**Process Documentation**

**Code Generation**

1. **Prompting the LLM:**

* **Task Definition:** Initially, the task was to develop a RESTful API using Node.js. The requirements included CRUD operations on to-do list tasks.
* **LLM Interaction:** Queries were made to the LLM to generate boilerplate code for Node.js using Express framework and MSSQL for database operations. Specific endpoints (GET, POST, PUT, DELETE) were requested to manage tasks.
* **Code Integration:** The generated code snippets were integrated into a structured Node.js application, ensuring that each function and middleware was correctly placed within the project architecture.

1. **Validation and Refinement:**

* **Testing:** The API was tested using Postman to ensure that each endpoint responded correctly to various HTTP requests. Adjustments were made based on the responses and errors noted.
* **Code Review and Refinement:** Code was reviewed for best practices in JavaScript and Express. Refinements included adding proper error handling and response statuses to improve the robustness of the API.

**Testing**

1. **Prompting the LLM:**

* **Task Definition:** The task involved developing comprehensive tests for the RESTful API managing to-do list operations, ensuring both functionality and security.
* **LLM Interaction:** Queries were initially made to the LLM to generate test cases using Mocha and Chai. However, based on subsequent requirements and encountered issues, the focus shifted to using Jest and Supertest. Specific endpoints (GET, POST, PUT, DELETE) were targeted for generating detailed test scenarios.
* **Code Integration:** The initial test cases provided by the LLM were integrated and then modified to adapt to the Jest and Supertest frameworks, ensuring that all API endpoints were thoroughly tested.

1. **Validation and Refinement:**

* **Initial Testing with Chai:** Initial tests were created using Chai, but issues with package compatibility and execution led to reconsideration of the testing approach.
* **Switch to Jest and Supertest:** Due to difficulties with Chai, you requested a switch to Jest and Supertest for a more integrated testing experience. The LLM then generated new test cases adapted to these technologies.
* **Testing:** After integrating the new testing framework, all endpoints were tested comprehensively to validate functional aspects and security implementations, such as JWT authentication.
* **Code Review and Refinement:** The test code underwent a thorough review to refine assertions and improve coverage. Adjustments were made based on initial test outcomes, focusing on HTTP status codes and error message handling. Tests were also extended to cover edge cases and failure modes.

**Iterative Improvement**

1. **Security Enhancement with JWT:**

* **Task Definition:** To add an authentication layer to the API, JWT was chosen to secure the endpoints.
* **LLM Interaction:** Specific prompts were given to the LLM to generate JWT implementation snippets for Node.js, focusing on creating tokens during user authentication and validating them on subsequent requests.
* **Implementation:** The generated snippets were adapted to fit the existing API structure, ensuring that the authentication flow was seamless and secure.

1. **Validation and Refinement:**

* **Testing:** After integrating JWT, the endpoints were tested again to ensure that access without a valid token was restricted. Adjustments were made based on the token expiration and error messages received.
* **Code Review and Security Checks:** The security implementation was reviewed to ensure it met security standards. Environment variables were used to store sensitive information like JWT secrets and hashed passwords, enhancing security.