Health and wellness management system

Functionalities we would like to implement -

Basic Functionality

- Basic Authentication
- Allow user to add medications
- Allow recurring medication reminders
- Reports generation
- Sending emails

Session Management

- When logging out of our platform on one device, if the user selects the option 'Log me out from all devices,' it should automatically log the user out from all other devices where they are currently logged in.

- For user's safety, we would like to have a special feature called 'Log me out from other devices.' When a user chooses this option, the user can keep using the platform on your current device while making sure you're logged out from all other devices. It's an easy way to make sure your account stays safe, giving you control over where you're signed in.

Cron Jobs

- Since - we are building a health wellbeing system, we would like users to add their medications as well.

- Following things should be kept in mind while creating this module.
  There are two types of "Add Medications" module.
  1. One Time 2. Recurring

  - **One Time**
    - Should be available for once only - i.e. if the user is adding a new medication for 29th May - 2:00 AM then it should be active for once only.

  - **Recurring**
    - While creating a recurring entry there won't be an option for selecting a date - there should be a drop down with two options only for now (Daily and Weekly) and there should be 2 date boxes - start date and end date. You can select two dates. Let's

take an example. **Start Date**: 30th May, 2024 **End Date**: 01st August, 2024 **Time**: 2:00 PM

- Starting from 30th May till 1st August I must receive notifications for that particular added medication. If the user selected "Daily" then the user must receive notifications every day at 2:00 PM.

- If the user selects "Weekly" then one more dropdown should be added which
contains all the days of a week. Let's say the user selected Thursday then the user must receive notifications (email) on every Thursday at 2:00 PM.

- Following - user must add these while adding a new medication:
  - Medicine Name
  - Description
  - Mark as done [User should be able to mark medicine as done - at the time when they receive a notification]
  - [and other necessary things we mentioned earlier - render everything according to the selected options]

Queues

- At the end of every week I would like to generate reports and send them to users. Now imagine generating a bunch of large resources at once would be a critical task. It may slow down our server or sometimes even crash it.

- This process contains reading the user's data - generating a csv and sending the user that sheet with appropriate body content.

- The CSV should contain - all the medicines I have taken this week and all the metadata [mark as done, time, date] related to it.

- Queues are important because they help manage background tasks efficiently, prevent blocking the main thread, and improve application performance. They allow tasks like email sending, report generation, and data processing to be handled asynchronously without slowing down the server. Using queues ensures better scalability, fault tolerance, and resource optimization.

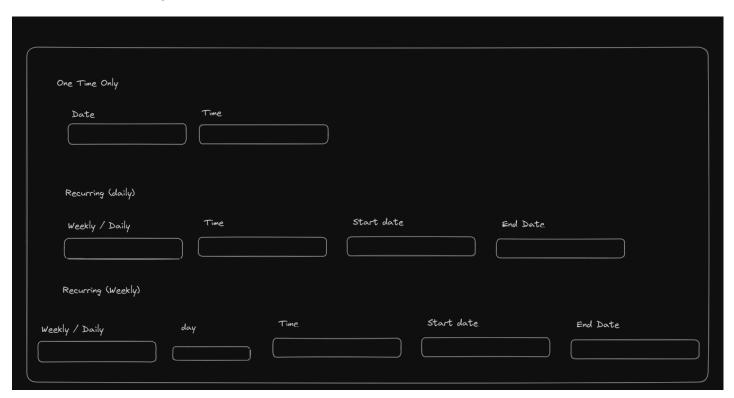- You can refer to BullMQ for managing queues.

Emails

- As discussed earlier - we would like to send users email about reports and medications reminders. At every place where I have mentioned notifications - we are discussing email notifications only.
- You may use the node-mailer package for this.

- For SMTP credentials you may generate that from your google accounts using App Password.

File Storage

- If you wish to use multer for file storing which stores directly to your server - it's alright.

- You can also consider using "cloudinary" which provides free storage up to some extent. This way - in future we can learn how we can work with other S3 storages.

Basic blueprint for add medication module:

Optional

- It would be great if you could dockerize the application for platform-independent execution. Create a Dockerfile to build an image and then push it to a container registry for easy deployment and scalability.