CHAPTER 1
INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The project Health Care Administration Capstone includes registration of patients, storing their details into the system and also computerized overall billing including patient admit room or admit ward charge. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. User can search availability of a doctor and the details of a patient using the id.

The Health Care Administration Capstone can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user friendly. The data are well protected for personal use and makes the data processing very fast. Health Care Administration is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals.

Health Care Administration is designed for multi specialty hospitals to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Health Care Administration that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting in a seamless flow.

Health Care Administration is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Health Care Administration enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes.

1.2 PROBLEM INTRODUCTION

1.2.1 Lack of immediate retrievals

The information is very difficult to retrieve and to find particular information, to find out about the patient’s history the user has to go through various registers. This results in convenience and wastage of time.

1.2.2 Lack of immediate information storage

The information generated by various transactions which takes time and efforts to be stored at right place.

1.2.3 Lack of prompt updating

Various changes to information like patient details or immunization details of child are difficult to make as paper work is involved.

1.2.4 Error prone manual calculation

Manual calculations are error prone and take a lot of time this may result in incorrect information. For example calculation of patient’s bill based on various treatments.

1.2.5 Preparation of accurate and prompt reports

This becomes a difficult task as information is difficult to collect from various register.

1.3 OBJECTIVE

1) Recording information about the Patients.
2) Recording information about the Doctors.
3) Recording information about Nurse and Ward boy.
4) Recording information related to diagnosis of Patients.
5) Generating Bills.

1.4 SCOPE OF THE PROJECT

- Information about Patients is done by just writing the Patients name, age and gender. Whenever the Patient comes up his information is stored freshly.
- Bills are generated by recording price for each facility provided to Patient on a separate sheet and at last they all are summed up.
- Diagnosis information to patients is generally recorded on the document, which contains Patient information. It is destroyed after some time period to decrease the paper load in the office.
- Information about various diseases is not kept as any document. Doctors themselves do this job by remembering various medicines.

1.5 ORGANIZATION OF THE CHAPTER

The report is organized as follows.

CHAPTER 1 describes the **Introduction:**

Overview-Objective of the project and scope of the project.

CHAPTER 2 describes the **Literature Survey:**

Techniques-**survey** of the related work-survey conclusion/ summary.

CHAPTER 3 describes the **System Analysis:**

Existing system-disadvantages-proposed system-advantages.

CHAPTER 4 describes the **System Requirements:**

System hardware and Software requirements.

CHAPTER 5 describes the **System Design:**

Techniques to analyze the requirements for a context.

CHAPTER 6 describes the **Modules:**

Proposed system and their working nature.

CHAPTER 7 describes the **System Testing:**

Testing process used to uncover errors from the system.

CHAPTER 8 describes the **Conclusion and Future Enhancement:**

The project report discussed the problem definition and its scope.

CHAPTER 2
LITERATURE SURVEY

2.1 SURVEY PAPERS (2016)

Management has been defined as the process, comprised of social and technical functions and activities, occurring within organizations for the purpose of accomplishing predetermined objectives through humans and other resources (Longest, Rakich & Darr, 2000). Healthcare quality and patient safety are the common mantra of all primary and secondary health care providers. In hospitals, over the years, a variety of models and schemes for hospital interventions and development have been deployed (Friesner, 2009). Hospital Management System provides the benefits of streamlined operations, enhanced administration & control, superior patient care, strict cost control and improved profitability.

Before computerized Hospital Management System came into practice, it was difficult to keep proper records of the daily activities of hospitals, patient information, maintenance schedule of equipments in the hospital, and how funds are being allocated and used. This resulted in waste of money, time and man power.Hospital Management System is an information management system designed to help manage the various aspects of a hospital (administrative, clinical and financial). It helps in monitoring and controlling the hospital’s daily transactions, as well as the hospital’s performance. It also helps to address the critical requirements of the hospital. Hospital Management System enables access to the right information and automation of complex task, there by allowing staff to spend more time caring for patients. Hospital Management System is custom built to meet the specific requirements of the medium and large size hospitals across the globe.

CHAPTER 3

SYSTEM ANALYSIS

3.1 SYSTEM ANALYSIS

For any project to be successful there is a need for an effective feasibility study. The purpose of feasibility study is not to solve the problem but to determine if the problem is worth solving. There are many feasibility studies to be conducted. But the three main feasibility to be performed are:

- Economical Feasibility
- Technical Feasibility
- Operational Feasibility

3.1.1 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

3.1.3 OPERATIONAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 EXISTING SYSTEM

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread throughout the hospital management infrastructure.

3.2.1 DISADVANTAGES OF EXISTING SYSTEM

- Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost.
- Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores.

3.3 PROPOSED SYSTEM

The Health Care Administration is designed for any hospital to replace their existing manual paper based system. The new system is to control the information of patients. Room availability, staff and operating room schedules and patient invoices. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks .

3.3.1 ADVANTAGES OF PROPOSED SYSTEM

- Computerization
- User Friendly
- Avoiding Redundancy
- Cost Effective and Data Security

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 REQUIREMENT ANALYSIS

In this section, all of the project requirements are involved and its best solution to implement the thesis. And also basic apparatus are here to be Implement for the effective project implementation are here.

4.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

➢ System	:	Intel core i3 Processor
➢ Hard Disk	:	500 GB
➢ Ram	:	2GB

4.3 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

➢ Operating system	:	Windows XP/7/8/10
➢ IDE	:	Net Beans 8.1
➢ Coding Language	:	JAVA
➢ Database	:	MS-Access

4.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations. Technical details, data manipulation and processing and other specific functionalities show how a use case is to be fulfilled. They are supported by non-functional requirements, which impose constrains on the design or implementation (such as performance requirements, security, or reliability).

4.4.1 TECHNICAL ISSUES

Many a software project has failed due to an incomplete or inaccurate a software application. Analysis process, especially technical issues. Technical issues are key step while developing.

4.4.2 RISK ANALYSIS

Project Risk Analysis is for Cost estimates of known accuracy and risk on capital investment projects. Their main challenge is to determine how to model and visualize the complex relationships between risks, define and monitor the risks impacts, analyze the probability of risk occurrence, mitigate the negative impact of risks, and monitor the course of the project with risks and uncertainties.

CHAPTER 5
SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

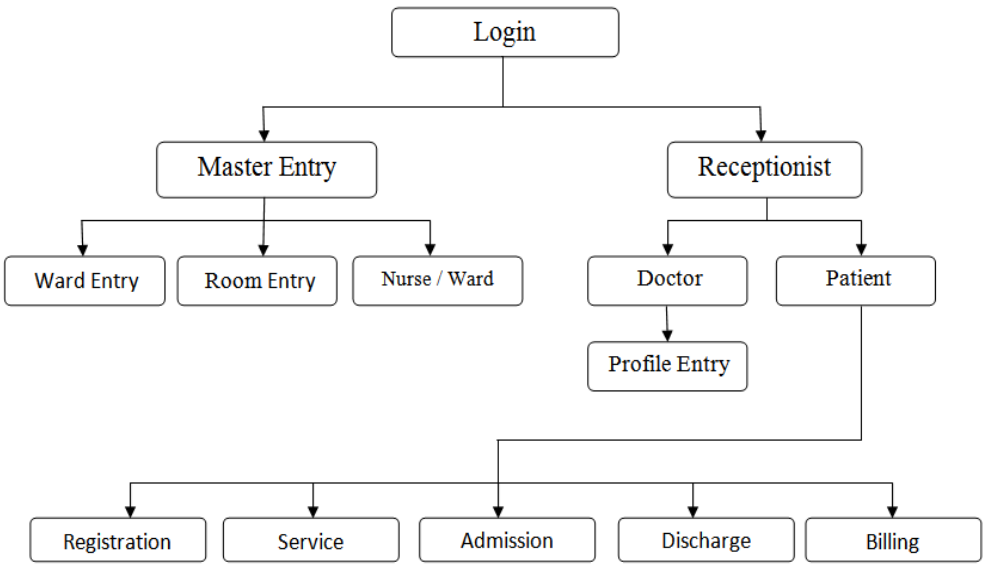

Fig 5.1 Architecture diagram of Health Care Administration capstone

5.2 INTRODUCTION TO UML

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

- Visualizing
- Specifying
- Constructing
- Documenting

5.2.1 VISUALIZING

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize, how the components of the system communicate and interact with each other.

5.2.2 SPECIFYING

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software.

5.2.3 CONSTRUCTING

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

5.2.4 DOCUMENTING

The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, desire, source code, project plans, tests, prototypes releasers, etc...

5.3 UML DIAGRAM

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices and arcs. You draw diagram to visualize a system from different perspective, so a diagram is a projection into a system. For all but most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams, only a few diagrams , or in no diagrams at all. In theory, a diagram may contain any combination of things and relationships. In practice, however, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of a software-intensive system. For this reason,

The UML diagram includes :

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. State chart diagram
7. Activity diagram
8. Component diagram
9. Deployment diagram

5.3.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language(UML) is a type of behavioral diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals(represented as use cases),and any dependencies between those use cases.

Use case diagrams are formally included in two modeling languages defined by the OMG: the unified modeling language(UML) and the systems modeling language(sysML)

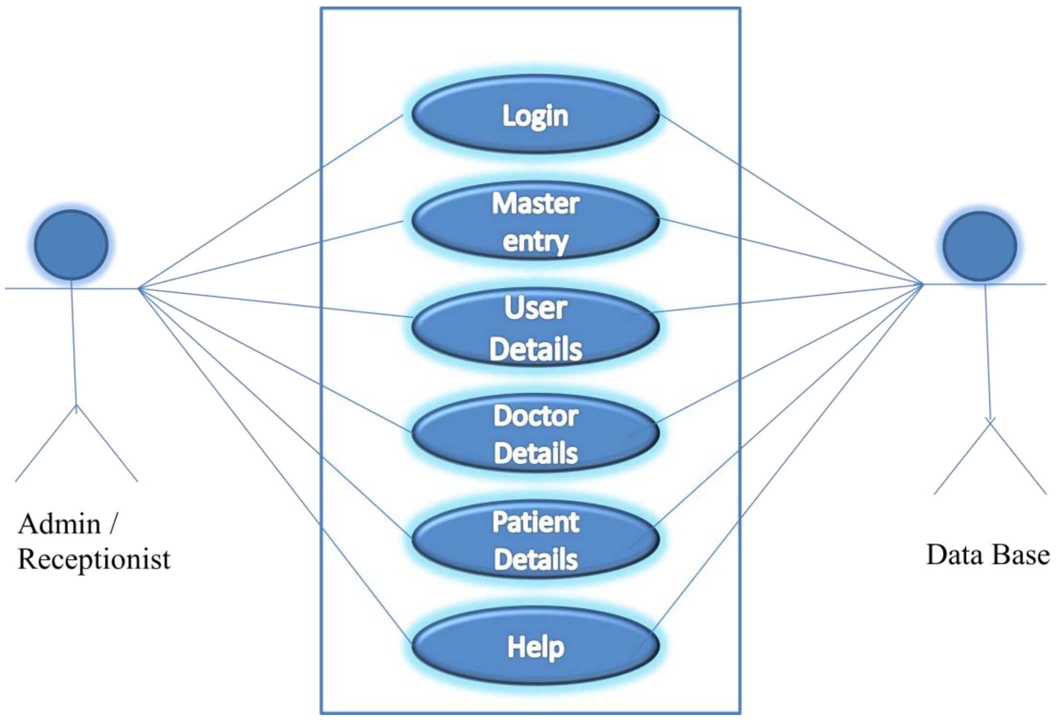

Fig 5.3.1 Use case diagram of Health Care Administration Capstone

5.3.2 CLASS DIAGRAM

A Class is a category or group of things that has similar attributes and common behavior. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle, area contains the attributes and the lowest areas show the operations. Class diagrams provides the representation that developers work from. Class diagrams help on the analysis side, too.

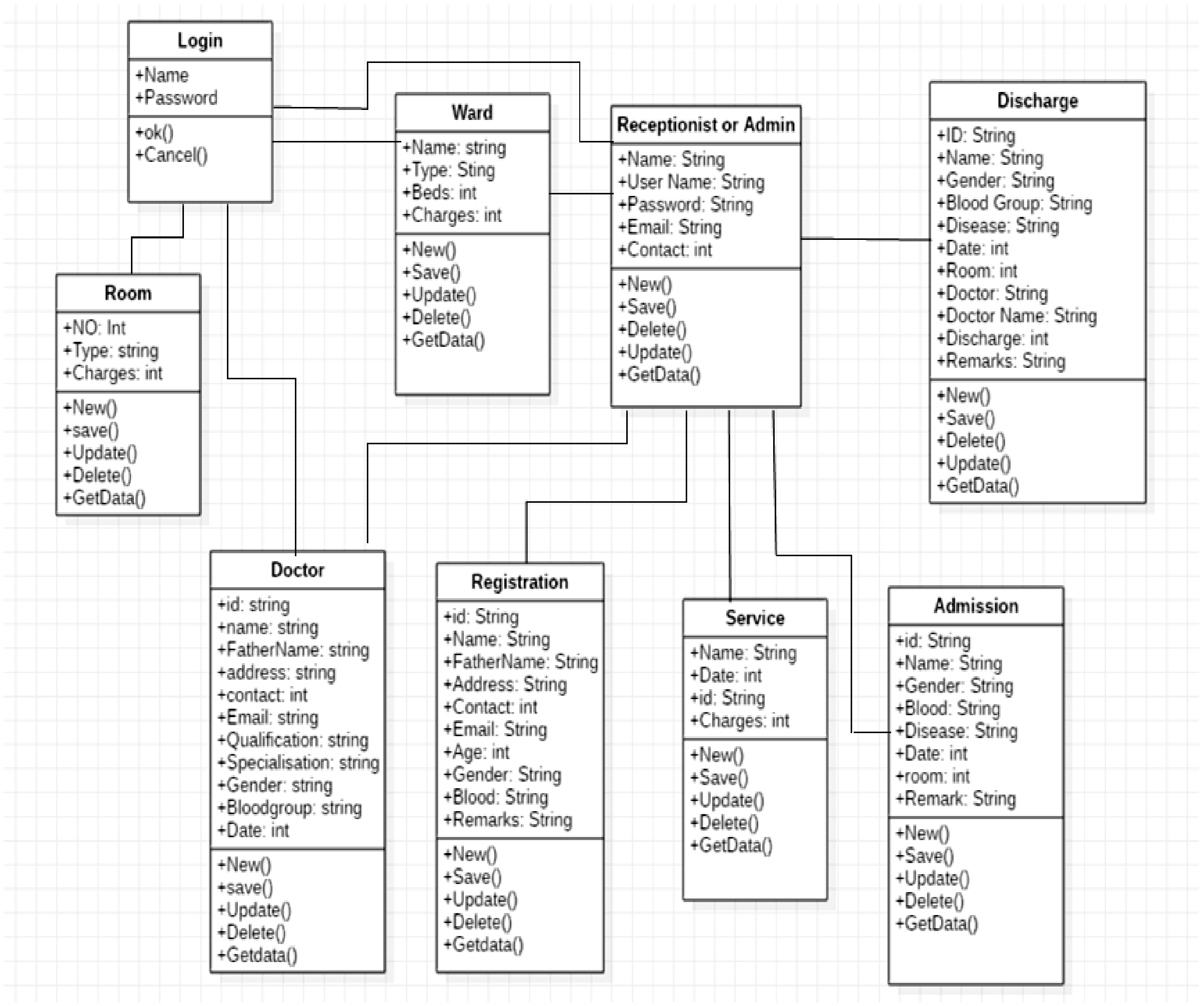

Fig No 5.3.2 Class Diagram of Health Care Administration Capstone

CHAPTER 6
MODULES

6.1 PROPOSED SYSTEM MODULES

1.Login
2.Main Menu
3. Master Entry
- Ward
- Room
- Nurse\Ward Boy

4.Receptionist\Admin
- Registration
- Change Password
- Login Details

5 .Doctor
- Profile Entry

6. Patient
- Registration
- Services
- Admission
- Discharge
- Billing

7.Help

6.1.1 LOGIN

Login is a module that contain user name and password of receptionist or administrator.

6.1.2 MAIN MENU

Main menu displays all the modules in the form of menu items.

6.1.3 MASTER ENTRY

Master Entry module used for selecting and displaying the ward, room and nurse or ward boy details.

6.1.3.1 WARD

Ward module contain operations like selecting new ward, storing , deleting, updating the ward details.

6.1.3.2 ROOM

Room module contain operations like selecting new ward, storing , deleting, updating the ward details.

6.1.3.3 NURSE\WARD BOY

In this module contain detail descriptions about the nurses and ward boys.

6.1.4 RECEPTIONIST\ADMIN

This modules contains the following functionalities.

6.1.4.1 REGISTRATION

This sub module is used for registering the new admin or receptionist..

6.1.4.2 CHANGE PASSWORD

This sub module is used for changing and updating the new password.

6.1.4.3 LOGIN DETAILS

This sub module holds the loin details such as user id and password.

6.1.5 DOCTOR

This module is used for entering the doctor's profiles.

6.1.5.1 PROFILE ENTRY

In this module we can enter the doctor's details such as doctor unique id qualification, specialization etc.,

6.1.6 PATIENT

This module is used for entering the patient details.

6.1.6.1 REGISTRATION

This sub module is used for registering the new patient's details.

6.1.6.2 SERVICES

This module shows the cost of the services provided to the patient's.

6.1.6.3 ADMISSION

This module contains the details about the patient in with date of admission.

6.1.6.4 DISCHARGE

This module contains the details about the patient out with date of discharge.

6.1.6.5 BILLING

This module generates the final billing based on the services provided to the patients.

6.1.7 HELP

This module show the project developers profiles.

CHAPTER 7
SYSTEM TESTING

7.1 INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.2.1 TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

7.2.2 TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

7.2.3 FEATURES TO BE TESTED

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.3 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.4 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.5 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.

Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs must be exercised.
Systems/Procedures	:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.6 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration testing System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.7 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.8 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

Since we are entering details of the patients electronically in the "Health Care Administration Capstone”, data will be secured. Using this application we can retrieve patient’s history with a single click. Thus processing information will be faster. It guarantees accurate maintenance of Patient details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

8.2 FUTURE ENHANCEMENT

We have planned to implement our proposed system for online.

CHAPTER 9

REFERENCES

[1] Hospital Management and Information System. Quintegra Solutions (2006).

[2] Hospital Management Information System, Gujarat Informatics System (2012).

[3] Praveen K.A and Gomes L.A, 2006: "A Study of the Hospital Information System (HIS) In the Medical Records Department of A Tertiary Teaching Hospital" Journal of the Academy of Hospital Administration, Vol. 18, No. 1 (2006-01 - 2006-12).

[4] Smith M. and Gert van der Pijl "Developments in Hospital Management and Information Systems". Tilburg University School of Economics, Netherlands.

[5] Java™ 2: The Complete Reference, Fifth Edition Herbert Schildt McGraw Hill/Osborne New York Chicago San Francisco Lisbon London Madrid Mexico City Milan New Delhi San Juan Seoul Singapore Sydney Toronto

WEBSITES

[1] https://www.healthcareadministrationedu.org/what-is-health-and-medical administration

[2] https://en.wikipedia.org/wiki/Health_administration

[3] https://www.healthcare-management-degree.net/faq/what-is-a-capstone project-and-why-are-they-required-by-most-masters-programs

[4] https://work.chron.com/basic-management-concepts-medical-front-office personnel-21834.html

[5] https://www.codejava.net/java-se/jdbc/java-jdbc-example-connect-to microsoft-access-database

[6] https://www.javatpoint.com/java-swing

[7] http://www.gujaratinformatics.com/hmis.html.

{

APPENDIX [1]

SOURCE CODE

USERS REGISTRATIONS

```
import java.awt.HeadlessException;

import java.awt.event.KeyEvent;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JOptionPane;


public class UsersRegistration extends javax.swing.JFrame

{

Connection con=null;

ResultSet rs=null;

PreparedStatement pst=null;

 public UsersRegistration()

{

 initComponents();

setLocationRelativeTo(null);

}


private void Reset()

{

    txtName.setText("");

    txtUserName.setText("");
```

```
   txtPassword.setText("");
   txtEmailID.setText("");
   txtContactNo.setText("");
   Save.setEnabled(true);
   Delete.setEnabled(false);
   Update.setEnabled(false);
   txtUserName.requestDefaultFocus();

}
private void initComponents()
{
 New = new javax.swing.JButton();
 Save = new javax.swing.JButton();
txtName = new javax.swing.JTextField();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
txtUserName = new javax.swing.JTextField();
txtPassword = new javax.swing.JPasswordField();
txtEmailID = new javax.swing.JTextField();
txtContactNo = new javax.swing.JTextField();
Delete = new javax.swing.JButton();
Update = new javax.swing.JButton();
GetData = new javax.swing.JButton();
jLabel6 = new javax.swing.JLabel();
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("User Registration");
```

```
setResizable(false);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

New.setText("New");

New.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt)

{

NewActionPerformed(evt);

}

}

);

getContentPane().add(New, new org.netbeans.lib.awtextra.AbsoluteConstraints(550,
230, 130, 40));

Save.setText("Save");

Save.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt) {

SaveActionPerformed(evt);

}

}

);

getContentPane().add(Save, new org.netbeans.lib.awtextra.AbsoluteConstraints(550,
280, 130, 45));

getContentPane().add(txtName, new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 230, 260, 30));

jLabel1.setText("Name");

getContentPane().add(jLabel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(210, 240, -1, -1));

jLabel2.setText("User Name");

getContentPane().add(jLabel2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 290, -1, -1));
```

```
jLabel3.setText("Password");

getContentPane().add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 340, -1, 10));

jLabel4.setText("Email ID");

getContentPane().add(jLabel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(210, 390, -1, -1));

jLabel5.setText("Contact No.");

getContentPane().add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 450, -1, -1));

getContentPane().add(txtUserName, new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 280, 260, 30));

txtPassword.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt)

{

txtPasswordActionPerformed(evt);

}

}

);

getContentPane().add(txtPassword, new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 330, 260, 30));

getContentPane().add(txtEmailID, new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 380, 260, 30));

txtContactNo.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyTyped(java.awt.event.KeyEvent evt) {

txtContactNoKeyTyped(evt);

}

}

);

getContentPane().add(txtContactNo, new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 440, 260, 30));

Delete.setText("Delete");
Delete.setEnabled(false);

Delete.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

DeleteActionPerformed(evt);

}

}

);

getContentPane().add(Delete, new
org.netbeans.lib.awtextra.AbsoluteConstraints(550, 340, 130, 40));

Update.setText("Update");

Update.setEnabled(false);

Update.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

UpdateActionPerformed(evt);

}

}

);

getContentPane().add(Update , new
org.netbeans.lib.awtextra.AbsoluteConstraints(550, 390, 130, 40));

GetData.setText("Get Data");

GetData.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt)

{

GetDataActionPerformed(evt);

}

}

);

getContentPane().add(GetData, new
org.netbeans.lib.awtextra.AbsoluteConstraints(550, 440, 130, 45));
```

```
jLabel6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/hospital-
2.jpg"))); // NOI18N

getContentPane().add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(-
100, 0, 800, 520));

pack();

}// </editor-fold>//GEN-END:initComponents

private void NewActionPerformed(java.awt.event.ActionEvent evt)

{

Reset();

}

private void SaveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_SaveActionPerformed

try{

con=Connect.ConnectDB();

if (txtName.getText().equals(""))

{

JOptionPane.showMessageDialog( this, "Please enter name","Error",
JOptionPane.ERROR_MESSAGE);

return;

}

if (txtUserName.getText().equals(""))

{

JOptionPane.showMessageDialog( this, "Please enter user name","Error",
JOptionPane.ERROR_MESSAGE);

return;

}

String Password= String.valueOf(txtPassword.getPassword());

 if (Password.equals(""))

{

JOptionPane.showMessageDialog( this, "Please enter password","Error",
JOptionPane.ERROR_MESSAGE);

 return;
}

if (txtContactNo.getText().equals(""))

{

JOptionPane.showMessageDialog( this, "Please enter contact no.","Error",
JOptionPane.ERROR_MESSAGE);

return;

}

Statement stmt;

stmt= con.createStatement();

String sql1="Select username from Registration where Username= '" +
txtUserName.getText() + "'";

 rs=stmt.executeQuery(sql1);

 if(rs.next())

{

JOptionPane.showMessageDialog( this, "User name already exists","Error",
JOptionPane.ERROR_MESSAGE);

txtUserName.setText("");

txtUserName.requestDefaultFocus();

 return;

}

String Password1= String.valueOf(txtPassword.getPassword());

String sql= "insert into
Registration(username,password,nameofuser,Email,ContactNo)values('"+
txtUserName.getText() + "','" + Password1 + "','" + txtName.getText() + "','" +
txtEmailID.getText() + "','" + txtContactNo.getText() + "')";
pst=con.prepareStatement(sql);

pst.execute();

String sql2= "insert into Users(username,user_password)values('" +
txtUserName.getText() + "','" + Password1 + "')";

 pst=con.prepareStatement(sql2);

 pst.execute();

 JOptionPane.showMessageDialog(this,"Successfully
Registered","User",JOptionPane.INFORMATION_MESSAGE);
```

```
Save.setEnabled(false);

}

catch(HeadlessException | SQLException ex)

{

 JOptionPane.showMessageDialog(this,ex);

}


private void DeleteActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

int P = JOptionPane.showConfirmDialog(null," Are you sure want to delete
?","Confirmation",JOptionPane.YES_NO_OPTION);

if (P==0)

{

con=Connect.ConnectDB();

String sql= "delete from Registration where Username = '" + txtUserName.getText()
+ "'";

 pst=con.prepareStatement(sql);

pst.execute();

String sql1= "delete from Users where Username = '" + txtUserName.getText() + "'";

pst=con.prepareStatement(sql1);

pst.execute();

JOptionPane.showMessageDialog(this,"Successfully
deleted","Record",JOptionPane.INFORMATION_MESSAGE);

Reset();

}

}catch(HeadlessException | SQLException ex)

{

 JOptionPane.showMessageDialog(this,ex);

}
```

```
 }

 private void UpdateActionPerformed(java.awt.event.ActionEvent evt)

{

   try{

 con=Connect.ConnectDB();

String Password1= String.valueOf(txtPassword.getPassword());

String sql= "update Registration set password='" + Password1 + "',nameofuser='" +
txtName.getText() + "',Email='" + txtEmailID.getText() + "',ContactNo='" +
txtContactNo.getText() + "' where Username='" + txtUserName.getText() + "'";

pst=con.prepareStatement(sql);

pst.execute();

String sql2= "update Users set user_password='" + Password1 + "' where username='"
+ txtUserName.getText() + "'";

 pst=con.prepareStatement(sql2);

 pst.execute();

JOptionPane.showMessageDialog(this,"Successfully updated","User
info",JOptionPane.INFORMATION_MESSAGE);

Update.setEnabled(false);

}

catch(HeadlessException | SQLException ex)

{

JOptionPane.showMessageDialog(this,ex);

 }

 }

 private void GetDataActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_GetDataActionPerformed

this.hide();

UsersRegistrationRecord frm = new UsersRegistrationRecord();

frm.setVisible(true);

}

 private void txtContactNoKeyTyped(java.awt.event.KeyEvent evt)
```

```
{
 char c=evt.getKeyChar();
 if (!(Character.isDigit(c)|| (c==
KeyEvent.VK_BACK_SPACE)||(c==KeyEvent.VK_DELETE)))
{
getToolkit().beep();
evt.consume();
}
 private void txtPasswordActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_txtPasswordActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_txtPasswordActionPerformed


    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Metal".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(UsersRegistration.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(UsersRegistration.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(UsersRegistration.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(UsersRegistration.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        }
        //</editor-fold>


        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new UsersRegistration().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    public javax.swing.JButton Delete;
    private javax.swing.JButton GetData;
    private javax.swing.JButton New;
    public javax.swing.JButton Save;
    public javax.swing.JButton Update;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    public javax.swing.JTextField txtContactNo;
    public javax.swing.JTextField txtEmailID;
    public javax.swing.JTextField txtName;
    public javax.swing.JPasswordField txtPassword;
    public javax.swing.JTextField txtUserName;
    }
```

CONNECT

```
import java.sql.*;
import javax.swing.*;
public class Connect {
Connection con=null;
public static Connection ConnectDB(){
 try
{
Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
Connection con = DriverManager.getConnection("jdbc:ucanaccess://C://Users//subu
star//Desktop//Ho//HMS_DB.accdb");
return con;
}
catch(Exception e)
{
JOptionPane.showMessageDialog(null, e);
return null;
}
}
}
```