Test Strategy Design

1. Product

Overview: A web-based to-do list application designed for task management. The application should allow users to:

Add Tasks: Users can create new tasks with a title, description and deadline.

Update Tasks: Users can change existing tasks.

Delete Tasks: Users can remove tasks.

Lists: Users can organize tasks into different lists or categories.

Deadlines: Users can assign deadlines to tasks.

Mark as Completed: Users can mark tasks as completed.

User Interface:

- List View: A view that shows all tasks in a specific list, their status and deadlines.
- Task View: A view for editing tasks.
- List Management: Interface for creating, editing or deleting lists.

Backend:

- Database: Stores tasks and lists in a MSSQL database.
- API: Provides endpoints for creating, updating, deleting and reading tasks and lists.

2. Scope

Inclusions:

Development of core functionalities for tasks (add, update, delete, categorize, set deadlines, mark completed).

Database schema design and integration.

User interface design and implementation.

API development and integration with the frontend.

Exclusions:

Advanced features like notifications or task prioritization.

User login and sign up.

3. Objective

Ensure that the application meets all functional requirements.

Ensure the application is stable and reliable.

Confirm that the application performs well with typical user interactions.

4. Test Types

Unit Tests:

Test individual components and functions.

Example: Test the functions for adding tasks, updating tasks and completing tasks.

Integration Tests:

Test the interaction between multiple components and functions.

Example: Test the interactions between the frontend and backend or between the application and the database.

Specification-Based Tests:

Test if the application fulfills the functional requirements and user stories.

Example: Test if a user can successfully create a task and add it to a list.

5. Test Tools

Use XUnit for writing and running unit tests, integration tests and specification-based tests.

Automate the CI/CD pipeline for building and testing the application with GitHub Actions.

6. Exit Criteria

- All unit tests, integration tests and specification-based tests pass successfully.
- The application fulfills all functional requirements and user stories.
- No critical bugs or issues.

Differences between each type of test:

Unit tests are used to test individual components or functions. This ensures that the individual components work and make it easy to identify where a bug or issue might be in the code. It guarantees that the core logic and functions work and makes it easier to refactor without being worried about breaking components or functions since it is easy to identify where the problem is.

Integration tests are used to test how multiple components interact with each other. This could be when the system gets data from a database or API. They ensure that there is no issue when combining components.

Specification-based tests are used to test the functional requirements or user stories. It ensures that the program works and specified in the user stories. They are seen from the perspective of the user and how they interact with the system. They reduce the chance of having missing or incorrect features.