Feature: Task Manager Hub

Developer: Kay Kayale

Submission Date: 04/14/24

Reviewer: Sarah Santos

Review Date: 04/15/24

Major Positives

- Comprehensive and uniform diagrams: All LLDs are concise and easy to follow how an authenticated user will navigate the Task Manager Hub feature. This demonstrates a clear understanding of the logic flow for Task Manager Hub and the operations an authenticated user can do within the feature. In addition, all LLDs provide corresponding response return objects after every function call in detail.
- Applied all four CRUD operations: All CRUD operations are displayed for each LLD (create, read/list, update/modify, delete), considering the Task Manager Hub as a dynamic application feature. This ensures the feature is comprehensible and adaptable to changes at any time. In addition, this demonstrates user interactions with the database and application.
- Demonstrated separation of concerns. Designs display organized sections of each class and their function within the feature. Furthermore, each design is separated by CRUD operation and displays different class functions based on the operation. This makes maintenance or additions to the feature design easy with minor changes being made to the diagrams.

Major Negatives

- Lack of detail with frontend sections: Frontend design from “index.html” to “TaskManagerHubController” are broad descriptions. There is a lack of detail to describe how each step is being performed. For example, with modify task, how does the user choose to modify a task in the index.html? From a list, with a button within the task, or a button specifically for a modifying tasks section? Providing the front-end operations (as you did in the backend) will make it easier for the future developer to correctly apply the design (see Design Recommendations).
- No backend failure cases: The only major failure designs provided were invalid inputs for the create task and modify task operations. These failure cases are only displayed within the frontend. However, according to the BRD, there must be backend failure cases for create, modify, and delete (see Unmet Requirements). In addition, the diagrams should display logging details of an error occurring for all failures.
    - Timeout Error LLDs: The timeout error diagrams are similar to their corresponding success diagrams. There is no distinction between them besides “ERROR: Operation took longer than 3 seconds” displayed above the diagrams. (see Design Recommendations)

Unmet Requirements

- Missing numbered steps: All diagrams are missing numbered steps. This is a requirement for all LLDs in order to sequentially understand the flow and logic of the design. Ensure that you go back and add all the numbers for each step of your LLDs.
- Missing failure outcomes from BRD: According to the BRD, there should be a few failure outcomes that should be displayed which include:
    - Authenticated users successfully create a new task, but the task isn't added to the to-do list and the system marks the outcome as failed. The system will display a “Failed to create new task. Please try again.” message.
    - Authenticated users make an editing error when attempting to update an existing task. The system will display an “Editing Error. Make Edits.” message.
    - Authenticated users make edits to a task, but these changes are not reflected in the task, but the system marks the outcome as failed. The system will display a “Failed to make edits. Please try again.” message.
    - Authenticated user deletes the task, but the task still appears and the system marks the outcome as failed. The system will display a “Failed to delete task. Please try again.” message.
- Completed Tasks: According to the BRD, the task manager hub is displayed as a to-do list and should allow users to edit their tasks as completed. Provide a design for how completed tasks will be handled and/or displayed within the feature.

Design Recommendations

- More details with frontend design: Provide detailed operations that must be executed within the frontend portion of the designs (from “index.html” to “TaskManagerHubController”).
    - Positives:
        - This helps the future developer to correctly apply the intended frontend design of the feature.
        - Provides clear logic of how the frontend operations are performed between each class.
    - Negatives:
        - Reduces flexibility in the design during the development process.
        - Increases complexity in understanding and implementing the frontend design.
- Display errors (logged) within the design: Demonstrate how errors are handled within the design, specifically logging the errors and how they are handled in the backend.
    - Positives:
        - Provides a distinct difference between success and failure diagrams which helps the developer how and where to handle the errors.
        - Improves debugging and testing for failure cases in the future, making it efficient to find where the error occurs within the feature.
    - Negative:
        - Possible overengineering which may cause unnecessary design complexities and efforts to noncritical elements if errors are minor.

- Create an LLD for notification settings: One of the properties of the task object is “notificationSetting.” Providing an LLD for the task manager hub notification operation will help the developer in designing how to build the different settings that can be set by the user.
    - Positives:
        - Clear implementation guidelines, increasing efficiency and reducing ambiguity.
        - Improves scalability and integration within the feature and possibly for other necessary features as well.
    - Negatives:
        - Increased initial effort can cause a delay in development and to finish within the deadline.
        - Over-specifying minor details can produce unnecessary efforts for noncritical operations. Instead, you can specify implementing MailSender service to save time and effort.

Test Recommendations

- End-to-end testing is important to ensure the user will have a seamless interaction with your feature.
    - Test task data is being properly fetched from the data store:
        - Ensure all tasks are displayed in a clear and organized manner.
        - Ensure correct tasks are fetched according to the user's preference.
        - Ensure all task details are accurate.
        - Ensure operation is reflected within 3 seconds.
    - Test updated tasks are changed in real time seamlessly:
        - Ensure users can modify tasks without any errors (or properly raised errors)
        - Ensure operation is reflected within 3 seconds.
    - Test the delete function and tasks are updated:
        - Test task will automatically be removed from view once deleted.
        - Ensure deleted tasks are deleted from the database and not just modified.
        - Ensure operation is reflected within 3 seconds.
- Unit testing variations of inputs (valid and invalid)
    - Test valid input due dates (ex. 2024-03-10)
    - Test invalid dates (ex. 2024-02-30)
    - Test valid/invalid Task titles for modifying or deleting
    - Test valid/invalid notification settings for creating or modifying
- Stress testing the feature's capabilities

- Determine the maximum number of tasks created the system can handle. Observe when the system's performance declines after a certain number of tasks.
- Determine how many tasks the list can display. Observe when the system's performance declines after a certain number of tasks.