Project Documentation: ServiceNow AI Agent System  
  
Prepared by: Sai Ganesh Vuppala

Date: May 27, 2025

**1. Introduction**

This document outlines the design, implementation, and operational details of the ServiceNow AI Agent System, a personal project I’ve been working on to automate IT ticket creation and management. The system leverages email monitoring, AI-driven analysis, and integration with ServiceNow to streamline workflows. I started this project out of curiosity about combining AI with IT service management, and it’s been an exciting journey so far! The goal is to have a functional prototype running on my Mac, which I can expand later if needed.

**2. Project Overview**

**2.1 Purpose**

The ServiceNow AI Agent is designed to automatically process emails received at projectdummy144@gmail.com, analyze their content, and create appropriate tickets (incidents or change requests) in my ServiceNow Personal Developer Instance (PDI). For change requests, it also sends approval emails to dummymanager144@gmail.com. This saves me time and helps simulate a real-world IT support scenario.

**2.2 Scope**

* **In Scope**: Email monitoring, AI-based ticket classification, ticket creation in ServiceNow PDI, and approval email notifications.
* **Out of Scope**: Advanced ticket updates (e.g., status changes beyond creation), integration with multiple email accounts, or production-grade scaling.

**2.3 Objectives**

* Automate ticket creation from emails every 30 minutes.
* Ensure secure handling of credentials and data.
* Provide logging for troubleshooting and tracking.

**3. System Architecture**

**3.1 Components**

* **AI Agent**: A Python script (ServiceNowAgent.py) running in PyCharm on my Mac, housed in /Users/saiganeshvuppala/PythonProject/ServiceNowAgent/. It’s the heart of the system, handling all logic.
* **Gmail API**: Fetches email and sends approval notifications. Authenticated via OAuth 2.0 using a credentials.json file I downloaded from Google Cloud.
* **Google Gemini API**: Analyzes email content to decide if it’s an incident or change request. I’m using the free tier (gemini-1.5-flash), which has a 500-request daily limit.
* **ServiceNow PDI**: My development instance at https://dev305029.service-now.com, where tickets are created.
* **Cron Job**: Schedules the script to run every 30 minutes using a cron job I set up on my Mac.
* **Configuration**: Stored in a .env file with sensitive details like API keys and passwords.
* **Logging**: Tracks actions in ticket\_log.txt for debugging.

**3.2 Data Flow**

1. The cron job triggers the script every 30 minutes.
2. The Gmail API checks for unread emails addressed to projectdummy144@gmail.com.
3. The AI Agent extracts the email body and, if it starts with “Change:”, creates a change request directly. Otherwise, it uses the Gemini API to classify the email.
4. Based on the classification, it sends a POST request to the ServiceNow PDI to create a ticket (e.g., INC0010110 or CHG00000XX).
5. For change requests, an approval email is sent to dummymanager144@gmail.com.
6. The email is marked as read, and the action is logged.

**3.3 Diagram**

1. I’ve attached a blueprint-style system design image (generated separately) that shows the flow between these components. It’s a bit rough, but it captures the connections nicely—Cron Job feeding into the AI Agent, which talks to Gmail, Gemini, and ServiceNow, with logs and config files in the background.

**4. Implementation Details**

**4.1 Technology Stack**

* **Language**: Python 3.9
* **IDE**: PyCharm (Community Edition)
* **Libraries**: requests, googleapiclient, google-auth-oauthlib, google.generativeai, dotenv, logging
* **OS**: macOS (my personal laptop)
* **Scheduler**: Cron

**4.2 Code Structure**

* **Main Script**: ServiceNowAgent.py contains functions like:
  + get\_gmail\_service(): Sets up Gmail API.
  + analyze\_email(): Classifies email content (with a fallback for “Change:” emails).
  + create\_ticket(): Interfaces with ServiceNow PDI.
  + send\_approval\_email(): Sends notifications.
* Environment: A virtual environment (.venv) keeps dependencies isolated.
* Config: .env file holds SNOW\_URL, SNOW\_USER, SNOW\_PASS, GMAIL\_ADDRESS, MANAGER\_EMAIL, and GEMINI\_API\_KEY.

**4.3 Setup Instructions**

1. Clone the repo (or just copy the files) to /Users/saiganeshvuppala/PythonProject/ServiceNowAgent/.
2. Create a .venv with python3 -m venv .venv and activate it with source .venv/bin/activate.
3. Install dependencies: pip install -r requirements.txt (I need to create this file still!).
4. Set up Google Cloud credentials for Gmail API and get a Gemini API key.
5. Add details to .env and secure it with chmod 600 .env.
6. Schedule the cron job: crontab -e and add \*/30 \* \* \* \* /Users/saiganeshvuppala/PythonProject/ServiceNowAgent/.venv/bin/python /Users/saiganeshvuppala/PythonProject/ServiceNowAgent/ServiceNowAgent.py >> /Users/saiganeshvuppala/PythonProject/ServiceNowAgent/cron\_log.txt 2>&1.

**5. Security Considerations**

* **Authentication**: Gmail uses OAuth, ServiceNow uses basic auth, and Gemini uses an API key—all stored in .env.
* **File Permissions**: I’ve set .env to read/write only for me to avoid accidental exposure.
* **Data Handling**: Logs don’t include passwords or keys, and email bodies are processed in-memory.

**6. Testing and Validation**

**6.1 Test Plan**

* **Manual Test**: Send an email to projectdummy144@gmail.com with Subject: “Test Incident 2025-05-27” and Body: “My laptop crashed!”. Run the script manually and check for an INC... ticket in PDI.
* **Change Request Test**: Send Subject: “Test Change 2025-05-27” and Body: “Change: Install new software”. Verify a CHG... ticket and approval email in dummymanager144@gmail.com.
* **Cron Test**: Wait for the next run at 9:30 AM CDT and send another test email.

**6.2 Expected Outcomes**

* Tickets should appear in PDI with correct details.
* Approval emails should arrive in the manager’s inbox.
* Logs should reflect all actions without errors.

**6.3 Issues Encountered**

* I hit a Gemini API quota limit (500 requests/day) yesterday, which caused some emails to be ignored. I’ve added a workaround to bypass Gemini for “Change:” emails, and I’ll wait for the quota to reset (around 11:00 PM CDT) or get a new API key if needed.

**7. Future Improvements**

* Add support for ticket updates (e.g., setting to “Resolved”).
* Implement a queue for handling multiple emails at once.
* Enhance security with environment variable encryption.
* Create a simple UI in PyCharm to monitor the system.

**8. Conclusion**

* This project has been a great learning experience! The ServiceNow AI Agent is functional for basic ticket creation and approval workflows. I’m planning to test it thoroughly today and might share it with a friend for feedback. Any suggestions would be appreciated

**9. Appendices**

**Appendix A: Sample Log Entry**

2025-05-27 09:00:00: Starting ServiceNowAgent...

2025-05-27 09:00:01: Gmail service initialized successfully.

2025-05-27 09:00:02: Fetching emails...

2025-05-27 09:00:03: Found 1 unread emails to process.

2025-05-27 09:00:04: Processing email ID: 1970f40755686f5f

2025-05-27 09:00:05: Fetched email subject: Test Incident 2025-05-27

2025-05-27 09:00:06: Email analysis result: {'action': 'create\_incident', 'priority': 'high', ...}

2025-05-27 09:00:07: Ticket Created: INC0010111 (ID: CHG0030002)

2025-05-27 09:00:08: Marking email 1970f40755686f5f as read.

**Appendix B: Contact**

* **Email**: [vuppalasaiganesh144@example.com](mailto:vuppalasaiganesh144@example.com)
* **GitHub**: https://github.com/vuppalasaiganesh