal:
        - **Diagnosis:** The application itself is slow (e.g., DB query, CPU starvation).
        - **Action: Execute Recovery Procedure 6.1 (Rolling Restart).**

SOP-03: Triage P1: High NLU Fallback Rate

1. **Check Logs:** Query logs for NLU_MODEL_LOAD_FAILURE or NLU_FALLBACK_TRIGGERED.
2. **Diagnosis:** The all-MiniLM-L6-v2 model artifact is either missing, corrupted, or failing to load into memory. The service is now "dumb" and providing a poor user experience.
3. **Action:**
    1. **Execute Recovery Procedure 6.1 (Rolling Restart)** to force a reload of the model artifact.
    2. **IF** the alert does not clear after 5 minutes:
        - **Diagnosis:** The deployed artifact is broken.
        - **Action: Execute Recovery Procedure 6.2 (Rollback)** and **Escalate to L3 (Botathon Team).**

6.0 Recovery Procedures

6.1 Procedure: Rolling Restart (Standard Mitigation)

- **Purpose:** Clears application state, reloads NLU model, and cycles pods. Use for non-dependency latency or unexplained errors.
- **Command (Kubernetes):**

```
Bash

# Verify deployment status

kubectl get deployment airline-support-chatbot -n prod

# Perform rolling restart

kubectl rollout restart deployment/airline-support-chatbot -n prod
```

6.2 Procedure: Deployment Rollback (Emergency)

- **Purpose:** Revert to the *previous stable* version of the application. Use for bad deployments or critical failures (like SOP-03).
- **Command (Kubernetes):**

```
Bash

# Check revision history

kubectl rollout history deployment/airline-support-chatbot -n prod

# Revert to the previous stable revision

kubectl rollout undo deployment/airline-support-chatbot -n prod
```

6.3 Database Migrations

- **CRITICAL NOTE:** Database migrations (python manage.py migrate) are applied automatically by the CI/CD pipeline during a deployment.
- **DO NOT** run migrations manually. A manual migration run will desynchronize the application and database schema, causing a **P0 outage**.

7.0 Escalation Path

1. **L1 (On-Call Operator):**
   - Receives P0/P1 alerts.
   - Follows this Runbook (SOPs 1-3).
   - Executes Recovery Procedures (6.1, 6.2) as directed.
2. **L2 (Dependency Teams):**
   - L1 escalates to DBA Team for database failures.

    - L1 escalates to API Partner Team for external API failures.
3. **L3 (Team Botathon):**
    - L1 escalates if:
        - The cause is identified as a code-level bug.
        - A rollback (6.2) is performed.
        - A CrashLoopBackOff state is observed.
        - The root cause is unknown after 15 minutes of triage.

8.0 Runbook Governance & Contact Matrix

8.1 Document Ownership

This runbook is a living document. The **L3 Support Team (Team Botathon)** is the designated owner and is responsible for its accuracy.

8.2 Review & Update Cadence

This document is subject to a formal review on a **quarterly basis** or **post-incident** (within 48 hours of any P0/P1 incident resolution), whichever comes first.

All updates must be version-controlled and peer-reviewed by the L3 team before publication. Outdated procedures are a direct risk to service availability.

8.3 Service Contact & Escalation Matrix

This matrix is the definitive source for service escalation.

Escalation Tier	**Team / Point of Contact**	**Pager / Slack Channel**	**Primary Responsibility**
L1 (Operator)	SRE On-Call (Primary)	[#sre-oncall]	Initial Triage, SOP Execution, Recovery (6.1, 6.2)
L2 (Dependency)	DBA Team	[#dba-support]	Database Failures (Ref: SOP-01)
L2 (Dependency)	API Partner Team	[#api-partners]	External API Latency / Failures (Ref: SOP-02)
L3 (Owner)	**Team Botathon**	[#botathon-support]	Code-Level Bugs, NLU Failures (SOP-03), Post-Rollback Investigation