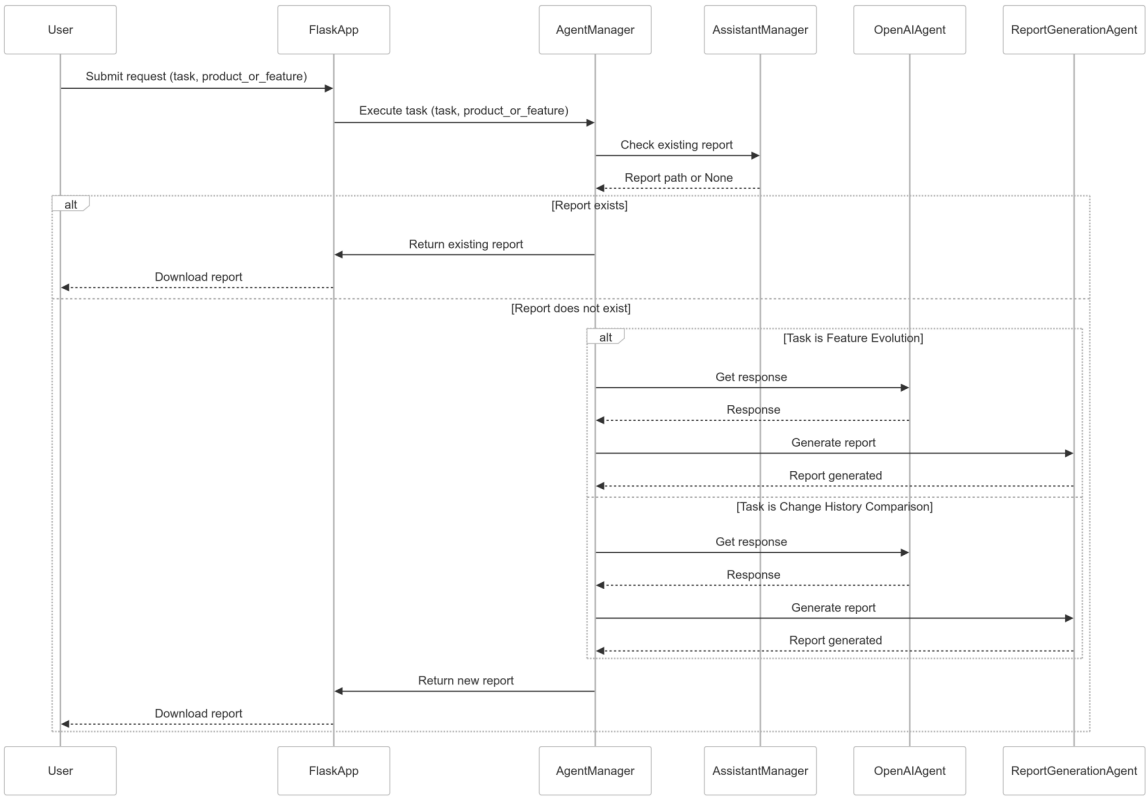
Doc Agent - Code Flow - Current - as of 25-Apr-2025

1. Sequence Diagram

The sequence diagram will illustrate the interactions between the different components of the system when a user requests to generate or fetch a report.

```
sequenceDiagram
    participant User
    participant FlaskApp
    participant AgentManager
    participant SharePointAgent
    participant OpenAIAgent
    participant ReportGenerationAgent
    participant AssistantManager
    User->>FlaskApp: Submit request (task, product_or_feature)
    FlaskApp->>AgentManager: Execute task (task, product_or_feature)
    AgentManager->>AssistantManager: Check existing report (task, product_or_feature)
    AssistantManager-->>AgentManager: Report path or None
    alt Report exists
        AgentManager->>FlaskApp: Report exists, download report
    else Report does not exist
        alt Task is Feature Evolution
            AgentManager->>OpenAIAgent: Get response (system_instruction, prompt, model)
            OpenAIAgent-->>AgentManager: Response
            AgentManager->>ReportGenerationAgent: Generate report (response, output_file, task)
            ReportGenerationAgent-->>AgentManager: Report generated
        else Task is Change History Comparison
            AgentManager->>OpenAIAgent: Get response (system_instruction, prompt, model)
            OpenAIAgent-->>AgentManager: Response
            AgentManager->>ReportGenerationAgent: Generate report (response, output_file, task)
            ReportGenerationAgent-->>AgentManager: Report generated
        end
        AgentManager->>FlaskApp: Report generated
    end
    FlaskApp-->>User: Report generated or downloaded
```



2. Overall Design Diagram

The overall design diagram will show the main components and their interactions.

```
graph TD
    A[User] --> B[FlaskApp]
    B --> C[AgentManager]
    C --> D[SharePointAgent]
    C --> E[OpenAIAgent]
    C --> F[ReportGenerationAgent]
    C --> G[AssistantManager]
    D --> H[SharePointService]
    E --> I[callOpenAPIAssistant]
    F --> J[EnhancedReportGenerator]
    G --> K[ProductAssistant]
    G --> L[FeatureAssistant]
```

3. Flow of Events

1. **User Interaction**:
    - The user submits a request through the Flask web application, specifying the task and product or feature.
2. **Flask Application**:
    - The Flask application receives the request and calls the AgentManager to execute the task.
3. **Agent Manager**:
    - The AgentManager checks if an existing report is available using the `AssistantManager`.
    - If the report exists, it downloads the report and returns it to the Flask application.
    - If the report does not exist, it proceeds to generate a new report.

4. **Generating a New Report**:
    - Depending on the task, the AgentManager calls the OpenAIAgent to get a response from OpenAI.
    - The OpenAIAgent interacts with the callOpenAPIAssistant function to get the response.
    - The AgentManager then calls the ReportGenerationAgent to generate the report using the response from OpenAI.
    - The ReportGenerationAgent uses the EnhancedReportGenerator to format and generate the PDF report.
5. **Returning the Report**:
    - Once the report is generated or downloaded, the AgentManager returns the report to the Flask application.
    - The Flask application then sends the report to the user.

Summary

These diagrams and the flow of events provide a clear understanding of how the system components interact to generate or fetch a report based on the user's request. The sequence diagram illustrates the step-by-step interactions, the overall design diagram shows the main components and their relationships, and the flow of events describes the process in detail.