CSC301 Documentation

Shun Chit Wong, Rahul Uppalapati, David Chen, Yuvraaj Chahil, Peter Ghobrial, Daksh Malhotra, Manjoy Malhotra

Frontend

/Timer.js:

- It is the frontend screen that provides the user to enter time for pomodoro and ability to time the user. This screen imports "@chakra-ui/react" for organizing the timer page.
- It has 5 buttons, "work" and "break" at the top for changing different times, "setting" for setting different times, "start" and "pause" for starting and pausing the timer.
- This also used useEffect to update the timer execution and useContext for setting different timer cases.
- This screen also used "CountdownAnimation.js", which imported a countdown circle for the timer display.
- SettingContext is another function that requires a timer which helps setting up all the functions and updating the timer according to the user interaction.
- There is also a specific button class for the timer, that takes in the active button the user hits.
- It also has sound for each button click, timer ticking, and timer time ends.

/Tasks.js:

- Displays the list of tasks the user has set. The user may add and remove tasks, as well as edit them.
- This page implements components AddTask.js, TaskCheck.js, and TaskList.js.js.
- AddTask.js defines how to add a task with its specific name and due date, and a built-in unique ID.
- TaskCheck.js is an icon component which displays either a checkmark or an X depending on if the task is completed or not. It can be toggled by clicking on it, and the state of the task is then saved.
- TaskList.js maps through the list of tasks and displays them, and includes the TaskCheck.js component and edit/delete buttons for each task, allowing you to remove finished tasks, or edit task names by popping up a modal menu to input a new taskname. When deleteTask or editTask is called, it iterates through the list of tasks, and deletes/updates the task with the given task's ID.

/Journal.js

- Displays journal entries. The user may add, delete, and edit entries here.
- It has two main components, a folder structure tree and a Slate editor.
- Uses useMemo and useState to update the editor contents in real time.
- Each formatting button uses Slate's node methods to apply text formatting to user selected text
- Entries are saved to the neo4j database in subfolders of a root folder designated to each user.

/LoginPage.js

- Login page with a login form
- Is able to authenticate the user's loginning by checking the database.

- Uses chakra UI components to create the login form.

/SignupPage.js
- Sign up page with a sign up form
- Is able to authenticate the user's information and store the information in the database.
- Uses chakra UI components to create the signup form

/LandingPage.js

/navbar.js
- A navbar for the landing page which has different buttons.
- Uses smooth scrolling to scroll to different parts of the page

/about.js
- A component where it describes the mission of the app

/features.js
- A component where it describes the features of the app

/FeedPage.js
- The home page for after the user has logged in to view at a glance information
- Has reorganizable components to allow the user to customize the home page
- Has a navbar component so the user can access other pages

/feedbar.js
- A navbar for switching between different features
- Is mobile responsive

/feed-calender.js
- A component for feed page
- Includes database fetching to show upcoming deadlines / events

/feed-notes.js
- A component for feed page
- Fetches data from the database to show a preview of a recently created note(s)

/feed-pomo.js
- A component for feed page
- Fetches data and should show a preview of the timer, potentially play sound, links to the pomodoro page

/feed-tasks.js
- A component for feed page
- Fetches data, and shows tasks that are not completed but due in the next seven days

/Calendar.js
- A calendar page that fetches the tasks and notes from a particular day by fetching information from the database
- Has a variety of views in including month, year and daily view

Backend

Please refer to Apollo
Run the backend, and view localhost:4000
The backend documentation is interactive and self documenting