CS460W The Residents (Alessandro, Fatima, Aden)
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.

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.): Connecticut Advanced Russell Emergency Services
- SRS: Software Requirement Specification
- GUI: Graphical User Interface
- MySQL: Database language
- 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
- SP: Seen Patient
- AP: Admitted Patient
- DI: Diagnosis
- LT: Lab Test

1.4. References

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

- HIPPA: 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 dependences, 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. Similarly, the MySQL database, pivotal for the system's data management, is hosted on a desktop or laptop. 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.
- Vitals and Medical History Recording: Documenting vital signs and medical history.
- 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:

- Front Desk 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.

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 on a MySQL database.

2.5. Assumptions and Dependencies

The (C.A.R.E.S.) System's operational efficacy is fundamentally tied to its integration with a MySQL 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

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 a person who walks into the Emergency Room and checks in at the front desk. They have to wait in the waiting area before becoming a patient and being seen by a nurse for their ailments.

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 integer value for a phone number. There will be a few methods implemented into the Person class, including a toString method that will display the value of a Person object, as well as a get and a set method for each inputed attribute. All attributes in the class will be inputted from a database source of example patients at a hospital emergency room.

3.1.1.3. Processing

The inputted values will be checked upon entry into the system for validity. The toString will take all inputted values of the Person class and form one String value.

3.1.1.4. Outputs

There will be one output of the Person class from the toString method. The output will be one String value of all the attributes, describing the person’s name, date of birth, address, and phone number. The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute.

3.1.2. Functional Requirement 2 (Patient)

3.1.2.1. Introduction

The Patient class functions as the foundation for an incoming patient to the emergency room. 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 a couple of inputs for insurance information, a value to store their diagnosis, and a space for an emergency contact. A patient has to wait to be seen by a nurse before getting measured for height, weight, etc. and a diagnosis from the doctor.

3.1.2.2. Inputs

The attributes in this class are as follows: a String for insurance plan, a String for an emergency contact, and a String for a diagnosis. At the time of inputting the insurance and emergency contact values, there will be no value inputted for the diagnosis until the patient is seen by the doctor. Methods include a get and set method for all three attributes. Also, the toString method for this class will be an extension of the toString method of the Person class, providing the information of the previous toString with the addition of displaying the attributes of the Patient class.

3.1.2.3. Processing

The inputted values will be checked upon entry into the system for validity. The toString will take all inputted values of the Person as well as the values of the Patient class and make them one String value.

3.1.2.4. Outputs

There will be one output of the toString method in the Patient class. The output will be one String value of all the attributes from the Person toString method, as well as the attributes of the Patient class, describing the person's intake information as stated before and adding the person's insurance information, emergency contact, and the diagnosis, if there is one inputted. The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute.

3.1.3. Functional Requirement 3 (SeenPatient)

3.1.3.1. Introduction

The SeenPatient functions as the basis for a patient in the process of being attended to by a nurse or have been seen by a Nurse or Doctor. The SeenPatient class is of direct inheritance of the Patient class, adding on more attributes to account for a Nurse recording down a person's measurements like height, weight, blood pressure, heart rate (beats per minute), and placement for any labs ordered and their bill when the visit has finished.

3.1.3.2. Inputs

The attributes in the SeenPatient class are as follows: a double value for the height as well as a double value for weight, a String value for blood pressure, a double value for heart rate, an object for any Lab (discussed further), and an object for the Bill (discussed further). At the time of inputting the measurements taken by the Nurse, there will not be any bill stored, as this occurs further into the patient's visit and is submitted by the ERS. Methods include a get and set method for each attribute. Also, the toString method for this class will be an extension of the toString method of the Patient class, providing the information of the previous toString with the addition of displaying the attributes of the Patient class.

3.1.3.3. Processing

The inputted values will be checked upon entry into the system for validity. The toString will take all inputted values of the Patient as well as the values of the SeenPatient class and make them one String value.

3.1.3.4. Outputs

There will be one output of the toString method in the SeenPatient class. The output will be one String value of all the attributes from the Patient toString method, as well as the attributes of the SeenPatient class, describing the patient's height, weight, blood pressure, heart rate, any labs that were ordered, and the bill (if it has been generated). 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 (AdmittedPatient)

3.1.4.1. Introduction

The AdmittedPatient class functions as the last form of a patient at the hospital emergency room staff. This class is of direct inheritance of the SeenPatient, adding on two attributes relating to getting admitted and receiving discharge instructions from the doctor and/or nurse.

3.1.4.2. Inputs

The attributes of the AdmittedPatient class are as follows: a String value for the date admitted and a String value for the discharge instructions. At the time of making the SeenPatient into an AdmittedPatient, the discharge instruction will not be filled, as the patient has to be seen by the doctor and labs have to come back before the doctor makes a diagnosis and decision on discharge. Methods include a get and set method for each attribute. Also, the toString method for this class will be an extension of the toString method of the SeenPatient class, providing the information of the previous toString with addition of displaying the attributes of the SeenPatient class.

3.1.4.3. Processing

The inputted values will be checked upon entry into the system for validity. The toString method will take all inputted values of the SeenPatient toString method, as well as the values of the AdmittedPatient class and make them one String value.

3.1.4.4. Outputs

There will be one output of the toString method in the AdmittedPatient class. The output will be one String value of all the attributes of the SeenPatient toString methods, and the attributes of the AdmittedPatient class, describing the patient's admittance date and the discharge instructions from the doctor/nurse. The get methods will print out the various attributes requested, and the set methods will change the respective value of the attribute.

3.1.5. Functional Requirement 5 (EmergencyRoomStaff)

3.1.5.1. 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 methods that access and change the information of the Person and Patient class.

3.1.5.2. Inputs

There will be no inputted attributes for the EmergencyRoomStaff class. There will be a few methods for the ERS to be able to create a new patient file, newPatient(), and a method to edit or view any existing patient's information, named editPatient().

3.1.5.3. Processing

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

3.1.5.4. 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.

3.1.6. Functional Requirement 6 (Nurse)

3.1.6.1. 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. As basic information of the nursing staff is unnecessary, the Nurse class will only contain methods that access and change information of the Patient, SeenPatient, and AdmittedPatient classes. The Nurse class will also contain a method that ends the discharge of a patient, and a method for searching patients in the system database.

3.1.6.2. Inputs

There will be no inputted attributes for the Nurse class. There will be a method for searching for a patient, searchPatient(), a method to edit any details of the patient's visit, editPatient(), a method for making a Patient into a SeenPatient, seePatient(), a method to order a type of lab, runLab(), a method to make a SeenPatient into an AdmittedPatient, admitPatient(), and a method for completing discharge of any patient, completeDischarge().

3.1.6.3. Processing

The searchPatient method will display the Patient Info Home Screen. The editPatient() method will allow the nurse to enter in information about the existing patient profile. The seePatient() method will entail the nurse entering in the new information and make the Patient into a SeenPatient object. The runLab() method will generate the lab requested and display the latest information from that lab, adding it into the SeenPatient object. The admitPatient() method will make a SeenPatient object into an AdmittedPatient object. Lastly, the completeDischarge() method will allow the nurse to add any remaining discharge information and then generate a summary for the patient to take out with them.

3.1.6.4. Outputs

For all methods described in the Nurse class, there will be no output apart from adding and/or editing information of existing Patient/SeenPatient/AdmittedPatient objects.

3.1.7. Functional Requirement 7 (Doctor)

3.1.7.1. 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.

3.1.7.2. Inputs

There will be no inputted attributes for the Doctor class. There will be a method for searching for a patient, searchPatient(), a method to edit any details of the patient's visit, editPatient(), a method to order a type of lab, runLab(), a method to make a diagnosis, diagnose(), a method to start the discharge of a patient, startDischarge(), a method to approve admission of a SeenPatient (making an AdmittedPatient), admitPatient(), and a method to complete the discharge of a patient, completeDischarge().

3.1.7.3. Processing

The searchPatient method will display the Patient Info Home Screen. The editPatient() method will allow the doctor to enter in information about the existing patient profile. The runLab() method will generate the lab requested and display the latest information from that lab, adding it into the SeenPatient object. The admitPatient() method will make a SeenPatient object into an AdmittedPatient object. The diagnose() method will take the information entered from the SeenPatient information, like blood pressure and heart-rate, and analyze it to match with one of four diagnoses. Then, if the info matches with one diagnosis, the method will add that information to the patient's history, and suggest the steps of treatment based on the diagnosis. The startDischarge() method will generate a discharge message in which the patient will be able to see all diagnosis, and review what the next steps to treating the ailment Lastly, the completeDischarge() method will allow the doctor to add any remaining discharge information and then generate a summary for the patient to take out with them.

3.1.7.4. Outputs

For all methods in the Doctor class, there will be no outputs apart from generating the discharge information and generating a summary that the patient will see when leaving the hospital.

3.1.8. Functional Requirement 8 (BillingStaff)

3.1.8.1. Introduction

The BillingStaff class will serve as the billing department for the emergency room and adjacent hospital departments assisting with admitted patients. Billing staff should be to access the CARES System in order to view what labs and medications were ordered for the patient and then be able to calculate the price of the patient's visit. Lastly, the billing staff need to generate/view/edit a bill for each patient after they become a seen patient. As basic information of the billing staff is unnecessary, the BillingStaff class will not have any inputted attributes, but will have methods that correspond to the various actions described in this section.

3.1.8.2. Inputs

There will be a few inputted attributes for the BillingStaff class, which include all base prices for equipment and medicine that will be used on any patients within the hospital emergency room. There will be double values for each piece of equipment and medicine. There will be a method to calculate the price of the patient's visit, which will entail accessing the patient's profile, checking what labs were ordered and other things that cost money for the patient, calculating the total bill price, and display all the itemized prices out in a neat and orderly fashion; the method will be called generateBill(). Also, there will be a method so that the billing staff can view the bill once it is generated or to see if it has been paid or not, which will be called viewBill().

3.1.8.3. Processing

The method generateBill() will take base prices for all equipment and medicine used, which will be stored in the class as double values, and calculate the total bill by adding the prices of each piece of equipment and medicine that were used during a specific patient's visit, whether that be a SeenPatient or an AdmittedPatient. The bill will be a generated display that only the billing staff and the patient will be able to see.

3.1.8.4. Outputs

To reiterate, the BillingStaff class has only one output, which will be the displayed bill for the patient and billing staff to see in the CARES System.

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 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 that has to complete said process.

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 from 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.

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 software will only run on computers

3.5. Attributes

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 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.

3.5.1. 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 IT personnel needs to have access to the database, including the login information for the database, to maintain the system in the ER.