# MEDICAL LAB AUTOMATION SYSTEM (MLAS)

Software Requirements Specification (SRS) Document

Software Engineering Assignment

2015

R Naresh (13CS30024)

Barnopriyo Barua (13CS30009)

## TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 PURPOSE

The purpose of this document is to give a detailed description of the requirements for the "Medical Lab Automation System" (MLAS) software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team. This will also address the constraints and assumptions made about the software.

## 1.2 PROJECT SCOPE

The software "MLAS" is a lab automation software which is used to automate various activities of a medical laboratory. The records of various tests (information regarding test name, normal values of test) are maintained. The administration has the authority to add/remove/edit information of tests. The records of various patients are also maintained along with the records of information of the Doctor under whom the test is being conducted. Whenever a patient comes for a test a bill (containing a unique id) is generated specifying the details of the test and the date and time of collection of test reports and the cost of the test. The generated test report shows the results of the test and the information regarding the patient and the normal value for the test. The management is notified when the equipment (like test tubes, syringes, etc.) are under-stock, so that they can place order for new stocks. The software depends heavily on a well maintained Database Management System. The GUI of the software is made in Java using Netbeans and is made user-friendly.

## 1.3 AUDIENCE AND READING SUGGESTION

The common audience of this document is The Management of the Medical Laboratory, and the Software Developer.

## 1.4 TERMS/ABBREVIATIONS AND ITS DEFINITION (READING SUGGESTION)

| TERM/ABBREVIATION | DEFINITION |
|---|---|
| **MLAS** | Medical Lab Automation System |
| **Patients** | People who come for medical tests |
| **Management** | The person who is in charge of the medical lab and is responsible for adding/removing/editing/listing medical tests and their normal values, and places order for new equipment upon notification of shortage of stock. And havs the authority to generate bills, test reports and check stocks of medical lab equipments (like syringes, test tubes , etc.) and |

| | |
|---|---|
| | notify the Management when there is a shortage of stock. |

## 1.5 References

IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirement Specifications. IEEE Computer Society, 1998

# 2 Overall Description

## 2.1 Product Perspective

This software “MLAS” is an automated version of the existing system which was performed manually. The management of report generation and maintenance of records of patients and various information regarding various tests were maintained in record books previously. The addition of new tests and editing of tests were difficult as the changes were required to be made in every traces of the record books, but now as the records are maintained together, it has become a lot easier. Also the process of notification when stocks were finished had to be done manually. This automated system replaces the previous manually performed system and is more efficient as the databases are maintained together in one place and is more secure. Also it reduces the dependency on manual labour as it is automated and it is easy to use.

## 2.2 Product Features

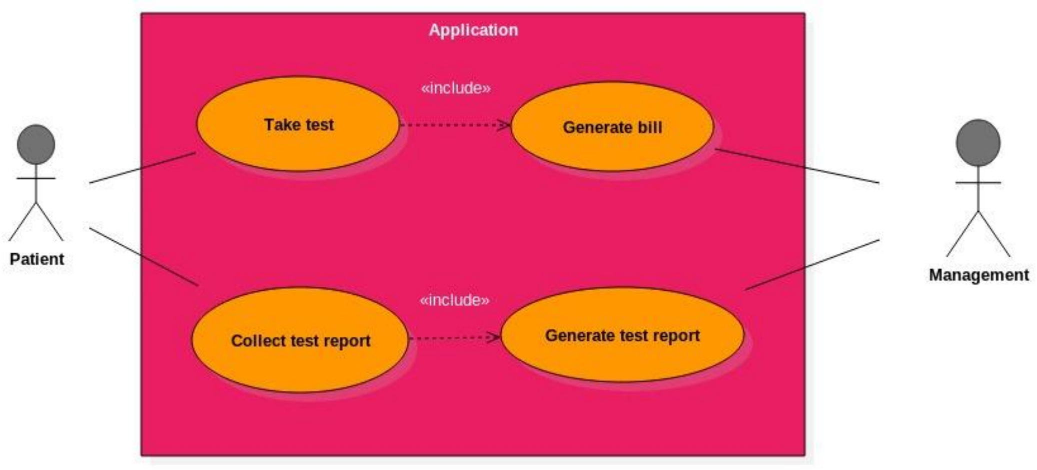
The user (Administrator) can add tests, view available tests, remove tests and edit the information of the tests (such as test name, normal values of the tests). The user (patient) can ask to generate the bill (which contains relevant information about the collection of test reports, cost for the test and the Doctor under whom the test is conducted). The user (patient) can ask for the test reports (which contains information about the test results, patient details and the normal values of the test) on production of the bill with unique id. The user (management) is notified when there is a shortage of stocks of devices or equipment (such as syringes, test tubes, medicines, etc.).The Administrator can check the status of the current stock also.

## 2.3 User Classes and Characteristics

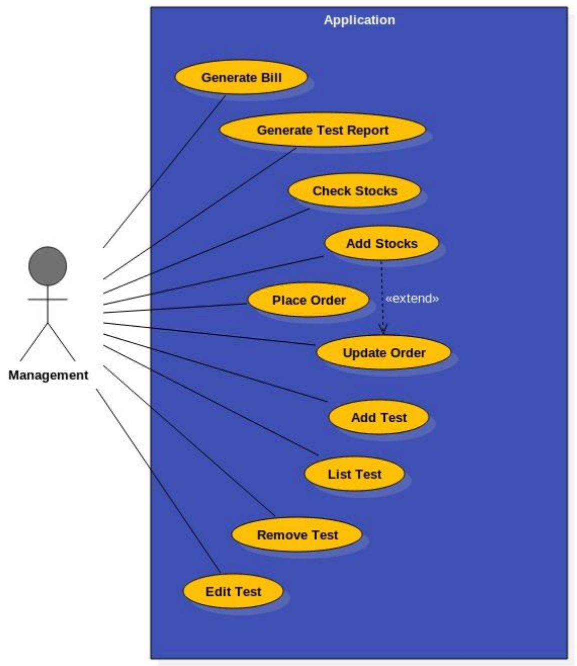
The Management of the Medical Laboratory is the user who will use this product. He has the access to every feature of the product. The Management has to generate the bill when a patient takes a test and generate the test report when the patient comes to collect the report. The management is the major user of the product and has the right to edit the tests available in the laboratory. The different use classes are:

- Management – Power to edit/add/remove test, list tests, and place order for new stocks. Power to Generate a Bill when a Patient takes a test and Generate Test Report when the patient comes to collect the test report, check stocks and notify Management when shortage of stocks.
- Patient – Power to take test and collect test report. He receives a bill after taking the test and he has to present that bill on or after the specified date and time to collect the test report.

*Use Case Diagram for user class Patient:*



*Use Case Diagram for user class Management:*



## 2.4 Operating Environment

It is operative in any operating system. The only requirement is that the environment should have the current version of the jdk (java development kit) (jdk 1.7 or above). As the programing is done in java (Netbeans ide), so there is a requirement of the java compiler (javac) pre-installed and working in the operating environment.

## 2.5 Design and Implementation Constraints

Connection is a constraint for the application. Since the application fetches data from the database (which will probably reside on a single centralized server), it is crucial that there is a connection for the application to function. The application will be constrained by the capacity of the database. The database may be forced to queue incoming requests and therefore increase the time it takes to fetch data.

## 2.6 User Documentation

The primary goal of the MLAS is to facilitate the process of Medical Lab Work. Consequently, the application will be designed to be as simple to use as possible. Nonetheless, users may still require some supplementary information about each component of the MLAS. The application will contain a Help Menu which will provide a tutorial on how to use the software.

## 2.7 Assumptions and Dependencies

The following is a list of assumptions made about the MLAS that affect the working of the application:

- The Management (Head of Medical Lab) has the right to add new test information, remove or edit already existing test details (test name, normal values for the test etc.).
- The Management generates the bill when a Patient takes a test and is also responsible for the test report generation before the patient comes for collecting the test report.

The following is a list of dependencies:

- A proper network connection exists between the server (on which the medical test database resides) and the various medical lab sub-systems which try to access the database.
- The server has a properly installed version of MySQL (used for the database).
- All the sub-systems using MLAS should have JDK 1.7 or above properly installed.

# 3 FUNCTIONAL REQUIREMENTS

## 3.1 USER CLASS – PATIENT

### 3.1.1 Take test

- Input: The patient comes for taking a Test in the Medical Laboratory.
- Output: The patient gets a Bill which contains information about the test and the date and time of collection of test report and has a unique id and the flow is passed to use case 3.2.1.
- Process: The Management checks if the test is provided by the Lab or not and then issues a bill with a unique id.

### 3.1.2 Collect test report

- Input: The patient comes with the bill (containing an unique id) and asks for the test report.
- Output: The Management gives away the test report (containing information about the test, patient's information, and test result and normal values of the test) corresponding to the bill id and the flow is passed to use case 3.2.2.
- Process: The Management checks whether the patient has come on or after the date and time mentioned in the bill for collection of test report and then gives away the test report.

## 3.2 USER CLASS – MANAGEMENT

### 3.2.1 Generate bill

- Input – The Management generates a bill using patient's name, doctor's name and the test information.
- Output – The Management generates the bill by specifying the date and time of collection of test report and assigns a unique id.
- Process – The Bill is generated by the Management after verifying the availability of the Doctor and whether the test is provided by the Lab.

### 3.2.2 Generate Test Report

- Input – The Management generates the test report using patient's details, doctor's name, test information and the results of the test.
- Output – The Test Report is generated corresponding to the bill of the Patient.
- Process – The Test Report corresponding to the bill id is generated after checking the date and time of collection and only given when the bill is produced by the Patient.

### 3.2.3 Check Stock Item

- Input – The stock item whose quantity is to be checked is the input.
- Output – The output is true if the quantity of it is greater than some threshold else the output is false.
- Process – This function checks the quantity of the stock item object and compares it with the threshold for it.

### 3.2.4 List Stocks

- Input – This function does not take any input.
- Output – This function prints the entire Stock array (name as well as the quantity of them).
- Process – This function accesses the Stock (array of Stock_Items) which is a member of the user class and then lists/prints all the stocks.

### 3.2.5 Check_all_Stocks

- Input – It does not take any input but uses the Stock array present in the user class.
- Output – The output is a Boolean array for each stock_item. Each of them is true if the quantity of that stock_item is greater than some threshold else the output is false for that particular stock_item.
- Process – This function checks the quantity of each of the stock item object and compares it with the threshold for it.

### 3.2.6 Add_Test

- Input – The test object is input which is to be added.
- Output – This function outputs the tests array added with the new test and returns false if the array already has the test added.
- Process – This function adds the test in the tests array after creating a test object with the given test details after checking if the test is not present in the tests array.

### 3.2.7 Remove_Test

- Input – The test object which has to be removed is the input.
- Output – This function outputs the tests array with the given test removed.
- Process – This function first checks if the test object given as input is present in the test array or not. If the test is present, this function removes the given test array by deleting the test object given as input which was added in the array before.

### 3.2.8 List_Tests

- Input – This function does not take any output.
- Output – This function lists/prints all the tests present in the tests array (i.e. all the tests added by the Management).
- Process – This function iterates over the tests array and lists/prints each of the tests at a time.

### 3.2.9 Edit_Test

- Input – This function takes two inputs:
  a) The test object (previous) which the user class wants to edit.
  b) The test object (new) constructed beforehand with the test details with which we want to edit it.
- Output – This function edits the test array by replacing the test object (previous) by the test object (new).
- Process – This function first finds the test (previous) array and then replaces with the test (new) object. If the test (previous) object is not found, this function adds the test (new) object in the tests array.

### 3.2.10 Place_new_Order

- Input – This function takes the stock_item whose stocks are in shortage.
- Output – This function adds the stock for the stock_item and increases the Quantity data member of the stock_item object.
- Process – This function places order and then increases the quantity of the stock_item by finding it in the Stock array. If the stock_item was not present, it adds the stock_item in the Stock array.

### 3.2.11 Place_bulk_Order

- Input – This function takes the check_stock array as input.
- Output – This function adds the stock for the stock_item which has check_stock as false as its value and increases the Quantity data member of that stock_item object.
- Process – This function places order for the stock_item which has check_stock as false as its value and then increases the quantity of the stock_item.

## 3.3 Data Dictionary

| DATA MEMBER | MEANING |
|---|---|
| **Patient::Name** | Is to be initialized with the name of the patient. |
| **Patient::Character** | Is to be initialized to 'M' if the patient is male and to 'F' if the patient is female |
| **Patient::Age** | Is to be initialized with the age of the Patient rounded down to the nearest Integer |
| **Test_Report::Patient_Details** | Instance of Patient which will store all the relevant details of the patient |
| **Test_Report::Doctor_Name** | Name of the doctor who prescribed the medical test to the Test_Report::Patient |
| **Test_Report::Test_Information** | Instance of Test which will denote the details concerning the test which the patient is undergoing |

| | |
|---|---|
| **Test_Result::Patient_Result** | Will store the test result of the patient for that medical test |
| **Management::Stock** | Array of instances of Stock_Item denoting denoting various medical lab inventory stocks |
| **Management::Num_Stocks** | Denotes the number of different type of items in the medical lab inventory |
| **Bill::Patient_Name** | Stores the name of the Patient for whom that bill is issued |
| **Bill::Doctor_Name** | Stores the name of the doctor who prescribed the medical test for which the bill is being issued |
| **Bill::Test_Performed** | An instance of the Test class. Comprises of all the details about the medical test for which the bill is being issued |
| **Bill::Date_Of_Test** | Stores the date on which the medical test was conducted |
| **Bill::Unique_id** | Integer denoting the unique id of that bill being issued |
| **Bill::Date_Of_Collection** | Date of the day when the report associated with that bill can be collected |
| **Bill::Time_Of_Collection** | Time of the day when the report associated with that bill can be collected |
| **Test::Test_Name** | Stores the name of the medical test |
| **Test::Test_Charges** | Stores the charge of the conduction of that medical test |
| **Test::Normal_Values** | String denoting the normal values for that particular medical test |
| **Stock_Item::String** | Stores the name of a particular medical stock item |
| **Stock_Item::Quantity** | Stores the number of pieces of that particular medical stock item |
| **Management::Tests** | Array of instances of Test class. Stores all the test which exists in the database as of that moment |
| **Management::Patients** | Array of instances of Patient class which stores the details of the patients who have taken at least one medical test in the lab |
| **Management::Num_Tests** | Integer denoting the total number of medical tests residing in the database at that moment |
| **Management::Num_Patients** | Integer denoting the total number of patients who have underwent at least one medical test in the lab |

# 4 External Interface Requirement

## 4.1 User Interfaces

User Interface is needed for every component of the software. User Interface will be made such that it is easy to understand for a person using for the first time. Most of the screens will be provided with a help menu and the GUI will be made good looking as well as interactive. The exception or error message will be displayed by a standard JMessageDialog box feature of Java. The interaction with the user interface will be made through mouse and keyboard only, no additional interacting devices are required, the rest hardware requirements are specified below.

## 4.2 Hardware Interfaces

- Hardware: Personal Computer
- Internet Connection: Either LAN connection or Wi-Fi connection

## 4.3 Software Interfaces

- Software Required: The programming will be done in Java using Netbeans ide. So, the latest version of JDK (jdk 1.7 or above must be installed). As the software depends heavily on databases, so MySQL is also required to be pre-installed.
- Operation System: WindowsXP or more, Linux, Mac OS.

# 5 Other Requirements

The system should have minimum amount of RAM to run this software (> 64 MB RAM). The system should have connectivity to access databases or should have the database in the same machine. There are no further requirements.