Documentation and Developer's manual

Health Centre (Medicine)

Submitted by

ARNA

Student Details

Ashwani Kumar (17JE003253)

Rahul Kumar (17JE003284)

Ayush Maheshwari(17JE003264)

Naman Patel (17JE003221)

Table of Contents

1. Introduction

The idea is to create an app for the Health Centre and provide major functionalities for helping the patients and people who keep counter.

1.1 Purpose

- Provide the Main Counter and Stock Counter Functionality.
- To provide an efficient implementation to Appointment Module And Patient Module

1.2 Scope

- We aim to provide an efficient working to Counter Module, Appointment module, and patient Module and make the UI as user-friendly as possible.
- With the help of this product, we can help the patients, DAO, and doctor by making their work easier and faster.

2. The Overall Description

2.1 Product Features

- **Stock Counter Module –**to transfer medicines to the main counter module and download a list of all medicines.
- **Main Counter Module-** to receive and return medicines. It also has functionalities to edit batch, give a patient medicine, and download the medicine table as an excel file.
- **Patient Module:** Edit patients details, doctor assigned, view medical history.

2.2 User Characteristics

There is a DAO who has access to this software and who can create or edit tables of the counter.

2.3 Assumptions and Dependencies

We need data for appointments beforehand.

3. Project Structure

The project has two folders backend and frontend.
Inside the backend folder, the most important one is the app folder which contains

1. Config (contains database connection related files)
2. Controller (contains the implementation of routes)

3. Models(contains the structure of table)
4. Routes (contains routes)
5. Server.js (main file)

The frontend folder contains src which has:
1. Core (contains all implementation of all components like the table, and button and forms)
2. Pages(Main pages related info)

3.1) Front End:

1. React: Front End Library (Running on Port 3000)

2. Third party used:

    react-router-dom: For Routing

3. Structure of Files:

    Public: contains assets (CSS and JS) files and index.html file.

    Src: contains files logic related files.

    Src->core: application core files like the menu, header, footer, layout, etc.

    Src ->pages: here we have put all module related file.

4. Route File:

    src->App.js: Here we have defined routes for each component of the application.

5. For Https request:
    We have used the Axios Library.
6. To create the UI, Gati(which was provided) and Material UI is used.

3.2) Back End:

Back end API (Running on port 8000)

1. Main File: server.js

2. We are using the Controller and Model concept in the backend. For every module, we create a Controller and Model file with the module name in the controller and model folder.

3. To achieve model (ORM) functionality we used the Sequelize library.
All database queries were done using Sequelize.

4. Folder structure:
    4.1. api
        4.1.1 Controllers
        4.1.2 Models
        4.1.3 Routes
        4.1.4 server.js

5. To Create API without having UI, Postman (https://www.postman.com/) was used.

4. Functionality and UI implementation

There are 3 tabs at the top implemented with routes(react-router) - StockModule tab, Appointment Module Tab, and Counter Module Tab. The click functionality on each tab is to switch between tabs. A history container is used to store the current route(URL) that the user is in.
Other functionalities and their way of implementation are listed below.

4.1 Adding Medicines to Stock

Major Functionality Provided: To add medicine to stock by clicking on the “Add medicine” Button.

Way of Implementation: On clicking on the Add Medicine Button a Form opens, and we enter the respective data in the Textfields of the Form component. The values of variables typed as input in Textfield are stored as react state variables and a function component is used throughout the code to the advantage of the speed it provides in comparison to the Class component. When we click the done button, an Axios PUT Request to the backend is triggered which updates the SQL tables in the backend. An API in the backend is built for this. The page is then refreshed, and the table is refreshed using data from the GET API request, to view the updated contents.
Similarly, other edit functionalities are implemented.

4.2 Transferring medicine to the main stock

Major Functionality Provided: To transfer the medicine to the main stock, click on the "SEND" button at the end of the row, besides the medicine you want to transfer. Then enter the quantity of the medicine you want to send and click "Transfer".
Way of Implementation: On clicking on the Send Button a Form opens(by setting its open value to true), and we enter the respective data in the Textfields of the form. The values of variables are stored as React State variables. When we click the done button an Axios PUT request to the backend is triggered which updates the SQL tables conditionally based on the quantity entered(whether that much quantity is available or not). If not available an alert pops up, and if available the put request is executed.

4.3 Search Functionality:

Major Functionality Provided: Search medicine by name.
Way of Implementation: This functionality is implemented using the onChange attribute of Textfield. Whenever there is a change in the Textfield(by typing what we need to search) the state variable corresponding to it is updated and then an API GET Request (with params as -the input typed in the Search Textfield) is made to fetch updated data. The backend has an API that returns the data matching the input params

provided. Using the response data of this API, the data is then displayed in the form of a table.

4.4 Table Implementation:

The material UI table is implemented wherever a table is used. The column names of the tables are added at the frontend and the rows are populated using a get API request to the backend using Axios for the respective tables. The JSON data then received is iterated over using the keys (as column names of the table) and each cell value is added one by one.

The color of columns is added using the style attribute.

4.5 Buttons:

The two types of buttons used are Gati buttons and material UI buttons. The only events that button handles are - onClick and Hover. On the occurrence of either event, a function associated with that event is called to achieve the desired functionality.

4.6 Download Functionality:

Major Functionality Provided: Click on the download button to get the table as an excel file.

Way of Implementation: The functionality is implemented using "react-export-excel" which is an export to excel library built with and for React. The column names and then each cell individually are assigned values to form the "ExcelSheet" which can be downloaded by clicking on the download button.