ITI123 Generative AI & Deep Learning Project (2025S2)

HDBAssist – AI-Powered Singapore HDB Resident Issue Guidance Application

Project Submitted By: Abirami Balachander / 9552195U on 26 February 2026

A person holding a phone and thinking about a problem

AI-generated content may be incorrect.

# Introduction & Motivation

HDBAssist addresses the confusion faced by Singapore Residents when reporting public estate issues.

* Residents often struggle to determine the correct authority (Town council, HDB, NEA, Police, SCDF) and how to prepare necessary information

## Problem Statement

*There is no unified intelligent assistant that interprets natural language complaints, determines urgency, routes to the correct authority, and generates structured guidance with responsible AI guardrails.*

The core problem addressed in this project is the **difficulty faced by HDB residents in correctly identifying the appropriate authority and next steps when reporting common estate-related issues**.

From a user’s perspective, estate issues are typically described in informal and unstructured terms (e.g. “the light outside my unit is not working” or “someone suspicious is loitering downstairs”). However, public service systems often require residents to first categories the issue correctly before selecting the appropriate reporting channel. This mismatch between **how residents describe problems** and **how services are organized** leads to frequent uncertainty.

Common challenges include:

* Ambiguity over which authority is responsible for a given issue
* Uncertainty about whether an issue is urgent or non-urgent
* Confusion caused by multiple reporting channels (hotlines, online forms, emergency numbers)

As a result, residents may:

* Submit reports to the wrong agency
* Delay in taking action while seeking clarification
* Abandon reporting altogether
* Escalate non-emergency issues unnecessarily

This project formulates the problem as a **decision-support and service navigation challenge**, rather than a prediction/classification problem in isolation. The goal is not merely to label an issue, but to **translate free-form problem descriptions into accurate, authoritative, and actionable guidance**.

## Overview of the Architecture

HDBAssist is built using a modular, multi-agent Generative AI architecture designed to provide structured, safe, and domain-specific public service guidance for Singapore residents. The system separates responsibilities across distinct architectural layers to ensure scalability, governance, maintainability, and deterministic routing.

*Presentation Layer –* ***Streamlit***

The Presentation Layer is implemented using Streamlit. It serves as the user interface where residents submit issues related to HDB estates. Streamlit handles user input capture, session management, and structured JSON rendering of responses. It communicates with the orchestration layer via REST API calls.

### *Orchestration Layer –* ***Flowise***

The Orchestration Layer is powered by Flowise. This layer coordinates the execution of multiple agents in a controlled workflow. It manages state variables, routing logic, conditional branching, and ensures deterministic execution across Supervisor, Condition, Domain RAG, and Report agents.

### *Intelligence Layer –* ***Supervisor Agent***

The Supervisor Agent performs structured first-intake processing. It extracts key fields such as country, location details, issue description, urgency hint, and assigns a single routing value (route). It does not provide recommendations. Instead, it outputs strict JSON that drives downstream routing.

### *Domain Layer –* ***RAG Agents***

Domain-specific Retrieval-Augmented Generation (RAG) agents are responsible for handling specialized issue categories including Police, Town Council, HDB, NEA, and Emergency. Each agent operates with domain-constrained instructions to prevent hallucination and ensure agency-appropriate guidance. These agents only activate when routed by the Supervisor.

### *Formatting Layer –* ***Report Agent***

The Report Agent consolidates outputs into a structured JSON schema suitable for frontend rendering. It standardizes urgency level, recommended authority, assessment, next steps, and preparation details. This ensures consistent and predictable UI display.

### Core Functionalities

• Deterministic routing based on structured 'route' field  
• Strict JSON schema enforcement to prevent malformed outputs  
• Multi-agent separation of concerns  
• Guardrail enforcement for Responsible AI  
• Session-based state tracking via sessionId  
• Scalable integration via REST API

### Advantages of This Architecture

### *Separation of Concern*

Each agent performs a clearly defined role. This improves maintainability and debugging since classification, domain reasoning, and report formatting are isolated.

### *Reduced Hallucination Risk*

By restricting domain agents and enforcing structured outputs, the architecture minimizes fabrication of contact details or cross-domain contamination.

### *Scalability*

New agencies or categories can be added by introducing new RAG agents without modifying the entire system. The route-based design supports easy extensibility.

### *Governance and Responsible AI*

The system includes safeguards such as country restriction logic, emergency prioritization, single-question clarification policy, and no professional advice constraints.

### *Deterministic Control Flow*

Using a single route value eliminates overlapping condition conflicts. This ensures predictable execution in Flowise.

### Conclusion

The HDBAssist architecture demonstrates a production-oriented, modular, and governance-aware multi-agent Generative AI system. Its layered design supports clarity, safety, scalability, and robust deployment for public service guidance applications.

# Multi-Agent Orchestration - Flowise

In HDBAssist, Flowise is the **control plane** for the full AI workflow. Streamlit is only a UI layer; Flowise is responsible for:

* **Executing agents in the correct order**
* **Maintaining a shared flow state** (variables that persist across nodes)
* **Branching to different domain agents** based on routing decisions
* **Standardizing outputs** so Streamlit gets a predictable response schema

This is important because an LLM alone is not deterministic, but Flowise provides deterministic orchestration and state management around it.

### Core orchestration pattern (multi-agent pipeline)

The workflow is a structured pipeline:

A screenshot of a computer

AI-generated content may be incorrect.

**Text Input node (Start)**

* Accepts the question from Streamlit via REST API.
* Initializes/sets key state variables (e.g., session\_id, chat\_input, placeholders for route, etc.)

**Supervisor Agent node**

* Performs **intake + classification** only:
  + Normalizes issue description
  + Extracts location fields
  + Sets urgency\_hint
  + Sets a single **route** value (e.g., town\_council, hdb, nea, etc.)
* Outputs **strict JSON** and updates Flowise state using “Update Flow State”.

**Condition node (Router)**

* Reads from Flowise state: {{$flow.state.route}}
* Branches to exactly one downstream domain agent:
  + route == "emergency" → Emergency agent
  + route == "police" → Police agent
  + route == "town\_council" → Town Council agent
  + route == "hdb" → HDB agent
  + route == "nea" → NEA agent
  + route == "outside\_sg" → OutsideSG response node
* This makes routing **deterministic** and avoids overlapping keyword conditions.

**Domain RAG Agent (one of the agencies)**

* Responsible for retrieving guidance from the knowledge base (vector DB).
* Produces domain-specific content only (no cross-agency guessing).

**Report Agent**

* Merges the supervisor’s extracted fields + RAG output into a single consistent schema.
* Returns predictable JSON to Streamlit so UI rendering is stable.

**How retrieval works (step-by-step)**

1. User enters issue in Streamlit
2. Supervisor sets route = "nea" (example)
3. NEA RAG Agent queries Pinecone with:
   * Query embedding from issue description
   * Filter: agency == "NEA"
4. Pinecone returns top-k relevant chunks
5. The agent uses only those retrieved chunks to answer (or to fill report fields).

This gives:

* **Higher accuracy**
* **Less hallucination**
* **Traceable evidence** via metadata sources

# Technical Stack

* Frontend: Streamlit
* Orchestration: Flowise Cloud
* LLM OpenAI GPT-40-mini
* VectorDB: PineCone
* Integration: REST API
* Deployment: Cloud based Flowise + Streamlit

# Core Features

* Natural language issue intake
* Supervisor-based classification and routing
* Domain-specific RAG agents
* Structured JSON output
* UI rendering of urgency, authority, next steps

# Responsible AI & Governance

Guardrails were implemented to prevent hallucinated contact details, incorrect emergency routing, and unsafe advice. Structured outputs enforce predictable JSON responses.

# Deployment

The system is deployed with Flowise cloud orchestration and Streamlit UI. End-to-end integration via REST API ensures functional prototype operation.

The requirements.txt file consists of all the requirements to be installed in my laptop.

Launch the app using streamlit run ui.py 🡪 the ui.py is the UI used to connect with flowise orchestrator

A screenshot of a computer

AI-generated content may be incorrect.

Launch of the UI

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Evaluation

Functional testing was conducted across emergency, police, town council, HDB, NEA, and outside-Singapore cases. Routing accuracy was validated for defined categories.

Few test scenarios screenshots:

**Scenario 1: Bicycle theft at Bedok.**

Expected outcome:

Urgency: Normal. Report to Police with details to report for the same.

Result: Achieved the expected result

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2: Fire Alert @ Jurong West**

Expected outcome:

Urgency: Emergency. Report to SCDF immediately.

Result: Achieved the expected result

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a black and white screen

AI-generated content may be incorrect.

Example of the internal flow for Leak issue reported in Tampines

Text Input 🡪 Supervisor Agent 🡪 Condition route 🡪 Town Council RAG 🡪 Report

A screenshot of a chat

AI-generated content may be incorrect.

# Limitations & Future Work

Future improvements include multilingual support, official API integrations, analytics dashboards, and expanded retrieval databases.

# Conclusion

HDBAssist demonstrates practical integration of Generative AI services, multi-agent orchestration, structured routing logic, responsible AI guardrails, and successful deployment.

Link for Github Repository of the files:

<https://github.com/Abirami121b/HDBAssist/tree/main>