SMU
SINGAPORE MANAGEMENT UNIVERSITY

SMU Academy

28 April 2026

Integrated Learning Project on AI Powered Town Council Feedback Response System

Group 3

Abdul Shariff Aboo Kassim
Kam Zhi Hao
Isabelle Khang Hui Wen
Siti Salwa Abdul Rahim

Contents

1. Abstract

This project aims to address inefficiencies in the existing Town Council feedback-handling process, which is often manual, time-consuming, and prone to inconsistency. Case-related actions are not readily accessible in a consolidated view, making it difficult for officers to quickly understand the status and history of a case. As a result, management personnel are often required to sift through multiple records to reconstruct the sequence of actions taken. Consequently, these limitations contribute to delayed responses for and pose challenges to officers triaging, tracking and managing cases.

The objective of this project is to develop an AI-powered prototype that supports case classification, workflow guidance, and response generation. The system adopts an AI-assisted (not AI-controlled) approach, ensuring that human officers remain accountable for all decisions.

An iterative development methodology was employed, beginning with workflow validation before integrating AI components such as natural language processing (NLP) and rule-based decision logic. The prototype was built using Python and Streamlit, with AI functionalities designed to assist in classification, prioritisation, and drafting of resident communications. Human oversight is preserved in the decision-making processes.

The prototype demonstrates the potential to improve efficiency in case triage, enhance case visibility for officers and management, reduce administrative workload, and promote greater consistency in resident responses. It also introduces mechanisms to support transparency and accountability, including structured workflows and comprehensive audit trails. However, these benefits are inferred from the system design and observed prototype behaviour and have not yet been validated through deployment in an operational Town Council environment.

In conclusion, this work illustrates how AI-enabled systems can be integrated into public service contexts in a responsible manner, provided that human oversight, governance controls, and transparency mechanisms are embedded by design. The prototype serves as a practical example of a human-in-the-loop approach to AI adoption, rather than a claim of operational effectiveness.

2. Background

2.1. Problem Statement

Town councils receive a high volume of resident feedback daily, ranging from routine maintenance issues to complex, multi-agency concerns. These feedback submissions are received through multiple channels, including phone calls, emails, the OneService application and walk-in enquiries. Currently, the feedback handling process is largely manual, relying on a centralised case repository system and dependent on individual Town Council officers' judgement. Each case requires the officers to review the submission, interpret the nature of the issue and determine the appropriate course of action, before the case can be resolved directly or escalated to the relevant external agencies for follow-up.

The manual nature of the existing process gives rise to several operational challenges. Firstly, it creates a high administrative workload as the officers are required to devote substantial time to repetitive tasks such as case categorisation, prioritisation and routing to the appropriate parties.

Secondly, the absence of structured decision support leads to delays in case triage and response times. Without automated workflows, even straightforward cases require additional processing time as officers must manually assess and determine appropriate actions. As a result, urgent cases may not be identified and prioritised in a timely manner.

Thirdly, inconsistencies arise in case categorisation and handling across different officers. As the decisions are largely based on individual interpretation, similar cases may be classified or managed differently. This variation results in inconsistent service delivery and uneven outcome for residents.

Lastly, the current system lacks clear auditability as feedback submissions are received through different channels. Actions taken on the cases are not always systematically and consistently logged, making reviewing decisions and ensuring accountability a major challenge.

Over time, these challenges create operational bottlenecks, contributing to increased cognitive load and fatigue among Town Council officers. The combined effects reduce operational efficiency, extend response times and negatively impact residents' experience and satisfaction with Town Council services.

2.2. Objectives

In response to the challenges identified above, the project establishes a set of objectives aimed at improving efficiency, consistency, and accountability within the feedback-handling process.

The primary aim of this project is to address the identified inefficiencies and limitations in the existing feedback-handling process through the development of an AI-assisted prototype system. To achieve this aim, the project objectives are structured into three key areas:

2.2.1. Improving Operational Efficiency

Streamline the end-to-end feedback lifecycle to reduce response and resolution times by automating repetitive and time-consuming tasks such as case categorisation and initial triage. By reducing manual overhead, the system allows Town Council officers to focus on their efforts on higher-priority and more complex cases.

2.2.2. Enhancing Decision Support

To provide officers with structured, AI-driven recommendations to support decision making throughout the case handling process. This includes automated case categories, priority assignment and suggested workflow actions based on the content of residents' submissions. These features aim to improve consistency in case handling, reduce officers' cognitive load, and enable faster and more informed decisions.

2.2.3. Strengthening Transparency and Governance

Accountability and governance are critical considerations in the public sector. Accordingly, the system is designed to ensure that all actions taken throughout the case lifecycle are fully traceable through comprehensive audit trails, including actions and recommendations generated by AI components. A human-in-the-loop approach is maintained, whereby AI functions strictly as a decision-support mechanism, while officers retain full authority and responsibility over final decisions. In addition, the system provides a single, consolidated interface that presents the complete lifecycle of each case, reducing context switching and supporting effective oversight.

3. Project Limitations

This project presents a proof-of-concept prototype designed to demonstrate the potential application of AI in improving Town Council feedback-handling processes. While the proposed system addresses key challenges identified in the existing feedback-handling process, it is important to acknowledge its current limitations. Understanding these constraints is essential for evaluating the system's effectiveness and determining its readiness for real-world implementation.

3.1. System Limitations

One of the primary limitations of the prototype is that it is not production-ready. The system was developed primarily as a proof-of-concept to demonstrate functionality and feasibility, and it has not undergone the extensive testing, optimisation, or security hardening required for deployment in a live operational environment. Additional development would be necessary to ensure system stability, robustness, and compliance with organisational IT and security standards.

The prototype also relies on a single-instance SQLite database, which limits its ability to support concurrent access by multiple users. In a real-world Town Council setting, multiple officers would be required to access, update, and manage cases simultaneously. The current database architecture is not designed to accommodate such multi-user workloads, thereby constraining scalability and potentially impacting system performance under operational conditions.

Another key limitation is the absence of a formal authentication and access control framework. The prototype does not currently implement role-based access control (RBAC), which is essential for enforcing differentiated user permissions. In practice, distinct roles such as administrators, officers, and supervisors would require varying levels of access to ensure data security, accountability, and proper governance. The lack of such mechanisms introduces both operational limitations and potential security risks.

3.2. AI-Specific Limitations

From an AI perspective, the system's performance is highly dependent on input quality. The accuracy and relevance of AI-generated outputs, such as case classification and recommended actions, are influenced by how clearly and consistently resident feedback

is expressed. Variations in language, ambiguity, or incomplete information may lead to less reliable results.

In addition, the AI model has not been fine-tuned using real operational data, which limits its effectiveness in handling edge cases. While the system performs reasonably well on common and predictable scenarios, atypical or complex cases may result in lower accuracy. This highlights the need for further model training and validation using domain-specific datasets.

There are also risks related to bias and lack of continuous monitoring. Without systematic oversight, AI systems may inadvertently produce biased or inconsistent outputs over time. In the absence of monitoring mechanisms in place, such issues may go undetected, potentially affecting fairness and service quality. This reinforces the importance of implementing governance structures for ongoing evaluation and refinement of AI performance.

4. Tools Used and System Architecture

To address the identified challenges within the scope of the prototype, a modular system architecture was designed to support efficient processing, scalability, and the integration of AI components. The architecture consists of four key layers:

4.1. A Streamlit-based web interface

4.2. A Python-based backend

4.3. AI decision support components

4.4. An SQLite database

This layered architecture enforces a clear separation of concerns, allowing each component to operate independently while collectively supporting the end-to-end feedback-handling workflow. Such an approach improves maintainability, facilitates iterative development, and enables future extensibility.

The frontend interface was developed using Streamlit and serves as the primary interaction layer for both residents and Town Council officers. It enables residents to submit feedback and track case status, while allowing officers to review submissions, assess case details, and determine appropriate actions based on AI-generated recommendations. A unified interface is provided to ensure usability and reduce context switching during case handling.

The backend is implemented in Python and functions as the core processing layer of the system. It manages application logic, orchestrates workflows, processes case data, and coordinates interactions between the user interface, AI components, and the database. Centralising core logic within the backend ensures consistency, reliability, and controlled enforcement of business rules across the system.

The AI decision support component enhances the system by analysing resident submissions and generating structured recommendations, including case categorisation, priority levels, and suggested workflow actions. These recommendations are advisory in nature and are presented to officers to support decision-making. Human oversight is retained at all stages, ensuring that final decisions remain under officer control.

The database layer, implemented using SQLite, is responsible for persisting all system data, including resident submissions, case updates, workflow actions, and audit logs. This systematic data storage supports transparency, traceability, and accountability, which are essential requirements in a public sector context.

In addition, given the team's limited prior experience in developing AI-based systems, Microsoft Copilot was used as a supporting development tool. It assisted in areas such as code generation, debugging, and code refinement. All AI-generated suggestions were reviewed and validated by the project team to ensure correctness, reliability, and alignment with project requirements.

5. Project Planning and Development Phases

Building on the defined system architecture, the project adopted a structured and iterative development approach to ensure that the proposed solution was practical, systematic, and aligned with operational requirements. Development was carried out in distinct phases, with validation conducted at each stage to guide subsequent design and implementation decisions.

5.1. Phase 1: Problem Identification and Research

The initial phase focused on analysing common types of resident feedback and identifying inefficiencies in existing Town Council workflows. This included examining pain points related to case triage, categorisation, routing, and response handling. Background research was conducted concurrently on relevant AI techniques and tools to assess their potential applicability in supporting case management and decision-making processes.

5.2. Phase 2: Workflow and System Design

In the second phase, a structured decision-making workflow was designed to formalise the processes of case categorisation, prioritisation, and routing. Emphasis was placed on clearly defining operational logic and decision points before introducing any AI components. This ensured that the system design remained grounded in real-world operational practices rather than being driven solely by technological considerations.

5.3. Phase 3: Identification of Key AI Components

Following the establishment of a well-defined workflow, key AI components were identified to support specific stages of the feedback-handling process. The role of AI was deliberately scoped as a decision-support mechanism rather than an autonomous system. The identified components include:

5.3.1. Natural Language Processing (NLP) to extract relevant information from resident submissions

5.3.2. Decision-tree logic to support structured and consistent case routing

5.3.3. Agency mapping to determine the appropriate handling authority

5.3.4. Case management integration to support assignment, tracking and status updates

5.3.5. AI-assisted communication to support the drafting of responses to residents

5.4. Phase 4: System Development and Iteration

An iterative development approach was adopted during implementation, where system features were developed, tested, and refined incrementally. Core workflow logic was implemented and validated prior to the integration of AI components, ensuring that functional correctness and operational relevance were maintained throughout the development process. Feedback from testing at each iteration informed subsequent enhancements and refinements.

Overall, this phased and iterative approach enabled the system to be developed in a controlled and systematic manner, with continuous validation ensuring alignment between system functionality, AI support, and operational requirements.

6. Step-by-Step Execution

Following the planning and system design phases, the prototype was implemented through a series of structured and sequential steps. These steps translated the conceptual workflows and architectural design into functional system components, ensuring alignment with operational requirements throughout the development process.

6.1. Workflow Design

The project commenced with the design of a structured feedback-handling workflow. The team analysed the typical lifecycle of resident feedback cases and developed a decision-tree logic to guide case processing in a consistent and systematic manner.

The decision-tree logic was designed to:

- Determine whether a case falls under Town Council jurisdiction or requires escalation to an external agency
- Classify cases based on complexity and severity
- Define appropriate follow-up actions, such as direct resolution, site inspection, or escalation

Establishing this workflow at an early stage ensured that the overall system design was grounded in real-world operational practices before implementation and AI integration.

6.2. System Development

Based on the validated workflow, the core system components of the prototype were developed. These include:

- A resident submission interface for feedback entry
- An officer dashboard for case review, decision-making, and follow-up
- A workflow management system to track case progress and actions

Together, these components support the end-to-end feedback-handling process, from initial submission to case resolution or escalation.

6.3. AI Implementation

AI capabilities were integrated to support decision-making at key stages of the workflow. The AI components were designed as decision-support tools and did not operate autonomously. Specifically, AI was used to assist with:

- Case classification based on keywords, context, and structured prompts
- Priority and severity assessment of submitted cases
- Recommendation of appropriate next actions within the workflow
- Drafting of preliminary response messages to residents

All AI-generated outputs were presented as recommendations, with officers retaining full control over final decisions.

6.4. Technical Implementation Details

The technical implementation of the prototype utilised the following technologies and design choices:

- Streamlit was used to develop a multi-page web interface for residents and officers
- Python was employed for backend logic, workflow orchestration, and AI integration
- SQLite was used as the database for storing resident submissions, case updates, workflow actions, and audit logs

AI prompt structures were designed to consistently extract the following attributes from resident submissions:

- Case category
- Priority level
- Suggested workflow action

This structured approach ensured that AI outputs could be reliably integrated into the workflow logic and user interface.

6.5. Testing and Refinement

The system was tested using simulated case scenarios that reflected common resident feedback situations. Testing focused on evaluating workflow correctness, AI recommendation quality and overall system usability.

An iterative refinement process was adopted, where:

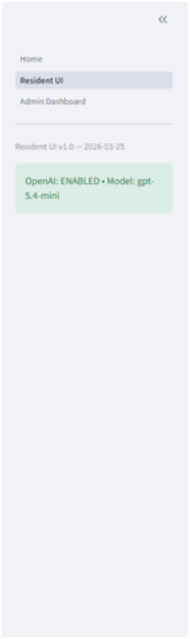
- System behaviour was reviewed based on test outputs
- AI prompts were adjusted to improve classification accuracy and consistency
- Workflow logic was refined to better reflect operational decision-making

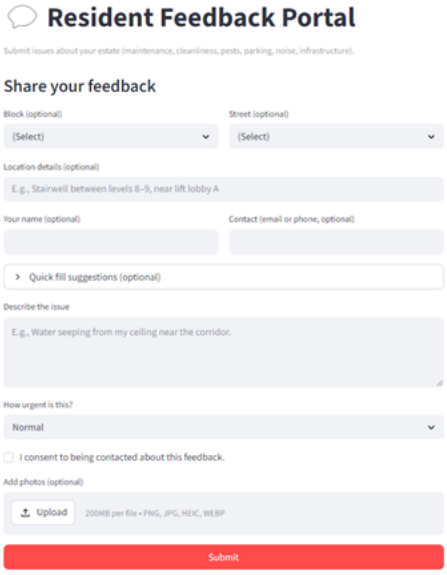
This iterative testing and refinement process helped ensure that the prototype functioned reliably within its intended scope and aligned closely with project objectives.

7. Screenshots of Outputs

This section presents the key system interfaces and outputs to illustrate how the implemented features function in practice. The screenshots and outputs demonstrate how the prototype supports residents and officers throughout the end-to-end feedback-handling workflow.

7.1. Resident Submission Interface





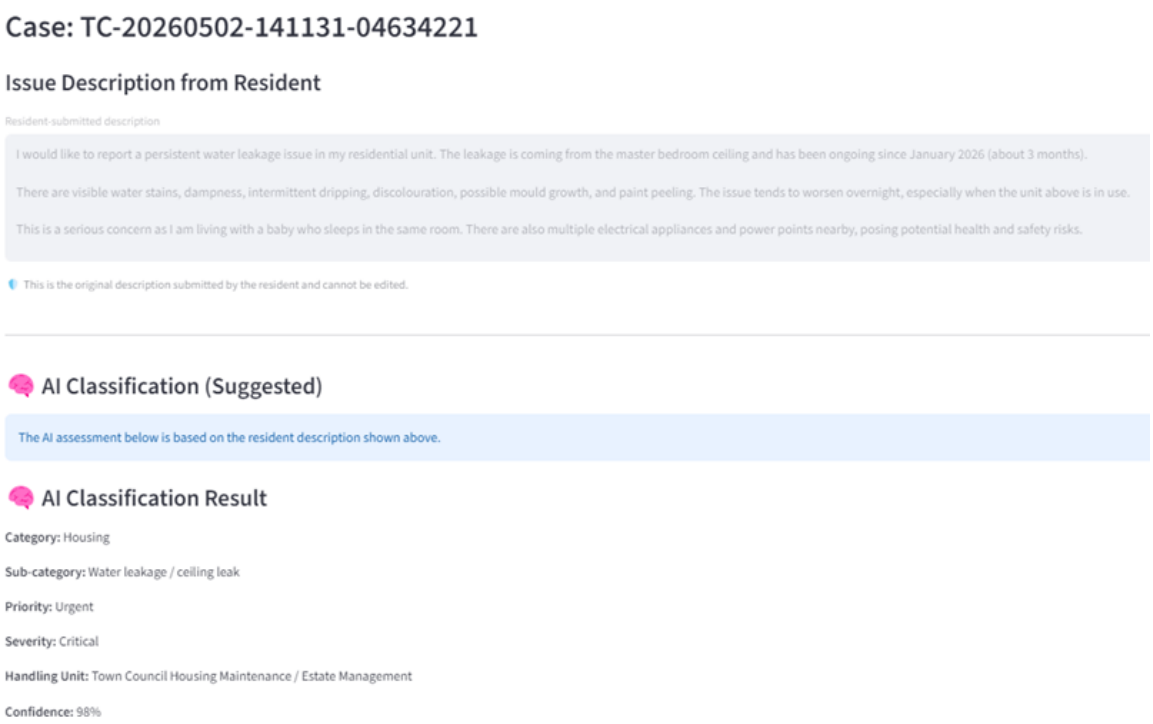
(Source: https://towncouncilapp-swt6768r9f8nz7yg3kczl2.streamlit.app/Resident_UI)

The resident submission interface serves as the primary entry point for feedback and captures essential information such as location, issue description, and perceived urgency through a structured form. This standardised input format ensures that relevant information is consistently collected, thereby reducing ambiguity and incomplete submissions.

To further assist residents, a *Quick Fill Suggestions* feature is provided. This allows users to select from common issue categories, which automatically populate a templated issue description. This feature helps residents articulate their concerns more clearly while reducing the effort required to submit feedback.

By guiding residents to provide more complete and structured information at the point of submission, the interface improves the quality of input data and supports more accurate downstream case processing and AI analysis.

7.2. AI Case Classification Output

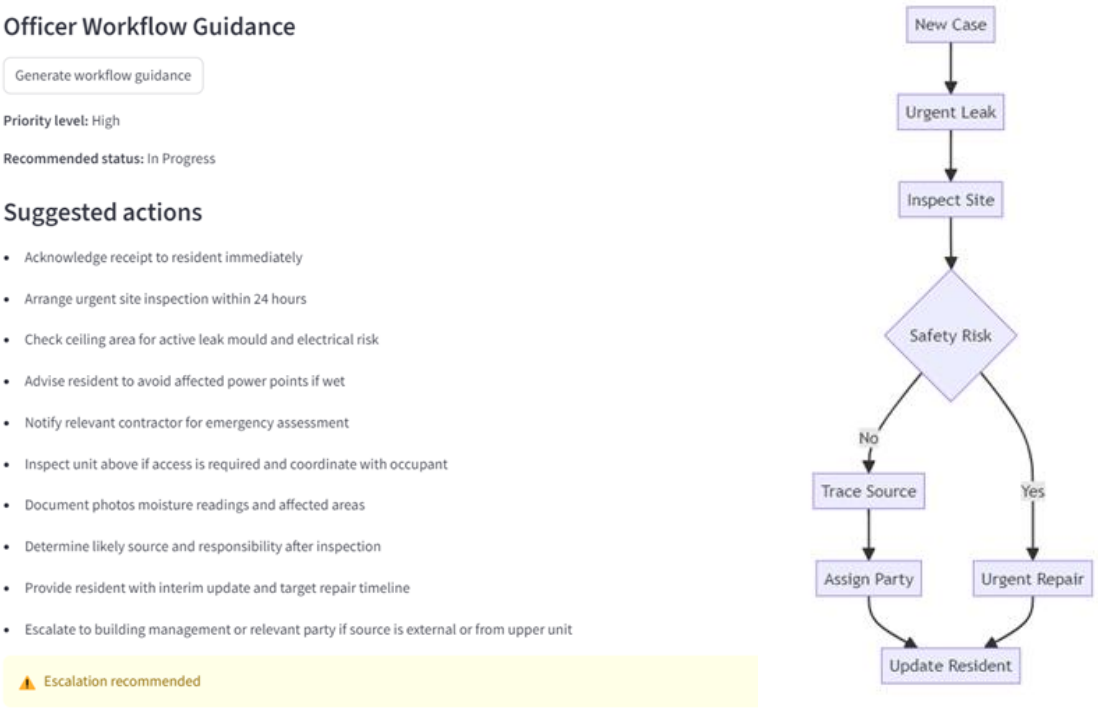


(Source: https://towncouncilapp-swt6768r9f8nz7yg3kczl2.streamlit.app/Admin_Dashboard. Password is “admin”. The link to unified portal is: https://towncouncilapp-swt6768r9f8nz7yg3kczl2.streamlit.app/)

Upon submission, the system automatically analyses resident feedback using AI to generate structured classification outputs. These include the recommended case category, assigned priority level and suggested handling unit or authority.

In addition, a confidence score is displayed alongside the AI recommendations to indicate the level of certainty in the classification. This transparency allows officers to better assess the reliability of the AI output and exercise appropriate judgement when reviewing case details.

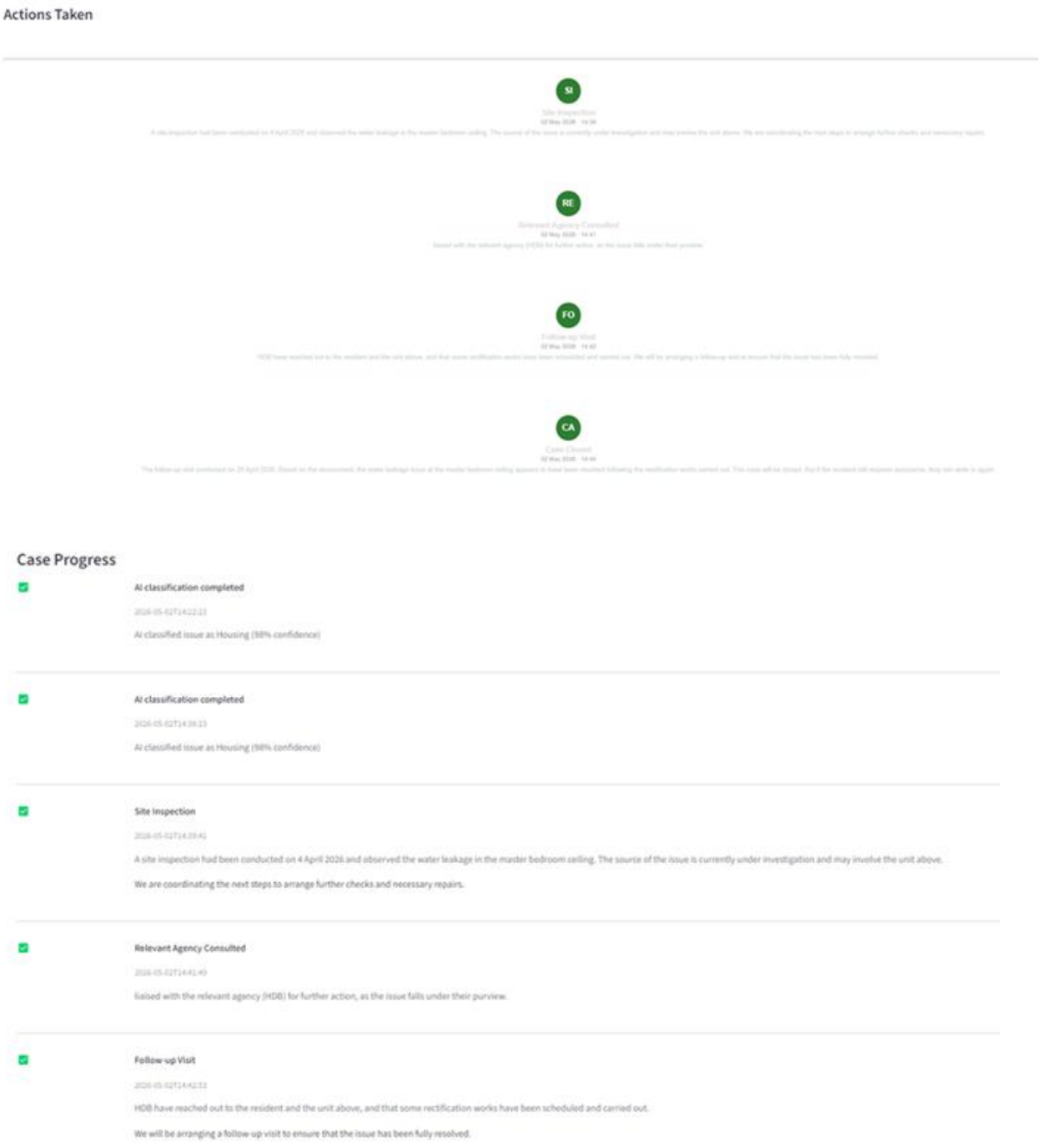
7.3. AI-Generated Officer Workflow Guidance



Based on the classified case information, the system provides AI-generated workflow guidance to support officer decision-making. This guidance includes recommended priority levels, suggested case status, and proposed follow-up actions such as inspections, on-site verification, or escalation to external agencies.

To further enhance clarity, a visual workflow diagram is generated to illustrate the sequence of actions and decision points. This visual representation enables officers to quickly understand the recommended process, reducing reliance on individual interpretation and improving consistency in case handling.

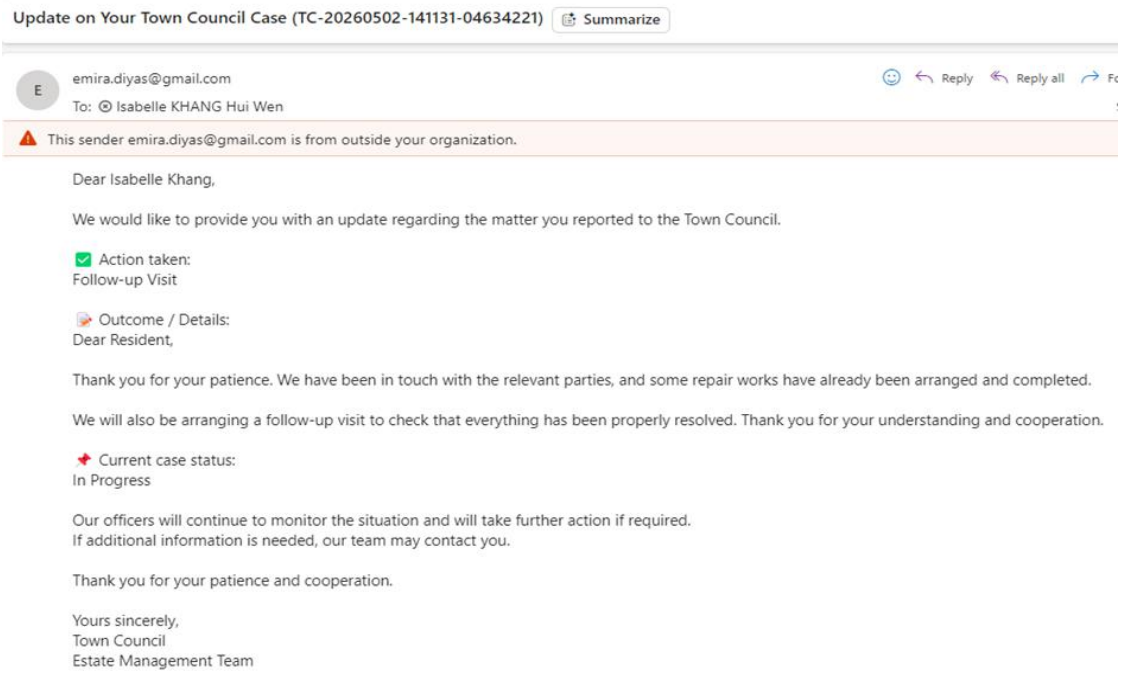
7.4. Case Tracking and Audit Trail



The system records all case-related actions in a structured, chronological timeline. This includes status updates, officer inputs, and AI-generated recommendations. The audit trail allows officers and supervisors to track case progress, review historical decisions, and ensure accountability throughout the case lifecycle.

This comprehensive logging mechanism enhances transparency and supports governance requirements, particularly in a public sector operational context.

7.5. AI-Generated Resident Communication



8. Results Interpretation

Based on the implemented features and observed system outputs, this section evaluates the effectiveness of the prototype in achieving its intended objectives.

Firstly, the use of structured input forms in conjunction with AI-assisted case classification facilitates faster and more efficient case triage. Upon submission, the system automatically generates recommended case categories, priority levels and suggested actions, thereby reducing reliance on manual interpretation. This enables officers to allocate more time to case resolution activities rather than administrative processing.

Secondly, the provision of AI-generated workflow guidance contributes to greater consistency in case handling. By offering structured recommendations and standardised workflow paths, the system reduces variability in decision-making across different officers. This promotes more uniform service delivery and supports the objective of consistent operational outcomes.

In addition, the prototype improves communication efficiency and enhances the resident experience. Automated acknowledgements and AI-assisted draft responses ensure that residents receive timely updates and clear guidance regarding their submissions. This helps manage expectations, reduces uncertainty, and improves overall satisfaction with the feedback-handling process.

Furthermore, the implementation of comprehensive case tracking and audit trails strengthens transparency and accountability. The structured, chronological recording of actions enables officers and supervisors to review decisions, monitor case progress, and ensure compliance with established workflows and governance requirements.

Overall, the results indicate that the prototype successfully demonstrates the potential of AI to support a more efficient, consistent, and transparent feedback-handling process. However, as the system has been evaluated using simulated data, further validation using real operational workloads and user testing would be required to fully assess its effectiveness and suitability for deployment in a live Town Council environment.

9. Recommendations and Future Work

While the prototype demonstrates strong potential in improving the feedback-handling process, several enhancements are required to support real-world deployment and long-term scalability.

A key recommendation is the integration of the proposed system with existing Town Council core systems. Such integration would enable seamless, end-to-end case management and real-time data exchange across operational units. In addition, the development of reporting and analytics capabilities would allow management to monitor performance metrics, identify recurring issues and support data-driven decision-making for continuous service improvement.

To improve accessibility and inclusivity, future iterations of the system should incorporate multi-language support. This would enable residents who are less proficient in English to submit feedback and receive responses in their preferred languages, such as Chinese, Malay, and Tamil, thereby improving overall service reach and resident satisfaction.

Strengthening system governance is also essential for operational deployment. The implementation of RBAC would ensure secure, role-specific access to system features and data. Differentiated access levels for administrators, officers, and supervisors would enhance security, accountability, and compliance with organisational governance policies.

Due to confidentiality requirements and compliance with the Personal Data Protection Act (PDPA), the current prototype was developed using sample data. As a result, further model training and calibration using real operational data would be necessary to improve the accuracy of AI-assisted classification and recommendation outputs. With appropriate safeguards in place, future enhancements could also include predictive and proactive capabilities, such as identifying emerging service issues or recurring problem hotspots.

The establishment of a formal AI governance framework is critical for responsible deployment. This framework should encompass ongoing performance monitoring, bias and risk management, regular audits of AI outputs, and clearly defined accountability structures. Such governance measures would ensure that AI continues to function as a transparent and trustworthy decision-support tool within the public service context.

Finally, this project is primarily practice-based and draws on one team member's professional experience working in a Town Council environment. Accordingly, system requirements, workflows and evaluation criteria were informed by firsthand operational knowledge rather than being derived solely from formal literature review.

Where applicable, general technical concepts were informed by established industry practices rather than specific academic sources.

10. Appendices

10.1. Selected Code Snippets with Emphasis on AI Components

10.1.1. Resident User Interface OpenAI Classifier Code

```python
# ------------------------------------------------------------
# Classification (rules + optional OpenAI)
# ------------------------------------------------------------

def rules_classify(text: str) -> Dict:
    t = (text or "").lower().strip()
    if not t:
        return {"category": None, "confidence": 0.0, "source": "manual"}

    scores = {
        cat: sum(1 for p in patterns if re.search(p, t))
        for cat, patterns in CATEGORY_PATTERNS.items()
    }

    best = max(scores, key=lambda c: scores[c]) if any(scores.values()) else None
    conf = 0.6 + 0.12 * (scores.get(best, 0)) if best else 0.0
    return {
        "category": best,
        "confidence": round(min(conf, 0.95), 2),
        "source": "rules" if best else "manual",
    }


def ai_classify(text: str) -> Optional[Dict]:
    """Optional OpenAI classifier; safe fallback if anything fails."""
    api_key = get_secret("OPENAI_API_KEY")
    if not api_key or not text.strip():
        return None

    try:
        from openai import OpenAI
        client = OpenAI(api_key=api_key)
        model = get_secret("OPENAI_MODEL", "gpt-5.4-mini")

        system_msg = (
            "You classify Singapore Town Council estate issues. "
            "Choose exactly one category from: maintenance, cleanliness, pests, "
            "parking, noise, infrastructure. "
            "Return JSON with keys 'category' and 'confidence'. "
            "Use 'infrastructure' for drains, footpaths, streetlights, roads."
        )
```

10.1.2. Admin Dashboard AI Classification Code

11.

TownCouncilApp / pages / 2_Admin_Dashboard.py

Code | Blame | 709 lines (558 loc) · 23.5 KB

```
163  def build_officer_action_timeline(case_id: str):
253
254      # ──────────────────────────────────────
255      # 🤖 AI Classification (Suggested)
256      # ──────────────────────────────────────
257      st.subheader("🧠 AI Classification (Suggested)")
258
259      st.info(
260          "The AI assessment below is based on the resident description shown above."
261      )
262
263      # ✅ Run AI only once per case (only when status is New)
264      if case.get("status") == "New":
265          if st.session_state.ai_classified_case_id != case["ref_id"]:
266              with st.spinner("Classifying issue using AI..."):
267                  try:
268                      ai_result = classify_issue_ai(
269                          description=issue_description,
270                          location=case.get("location", "")
271                      )
272
273                      # ✅ Persist AI result
274                      st.session_state.ai_classification = ai_result
275                      st.session_state.ai_classified_case_id = case["ref_id"]
276                      st.session_state.ai_suggested_category = ai_result.get("category")
277
278                      # ✅ Log AI classification to timeline once
279                      if not st.session_state.ai_logged_to_timeline:
280                          create_progress_entry(
281                              ref_id=case["ref_id"],
282                              step_code="AI_CLASSIFICATION",
283                              step_label="AI classification completed",
284                              notes=(
285                                  f"AI classified issue as "
286                                  f"{ai_result.get('category', 'Unknown')} "
287                                  f"({ai_result.get('confidence', 0):.0%} confidence)"
288                              )
289                          )
290
291                          st.session_state.ai_logged_to_timeline = True
```

11.1.1. Database AI Governance Code

TownCouncilApp / db.py

Code | Blame | 479 lines (399 loc) · 12.8 KB

```
256     def update_case_category(
275         conn.close()
276
277     # ------------------------------------------------------------
278     # Log officer workflow decision (AI governance)
279     # ------------------------------------------------------------
280     def log_workflow_decision(
281         ref_id: str,
282         workflow: Dict,
283         officer_decision: str,
284         officer_notes: Optional[str],
285     ):
286         conn = _get_conn()
287         cur = conn.cursor()
288
289         cur.execute("""
290             INSERT INTO workflow_decisions (
291                 ref_id,
292                 ai_used,
293                 priority_level,
294                 recommended_status,
295                 actions_json,
296                 officer_decision,
297                 officer_notes,
298                 created_at
299             )
300             VALUES (?, ?, ?, ?, ?, ?, ?, datetime('now'))
301         """, (
302             ref_id,
303             1 if workflow.get("notes", "").lower().startswith("ai") else 0,
304             workflow.get("priority_level"),
305             workflow.get("recommended_status"),
306             str(workflow.get("actions")),
307             officer_decision,
308             officer_notes,
309         ))
310
311         conn.commit()
312         conn.close()
313
```

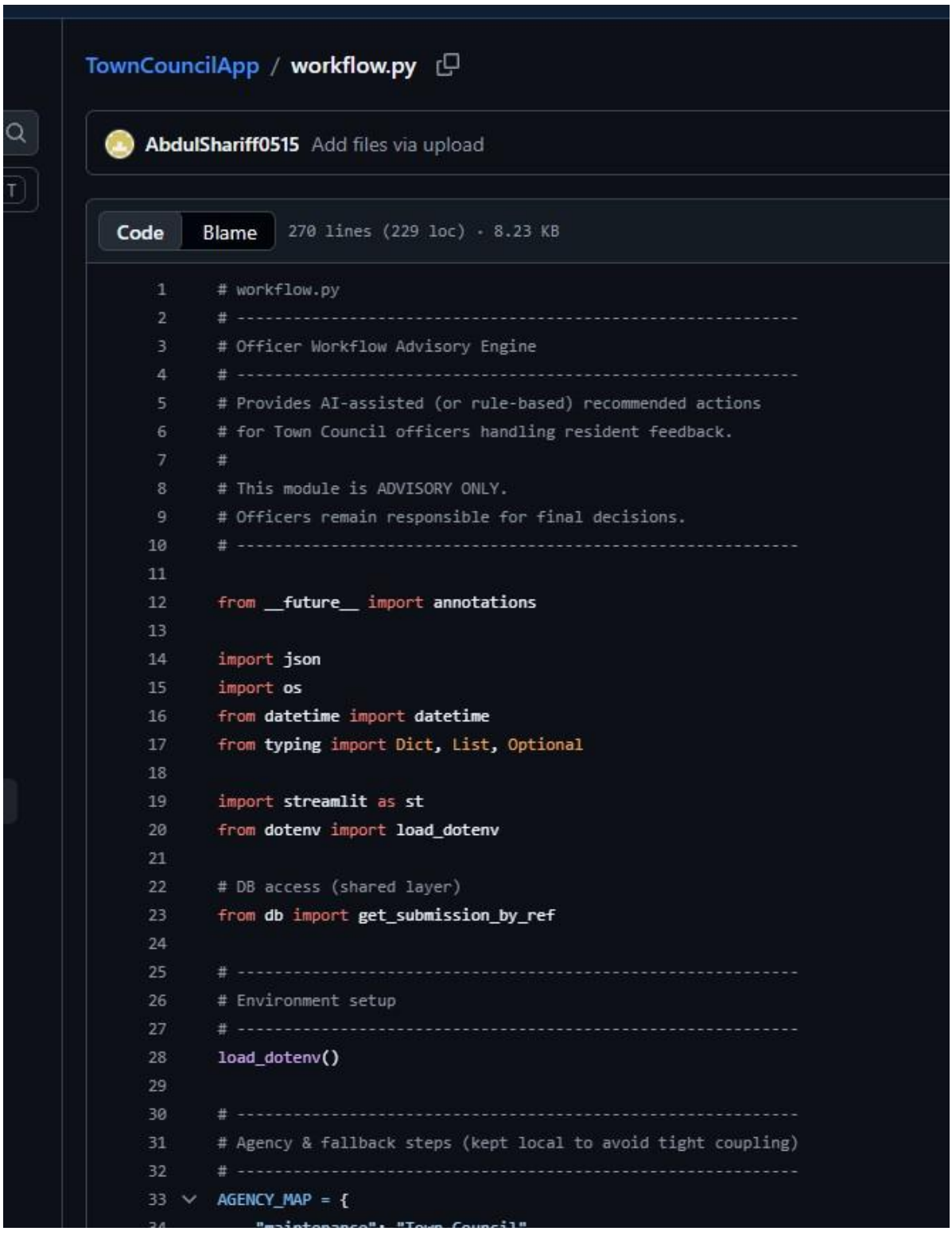
11.1.2. AI-Assisted Rewriting of Internal Notes for Resident-Facing Notifications

TownCouncilApp / notifications.py

Code Blame 107 lines (81 loc) · 3.15 KB

```python
def rewrite_action_notes_for_resident(action_notes: str) -> str:


# ------------------------------------------------------------
# Existing function (MODIFY INSIDE, not move)
# ------------------------------------------------------------

def notify_resident_of_action(
    case_id: str,
    action_type: str,
    action_notes: str,
    new_status: str,
):
    """
    Sends a resident an email update when an officer records an action.
    """

    case = get_submission_by_ref(case_id)
    if not case:
        return

    # Respect consent
    if not case.get("consent"):
        return

    resident_email = case.get("contact")
    if not resident_email:
        return

    # ✅ AI rewrites internal notes into resident-friendly language
    resident_friendly_notes = rewrite_action_notes_for_resident(action_notes)

    # ✅ Generate email using AI-rewritten content
    subject, body = generate_case_action_update_email(
        resident_name=case.get("name"),
        ref_id=case_id,
        action_type=action_type,
        action_notes=resident_friendly_notes,
        new_status=new_status,


    )
```

11.1.3. Workflow – AI-Assisted Recommended Actions



TownCouncilApp / workflow.py

AbdulShariff0515 Add files via upload

Code | Blame | 270 lines (229 loc) · 8.23 KB

```python
# workflow.py
# -------------------------------------------------------------
# Officer Workflow Advisory Engine
# -------------------------------------------------------------
# Provides AI-assisted (or rule-based) recommended actions
# for Town Council officers handling resident feedback.
#
# This module is ADVISORY ONLY.
# Officers remain responsible for final decisions.
# -------------------------------------------------------------

from __future__ import annotations

import json
import os
from datetime import datetime
from typing import Dict, List, Optional

import streamlit as st
from dotenv import load_dotenv

# DB access (shared layer)
from db import get_submission_by_ref

# -------------------------------------------------------------
# Environment setup
# -------------------------------------------------------------
load_dotenv()

# -------------------------------------------------------------
# Agency & fallback steps (kept local to avoid tight coupling)
# -------------------------------------------------------------
AGENCY_MAP = {
```

11.1.4. AI Classifications

TownCouncilApp / ai_classification.py

AbdulShariff0515 AI Classification

Code Blame 73 lines (55 loc) · 1.9 KB

```python
import os
import json
from openai import OpenAI

# Initialise OpenAI client
client = OpenAI(
    api_key=os.getenv("OPENAI_API_KEY")
)

MODEL_NAME = os.getenv("OPENAI_MODEL", "gpt-5.4-mini")


def call_llm(prompt: str) -> str:
    """
    Sends a prompt to the LLM and returns raw text output.
    """

    try:
        response = client.chat.completions.create(
            model=MODEL_NAME,
            messages=[
                {"role": "system", "content": "You are a helpful and precise assistant."},
                {"role": "user", "content": prompt}
            ],
            temperature=0.2,
            max_completion_tokens=500
        )

        return response.choices[0].message.content.strip()

    except Exception as e:
        # Centralised AI failure handling
        raise RuntimeError(f"AI service error: {str(e)}")


def classify_issue_ai(description: str, location: str | None = None):
```