Reminders

Developer: Haley Vy

Submitted: April 14, 2023

Reviewer: Githel Suico

Reviewed: April 15, 2023

Criteria

Major Positives & Negatives

Positives

- Sequence diagrams for all LLDs include logging for both errors and successful operations.
- Success LLDs include database queries, axios calls to the web API, detailed return values, and method signatures.
- Organization of designs are easily understandable with proper notation such as object lifelines and a request/response structure.

Negatives

- Success LLDs do not include HTTP response codes indicating a successful operation.
- Designs do not entirely follow MVC architecture as it is missing a Controller layer between the user interface and backend Manager layer classes. The Controller layer is necessary for accessing the service's API endpoints.
- Designs not following MVC architecture as it does not include a model to define the datastructure for a reminder.
- Missing proper message feedback for successful operations, which are necessary for the user to receive a visual result of the action just committed.
- Logging messages are not consistent with the feature being used, i.e. logging the message "Bill notification email success" for the chore list feature.

Unmet Requirements

- *Adding or changing reminders will be logged to a database in the form of activity and/or errors (if any). (githelsui/WeCasa: @ Team HAGS JP.)*
    - Missing LLD for editing an existing reminder.

Design Recommendations

- Include HTTP response codes indicating a successful operation.
    - **Positives**: Returning successful HTTP response codes, such as 200 OK or 201 Created provides confirmation for the user and easy troubleshooting for the developer. Furthermore, this can help support API documentation for the application.
    - **Negatives:** None**.**
- Following an MVC architecture entirely by including a Reminder model for the backend, a Controller layer, and data transfer objects (DTOs) for the Web API.
    - **Positives**: By specifying a Controller layer with its respective objects, the separation of concerns is being implemented allowing for modularization and adheres to the application's microservice architecture.
    - **Negatives:** None**.**
- Include proper message feedback for successful operations, not just for failed operations and errors.
    - **Positives**: It is necessary for the user to receive a visual result of the action just committed for better user experience.
    - **Negatives:** None**.**
- Ensure that Logging messages are consistent with the feature being used, i.e. logging the message "Chore notification email success" for the chore list feature.
    - **Positives**: Ensures consistency and less ambiguity for the development of the feature.
    - **Negatives:** None**.**
- Creating a separate LLD for updating an existing reminder.
    - **Positives**: Having a separate user story for editing an existing reminder reduces ambiguity for the developers. With this extra sequence diagram, developers will be able to create an interface specifically for updating a reminder that has already been created.
    - **Negatives:** None**.**

Test Recommendations

Integration Tests

- To test the successful creation and sending of a reminder, the developer should programmatically create reminders for a test user using manually set configurations for the time interval of the notification.
    - Arrange the test reminder to create with a specified time interval of when to send.
    - After creating the reminder, use the NotificationService to send out the reminder via email.
    - Manually set the local machine's time to the set time interval of the reminder.
    - Verify if the email is present in the test user's inbox.
    - In regards to the BRD, this test ensures that the reminder was successfully created, sent, and delivered at the correct time.
- To test the update of an existing reminder, the developer should programmatically create and update the reminder for a test user using manually set configurations for the time interval of the notification.
    - After creating and updating the reminder, use the NotificationService to send out the reminder via email.
    - Manually set the local machine's time to the set time interval of the reminder.
    - Verify if the email is present with the updated content in the test user's inbox.
    - In regards to the BRD, this test ensures that the reminder was successfully updated, sent, and delivered at the correct time.
- To test the failed creation and sending of a reminder, the developer should programmatically create an invalid, nonexistent test user.
    - After creating the test reminder, use the NotificationService to send out the reminder to the invalid test user.
    - A successful test means that the SendGrid API returned an error from the Service layer.
    - In regards to the BRD, this test ensures that the failure of the SendGrid API will be handled by the Service/Manager layer and therefore logged as an error.

Unit Tests

- To test input validation, I would attempt to create a reminder with invalid time intervals and invalid recurring event options with the NotificationService object.
    - I would also attempt to create a reminder with a user that does not exist. This error should be thrown by the Manager/Service layer as it follows the business logic of the feature.
    - In regards to the BRD, this test ensures that the failure of a reminder being sent will be handled by the Service/Manager layer and therefore logged as an error.

References

githelsui/WeCasa: @ Team HAGS JP. (2023, Feb 17) *GitHub* https://github.com/githelsui/WeCasa/blob/main/docs/BRD%20v2.1.pdf