The 3NGINE – Autonomous AI Productivity Agent

Course: ITAI2376 – Intelligent Agents and Interfaces

Group Name: The3NGINE Team

Team Members: Kolapo Mogaji

Date: May 5, 2025

1. Project Overview

The 3NGINE is a generative AI agent designed to act as a personal productivity engine. It interprets natural language prompts and autonomously executes digital tasks such as sending emails, scheduling calendar events, and summarizing information. It offers a streamlined user experience, eliminating the need to manually organize and track digital tasks.

By simulating the behavior of a human assistant, The 3NGINE can manage tasks with a level of autonomy that saves time and reduces mental load. The goal of the project is to create a system that seamlessly integrates AI-driven reasoning with practical, real-world productivity tools such as Google Calendar and (planned) Gmail integration.

2. System Architecture

The agent system follows a modular architecture comprising the following components:

- Input Processing: Captures and interprets user prompts written in natural language.
- Classification Logic: Uses keyword-based rules to determine the type of task (e.g., email, calendar, unknown).
- Reasoning Module: Applies the ReAct (Reason + Act) pattern to map decisions and tool usage.
- Tool Integration Layer: Currently integrates with Google Calendar via OAuth2. Gmail and memory modules are planned.
- Output Generation: Produces system responses and executes real actions like scheduling events.

- Safety and Control: Ensures boundary enforcement through fallback messages and input validation.

3. Implementation Details

The agent is implemented using Python and organized into separate modules for clarity and scalability. The `main.py` script serves as the entry point and routes user prompts through the classification and reasoning logic.

The Google Calendar API is integrated through OAuth2 authorization and Python's `google-api-python-client` package. A simulated email module is in place, and a future version will implement Gmail API support for live emailing. Tasks are logged and interpreted through defined keyword-based flows, and a memory system is outlined for tracking previous tasks.

4. Evaluation Results

The following test scenarios were used to evaluate agent behavior:

Test Case 1: 'Schedule a meeting next Friday'

→ Expected: Create a calendar event at 2 PM CST

→ Result: ✅ Successful

Test Case 2: 'Send follow-up email to project team'

→ Expected: Simulated email confirmation

→ Result: ✅ Successful

Test Case 3: Invalid prompt ('cd /Downloads')

→ Expected: Return fallback message

→ Result: ✅ Successful

5. Challenges and Solutions

- Parsing temporal phrases such as 'next Friday' required natural language date parsing (currently hardcoded).
- OAuth setup with Google APIs required testing and debugging browser-based consent flows.
- Differentiating between shell commands and user prompts necessitated strict input handling.
- Addressed tool detection errors through improved keyword mapping and validation logic.

6. Lessons Learned

This project reinforced core AI agent concepts such as modular architecture, external tool integration, and ReAct-based planning. It also highlighted the complexity of translating vague natural input into precise action.

Practical lessons included:

- How real APIs like Google Calendar function and authenticate via OAuth.
- The importance of fallback logic and user input safety.
- How essential it is to iterate quickly with a working prototype before scaling features.

7. Future Improvements

- Incorporate natural language time parsing with the `dateparser` or `duckling` libraries.

- Expand memory system using vector databases (e.g., FAISS) to log, retrieve, and reason over past interactions.

- Integrate Gmail API for live email execution.

- Explore LangChain or OpenAI tool use for structured task orchestration.

- Add UI for user-friendly web-based interaction with the agent.