CS 460W – Software Development

Project Deliverable #1

Team Members: Alisa Vongsarasinh, Benjamin Bahadory, Daniel Baqaeen

3.1 **Functional Requirements**

3.1.1 **CreateRecord**

3.1.1.1 **Introduction** This function will mimic the responsibilities and the tasks done by an ER registrar. This function will prompt the user acting as the registrar through a list of required inputs. The inputs will be listed below. Only the registrar is allowed to access the CreateRecord function.

3.1.1.2 **Inputs** The inputs for this function will only be taken from the registrar user and they will be as follows: Name, Address, Number, Email, Data of Birth, Symptoms, Primary Physician, Health Insurance, Covid Vaccination Name, Covid Vaccination Date, Emergency Contact Name, Emergency Contact Number, Allergies, Previous Medical Conditions, Previous Medications, Height, Weight, Religion, Race, Pregnancy, Gender, SSN, Sexually Active, Blood Type

3.1.1.3 **Processing** First a check will be done to make sure that the required values are non-empty. For each input the type will be checked as will, for example: for the number field, there will be a check to make sure only numbers are input in that field and not any other characters. This function will also create a unique patient ID.

3.1.1.4 **Outputs** Once the input values are verified, the user will be able to submit the inputs where they will be saved into a database. If there is an error in any of the inputs, an error message will appear indicating where the error is such as empty field or not correct input type.

3.1.2 **ViewRecord**

3.1.2.1 **Introduction** This function will be used just to view the data and not edit anything. The registrar will have limited access to only view general information, however, the nurse and the doctor will be able to access the medical information of the patient.

3.1.2.2 **Inputs** There are no inputs for this function since this is a view only function.

3.1.2.3 **Processing** This function will retrieve the correct record from the database by doing a query search for the patient’s information like date of birth and name or by patient ID.

3.1.2.4 **Outputs** If there is no record with the provided information, then nothing is displayed. If there is a match then the information is displayed in an understandable and structured format.

3.1.3 **EditRecord**

3.1.3.1 **Introduction** This function will be used to edit the patients record in order to either update previous information or add data collected by the nurse or the doctor later on. This is accessible by the registrar, nurse, and doctor. Each will be allowed a different access and what information they can edit. For example, registrar can't view or edit medical information.

3.1.3.2 **Inputs** The inputs for this function will only be taken depending who the user is. If the user is the registrar then the inputs would be the same as for the CreateRecord function. If the user is the nurse, the inputs are as follows:

Vitals (blood pressure), Height, Weight, Nurse Notes, Admitted, Number of Nights, Assigned Physician, IVs

If the user is the doctor, the inputs are as follows:

Doctor Notes, Diagnosis , Requested Tests (Xray, bloodtest, CAT scan, urine test, MRI)

3.1.3.3 **Processing** The previous information for the patient is retrieved using a query, then the data is displayed and can be edited. Once the data is filled a check will be done to make sure that the required values are non-empty. For each input the type will be checked as will, for example: for the height input, there will be a check to make sure only numbers are input in that field and not any other characters.

3.1.3.4 **Outputs** Once the input values are verified, the user will be able to submit the inputs where they will be saved into the database. If there is an error in any of the inputs, an error message will appear indicating where the error is such as empty field or not correct input type.

3.1.4 **WriteDischargeInfo**

3.1.4.1 **Introduction** This function will be only accessible by the doctor and will contain information relating to the diagnosis, tests provided, medications, and other possible treatments if not medication.

3.1.4.2 **Inputs** The inputs for this function will be as follows:

Diagnosis, test provided, medications and other possible treatments

3.1.4.3 **Processing** Once the data is filled a check will be done to make sure that the required values are non-empty, for example Diagnosis.

3.1.4.4 **Outputs**

Once the input values are verified, the user will be able to submit the inputs where they will be saved into the database. If there is an error in any of the inputs, an error message will appear indicating where the error is such as empty field. The discharge info will also be viewed in a detailed and structured format.

3.1.5 **CreateBill**

3.1.5.1 **Introduction** This function will be only accessible by the billing department. It will be responsible for going over the services provided and calculating what the patient owes.

3.1.5.2 **Inputs** The inputs will be the services provided like tests, medication, IVs and number of nights patient stayed if the patient stays overnight (date of check in and checkout) Also, the name, address, and patient details will be needed.

3.1.5.3 **Processing** There will be no check for empty values here since the patient may receive no services and will only be billed with the default visit charge. Also, a calculation will be done to find the total amount the patient owes. The prices for each service will be already stored and then depending on what services were provided, the patient will be charged accordingly, and a default visit charge will be always added. This data and the total amount due will be stored in the database.

3.1.5.4 **Outputs** Once the input values are verified, the user will be able to submit the inputs where they will be saved into the database. If there is an error in any of the inputs, an error message will appear indicating where the error is such as empty field. The bill will also be viewed in a detailed and structured format with the deadline and instructions on how to pay clearly stated.

3.2 **External Interface Requirements**

3.2.1 **User Interfaces**

- Login Screens: Every user will be presented with a hospital login screen, with a box for usernames and passwords, and a login button. Possibly a different portal for patients and employees, selected from a home page.
- Personal interfaces: Depending on credentials entered, users of different types (Patient, doctor, nurse) will be directed to an appropriate page containing their necessary functions.

- Patient - Intake form: Input all necessary data for the hospital, including symptoms/chief complaint
  - Text boxes for input
  - Account page, shows bills
- Nurse - Patient database: View all patient data and input new data such as vitals, notes, nights stayed, IV status. Doctors write prescriptions and give diagnoses, nurses help administer treatment and log it.
  - Lists for all information, text boxes for specific inputs. Buttons for confirmation and deletion.
  - Tabs with patient names: Click patient, opens customizable data table
- Doctor - Patient database: View all patient data including info added by nurse. Add diagnosis and medication prescription.
  - Lists for all information, text boxes for specific inputs. Buttons for confirmation and deletion.
  - Nearly the same as nurse but with different input permissions
- Billing - Input prices for each service, posts to patient account page

3.2.2 **Hardware Interfaces**

- Desktop and laptop computers (with their necessary peripherals) connected wirelessly or via ethernet to a common hospital network. Servers needed to host network and web pages.
- Screen-oriented terminal control

3.2.3 **Software Interfaces**

- Windows 10 or later, with Java Virtual Machine installed

3.2.4 **Communication Interfaces**

- Website hosted on servers, or an app which is only usable on computers on the hospital network, also hosted by servers.

3.3 **Performance Requirements**

- Static Requirements - One terminal per nurse and doctor, plus a few public terminals for patients who do not submit intake forms from their own machine.
- Dynamic Requirements - Logging in and submitting intakes and billing statements can take a few seconds or more, can take up to 10-15 seconds without causing any major disruptions. Editing patient data must be done quickly however, as it will need to be done by multiple accounts and at high frequency.

3.4 **Design constraints**

3.4.1 **Standards compliance** We will be following the HIPAA rules and regulations. We will be ensuring that the patient's data is protected and only viewed by the allowed parties. For example a patient's medical record will only be accessible by the doctor, nurse, and any other parties the patient gives permission to. As for report format, we will ensure that the bill will be in a detailed and understandable format for the patient.

3.4.2 **Hardware limitations** Enough memory should be dedicated to store all patients records (Petabytes of memory) and have a server that can handle all the requests and functions

3.5 **Attributes**

3.5.1 **Availability** The system should reasonably be available to any user 95% of the time, meaning that there is a 5% fault tolerance, or 18.25 days of the year where the system will be down. The availability relies heavily on the reliability of the software, hardware, and security.

3.5.2 **Security** As personal information will be inputted into the database, encryption can be implemented as well as activity logging to monitor for any suspicious activity, especially in areas where there needs to be a high level of security (e.g., billing area as credit card information will be needed).

3.5.3 **Maintainability**

- Traceability. The code will be able to be traced through its life, from its inception to development to deployment as well as through any changes made after deployment.
- Coupling. Based on the role of the user, certain items will be activated on the GUI (e.g., nurses and physicians will be able to see medical history, but the billing department will not) which can be altered if necessary. Since many aspects of the program are linked, those changes will be displayed in the update version of the medical report GUI.
- Cohesion. The classes used will only focus on their specific function making the program highly cohesive, especially due to the fact that it will be built with Java, an object-oriented language.
- Portability. Program will be available on any device that has internet access.

3.5.4 **Reliability** The system will be tested multiple times by all members of our team to ensure that our program works correctly.