The Residents: Aden D'occhio, Alessandro DiCesare, and Fatima Chowdury
CS460W Software Development - Professor Ingrid Russell
Github: https://github.com/alesso425/CS460W-ER-Project
Software Requirement Specification (SRS) Document

C.A.R.E.S. System (Russell-Stover Memorial Hospital)

Because We Care

# 1. Introduction

## 1.1. Purpose

This document for the Connecticut Advanced Russell Emergency Services (C.A.R.E.S.) System delineates the functional capabilities, non-functional requirements, system architecture, performance criteria, and user interface design of the (C.A.R.E.S.) System, aiming to deliver a comprehensive understanding of both its functionalities and limitations. This document serves as a vital communication tool between the development team (the Residents) and the client, ensuring a mutual understanding of the deliverables for the final software package. The detailed specifications within are intended to guide the development process and align the project outcomes with the client's expectations. The primary purpose of the C.A.R.E.S. System is to provide a robust and reliable software solution for managing patient information and emergency room operational flow in a streamlined and efficient manner. It facilitates the prompt and accurate check-in, assessment, treatment, and billing of patients within emergency departments, bolstering the overall quality of emergency care. The C.A.R.E.S. System aims to empower healthcare professionals with real-time access to critical patient data, enhance the coordination of care, and expedite the delivery of medical services, thereby improving patient outcomes and operational efficiency in high-pressure emergency room environments.

## 1.2. Scope

This document articulates the Software Requirement Specification (SRS) for the Connecticut Advanced Russell Emergency Services (C.A.R.E.S.) System, designed to enhance patient management and care in emergency rooms. The (C.A.R.E.S.) System is engineered to efficiently manage patient information from their entry to discharge, prioritizing effective healthcare delivery without the need for long-term record storage. It is capable of storing multiple patients concurrently and enables hospital staff, including front desk personnel, nurses, doctors, and billing staff, to perform critical functions such as patient check-in, vital sign recording, ordering and recording of lab tests, diagnosis, prescription management, and generating comprehensive discharge reports including billing details.

Key features of the (C.A.R.E.S.) System include:

- Multi-patient handling with real-time data display.
- Role-specific access for various hospital staff, ensuring efficient task execution.
- Support for a limited range of lab and diagnostic technologies, focusing on four core types for streamlined processing.
- Provision for five distinct diagnoses, aligning with the most common emergency room scenarios.
- Single-user access at any given time to maintain data integrity and streamline workflow.

### 1.3. Definitions, Acronyms, and Abbreviations

- (C.A.R.E.S. or CARES System): Connecticut Advanced Russell Emergency Services
- SRS: Software Requirement Specification
- GUI: Graphical User Interface
- RBC: Red Blood Count
- WBC: White Blood Cell Count
- MRI: Magnetic Resonance Imaging
- CT: Computed Tomography
- IM: Intramuscular injection
- IV: Intravascular injection
- SC: Subcutaneous injection
- D: Doctor
- RN: Nurse
- ERS: Emergency Room Staff
- BS: Billing Staff
- PE: Person
- PA: Patient
- DI: Diagnosis
- LT: Lab Test

### 1.4. References

All external documents referenced by the development team will appear in this section:

- HIPAA: https://www.hhs.gov/hipaa/index.html

### 1.5. Overview

This SRS document will further elaborate on these capabilities, detailing functional and non-functional requirements, system architecture, performance criteria, hardware requirements, software requirements plus dependencies, security features, system use, and user interface design.

## 2. General Description

### 2.1. Product Perspective

The (C.A.R.E.S.) System is designed to function optimally on desktop or laptop computers, which are the primary hardware for accessing the program by emergency room staff. The database for the CARES System will be utilizing a hashtable, which is a simple key/value data structure in which Patient information can be stored and assigned a key. Whole patient's info can be inserted, edited, or removed at any time. There will be a Database class that creates a hashtable for each of the following groups of information: login information (loginTable) and patients (patientTable). Having one database for all of a patient's information leads to an easier functionality of the program, as there are less objects and methods to maintain. In terms of keys for searching purposes, they will be a concatenation of the patient's full name and the eight digits of their birthday (MM/DD/YYYY). This setup offers flexibility, allowing the database server to potentially operate on the same hardware used to access the system. This dual-purpose use of hardware ensures that the (C.A.R.E.S.) System is both accessible and economical for healthcare facilities, streamlining emergency room operations through a cohesive software and hardware solution.

### 2.2. Product Functions

The system provides a range of critical functionalities, which are described in the following list:

- Patient Check-In: Capturing essential information like name, address, insurance, and emergency contact.
    - Last Name
    - First Name
    - Permanent Address
    - Cell Phone
    - Birthday
    - Insurance Plan
    - Emergency Cell
- Vitals and Medical History Recording: Documenting vital signs and medical history.
    - Height
    - Weight
    - Blood Pressure

    - Heart Rate
    - SpO2 (Blood Oxygen Level)
    - Body Temperature
    - BMI (Calculated with Height and Weight)
- Lab Tests: Performing specific lab tests such as Hematologic (e.g., RBC, WBC counts), Radiologic (e.g., X-ray, CT, MRI), Urinary, and Stool Tests.
- Diagnosis and Prescription: Limited to five diagnoses (e.g., High Blood Pressure issues, High Cholesterol abnormalities, Kidney Disease, Liver Disease, Orthopedic condition,  Broken Arm. Prescriptions are linked to diagnoses.
- Billing and Discharge: Generating detailed billing reports including charges for services and hospital stay.

### 2.3. User Characteristics

The system is tailored for use by four different classes of individuals who work in the hospital setting:

- Emergency Room Staff: Handling initial patient data entry, such as name address, etc...
- Nurses: Full access to patient data, ordering labs, declaring a patient ready to be admitted, providing discharge instructions, and completing discharge of patients.
- Doctors: All nursing capabilities plus diagnosing, prescribing, admitting a patient, and initiating discharge.
- Billing Staff: managing billing information.

Minimal training will be required to use the system as GUI should be intuitive to use for any users who actively use the internet or handheld devices. The CARES System will feature one page in which all users will have displayed to them. Each different user will have specific permissions to access various panels in the same display based on their role as ERS, Nurse, Doctor, or Billing Staff.

### 2.4. General Constraints

The C.A.R.E.S. System operates within specific constraints, including its ability to support only one active user session at a time, which is a critical consideration for workflow management in high-demand emergency room settings. Additionally, the system's functionality is designed around a limited set of lab tests and diagnoses to streamline emergency care processes. Lastly the C.A.R.E.S. System itself is structured to operate independently with its primary dependency being a HashTable class creating the database source of information.

### 2.5. Assumptions and Dependencies

The C.A.R.E.S. System's operational efficacy is fundamentally tied to its integration with a Hashtable database hosted on a server. This dependency underscores the need for a stable and

accessible server environment to ensure uninterrupted system performance and data management capabilities. If the database system goes down the program will fail to operate as all data will be offline and inaccessible.

# 3. Specific Requirements

## 3.1. Functional Requirements

### 3.1.1. Functional Requirement 1 (Person)

#### 3.1.1.1. Introduction

The Person class functions as the basis for the database of people used in this project. The design of Person will be simple yet sufficient for the use of future classes in the CARES software. This class represents any person that is inputted into the CARES System, whether that be a staff member (emergency room staff, billing staff, nurse, and doctor) or a patient. The person class stores basic information and can be utilized by more than one class in the system.

#### 3.1.1.2. Inputs

The attributes included in this class are: a String for a first name, a String for a last name, a String for a date of birth, a String for a permanent living address, and an String value for a phone number. There will be a few methods implemented into the Person class, including a get and a set method for each inputed attribute. Utilizing test cases, a Person object's attributes are filled when a person is checked into the CARES System.

#### 3.1.1.3. Processing

The inputted values will be checked upon entry into the system for validity. In detail, the String for date of birth will always be checked to make sure the inputted value will be in MM/DD/YYYY format. Also, the String value for a phone number will be checked to make sure the inputted value is in ###-###-#### format.

#### 3.1.1.4. Outputs

The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute. The set methods for date of birth and phone number is where the exceptions check for the correct formatting.

### 3.1.2. Functional Requirement 2 (Patient)

#### 3.1.2.1. Introduction

The Patient class functions as the sole class for all patients in the hospital emergency room department. After a Person is entered into the CARES system, they can get accepted by an ERS, who makes them a Patient. The Patient class is of direct inheritance from the Person, adding on attributes for their calculated patient identification, an insurance plan, any emergency contact, an admitted date, measurements (height, weight, blood pressure, heart rate, blood oxygen level, body temperature), diagnoses, lab panels, and their bill. A patient has to wait to be

seen by a nurse before getting measured for height, weight, etc., and a diagnosis from the doctor. A patient also has to wait for their bill until after their visit, when all costs of their visit are calculated and displayed on one page.

#### 3.1.2.2. Inputs

The attributes in this class are as follows: a String for a patient ID, a String for insurance plan, a String for an emergency contact, a LocalDate object for time admitted, a boolean value for admittance, a boolean value for discharge, a double value for height, a double value for weight, a String value for blood pressure, a double value for heart rate, a double value for blood oxygen level, a double value body temperature, a double value for body mass index, a Diagnosis object for any diagnosis, a LabPanel object for any labs run, a Bill object for the bill, a LocalDate object for the discharge date, and a String value for the discharge instructions. At the time of creating a Patient object, which will occur when a person is checked in, most of the measurements of the person will not be inputted until a nurse or doctor will see them. Methods include a get and set method for all attributes. Additional methods include one that calculates the time a patient has stayed at the hospital and a couple methods that check for correct formatting.

#### 3.1.2.3. Processing

The inputted values will be checked upon entry into the system for validity. In detail, the String for an emergency contact will always be checked to make sure the inputted value is in ###-###-#### format. The String for height will always be checked to make sure the inputted value is within the normal range for height of a human. The String for weight will be checked to see that the weight of the person is within normal grounds. Blood pressure will also be checked so that the systolic pressure is within 0-200 and the diastolic pressure is within 0-150. Heart rate will be checked so that the inputted rate is a positive number; likewise with blood oxygen level and body temperature.

#### 3.1.2.4. Outputs

The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute. The set methods for emergency contact, height, weight, blood pressure, heart rate, blood oxygen level, and body temperature is where the exceptions check for the correct formatting.

### 3.1.3. Functional Requirement 3 (Staff)

#### 3.1.3.1. Introduction

The Staff class serves as the foundation class for all following staff members and is of direct inheritance to the Person class. A staff member will have a username and password to be able to log into the CARES System. From there, every member has specified roles through inherited classes later in the program. Each Staff object will store basic information of each member and their username and password.

#### 3.1.3.2. Inputs

The attributes in this class are as follows: a String value for the staff member's username and a String value for the password. Methods include one that searches the database for a specified patient within the database, as well as get and set methods for the specified attributes.

#### 3.1.3.3. Processing

There are no exception checks for any of the attributes within this class as there are no constraints towards them. The searching patient method will take parameters and access the database to see if the patient is stored there.

#### 3.1.3.4. Outputs

The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute.

### 3.1.4. Functional Requirement 4 (EmergencyRoomStaff)

#### 3.1.4.1. Introduction

The EmergencyRoomStaff class functions as the foundation for the staff assignments in a typical hospital emergency room department. ERS will be able to conduct intake for all incoming people into the emergency room, being able to add/edit/view basic information from the Person and Patient classes in the CARES System. As basic information about the ERS is unnecessary, the ERS class will only contain a method that will serve as an ERS checking-in a patient.

#### 3.1.4.2. Inputs

The inputted attributes for the ERS class include an extension of the Staff constructor, creating an ERS member with a username and password. The only method within the class is the createPatient() method, that takes in all of the preliminary data and creates a Patient object and stores it within the database.

#### 3.1.4.3. Processing

In terms of the methods utilized in this class, createPatient() will take inputted attributes from the person getting entered into the CARES System and create a Patient object with said attributes. Once the user is done editing information, the method will end on the user's prompt that they are finished.

#### 3.1.4.4. Outputs

For all the methods described for the ERS class, there will be no output apart from adding new information and creating a new Patient object or being able to view a display of the Patient object requested.

### 3.1.5. Functional Requirement 5 (Nurse)

#### 3.1.5.1. Introduction

The Nurse class serves as the class for nurses in the emergency room department. Nurses should be able to access the CARES System to view and edit any patient information, as well as order labs for any of the patient's ailments and end the discharge of the patient when it is time for them to be on their merry way. The Nurse class will be of direct inheritance to the ERS class. Capabilities of a Nurse object are handled in the user interface, there is only the constructor creating a Nurse object.

#### 3.1.5.2. Inputs

There will be no inputted attributes for the Nurse class apart from and a username and password that will be a Nurse's login information. All utilization of the Nurse class is handled at the user interface level, where a Nurse object is given permissions to access various sections of the portal. Nurses will be able to edit patient information, add measurements taken, conduct admittance and discharge.

#### 3.1.5.3. Processing

Due to the structure of the user interface, the Nurse's capabilities will be specified further.

#### 3.1.5.4. Outputs

For all methods described in the Nurse class, there will be no output.

### 3.1.6. Functional Requirement 6 (Doctor)

#### 3.1.6.1. Introduction

The Doctor serves as the class for doctors in the emergency room department. Doctors should be able to access the CARES System to view and edit any patient information as well as run labs and make diagnoses of the patient. The doctors should also be able to start the discharge of a patient and complete the discharge. Lastly, the doctors have to be able to approve the admission of the patient for more care. The Doctor class will be of direct inheritance to the Nurse class. Capabilities of a Doctor object are handled in the user interface, there is only the constructor creating a Doctor object.

#### 3.1.6.2. Inputs

There will be no inputted attributes for the Doctor class apart from a username and password that will be a Doctor's login information. All utilization of the Doctor class is handled at the user interface level, where a Doctor object is given permissions to access various sections of the portal.

#### 3.1.6.3. Processing

Due to the structure of the user interface, the Doctor's capabilities will be specified further.

#### 3.1.6.4. Outputs

For all methods described in the Doctor class, there will be no output.

### 3.1.7. Functional Requirement 7 (BillingStaff)

#### 3.1.7.1. Introduction

The BillingStaff class will serve as the billing department for the emergency room and adjacent hospital departments assisting with admitted patients. The class is of direct inheritance of the Staff class. Billing staff should be able to access the CARES System in order to calculate the billing for a patient's visit. Lastly, the billing staff needs to view a bill for each patient.

#### 3.1.7.2. Inputs

There will be no inputted attributes for the Billing Staff class apart from a username and password that will be a Billing Staff's login information. All utilization of the Billing Staff class is handled at the user interface level, where a Billing Staff object is given permissions to access various sections of the portal.

#### 3.1.7.3. Processing

Due to the structure of the user interface, the Billing Staff's capabilities will be specified further.

#### 3.1.7.4. Outputs

For all methods described in the Doctor class, there will be no output.

### 3.1.8. Functional Requirement 8 (Lab and LabPanel)

#### 3.1.8.1. Introduction

The Lab and LabPanel classes serve as the mechanism to conduct labs for a given patient. The Lab class contains a constructor to utilize many different lab types as objects. The LabPanel is of direct inheritance of the Lab class and serves to specify all lab types, creating an array of them to be utilized by each Patient object.

#### 3.1.8.2. Inputs

In terms of attributes, the Lab class contains an enumeration called LabResult, which contains the three values of NotRun, Normal, or Abnormal, a Random object, a LabResult object, a String value for the name of the lab, an integer value for the amount of each lab run, and a double value for the cost of each lab. Methods for the Lab class include a get method,

getTotalCost(), that gives the total cost of one lab type, traditional get methods for all other attributes described, and a method that generates a random result of a specified lab.

Attributes of the LabPanel class include a Lab object for each lab type the emergency room department is capable of conducting, those being a red blood cell test, white blood cell test, liver test, kidney test, x-ray, cat scan, mri, urine sample test, and stool sample test. The last attribute in this class is an array of the specified lab objects. At the time of creating a Patient object, there will be a LabPanel object made so that all labs will be able to be run on the patient no matter what ailment the patient has. Methods within the LabPanel class include runLab(), which runs a specified lab, getCurrentResult(), which returns the result of one specified lab, and getLabs(), which returns the Lab[] array.

#### 3.1.8.3. Processing

The getTotalCost() method multiplies the cost of the lab by the number of times the lab was run. All other get methods in the Lab class return their respective attributes. The reset() method resets a specified LabResult enumeration to NotRun. The run() method creates a random lab result by randomizing an integer value between 0 and 6, and if the integer is 0, then the lab result is Abnormal; any other number will make the lab result Normal.

The runLab() method takes a parameter of a number corresponding to a lab type in the labPanel array, and returns the generated result for that lab. The getCurrentResults() method returns the entirety of the labPanel array as one String. The getLabs() method returns the entire array of Lab objects.

#### 3.1.8.4. Outputs

The get methods of the Lab class will print out the various attributes requested. The run() method has no output apart from changing a lab result.

The runLab() method returns the result of a lab (or Lab.LabResult). The getCurrentResults() method returns a String array of all lab results at the time of accessing a Patient's information on the dashboard. The getLabs() method returns the Lab object array to be called by the Bill class.

### 3.1.9. Functional Requirement 9 (Condition and Diagnosis)

#### 3.1.9.1. Introduction

The Condition class serves as the basis for any prescriptions for ailments the patient might have. The class contains a constructor to specify three different types of prescription medication that the patient could be offered, and a get method for the entire array of prescription options.

The Diagnosis class functions to provide all the specified Conditions that a patient may have in the emergency room department. Not necessarily an extension/inheritance of the Condition class, the Diagnosis class holds a Condition object for each of the diseases/ailments will be utilized.

#### 3.1.9.2. Inputs

The Condition class takes one attribute, a Prescription object array is made to store three valid prescriptions a Doctor might prescribe to a Patient. The method getValidPrescriptions takes on no inputs, simply returning the array of prescriptions.

The Diagnosis class contains five Condition objects specifying each diagnosis used in the CARES System. Then there is a Condition object array storing said objects together in one array. Lastly for attributes, there is a boolean array, isDiagnosed, to assess if the patient is diagnosed for one or more of the five diseases. Methods of this class include a get method for the Condition object array, getConditions(), a get method for the boolean array isDiagnosed, getIsDiagnosed(), and a set method for the boolean array, to manage what diagnosis a patient might have.

#### 3.1.9.3. Processing

The get method of the Condition class returns its respective attribute, a Prescription object array.

The get methods of the Diagnosis class returns their respective attributes. The setIsDiagnosed() method sets a specified diagnosis, taking an integer number corresponding to the Condition in the isDiagnosed array and a boolean value to change the diagnosis to, and changes the respective diagnosis.

#### 3.1.9.4. Outputs

The lone output of the get method in the Condition class will be a Prescription object of three valid prescription options for a Patient.

The get methods of the Diagnosis class return their respective attributes, and the set method does not return an attribute.

### 3.1.10. Functional Requirement 10 (Prescription)

#### 3.1.10.1. Introduction

The Prescription class functions as the basis for all prescriptions being assigned to patients at an emergency room visit. The class contains an enumeration that describes the type of medications and the method of administration for each medication, as well as constructors corresponding with the enumeration.

#### 3.1.10.2. Inputs

Attributes of the Prescription class include an enumeration of all medications available to the Doctor with the administration type, specified options being declared in another enumeration in the same class. There is a boolean to see if a patient was prescribed something already, and a Medication object that is called upon for the use of the enumeration. Methods include a boolean get method for isPrescribed() and a set method for the same attribute.

#### 3.1.10.3. Processing

The get method of this class returns its respective attribute. The set method for the boolean changes the respective boolean value.

#### 3.1.10.4. Outputs

The output for the get method is a boolean value of if the Patient was prescribed the specified medication.

### 3.1.11. Functional Requirement 11 (Bill)

#### 3.1.11.1. Introduction

The Bill class serves as the bill for the entirety of the patient's visit, accounting for the time they spent at the emergency room, the treatments they received, and any medications or treatment they take with them out of the hospital. The bill will be generated automatically on discharge.

#### 3.1.11.2. Inputs

There will be no inputted attributes for the Bill class. This class is utilized through its methods by the MainViewController class when a patient is discharged by a Nurse or Doctor. Methods contained within the class include the labCosts() method that takes all labs run - also taking account of how many instances - for a specified patient and calculates the total cost for each lab, the formatBill() method formats each line of a bill to be within the same width, ensuring that the bill is completely one page and seamlessly aligned and the toBill() method, which creates the whole bill to be displayed onto the patient dashboard utilizing the previous two methods.

#### 3.1.11.3. Processing

The labCosts() method takes on an inputted array of Lab objects from a patient's visit and then traverses the array, adding each total cost as a line to the returning StringBuilder object. The StringBuilder's toString is then called as the return so that the toBill() method will be able to add that into the entire bill. The formatBill() method takes each inputted line of the bill and makes sure they are within the bound of 40 spaces, the width of the bill being displayed. The toBill() method takes an inputted Patient object and builds upon one String value, adding every line of the bill and returning the String to be displayed on the dashboard.

#### 3.1.11.4. Outputs

The output of labCosts() is a String value of all labs of the inputted patient. The output of formatBill() is a String value of one lab of the given attributes. The output of toBill() is a String value of the entire bill.

### 3.1.12. Functional Requirement 12 (Database)

#### 3.1.12.1. Introduction

The Database class serves as the storage mechanism for all patient information as well as all login information for existing staff members of the emergency room department. The class utilizes Hashtable data structure to store each type of information. To save the stored information, the Hashtables are serialized to a binary file under a specified name whenever someone logs out of the CARES System. Then, whenever someone logs back into the system, the file will be called and the Hashtables extracted to be used further. This way, no information is tampered with and the information can easily be transferred and accessed between different types of users.

#### 3.1.12.2. Inputs

To start, there are two String values that hold permanent, unalterable names for the two database Hashtables being used, one for patients and one for user login data. Then, two Hashtables are declared, one for each database. The Patient database will hold Patient objects with a String value (later defined as a concatenation of the patient's name and date of birth) for the key, while the login database will hold Staff objects with a String value (the username and password of the user) for the key. Methods include a get method for the Patient Hashtable and one for the Login Hashtable. Then, updateKey() changes a key in the Patient Hashtable, saveDatabase() saves both the Login and Patient Hashtables to a binary file stored with the program, with the name being their respective String names. The loadDatabase() method loads both the Login and Patient Hashtables, but if there are not any tables stored within the program, the method makes a skeleton table to be filled for new patients. The Login Database that loads has four Staff objects, one each for the type of staff member at the ER with example login information, as the program has not been submitted for real-world use yet.

#### 3.1.12.3. Processing

The updateKey() method takes in the parameters of two String values, one for the old key and one for the new key, and assigns the new key to the Patient object and re-adds the object into the Patient Hashtable. Both the saveDatabase() and loadDatabase() take on no parameters when loading the two Hashtables from a binary file.

#### 3.1.12.4. Outputs

The two get methods return their respective Hashtable Databases. Apart from those, there are no other returned values in the methods following.

### 3.1.13. Functional Requirement 13 (Warning Manager and Warning Listener)

#### 3.1.13.1. Introduction

The WarningManager class and WarningListener class manage all warnings within the CARES System that may occur during use of the program. The WarningManager handles errors and exceptions that are thrown in the MainViewController class, the main class of the user interface, where most functions that staff members will use are located. The Warning Listener is an interface called upon to make various warning messages to the program if something is inputted incorrectly or the program cannot function further.

#### 3.1.13.2. Inputs

The WarningManager class holds a WarningManager object and an ArrayList of WarningListener objects. The methods in the class include a get method for the WarningManager object, getInstance(), a method that adds a WarningListener object to the ArrayList, addListener(), and a method that traverses the WarningListener ArrayList and displays each one, showWarningToAll().

The WarningListener interface has no attributes but includes a method to be overridden in the user interface to show the message of each warning/error.

#### 3.1.13.3. Processing

The get method of the WarningManager class returns its respective object. The addListener() method takes a parameter of a WarningListener instance and adds it to the ArrayList of listeners. The showWarningToAll() method takes a parameter of a String value, being a warning message, and displays the message into the user interface, called on in the MainViewController class.

The showWarning() method of the WarningListener interface takes on a parameter of a String value, to be overridden in the MainViewController class.

#### 3.1.13.4. Outputs

The output of the getInstance() method is a WarningManager object. There is no other returnable output for the methods of the class. Methods within an interface, like WarningListener, can only be instantiated.

### 3.2. External Interface Requirements

#### 3.2.1. User Interfaces

The user interface will feature an easy organized system with concise layouts of buttons for each user that logs in. All emergency room staff will have similar sections labeled and access-ready once they log in to their respective account types. Patients will not have any access to the CARES System, as the only output they see will be a generated summary of their visit to the emergency room. For each action sending information to another department, like starting discharge or ordering labs, there will be a specified button that starts the process and sends the information to the next staff member who has to complete said process.

##### 3.2.1.1. Login Controller

The LoginController class features the launching point of the login page. When called upon with the Main class, the LoginController launches corresponding FXML files for a username field, password field, login button, and a label (in case a login error is thrown).

The checkLogin() method, which runs when the Button loginButton is clicked by a user, checks to see that the user's login information is correct (accessing the loginTable Hashtable), and then launches the main view FXML file, corresponding with the MainViewController class.

##### 3.2.1.2. Main View Controller

The MainViewController class contains the entirety of the patient dashboard when a staff member logins correctly. Depending on the permissions of said user, whether that be an Emergency Room Staff or Billing Staff member, a Nurse or Doctor, there will be some information panes unavailable.

The method initialize() launches all of the information panes corresponding to the staff member that is logged in. The logOut() method is very simple, relaunching the login FXML file, which rolls back to using the LoginController class. The setBillingStaffView method declares what information panes and buttons are available for Billing Staff members to use, which includes the billing field. The setDeskStaffView() method declares what information panes and buttons are available to a Front Desk Staff (ERS) member, which includes the basic information pane and the admit button. The setNurseView() method declares that any Nurse will be able to access all mechanisms of the CARES System patient dashboard, except for the Diagnosis panel, where only doctors are able to access and edit. The setDoctorView() method declares that any Doctor will be able to access all panels of the CARES System patient dashboard.

The loadPatient() method checks to see if a current patient is loaded (in that case, clears the Patient object on the dashboard) and then loads the called Patient from the Database, while also loading all corresponding fields. The loadFields() method traverses the loaded patient's information and assigns them to a TextInputControl array to be accessed in the loadPatient()

method. The unloadPatient() method simply loads an empty Patient object to reset the patient dashboard when called. The loadLabs() launches to load a current Patient's lab results if they have been run already. The searchPatient() function, triggered when a user clicks the search button, searches for the Patient object specified by name and date of birth fields, and loads it into the patient dashboard. If no Patient object exists, the program throws an error message and the user can try again.

The runLabs() method, triggered by the corresponding button event, runs whatever Labs are labeled to be called and then updates the LabPanel view to show the results of said labs. The method that follows, loadValidScripts(), produces the appropriate medications recommended to a Patient with a certain diagnosis. Then the updateScripts() method changes the prescription view based on what medication(s) were chosen by the user.

The pair methods of addListenerToTextField() and removeListenerFromTextField() handle when the user interface has to "listen" to the keyboard of the user's computer. RemoveListenerFromTextField() is utilized when unloading a patient from the dashboard.

For the admitButton() method, it checks to see that all measurements of a Patient were entered before continuing, then admits the Patient to the next step of the hospital visit, with a message displaying so. The dischargeButton() method, triggered by the corresponding button, conducts both starting discharge and completing discharge. A Doctor object has to start the discharge, and a Nurse object has to complete discharge, adding any extra treatment notes.

Lastly, the showWarning() is an overridden method of the WarningListener interface, adding to the method an alert that launches with the called warning/error message in a new small window.

### 3.2.2. Hardware Interfaces

The CARES System will use a combination of screen-oriented terminal control and line-oriented terminal control. Screen-oriented terminal control will be in use when users click buttons that activate certain processes, like admitting a patient or ordering a specific lab. Line-oriented control will be utilized in every aspect of entering a person's or patient's information into the CARES System.

### 3.2.3. Software Interfaces

In terms of software interfaces, the program will be designed to run on the most used operating systems in the technology field, including Windows, MacOS, and Linux. This entails no extra equipment or costly measures enforced on the CARES System for any hospital emergency room department that would like to use the system.

#### 3.2.4. Communications Interfaces

For communication interfaces, the hospital emergency room department must be able to support LAN connection for any non-mobile computers that are used and must be able to support wireless network connectivity for any mobile computers, like tablets, that are used.

### 3.3. Performance Requirements

As users will be logged in for long periods of time, the CARES System must be able to handle a huge amount of information as well as requests from the user. The program should be able to run continuously just as the hospital emergency room department is run twenty-four hours a day, seven days a week. And until the user logs out, the CARES System should be able to process requests and take in new information without error/overload.

### 3.4. Design Constraints

#### 3.4.1. Standards Compliance

Frameworks and guidelines will be used to ensure that the software is developed and maintained in a high-quality manner and with consistency. They will help maintain the quality, reliability, security, maintainability, and cost-effectiveness of the project. Compliance with regulatory standards such as security compliance standards like scanning for security vulnerabilities and encrypting data will eliminate risks of cyber attacks.

Recording the requirements and the changes in requirements throughout development is another standard that must be followed during design.

Due to the lack of budget, some design aspects cannot be integrated into the system, resulting in limitations on aspects such as the number of times the project can be updated and tested, and what tools can be used. This may affect the efficiency of the software due to having access to minimal resources. However, this may also make the program simpler to code, and easier for the doctors/nurses to go through the system, hence improving usability/ user-friendliness.

Individual designers' workflow and creative decision-making may lead to some constraints in the design. Also, the designers' skills and abilities are a factor in limiting the design features of the project. Since there is a time limitation, not many features can be integrated into the system within the limited time. These design limitations may be in the style, usability, and functional constraints such as the features that the software needs to include, and the accuracy of the tasks and features of the software.

Another factor that may affect the design is the specific set of requirements and standards for the project which may also lead to constraints in some other aspects of the design such as the technology used to build it, which may reduce the performance reliability.

System constraints such as the system not accepting updates to the user interface will also limit the design advancement.

#### 3.4.2. Hardware Limitations

Tech stack limitations, choosing the right set of technologies to use, and the availability of the technology are additional constraints that will limit the design of the software. In addition, this decision must be made at the start of the project as it is expensive to change later on. However, this makes the program simpler to code, and user-friendly for the doctors/nurses.

The designed software is only capable of running on specific hardware with the following requirements:

- Windows/any server-based OS,
- Minimum memory of 16GB
- Minimum storage of 1TB
- Minimum CPU 3.0GHz.
- The CARES software will only run on computers

## 3.5. Attributes

### 3.5.1. Availability

Ensuring the system can continue to function even in the presence of system failures and faults will be difficult for this project due to the limitations in resources and time. If the database goes offline, the site will not run, therefore, the hospital must have backup power to run the software in case of emergency power failures.

Factors such as early detection of potential problems, ease of diagnosis, and repair are important to maintain availability, which will need to be integrated into the design of this software to ensure availability.

### 3.5.2. Security Requirements

Due to HIPAA, data confidentiality must be maintained through the use of appropriate safeguards and security measures. This includes encryption, access controls, and regular risk assessments to ensure the protection of sensitive health information.

In addition to the integration of security measures, we must ensure the early integration of these features by implementing Symmetric and Asymmetric Key Cryptography, to avoid complications in the development process.

### 3.5.3. Maintainability

To make sure the software can be updated and adapted to changing needs while reducing the cost of ongoing development and extending the lifespan of the software, multiple versions of the project will be documented on Github as a Version Control System to track version updates to be able to return to previous versions if need be.

Code quality and documentation are also important factors for the maintainability of the software.

Frequent testing of the software to debug issues to maintain the user-friendliness of the software is also required for the development of this software. However, due to the limited resources and time, this may not be possible.

### 3.6. Other Requirements

An authorized individual at the hospital or information technology personnel needs to have access to the database, including the login information for the database, to maintain the system in the ER.