**WeCare Hospital Management System**

****

Hospital Management System Proposal

Team #19: Nicholas Hsiao, Huynh Phan, Diana Sok

September 5, 2019

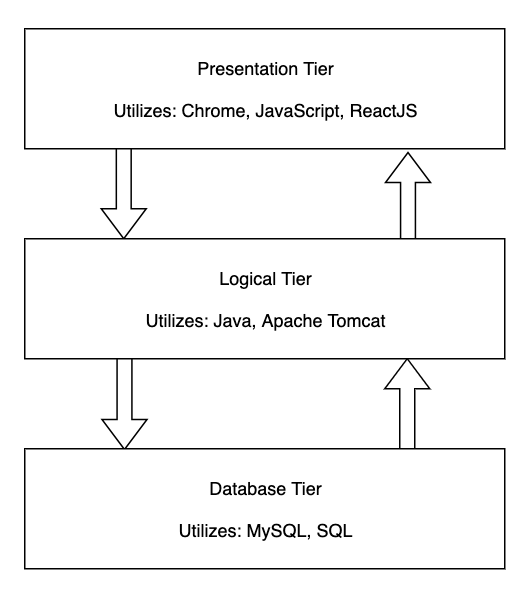
Instructor: Dr. Mike-Wu

## **1. Project Overview**

For this database application project, we will be developing a hospital management system with both user and administrative functions. Users (patients) will be able to register as a new user and fill in a medical profile or continue as a registered user and have access to their medical profile and appointments. Users will also be able to schedule and update appointments. Administrators (doctors and nurses) will have read and write access patient medical records and be able to confirm, view and update all upcoming appointments. This project will be a great opportunity to build a real-world full stack application.

## **2. System Environment**

Our application will follow the three-tier architecture structure presented below. For client browser we will be using Google Chrome. The front-end and user interface will be built using JavaScript and ReactJS as the framework. Our project will use Apache Tomcat and Java for hosting the web server. Finally, we will be using SQL and the MySQL RDBMS for our database tier.



We will be hosting the Apache Tomcat server on a MacBook laptop. The specifications are listed below:

|  |  |
| --- | --- |
| Component | Specification |
| Central Processing Unit | 2 GHz Intel Core i5 |
| Number of Processors | 1 |
| Memory | 8 GB 1867 MHz LPDDR3 |
| Operating System | macOS Mojave (10.14.5) |

## 

## **3. Functional Requirement**

**Patient Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement | ID | Detail Leveled Requirements |
| Create Account | 1P | Allow user to register and create an account |
|  | 2P | Allow user to enter in personal information and fill out medical history |
|  | 3P | Assign new users a unique patient id |
| Login/ Logout | 4P | Allow users to log in after providing valid credentials |
|  | 5P | Allow users to log out of their account |
|  | 6P | Allow users to recover or reset their password via email |
| Personal Info and Medical History/Profile Access | 7P | Allow users to edit or update their personal information and medical history |
|  | 8P | Allow users to review their personal information and medical history |
|  | 9P | Allow users to have read access to their own medical profiles written by doctors |
| Schedule appointments | 10P | Allow user to schedule an appointment based on doctor. |
|  | 11p | Allow user to schedule appointment based on time. |
|  | 12P | Prevent scheduling an appointment conflicting with patient’s existing appointments. |
|  | 13P | Prevent scheduling an appointment conflicting with the doctor’s schedule. |
|  | 14P | Allow patient to schedule multiple appointments. |
| View/Update appointments | 15P | Allow user to view upcoming appointments. |
|  | 16P | Allow user to view past appointments. |
|  | 17P | Allow user to cancel an appointment. |
|  | 18P | Allow user to change appointment date. |
|  | 19P | Allow user to change appointment time. |
|  | 20P | Updating appointment date cannot conflict with scheduled doctor’s existing schedule. |
|  | 21P | Updating appointment date cannot conflict with patient’s existing schedule. |
|  | 22P | Updating appointment time cannot conflict with scheduled doctor’s existing schedule. |
|  | 23P | Updating appointment time cannot conflict with patient’s existing schedule. |
|  | 24P | [perhaps] ??!???!?? Penalize appointment changes done [this time frame] before the appointment. ???!?? [perhaps] |

**We don’t have a billing function so idk about charging^**

**Administrative (Doctors/Nurses) Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement | ID | Detail Leveled Requirements |
| Create Account | 1A | Allow administrator to register and create an account upon providing their Employee ID |
|  | 2A | The Employee ID must be 9 digits long |
|  | 3A | Allow administrators to provide their first and last name, pick a department, and create a password |
|  | 4A | Upon creation, a generated work email will be provided from the information and added to the profile ([NAME@kaiser.org](mailto:NAME@kaiser.org) example) |
| Login/Logout | 5A | Enable administrators to log in upon providing their Employee ID and password |
|  | 6A | Allow administrators to log out of their account |
|  | 7A | Enable administrators to submit a support ticket to change their password (WE CAN JUST SIMULATE THIS AND CHANGE THE PASSWORD AFTER) |
| Access medical profiles | 8A | Administrators will be allowed to view all patient profiles |
|  | 9A | Administrators will be allowed to add to the medical history of a patient profile |
|  | 10A | Administrators will be allowed to view updates to a patient profile given by the patient |
|  | 11A | Administrators will be allowed to view all employee profiles |
|  | 12A | Administrators will be allowed to provide their own hours of restricted availability I think it belongs here 😊 |
|  | 13A | Administrators will be allowed to add a prescription to a patient profile (NOTHING MORE COMPLICATED THAN A NOTE) |
|  | 14A | Administrators will be allowed to view the appointments of other employee profiles (maybe this can be added below) |
| View/Update appointments | 15A | Allow user to view upcoming appointments. |
|  | 16A | Allow user to view past appointments. |
|  | 17A | Allow user to cancel an appointment. |
|  | 18A | Allow user to specify chunks of time that cannot be scheduled. |
|  | 19A | Update affected patient of appointment cancellation. |

## 

## **4. Non-Functional Requirements**

1. Execution qualities (Qualities which are observable during operation)
   1. Security
      1. There will be no broken authentication or broken access control points through which admin privileges are given to non-admin users
      2. Patients will not be able to access restricted data
   2. Privacy
      1. Patients cannot view other patients’ data
      2. Patients cannot view any of the doctor’s private data, such as their patient list
   3. Performance
      1. The database program shall execute in timely fashion, returning queries in a reasonable amount of time
   4. Constancy
      1. The program will not need to be executed separately or restarted in a single user’s session as well as a series of concurrent users of variable permission levels. The program will run in a perpetual state throughout its use.
2. Evolution qualities
   1. Documentation
      1. The system will feature an organization of code with descriptions such that each component can be easily understood as a constituent of the system
   2. Testability
      1. The system source code will follow the 3-tier architecture. Using that to the advantage of self-testing, the divided infrastructure will enable the project team to easily identify the point of vulnerability or error as one of the three classifications once error/exception handling is implemented