Team Project

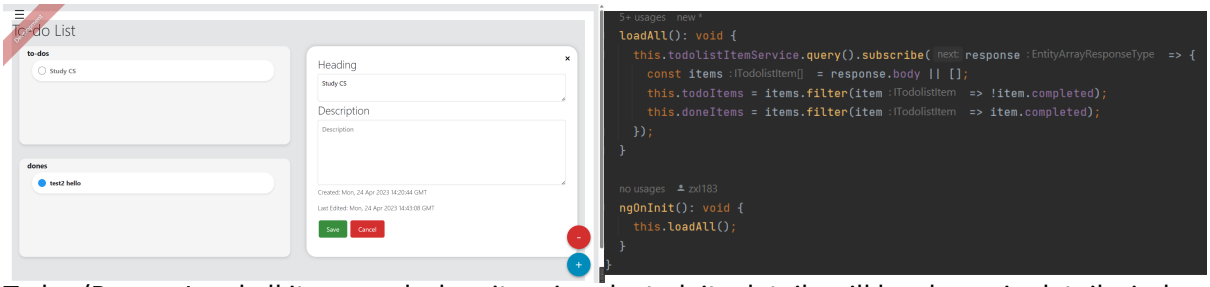
for

Submission 3

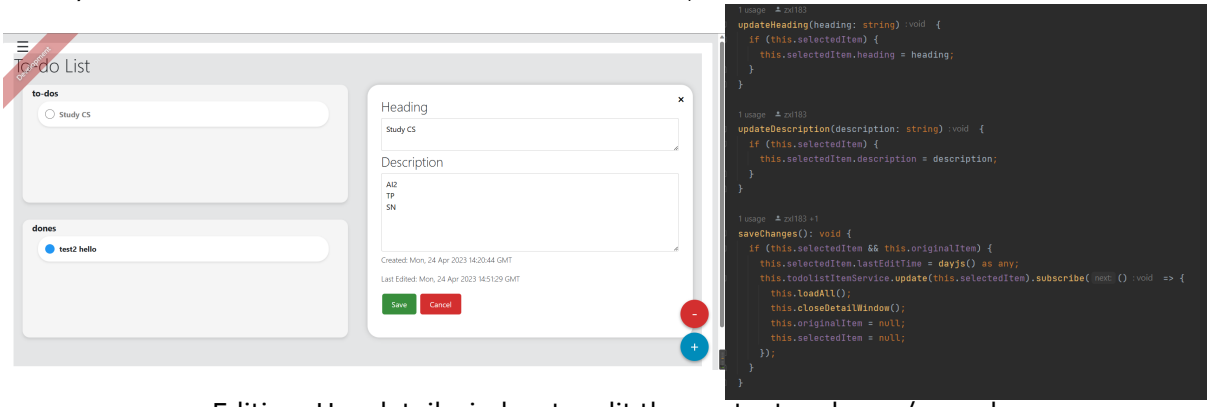
By Team 23-22

Zijun Li_2272583_zxl183

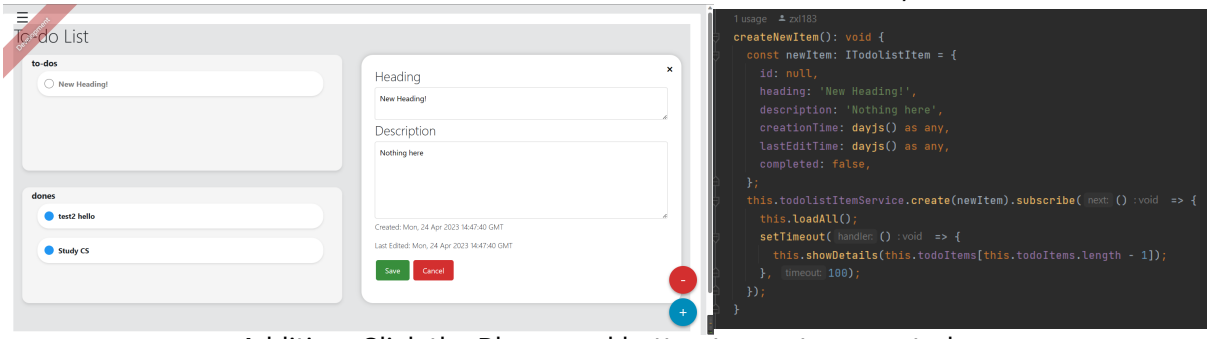
1 Screenshots of the vertically sliced features: To-do List



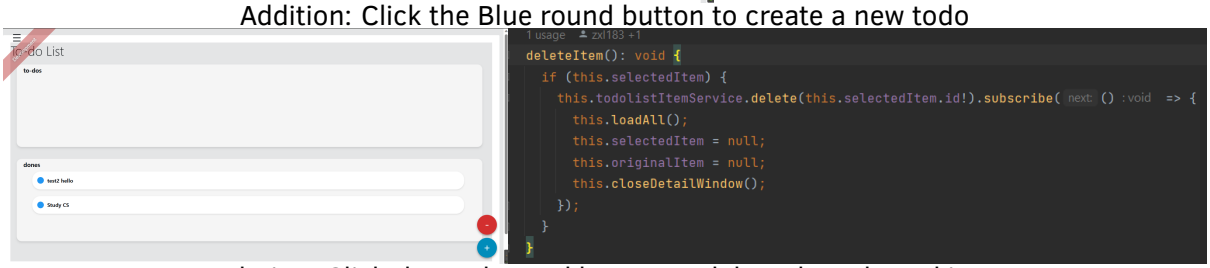
Todos/Dones: Load all items and when item is selected, its details will be shown in detail window



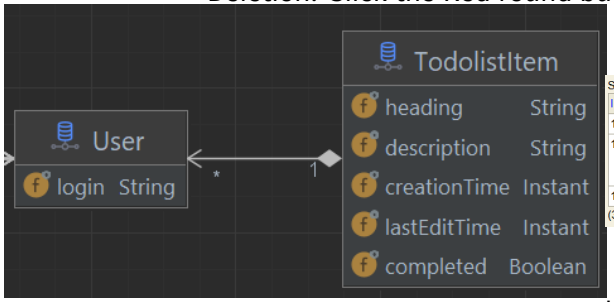
Edition: Use detail window to edit the content and save/cancel



Addition: Click the Blue round button to create a new todo



Deletion: Click the Red round button to delete the selected item



SELECT * FROM TODOLIST_ITEM;

ID	HEADING	DESCRIPTION	CREATION_TIME	LAST_EDIT_TIME	COMPLETED	USER_ID
1501	test2 hello	Nothing here	2023-04-24 13:26:32.563	2023-04-24 14:19:45.093	TRUE	1
1551	Study CS	AI2 TP SN	2023-04-24 14:20:44.207	2023-04-24 15:18:15.44	FALSE	1
1651	New Heading!	Nothing here	2023-04-24 15:18:18.648	2023-04-24 15:18:18.648	TRUE	1

(3 rows, 0 ms)

Database

2 Description of the development and integration of To-do List feature

I developed a to-do list feature within the Time Management application. This feature can display to-do and done items from the database, toggle their completion status, add new items, delete old items, and modify their content.

We used JHipster for development, which has already integrated many features, making it very easy to create entities that connect to the database.

First, I used a JDL file to create a to-do list item entity connected to the user (Commit). This established a relationship between the user entity and the to-do list item entity in the backend, defining the necessary fields and database schema for the to-do list items.

Then, I modified some Java files based on this, allowing users to display only their own items instead of all items (Commit). I updated the backend service layer and repository to fetch to-do list items filtered by the user, ensuring that each user only has access to their own items.

Next, by adding and modifying TypeScript files, I implemented the switch between to-do and done items, as well as the modification, saving, and canceling of item content (Commit). I updated the frontend components and services to handle these actions and communicate with the backend API. This allowed the frontend to display, update, and switch the status of items, while the backend took care of persisting the changes in the database.

This comprises the content of MVP_TodoList.

After that, I added the functionality for adding (Commit) and deleting (Commit) items and further improved the aesthetics of the user interface. I extended the frontend components and services to include add and delete actions, while updating the backend API to handle these new requests. As a result, the frontend can now send requests to add or delete items, and the backend takes care of creating or removing records in the database accordingly.