# Gen AI for Platform Support – Integrated Platform Environment

**Team:** train\_test\_split

## Introduction

The **Generative AI Integrated Platform Support Environment** is designed to make the life of a platform support person easier by providing a **one-stop solution** for managing all tasks efficiently. The system includes:

* **An agentic AI-powered interactive chatbot** to assist with issue resolution, troubleshooting, and knowledge retrieval.
* **A dashboard** that displays all **incidents, change tasks (Ctasks), and relevant updates** in a structured and user-friendly way.
* **Real-time alerting system** that detects anomalies and immediately notifies the user for quick decision-making and response.

By integrating AI-driven automation and real-time insights, the solution aims to **reduce manual workload**, **speed up incident resolution**, and **improve operational efficiency**.

## Solution Overview

Our **Gen AI Integrated Platform Support Environment** provides a **one-stop solution** for platform support engineers, combining **real-time monitoring**, **intelligent automation**, and **interactive AI-powered assistance**. The system consists of two primary components:

### **1. Dashboard** (Real-time Monitoring & Incident Management)

The dashboard is the **central hub** where platform support personnel can monitor key system metrics, incidents, and automation processes. It is divided into several panels:

* **Synthetic Monitoring Panel**: Displays real-time status of application monitors (success/fail).
* **Autosys Jobs Panel**: Shows job statuses, including running, completed, and failed jobs.
* **BigPanda Alerts Panel**: Highlights active alerts and incidents detected by anomaly detection.
* **Incidents Panel**: Categorizes open incidents and Major Incident Management (MIM) cases.
* **Change Management Panel**: Displays scheduled changes and pending approvals.
* **Automation Panel**: Lists available automation scripts and their last execution status.

This dashboard provides a **holistic view** of the platform's health, helping engineers **quickly assess** and **respond** to critical issues.

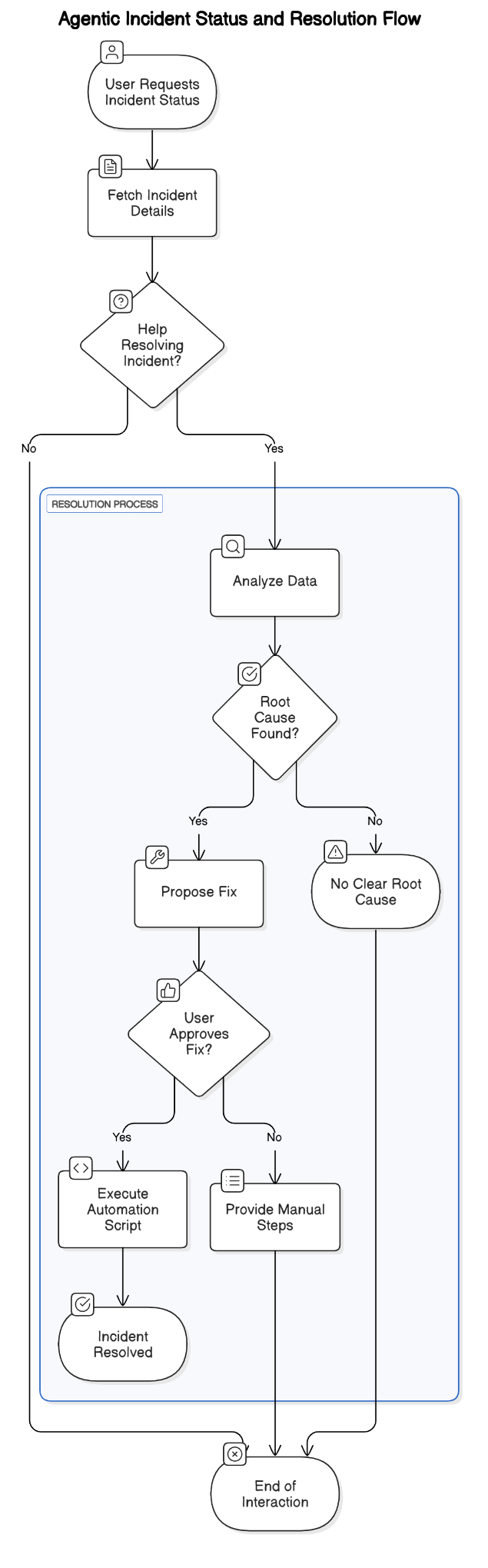
### **2. Chatbot** (AI-Powered Virtual Support Assistant)

The **AI-driven chatbot** acts as a **virtual support assistant** with **agentic capabilities**. It enables engineers to **interact conversationally** and perform various support tasks:

* **Incident Management**:
  + Retrieve incident details (priority, status, description).
  + Suggest resolutions using log analysis and monitoring insights.
  + Execute predefined automation scripts for resolution.
* **Automation Execution**:
  + Runs scripts for diagnostics and troubleshooting.
  + Launches health checks and remediation tasks.
* **Change & Task Management**:
  + Provides updates on change requests and approvals.
  + Lists pending tasks assigned to the user.

The chatbot follows an **agentic workflow**, where a **parent agent** understands the user’s query and delegates tasks to **worker agents** responsible for execution.

Below is an example decision diagram of the agentic chat bot for incident resolution.



### **3. Real-Time Monitoring & Alerting** (Proactive Issue Detection)

#### **Overview**

The **Health Check Module** is responsible for monitoring system and application metrics, detecting anomalies using machine learning, and providing real-time health status reports. It integrates with **Prometheus** to collect system metrics and uses an **AI-powered anomaly detection system** to identify potential issues before they impact operations.

#### **Key Features**

1. **System Health Monitoring**
   * Monitors CPU, memory, and disk usage using the psutil library.
   * Collects real-time system health metrics and logs performance statistics.
2. **Application Health Monitoring**
   * Integrates with **Prometheus** to collect and analyze application-level metrics.
   * Provides insights into service availability, API response times, and resource utilization.
3. **Anomaly Detection**
   * Uses the **Isolation Forest** machine learning model to detect anomalies in system and application metrics.
   * Applies statistical analysis, such as standard deviation thresholds, to flag abnormal behavior.
   * Processes metric data using a sliding window approach for real-time anomaly detection.
4. **Real-Time Alerts & Reporting**
   * Exposes a /check\_health API endpoint via Flask for retrieving system health status.
   * Alerts support engineers when critical thresholds are breached.
   * Integrates with an **AI agent** to notify external monitoring systems when anomalies are detected.

#### **Components & Implementation**

1. **Health API (health\_api.py)**
   * Provides an API endpoint (/check\_health) to:
     + Fetch Prometheus metrics.
     + Detect anomalies in real time.
     + Return health status for each monitored parameter.
2. **Agent Model (executeScript.py)**
   * Continuously monitors system metrics.
   * Detects anomalies using the **Isolation Forest** model.
   * Sends anomaly reports to an external agent (autogen).
3. **Prometheus Polling Agent (prometheus\_polling\_agent.py)**
   * Periodically fetches all metrics from Prometheus.
   * Detects anomalies across multiple monitored parameters.
   * Sends anomaly data to the **Master Agent** for further processing.
4. **System Health Check (healthcheck.py)**
   * Collects and logs system metrics such as CPU, memory, and disk usage.
   * Generates periodic health status reports.

#### **How It Works**

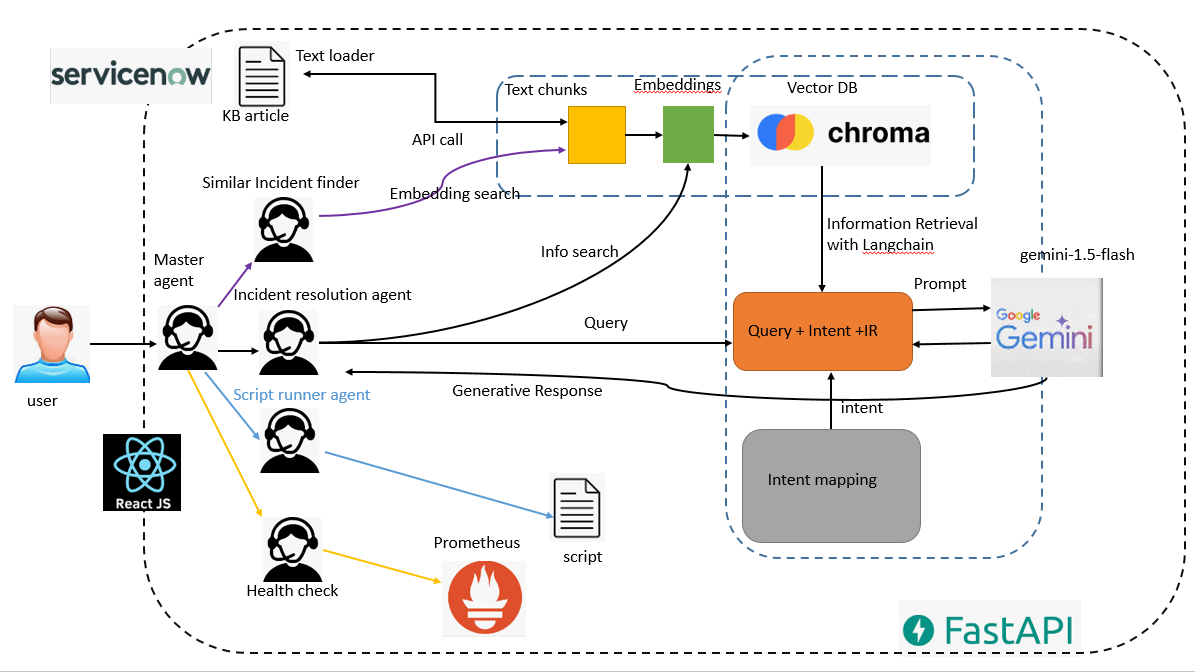
1. Prometheus collects system and application metrics.
2. The **Health API** retrieves metrics and runs anomaly detection using the **Isolation Forest** model.
3. If anomalies are detected, alerts are triggered and sent to the AI agent for further action.
4. Engineers can check real-time health status via the /check\_health API.

This module ensures **proactive issue detection**, reducing downtime and improving overall system reliability.

## Technology Stack and Usage

* Programming Languages and Frameworks
  + Back end: Python
  + Front end: JavaScript, TypsScript
* React – Front end Framework
* Embedding model – models/embedding-001 – to generate vector embeddings
* LLM model - gemini-1.5-flash
* Fast API – High performance web framework for building APIs
* Docker
* Autogen agents

## Design Diagram



## Implementation

### Dashboard

* The dashboard UI is generated using Loveable.dev AI tool
* The UI will be updated with details periodically.
* The incidents panel is
* All other panels are implemented as wireframes. It can easily be modified to fetch details in real time.

### Chat bot

### **1. Master Agent**

The master agent is built using AutoGen agents. Once the agent receives a query, it directs it to either the **Incident Resolution Agent** or the **Script Runner Agent**. All agents are implemented using AutoGen agents.

### **2. Incident Resolution Agent**

When a user requests incident resolution, the query is processed through the **RAG (Retrieval-Augmented Generation) chain** to fetch relevant information from stored embeddings and provide a response using a **large language model (LLM)**.

#### **RAG Chain Setup**

1. An API call is made to **ServiceNow** to collect all **Wells Fargo KB (Knowledge Base) articles**.
2. The text content is split into smaller segments using the **RecursiveCharacterTextSplitter** from LangChain.
3. A **ChromaDB vector database** is created, and text embeddings are stored using **models/embedding-001**.
4. An **embedding model instance** is configured to generate embeddings from text segments.
5. A **RAG chain** is constructed, integrating the previously configured models and designating them as **retrievers**.

### **3. Script Runner Agent**

When the **Master Agent** delegates a task to the **Script Runner Agent**, the script is executed, and the response is passed back to the **parent agent** and displayed to the user through the **chatbot UI**.

### **4. Similar Incident Finder**

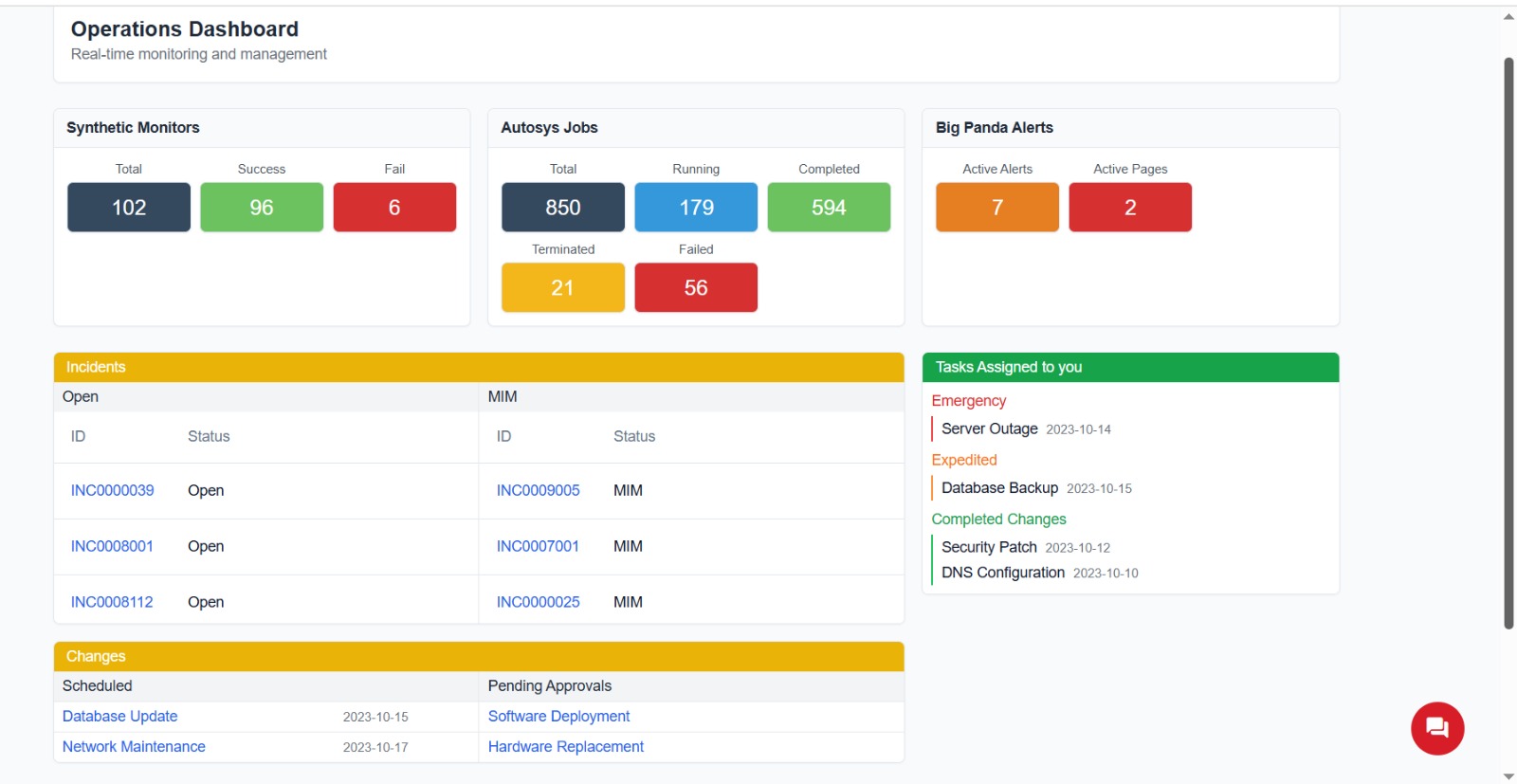
All **Wells Fargo incidents** are fetched from **ServiceNow** and stored in **ChromaDB** as embeddings. When a user queries the chatbot for **similar incidents**, a **similarity search** is performed on the stored embeddings, and the most relevant incidents are presented to the user.

### **5. Health Checker**

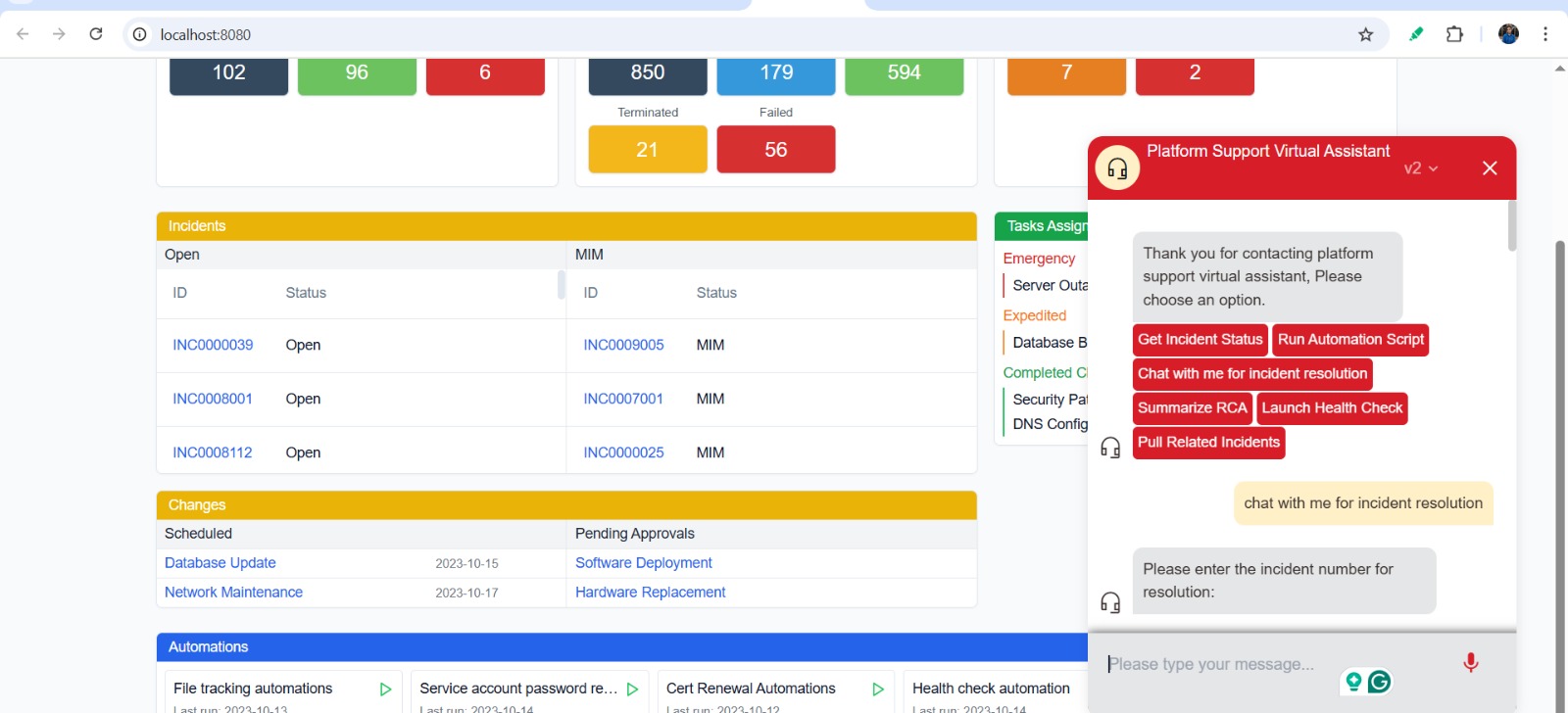
Two types of **health checks** are implemented as part of this system:  
1. **API-based Health Check** – Provides a **collective health check report** that the support engineer can access.  
2. **Agent-Based Monitoring** – An **AutoGen agent** continuously polls **Prometheus** and **triggers alerts** if an anomaly is detected.

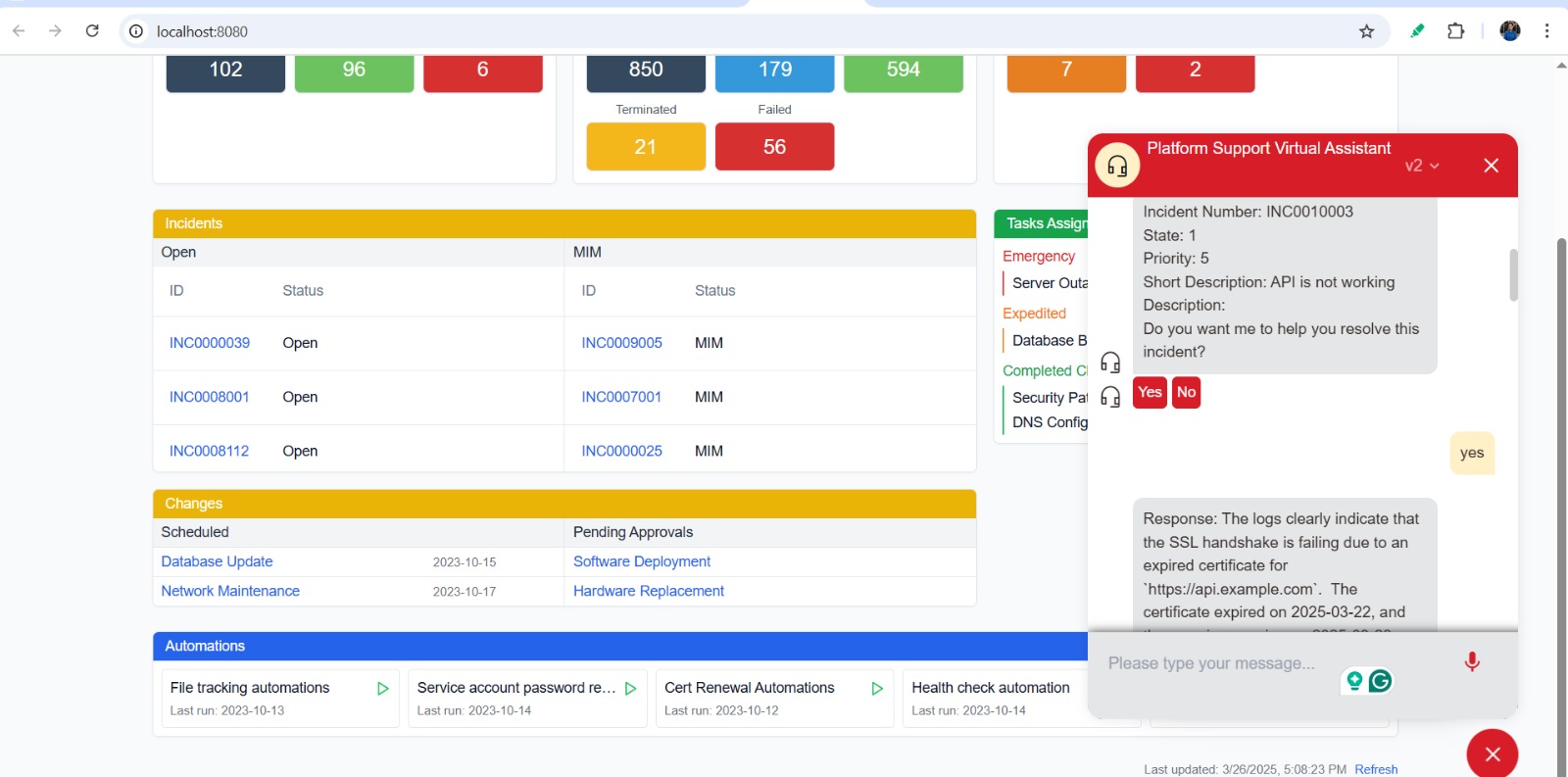
## Prototype

Dashboard



Chat bot





## Deployment Plan

Please refer deployment playbook added to git repository.

## Bill of Materials (BOM)

All the resources used are free of cost

## Conclusion

The **Gen AI Integrated Platform Support Environment** enhances platform support operations by integrating **AI-driven automation, real-time monitoring, and intelligent virtual assistance**. The system streamlines **incident resolution, reduces manual workload, and improves response times** through its interactive chatbot, real-time monitoring dashboard, and automated anomaly detection.

### **Key Benefits**

* **Faster Incident Resolution** – AI-powered incident analysis and automation scripts accelerate troubleshooting.
* **Proactive Monitoring & Alerting** – Real-time anomaly detection helps prevent failures before they escalate.
* **Knowledge-Driven Support** – The **Retrieval-Augmented Generation (RAG) pipeline** provides relevant and accurate solutions.
* **Seamless Integration** – Connects with **ServiceNow, ChromaDB, and Google Gemini** to provide a centralized support experience.
* **Scalability & Extensibility** – The system is **modular** and can be expanded with additional automation scripts, AI models, and integrations.

By leveraging AI and automation, this **intelligent support ecosystem** ensures that platform engineers can efficiently manage incidents, automate routine tasks, and proactively address system issues—ultimately driving higher **operational efficiency and reliability**.

## References

1. <https://chatgpt.com/>
2. <https://microsoft.github.io/autogen/dev//user-guide/agentchat-user-guide/tutorial/agents.html>
3. <https://docs.crewai.com/introduction>
4. <https://aws.amazon.com/what-is/retrieval-augmented-generation/>