**ER Admission Software CS 460W**

**Envision Health**

**Will Pitney and Katherine Burgan**

**GitHub:** [**https://github.com/Kburgan1006/CS460W-Project**](https://github.com/Kburgan1006/CS460W-Project)

1. **Introduction**: This section contains an overview of the complete document.
   1. **Purpose** The main purpose of this document is to provide an overview of the specifications and requirements for the proposed software, to be read by the client and the software design team.
   2. **Scope** This document specifies requirements for an application to manage Emergency Room (ER) patients.

The application allows users to:

* Input and read basic information about a patient
* Order labs, and record and read the results
* Diagnosis problems with patient
* Admit patient for extended stay
* Prescribe medication for patients
* Label discharges
* Bill patients upon discharge in an itemized bill

(How is the application storing information/ documents?)

(Does the application run offline and store locally or were we going to have a database?)

* 1. **Definitions, acronyms and abbreviations** (As Needed)
  2. **References**
* Textbook - Object-Oriented Software Engineering Using UML, Patterns and Java, Bernd Bruegge and Allen H. Dutoit, 3rd edition, Prentice Hall
  1. **Overview** The rest of this document will provide a general description of the product, including the product perspective, the product functions, the product user characteristics, the general constraints, as well as any assumptions and dependencies. This document will also describe any specific requirements, including functional requirements, external interface requirements, performance requirements, design constraints, and quality attributes of the application.

1. **General** **Description** This section contains a description of matters that concern the overall product and its requirements. It provides a perspective for understanding the specific requirements from section 3 of this document.
   1. **Product** **perspective** This product is independent and stands alone. It will be its own executable file independent of other hospital software and software resources. Hardware used for this product will be the existing hardware already used by the hospital and its staff.
   2. **Product** **functions** There are 3 major functions for our product: Admitting, Testing/Diagnostics, and Billing/Discharge. Admitting will deal with taking in data about the patient and entering into the system. Testing and Diagnostics will serve to order tests, view results, and a section to enter a diagnosis. Billing/Discharge will generate a document of discharge instructions for the patient given by the doctor. It will also generate a bill compiled of all tests performed and medications given.
   3. **User** **characteristics** The users for this product will be general ER staff, nurses, doctors, and billing employees. There will be no special training, experience, or technical expertise required to use our product. The only requirement should be a general walkthrough with client and final product.
   4. **General** **Constraints** The only constraints for this product will be concerning access to different parts of the software and what sensitive information is being stored. Some users should not have access to read or write in certain areas. This is due to sensitive information about the patient and giving proper access to those that need it.
   5. **Assumptions** **and** **dependencies** (TBD)
2. **Specific** **Requirements** This section contains all the details which are relevant for the design phase to follow. The ordering given here is just one way to present the specific requirements in a logical way. Specific requirements should be such that one may objectively determine whether they are fulfilled or not.
   1. **Functional** **Requirements** In this subsection, a description is given of how the transportation of inputs to outputs is achieved. The description given for each class of functions, and sometimes for each individual function. To a certain extent, this description can be seen as a solution to the user. This component of the requirement specification is the main starting point for the design phase.
      1. **Admitting and entering Patient Information**
         1. **Introduction** This function will allow ER staff to admit patients and enter in all required information. This will be done through a combination of textboxes, combo boxes, selection boxes etc.
         2. **Inputs** Required inputs will be: Name, address, phone number, prior medical history, time and date, and admission status. Address should be in standard format along with phone number. Prior medical history will just be general text entered by user. Time and Date will automatically be added upon submission. Admission status will be entered/updated by user.
         3. **Processing** Before submitting data to the database we will check for all required lengths and formats. Functions to count length of strings will be performed as well as checking that required information was entered. If there is anything incorrect it will prompt the user to correct it before attempting to submit again.
         4. **Outputs** The output for this function will be the submitting/entering the given data into the database.
      2. **Ordering Labs and Tests**
         1. **Introduction** The application shall allow users of certain credentials to place orders for blood tests, x-rays, and other exams to be performed on a patient by a technician
         2. **Inputs** The application will take both the patient to be administered as a string tests and the type of test to be performed, as a string listed within a predefined enumeration of available tests.
         3. **Processing** The application will check to ensure that the patient exists within the database and that the ordered test is one of the available types. The application will respond to abnormal inputs by refusing to place and order and warn the user about correct inputs. The application will function with an all-or-nothing paradigm, either completing the order or doing nothing at all.
         4. **Outputs** The application will output a form or ticket with the patient name and ordered test as a (raw text file? Pdf? Not sure.)
      3. **Writing Discharge Instructions**
         1. **Introduction** The application will allow users of specific credentials to write documents containing information regarding patient discharges.
         2. **Inputs** The application will take a long string from the user.
         3. **Processing** (N/A)
         4. **Outputs** There will be no output for this specific requirement. (Addressed in a later Functional Requirement)
      4. **Prescribe Medications**
         1. **Introduction** The application will allow specific user classes to prescribe medications from a predefined list to patients in order to help treat illnesses and other health issues. The application will allow the user to prescribe multiple medications to a single patient.
         2. **Inputs** The application will take a string containing the name of the medication and an integer containing the amount of medication to prescribe.
         3. **Processing** The application will check that the name given is within the predefined list of medications, and warn the user if the provided name is not found. The application will check that the amount of medication is within the valid range of integers no less than one.
         4. **Outputs** There will be no output for this specific requirement. (Addressed in a later Functional Requirement)
      5. **Diagnosing**
         1. **Introduction** The application will allow specific user classes to diagnose diseases and other health problems of a patient from a predefined list of issues.
         2. **Inputs** The application will take a string containing the name of the health issue. The application will take a string containing the patient’s name.
         3. **Processing** The application will check that the provided issue is found within the predefined list, warning the user and failing to process if it is not. The application will check that the provided patient name is found within the patient database, warning the user and failing to process if it is not.
         4. **Outputs** There will be no output for this specific requirement. (Addressed in a later Functional Requirement)
      6. **Ordering Discharge**
         1. **Introduction** The application will allow users of a certain class to order a discharge for a patient from the hospital.
         2. **Inputs** The application will take a string containing the patient’s name. The application will take the date and time of the discharge.
         3. **Processing** The application will check that the patient exists within the database, warning the user and failing to process if otherwise. The application will check that the date and time are valid, warning the user and failing to process if otherwise.
         4. **Outputs** There will be no output for this specific requirement. (Addressed in a later Functional Requirement)
      7. **Generating Bill**
         1. **Introduction** This function will generate a bill document based on all tests ordered, prescriptions/medications given, and total nights stayed. These will all have set fees that will be listed on the itemized bill.
         2. **Inputs** There will be no actual direct input for the required information. This information will be pulled from the database and formatted into a document.
         3. **Processing** After pulling data we will need to ensure calculations for the total bill are done correctly based on the data. We will also need to ensure that the total is properly formatted.
         4. **Outputs** The output for this will be a generated document for the ER staff to print and give to patient. This document will clearly state what the charge is and how much it was. It will also list the total amount due to be paid.
      8. **Discharge Papers**
         1. **Introduction** We will need to generate discharge papers using instructions and medications given by the doctor/nurse. These will be given to the patient upon discharge with their bill.
         2. **Inputs** There will be no direct inputs and information will be requested from the data base. There should only be discharge instructions, a diagnosis from doctor, and prescriptions if given.
         3. **Processing** There should be no need to process or check data pulled from the database as no changes to the data will be made. We will only need to request the information from the data base and then format it into a discharge document.
         4. **Outputs** The output for this function is the discharge document with instructions, diagnosis, a prescription information. This will be provided to the patient.
   2. **External** **interface** **requirements**
      1. **User** **interfaces** Should have single basic login in screen. After logging in there will be different sections for different categories of functional requirements. There will be different access levels, so depending on the user access they may be prevented from accessing different areas. (TBD)
      2. **Hardware** **interfaces** There will be no direct interfaces with other hardware.
      3. **Software** **interfaces** There will be no required software for this product. It will be able to run all computers in the ER department with the already existing operating systems.
      4. **Communications** **interfaces** (N/A)
   3. **Performance** **requirements** The goal for this program is for it to be able to concurrently handle multiple users at one time. If this is not possible we will go for one user at a time.
   4. **Design** **constraints** Design constraints may result from such things as the prescribed use of certain standards or hardware. (Ask about it, possibly comparable to max number of people admitted to ER) (TBD)
      1. **Standards** **compliance** There are no existing standards or specific regulations for discharge papers and will be discussed with client as we learn more information.
      2. **Hardware** **limitations** A description of the characteristics of the hardware environment, in as far as they lead to software requirements. An example of this might be the amount of memory available.
   5. **Attributes** (TBD)
      1. **Availability** Software must withstand regular use and should not have an issues during its use. There should be no major breakdowns or data loss.
      2. **Security** **Requirements** We need to ensure that proper access and security is maintained with different access groups. Example: ER staff should not be able to see testing. Passwords should also be encrypted and stored properly.
      3. **Maintainability** (N/A)
   6. **Other** **requirements** (N/A)