**WeCare Hospital Management System**

****

Hospital Management System Proposal

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

September 5, 2019

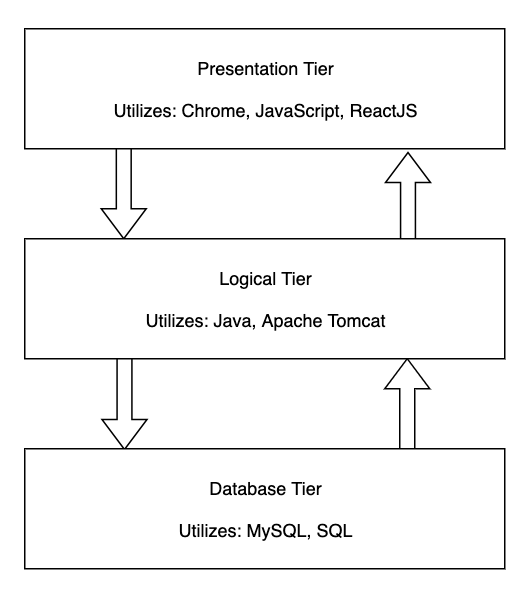
Instructor: Dr. Mike-Wu

## **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 desktop PC. The specifications are listed below:

|  |  |
| --- | --- |
| Component | Specification |
| Central Processing Unit | AMD Ryzen 5 1400 Quad-Core Processor (8 CPUs), ~3.2 GHz |
| Memory | 12288MB (12GB) DDR4 2400 MHz |
| Operating System | Windows 10 Home 64-bit |
| Systems Manufacturer | Micro-Star International Co., Ltd |
| Download Speed | 298.70 Mb/s |
| Upload Speed | 11.87 Mb/s |
| Ping | 10 ms |

## 

## **3. Functional Requirement**

**Patient Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement | ID | Detail Leveled Requirements |
| Create Account | 1P |  |
|  | 2P |  |
|  |  |  |
| Login/ Logout |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Access personal information and medical profile |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Schedule appointments |  |  |
|  |  |  |
|  |  |  |
| View/Update appointments |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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

|  |  |  |
| --- | --- | --- |
| Requirement | ID | Detail Leveled Requirements |
| Create Account | 1A |  |
|  | 2A |  |
|  |  |  |
| Login/ Logout |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Access patient medical profiles |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| View/Update appointments |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 

## **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