MediForm

Team NTP

Nicholas Reilly, Timothy Kodz, Paul Fundaro

<https://github.com/TimothyKodz/MediForm>

1.1: Purpose

Dear client, we would like to introduce and give you the specifications for the Mediform.

1.2: Scope

The Mediform is a multi-purpose application that allows the users to have access to the information based on their log in. This application was made to ensure accurate information goes to the correct hospital employees. This application will make it easy for medical users to access existing information, get updated information from the patient, and communicate from the receptionist to the billing department with ease. Our goal is to make hospital employees' lives easier by allowing fast patient medical updates with quick and easy information access for the appropriate people.

1.3: Definitions, Acronyms, and Abbreviations

No definitions, acronyms, or abbreviations were used.

1.4: References

No references were used.

1.5: Overview

This document will discuss the details and requirements intended to be fulfilled by the system. Due to the limited scope of the project there are some aspects of development that will not be covered, or only briefly covered. The main focus of the document is on the specific functions of the software and how those will be implemented and executed. Other topics that are discussed include some hardware requirements, and database connectivity.

2.1: Product Perspective

Mediform is an independent product that doesn’t rely on other hospital systems other than a computer to run it.

2.2: Product Functions

Our system will provide access to a database containing patient information for an emergency room in a hospital. It will provide 4 different levels of access to patient information, based on role within the hospital: registration, nurse, physician, and billing. Access level will be determined through each user's login information which will grant each user the appropriate level of access. Depending on role, users will have access to: view patient information, input new patient information, edit patient information, order medical tests, create prescriptions, and compile pricing information. Each role should have the following permissions:

Registration: input, edit, and view patients' personal info; this encompasses things such as name, address and date of birth. Input patient symptoms.

Nurses: input and edit some medical information, such as basic vital signs, and view all medical information. Add to and view patient notes, created by the physician and other nurses.

Physician: input, edit, and view all medical information. Create diagnoses. Order medical tests and procedures. Order prescriptions. Create, edit, and view patient notes, created by the physician and nurses.

Billing: View the tests and procedures performed and any medication the patient took during their stay. Determine the length of stay. Compile a bill based on: quantity and price of procedures and medication, and duration of stay. Print the bill.

2.3: User Characteristics

This system is a user-friendly environment and does not require past experience and/or any extra training in order to operate the system to its full functionality.

2.4: General Constraints

Minimum System requirements needed to run all software involved in the system. Password Encryption will be added to ensure security regulations.

2.5: Assumptions and Dependencies

We are under the assumption that all medical employees are trained in their specific roles.

We are dependent on the fact that each user will be classified under the correct role so that they will be able to access the information that they need. Also that the machine being used is a windows machine and has to have java.

3.1.1 Function: Login

3.1.1.1

All users should have a unique username and password to log into the system. Their login credentials and staff information, such as their role in the hospital, will be stored in a staff database.

3.1.1.2

The user will use a keyboard to input their login. The username and password will be type String, but the password will be stored encrypted in the database.

3.1.1.3

The system will pull the information from the database to verify if the input credentials are correct. The system will first take the username to search the database for the correct staff entry. To check if the password entered was correct, the password, stored in the database, will be decrypted and then compared to the password entered by the user.

3.1.1.4

If the users login credentials are correct then the user will be logged into the system with their defined role, if either of the credentials are not correct then the system will prompt the user to re-enter their information.

3.1.2 Function: Create registration form

3.1.2.1

This form should be filled out by the user upon the patient's arrival to the hospital. The information that is entered into the form will be stored in a database containing all patients registration information. Patients' various information will be identified by a patient ID that will be assigned by the system, and used to retrieve patient information by the staff.

3.1.2.2

The registration form should accept the following [value : type] pairs:

Name : String

Address : String

Phone : String

Date of birth : String

Emergency contact name : String

Emergency contact phone : String

Insurance : Boolean

If true:

Insurance provider : String

Insurance ID : String

Primary physician : String

Current medication : Boolean

If true:

Medication name : String

Allergies : Boolean

If true:

Allergen : String

Medical history : String or null

Date of visit: String

Symptoms/Reason of visit : Boolean

Symptoms will be presented as a checklist.

Symptoms should also be viewable in the medical information

3.1.2.3

All fields should be completed unless specified as accepting null. The software will be able to generate its own IDs (type Int) for each new patient. IDs will be generated beginning at 1 and counting upward by 1 for each new patient. The system will check the total number of entries in the collection containing patient registration, then create a new ID that is one more than the total number of entries.

3.1.2.4

Upon submitting the form, the user will be returned to the menu screen, with a popup informing that the patient entry was created. If the form fails to submit due to an error in one of the entry fields the user will be notified and the system will remain on the form page.

3.1.3 Function: View registration form

3.1.3.1

Allows the user to view the registration information for a patient that has already been created.

3.1.3.2

The user will need to input the patient ID.

3.1.3.3

Using the patient ID number that the user inputs, the system will search the database for the entry with the corresponding patient ID.

3.1.3.4

The information will be displayed in the same format that new entries are created.

3.1.4 Function: Enter vitals

3.1.4.1

The user will take and record vital signs of the patient. This will be done through manual entry into a form, and then submitted to the database

3.1.4.2

The viral sign information that will be recorded is as follows: temperature : int, pulse rate : int, respiration : int, blood pressure : String; they will also record weight : int, and height : float, if able.

3.1.4.3

The system will create the entry into the medical info database using the patient ID that the user would enter before being able to take vitals.

3.1.4.4

Upon submitting the form, the user will be returned to the menu screen, with a popup informing that the patient entry was created. If the form fails to submit due to an error in one of the entry fields the user will be notified and the system will remain on the form page.

3.1.5 Function: Medical information

3.1.5.1

The user will be able to create, edit, and view a patient's medical information. This information will pertain directly to their care and treatment at the emergency room, and will direct diagnoses for the patient. The patients symptoms/reason for visit, and vital signs are included in this grouping of information. The user will also be able to diagnose the patient's problem and record it with this function. The diagnosis/es will be recorded in the patient's medical information and stored in a database for medical info, using the patient ID as the key for the entry. The user will also be able to admit or deny the patient residence in the emergency room.

3.1.5.2

The diagnosis will be selected from a list of possible ailments suggested by the system based on presented symptoms. This will be done using check boxes and the results will be recorded as type Boolean. The choice to admit or deny the patient will also be done using check boxes and recorded as type Boolean.

3.1.5.3

This function will update the existing medical information entry in the correct database that was previously created by the user that input the patient's vitals.

3.1.5.4

Upon submitting the form, the user will be returned to the menu screen, with a popup informing that the patient entry was properly updated and saved.

3.1.6 Function: Order medical test

3.1.6.1

The user will select from a variety of medical tests to be performed on the patient based on symptoms and diagnosis. Tests ordered will be recorded and stored in the patient's medical information, and results will be available when viewing medical information. If the same test is ordered multiple times for the same patient the number of times the test is run will be recorded. This number should be stored as an int. The tests available for selection are broken into four categories: blood tests, radiology, urine, and stool.

3.1.6.2

Medication will be ordered through selection from a list, by checking boxes that are . Multiple tests can be ordered at the same time.

3.1.6.3

Test results will be automatically generated using an algorithm.

3.1.6.4

Upon submitting the order, the user will be returned to the menu screen, with a popup informing that the test was ordered.

3.1.7 Function: Order medication/prescription

3.1.7.1

The user will select from a variety of medication for the patient to receive based on diagnoses. Medication ordered will be recorded and stored in the patient's medical information, including the dosage and total amount taken. This number should be stored as an int.

3.1.7.2

Medication will be ordered through selection from a list, followed by an input regarding the amount of medication to be ordered. Multiple medications can be ordered at the same time.

3.1.7.3

The function will check that each medication ordered, has an int value in the correct field.

3.1.7.4

Upon submitting the order, the user will be returned to the menu screen, with a popup informing that the medication was ordered. If the form fails to submit due to an omission in one of the quantity fields the user will be notified and the system will remain on the form page.

3.1.8 Function: Notes

3.1.8.1

The user will have the ability to create new notes on a patient, and view the collection of created notes for that patient.

3.1.8.2

The user should be able to enter any note as a String using the keyboard.

3.1.8.3

Notes will be stored directly in a separate database, and will be retrieved using patient ID. The user who created the note will be identified on the note itself. When retrieving notes to view, the system will pull all entries with the matching patient ID.

3.1.8.4

Upon submitting the note, the user will be returned to the menu screen, with a popup informing that the patient note was properly saved.

3.1.9: Function: Compile bill

3.1.9.1

This function will be utilized by the billing staff. The function will pull data on the patient from the database, this data should include: the type and quantity of tests performed on the patient, the type and quantity of medication used on the patient, the length of the patient's stay.

3.1.9.2

There would not be any input from the user as all data is drawn from the databases and algorithms.

3.1.9.3

The function will employ an algorithm that sums the products of the cost of procedures/medication and quantity of procedures/medication.

(Cost of med X \* Amount of med X taken) + (Cost of test Y \* Amount of times test Y was taken) + …

3.1.9.4

The function should output to a print friendly format, for hard copy sharing with the patient.

3.2.1 User Interfaces

The style of our software should be simple and clean, with the attempt to reduce potential confusion and training time for the staff that is using it. Upon launching the software, all staff will be taken to a login page, containing: the software name, and text boxes for the user to input their username and password. They will then be taken to a page that requests a patient ID. After entering a valid ID, they will then be taken to a main menu page which contains buttons for the functions respective to their role in the hospital.

Registration staff, because they often will be creating entries for new patients, they will bypass the page requesting patient ID and instead will be taken directly to the main menu. They will have buttons to: create a new patient in the system, and view a patient's registration information, which will then prompt for the patient ID.

Nurses will have buttons to: input a patient's vitals, view a patient's medical information, and view a patients notes.

Physicians will have buttons to: view a patient's medical information, order medical tests and procedures, order prescriptions, and view a patient's notes.

Billing staff, because they only have one function, can complete all their work from a single page, will be taken to a page to enter a patient's ID to access the information they need for billing.

In the top left of the main menu, there will be a button to return to the screen that prompts for the patient ID.

All of the data that will be input and viewed will be in a basic form format. The goal is to maintain similarity with other paper forms that may have been used in similar institutions before. This could help to expedite user familiarity with the system, as it should be a format they are somewhat accustomed to interacting with.

In the top left of all of the form pages, there will be a button to return the user to the main menu; this should be located in the same position as the main menu button to return the user to the patient ID page. In the top right of the form pages will be a button to access additional functions such as editing or printing the form. These options will only be available if the staff member's role should have access to that function.

3.2.2 Hardware Interfaces

The system will operate using a screen-oriented terminal mode, sending completed forms to the database. Since the scope of the project contains the system to a single machine, interface protocol should not be a concern.

3.2.3 Software Interfaces

There will be one software interface for the system. To maintain patient information, a database will be created and maintained using the database program MongoDB.

3.2.4 Communication Interfaces

Due to the scope of the project this is not a relevant concern for the system.

3.3: Performance Requirements

Although there are no direct performance requirements, seeing that this is a software that will be used in a professional medical setting, where rapid response may be necessary, performance should be considered. Entries and edits to patient information in the system should come infrequently, as our form system of entering information allows staff to submit collected relevant information at the same time. Requests to view information should also come infrequently, as patient information should not need to be updated rapidly. We expect the response time of the system to process completed forms that have been submitted to be less than one second. Similarly, the response time for the system to retrieve and display information for the user should be less than one second. Although the scope of this project does not encompass things such as external terminals, there should be no limits to the number of users that could connect to the system.

3.4.1: Standards Compliance

Due to the scope of the project there are no limitations from standards.

3.4.2: Hardware Limitations

Recommended 1GB of RAM for database functionality as well as an appropriate amount of extra storage given the number of expected entries into the database. A Windows machine is also necessary.

3.5.1: Availability

Use of an external database and backed-up code in case of any hardware or software failures.

3.5.2: Security Requirements

For security purposes, a login with a username and password will be required to enter the software. If unknown or incorrect credentials are given, the software will not allow the user to access any data or information and will not pass the login screen. Certain login users will also be limited on what information they can view in the system.

3.5.3: Maintainability

Use of the MongoDB database to maintain all information safely and securely via cloud-based technology.

3.6: Other Requirements

There will be requirements for our data basing tool (MongoDB) that we will be using to store patient data. One of the requirements is that they must use a windows 10, 8.1, or 7 sp1.