Lebanese American University

School of Arts and Sciences

Software Engineering

CSC 490

Instructor: Dr. Nashat Mansour

Testing Results & Code
Hospital Management System

Bahaa Thebian (201904844) - Christopher Chalhoub (201804127)
Hassan Abed Ali (201905280) - Moussa Haidous (201902042)

10/05/21

Table of Contents

Abstract:

After drafting both the software requirements specification and the design report, we introduce below our software development & testing results.

We build the project abiding by our code design constraints and abstractions introduced in our design report. We made sure to follow software engineering practices, by first unit testing each class method in isolation, then proceeding with integration testing after we integrated all the methods together in each class, then moved on to component testing and finally, overall testing for all of the subsystems together, of which we'll present the results of below.

During our testing, we covered all possible cases, in addition to edge cases, such as inputting wrong data types, invalid values, and such.

Following our testing results, we present all of our code for each subsystem.

Since we were able to see the code during our testing, it is a white box testing which helped us to discover possible inputs that may cause the code to not work properly.

Testing Results:

Subsystem 1: Reception Subsystem

Registering a Patient

Testing Objective	Input	Output	Error Found	Change Implemented
Testing if input is validated when input is not of appropriate data type	ID = 1 Age = “str” First Name= Hassan Last Name = Abed Ali	Please Enter an integer value in the Age Field	-	-
Testing if the service reject an ID that already exists	ID = 1 Age = 24 First Name= Hassan Last Name = Abed Ali	This ID already exists for other patient, please enter a new unique one	-	-
Testing when all inputs are valid	ID = 2 Age = 24 First Name= Hassan Last Name = Abed Ali	Patient has been successfully registered	-	-

Search for a Patient

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when a patient is in the hospital	First Name = Hassan Last Name = Abed Ali	RoomNumber: 123 Status : good	-	-
Testing when a patient is not in the hospital	First Name = “John” Last Name = “Smith”	Patient not found	-	-

Remove Patient

Testing Objective	Input	Output	Error Found	Change Implemented
Testing if input is validated when data is invalid	ID=”hsn”	Please Enter an integer in the ID Field	-	-
Testing when Patient is present	ID = 1	Patient has been removed successfully	-	-
Testing when there is no patient with that ID	ID = 3	Patient has been Removed Successfully	yes	If the number of rows affected in the database is 0, then the ID did not exist. so we checked the number of affected rows, if it was 0 we print “ This ID does not correspond to a patient in the hospital” Then everything worked fine!

Subsystem 2: Test Management

Adding Test

Testing Objective	Input	Output	Error Found	Change Implemented
Testing if input is validated when it is invalid	Test ID = “pat” Patient ID = 1	Please Enter An Integer in the Test ID field	-	-
Testing when input is valid	Test ID = 10 Patient ID = 1	test has been successfully added	-	-
Testing when input is a Test ID that already exists	Test ID = 10 Patient ID = 2	This Test ID already exists in the database, please enter a new unique one	-	-
Testing when input is a Patient ID that does not exist in the database	Test ID = 11 Patient ID = 3	No Patient with the specified ID exists	-	-

Insert Test Result

Testing Objective	Input	Output	Error Found	Change Implemented
Testing if input is validated when it is invalid	Test ID = “bbb” Result = “Positive”	Please Enter An Integer in the Test ID field	-	-
Testing when input is a Test ID that does not exist in the database	Test ID = 200 Result = “Negative”	No Test with ID = 200 Exists	-	-
Testing when input is a Test ID that exists in the database	Test ID = 10 Result = “Negative”	Test Result was successfully added	-	-

Delete Test

Testing Objective	Input	Output	Error Found	Change Implemented
Testing if input is validated when it is invalid	Test ID = “what”	Please Enter An Integer in the Test ID Field	-	-
Testing when input is a Test ID that does not exist in the database	Test ID = “1000”	No Test with ID = 1000 Exists	-	-
Testing when input is a Test ID that exists in the database	Test ID = 10	test was successfully deleted	-	-

Generate Test Report

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when input is valid (an existing Test ID)	Test ID = 10	Patient ID : 1 Test Name : PCR Result : Negative Date Done : 5/7/2021	-	-
Testing if input is validated when it is invalid	Test ID = “aaa”	Please Enter An integer in the Test ID Field	-	-
Testing when input is a Test ID that does not exist in the database	Test ID = 500	No Test with ID = 500 Exists	-	-

View All Tests

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when no tests are in the system yet	Clicking "View All Tests" Button	No Tests in the system	-	-
Testing when there is tests in the system	Clicking "View All Tests" Button	1) Test ID = 10 Test Name = "PCR" Result = "Negative" Date Done = 5/2/2021 2) Test ID = 15 Test Name = "Blood" Result = "O+" Date Done = 7/3/2021	-	-

Subsystem 3: Inventory Management

Inserting New Product

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when input is valid	Product Name = "AB+ blood" Type = Blood Amount = 10 Price = 5	Product was successfully added	-	-
Testing if input is validated when it is invalid	Product Name = "ventilators" Type = Equipment Amount = "trs" Price = 100	Please Enter An Integer value for Amount field	-	-
Testing if the Amount field is a Negative Number	Product Name = "ventilators" Type = Equipment Amount = - 8 Price = 100	Please Enter a positive value for Amount field	-	-

Update Product Amount

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when input is valid	Product Name = “AB+ blood” Amount = 50 Operation = +	Product Amunt was successfully Updated	-	-
Testing if input is validated when it is invalid	Product Name = “AB+ blood” Amount = 80 Operation = “asdasd”	Please Enter a correct Operation type	-	-
Testing when input in amount is a negative number	Product Name = “AB+ blood” Amount = - 80 Operation = +	Please Enter a positive value for Amount field	-	-
Testing when the input amount makes the available less than 0	Product Name = “AB+ blood” Amount = 80 Operation = -	Deducting 80 makes the available amount less than 0, please lower the amount value to be deducted	-	-

View All Products

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when no products are in the system yet	Clicking “View all Products” Button	There is no products in the system	-	-
Testing when there is products in the system	Clicking “View all Products” Button	1) Product Name : “AB+ blood” Amount : 15 Price : 5 Type : blood 2) Product Name : “ventilators” Amount : 6 Price : 78 Type : Equipment	-	-

Notification for Low Amounts

Testing Objective	Input	Output	Error Found	Change Implemented
Testing when there are no products in the system	Clicking “View Products with Low Amount” Button	There is no products in the system	-	-
Testing when there is a product with amount less than 10	Clicking “View Products with Low Amount” Button	Product Name : “ventilators” Amount : 6 Price : 78 Type : Equipment	-	-
Testing when there is no products with amount less than 10	Clicking “View Products with Low Amount” Button	There is no products in the system with low amount	-	-

Subsystem Code

DatabaseConnection:

```
import java.sql.*;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DatabaseConnection
    {

        private static Connection con = null;
        String user = "Hassan@hospital247";
     String pass = "project";
        public static  DatabaseConnection Instance;
     static
     {
        String url
="jdbc:sqlserver://hassan.database.windows.net:1433;database=hospital
;user=Hassan@hassan;password={project};encrypt=true;trustServerCertif
icate=false;hostNameInCertificate=*.database.windows.net;loginTimeout
=30;";

                        try {
                                con = DriverManager.getConnection(url);
```

```
                        System.out.println("Successefully
connected to the database");
                    } catch (SQLException e) {

                        e.printStackTrace();
                    }


    }

    public static DatabaseConnection GetDatabaseConnection()
    {
       if(Instance==null)
       {
              System.out.println("im here");
              return Instance=new DatabaseConnection();
       }
       else
              return Instance;
    }

    public int Insert(String Query) throws SQLException
    {

        PreparedStatement QueryStatement
            = con.prepareStatement(Query);

        int n = QueryStatement.executeUpdate();
        return n;
    }

    public int Delete(String Query) throws SQLException
    {

        PreparedStatement QueryStatement
        = con.prepareStatement(Query);

        return QueryStatement.executeUpdate();
    }


    public ResultSet Select(String Query) throws SQLException
    {


        PreparedStatement QueryStatement
        = con.prepareStatement(Query);


        ResultSet ResultSet = QueryStatement.executeQuery();

              return ResultSet;
```

```
    }


    public int update(String Query) throws SQLException
    {


       PreparedStatement QueryStatement = con.prepareStatement(Query);
       int n= QueryStatement.executeUpdate();
       return n;
    }



    }
```

Subsystem 1: Reception Subsystem

1) Receptionist

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Receptionist extends JPanel {
    private static JButton PatientRegister;
    private JLabel jcomp2;
    private JButton PatientSearch;
    private JButton PatientRemoval;


    public Receptionist() {


        //construct components
        PatientRegister = new JButton ("Register Patient");
        jcomp2 = new JLabel ("Please Choose one of the following
Operations");
        PatientSearch = new JButton ("Search for a Patient");
        setBackground(Color.CYAN);
        PatientRemoval = new JButton ("Remove Patient");
        jcomp2.setFont(new Font("Arial", Font.PLAIN, 18));
        //adjust size and set layout
        setPreferredSize (new Dimension (667, 366));
        setLayout (null);
```

```
        //add components
        add (PatientRegister);
        add (jcomp2);
        add (PatientSearch);
        add (PatientRemoval);

        //set component bounds (only needed by Absolute Positioning)
        PatientRegister.setBounds (0, 105, 225, 55);
        jcomp2.setBounds (150, 30, 390, 35);
        PatientSearch.setBounds (225, 105, 225, 55);
        PatientRemoval.setBounds (450, 105, 225, 55);

        PatientRegister.setBackground(Color.green);
        PatientSearch.setBackground(Color.green);
        PatientRemoval.setBackground(Color.green);



        PatientRegister.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Register
Patient");
                    d.setSize(700, 400);
                    d.getContentPane().add(new
RegisterPatientService());
                    // set visibility of dialog
                    d.setVisible(true);
                }
              });

        PatientRemoval.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Remove
Patient");
                    d.setSize(700, 400);
                    d.getContentPane().add(new
RemovePatientService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
```

```
            });

        PatientSearch.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Search
Patient");
                    d.setSize(700, 500);
                    d.getContentPane().add(new
PatientSearchService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
            });




        }


    public static void main (String[] args)
    {
       JFrame  frame = new JFrame ("Reception System");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add (new Receptionist());
        frame.pack();
        frame.setVisible (true);

    }
}
```

2) Patient

```
import java.sql.Date;

public class Patient {

        public String FirstName;
        public String LastName;
        public int ID;
        public String DateOfEntry;
        public int PhoneNumber;
        public int EmergencyContactPhone;
        public int Age;
        public String ReasonOfEntry;
```

```
        public String Email;
        public String Gender;


        public Patient(String firstName, String lastName, int iD,
String dateOfEntry, int phoneNumber,
                       int emergencyContactPhone, int age, String
reasonOfEntry, String email, String gender) {
                super();
                FirstName = firstName;
                LastName = lastName;
                ID = iD;
                DateOfEntry = dateOfEntry;
                PhoneNumber = phoneNumber;
                EmergencyContactPhone = emergencyContactPhone;
                Age = age;
                ReasonOfEntry = reasonOfEntry;
                Email = email;
                Gender = gender;
        }
}
```

3) RegisterPatientService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.Date;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class RegisterPatientService extends JPanel {
    private JTextField FirstName;
    private JTextField ID;
    private JTextField EmergencyContactPhone;
    private JTextField Age;
    private JTextField PhoneNumber;
    private JTextField DateOfEntry;
    private JTextField LastName;
    private JTextField Email;
    private JTextField ReasonOfEntry;
    private JTextField Gender;
```

```
    private JLabel jcomp11;
    private JLabel jcomp12;
    private JLabel jcomp13;
    private JLabel jcomp14;
    private JLabel jcomp15;
    private JLabel jcomp16;
    private JLabel jcomp17;
    private JLabel jcomp18;
    private JLabel jcomp19;
    private JLabel jcomp20;

    private JButton Register;
    private JLabel Result;
    public Patient patient;
    JRadioButton Male;
    JRadioButton Female;

    public RegisterPatientService() {
        //construct components
        FirstName = new JTextField (5);
        FirstName.setBackground(Color.green);
             FirstName.setBorder(new LineBorder(Color.BLACK, 2));
        ID = new JTextField (5);
        ID.setBackground(Color.green);
        ID.setBorder(new LineBorder(Color.BLACK, 2));
        EmergencyContactPhone = new JTextField (5);
        EmergencyContactPhone.setBackground(Color.green);
        EmergencyContactPhone.setBorder(new
LineBorder(Color.BLACK, 2));
        Age = new JTextField (5);
        Age.setBackground(Color.green);
        Age.setBorder(new LineBorder(Color.BLACK, 2));
        PhoneNumber = new JTextField (5);
        PhoneNumber.setBackground(Color.green);
        PhoneNumber.setBorder(new LineBorder(Color.BLACK, 2));
        DateOfEntry = new JTextField (5);
        DateOfEntry.setBackground(Color.green);
        DateOfEntry.setBorder(new LineBorder(Color.BLACK, 2));
        LastName = new JTextField (5);
        LastName.setBackground(Color.green);
       LastName.setBorder(new LineBorder(Color.BLACK, 2));
        Email = new JTextField (5);
        Email.setBackground(Color.green);
       Email.setBorder(new LineBorder(Color.BLACK, 2));
       ReasonOfEntry = new JTextField (5);
       ReasonOfEntry.setBackground(Color.green);
       ReasonOfEntry.setBorder(new LineBorder(Color.BLACK, 2));
```

```
        Gender = new JTextField (5);
        Gender.setBackground(Color.green);
       Gender.setBorder(new LineBorder(Color.BLACK, 2));


        jcomp11 = new JLabel ("First Name");
        jcomp12 = new JLabel ("Last Name");
        jcomp13 = new JLabel ("Date of Registration");
        jcomp14 = new JLabel ("ID");
        jcomp15 = new JLabel ("Phone Number");
        jcomp16 = new JLabel ("Reason of Entry");
        jcomp17 = new JLabel ("Email");
        jcomp18 = new JLabel ("Age");
        jcomp19 = new JLabel ("Gender");
        jcomp20 = new JLabel ("Emergency Contact Phone");
        Register = new JButton ("Click Here to Register the
patient");

        Result = new JLabel ("");

        //adjust size and set layout
        setPreferredSize (new Dimension (667, 366));
        setLayout (null);
        setBackground(Color.CYAN);




        JButton b = new JButton("...");
        b.setBounds(430, 100, 50, 30);
             //p.add(text);
             add(b);
             b.setBackground(Color.green);
             JFrame frame = new JFrame ("Add Patient");
             b.addActionListener(new ActionListener() {
                   public void actionPerformed(ActionEvent ae) {
                         DateOfEntry.setText(new
DatePicker(frame).setPickedDate());
                   }
             });


        ButtonGroup buttonGroup = new ButtonGroup();
         Male = new JRadioButton("Male");
        buttonGroup.add(Male);
        add(Male);
```

```
         Female = new JRadioButton("Female");
        buttonGroup.add(Female);
        add(Female);
        Male.setBounds(480, 240, 175, 30);
        Female.setBounds(480, 270, 175, 30);
        Male.setBackground(Color.cyan);
        Female.setBackground(Color.cyan);
        Male.setBorder(new LineBorder(Color.BLACK, 2));
        Female.setBorder(new LineBorder(Color.BLACK, 2));



        //add components
        add (FirstName);
        add (ID);
        add (EmergencyContactPhone);
        add (Age);
        add (PhoneNumber);
        add (DateOfEntry);
        add (LastName);
        add (Email);
        add (ReasonOfEntry);
        add (jcomp11);
        add (jcomp12);
        add (jcomp13);
        add (jcomp14);
        add (jcomp15);
        add (jcomp16);
        add (jcomp17);
        add (jcomp18);
        add (jcomp19);
        add (jcomp20);
        add (Register);
        add (Result);
        Register.setBackground(Color.green);

        //set component bounds (only needed by Absolute
Positioning)
        FirstName.setBounds (130, 50, 175, 30);
        ID.setBounds (130, 100, 175, 30);
        EmergencyContactPhone.setBounds (480, 150, 175, 30);
        Age.setBounds (130, 200, 175, 30);
        PhoneNumber.setBounds (130, 150, 175, 30);
        DateOfEntry.setBounds (480, 100, 175, 30);
        LastName.setBounds (480, 50, 175, 30);
        Email.setBounds (130, 250, 175, 30);
        ReasonOfEntry.setBounds (480, 200, 175, 30);
```

```
        jcomp11.setBounds (15, 50, 100, 25);
        jcomp12.setBounds (350, 50, 100, 25);
        jcomp13.setBounds (310, 100, 100, 25);
        jcomp14.setBounds (15, 100, 100, 25);
        jcomp15.setBounds (15, 150, 100, 25);
        jcomp16.setBounds (350, 200, 100, 25);
        jcomp17.setBounds (15, 250, 100, 25);
        jcomp18.setBounds (15, 200, 100, 25);
        jcomp19.setBounds (350, 250, 100, 25);
        jcomp20.setBounds (320, 150, 165, 25);
        Register.setBounds (180, 300, 300, 65);
        Result.setBounds (245, 320, 155, 35);

        setBackground(Color.CYAN);

        Register.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {
     try {
       RegisterPatient();
      }
catch (ParseException e1) {
            e1.printStackTrace();
      }
            }
        });
    }

    public String ValidateAndAssignValues() throws
ParseException
    {
      String gender="";
       if(Male.isSelected())
                   gender ="Male";
       if(Female.isSelected())
                   gender="Female";

        patient=new Patient(
        FirstName.getText(),
        LastName.getText(),
        Integer.parseInt(ID.getText()),
        DateOfEntry.getText(),
        Integer.parseInt(PhoneNumber.getText()),
        Integer.parseInt(EmergencyContactPhone.getText()),
```

```
        Integer.parseInt(Age.getText()),
        ReasonOfEntry.getText(),
        Email.getText(),
        gender
        );

        return "Information Validated and Assigned";
    }


  public String RegisterPatient() throws ParseException
  {
        try
        {
              ValidateAndAssignValues();
        }

        catch(NumberFormatException exc)
          {
            JOptionPane.showMessageDialog(new JDialog(),
                        "Please enter a number not a string in
the fields where a number is required.",
                        "Number format error",
                        JOptionPane.WARNING_MESSAGE);
          }

        String query = FormulateQuery();
        DatabaseConnection.GetDatabaseConnection();
        try
        {
              DatabaseConnection.Instance.Insert(query);
        }

        catch(SQLException exc)
          {
            JOptionPane.showMessageDialog(new JDialog(),
                        exc.getMessage(),
                        "Database error",
                        JOptionPane.WARNING_MESSAGE);
          }

        JOptionPane.showMessageDialog(new JDialog(),
                "Patient has been successfully registered",
                "Success",
                JOptionPane.WARNING_MESSAGE);
      return null;
```

```
   }
```

```
private String FormulateQuery() {
       return "INSERT INTO PATIENT VALUES("+patient.ID +
",\'"+patient.FirstName +"\',\'" + patient.LastName +"\'," +
"\'"+patient.DateOfEntry
                     +"\',"+
patient.PhoneNumber+","+patient.EmergencyContactPhone+","+patien
t.Age+",\'"+patient.ReasonOfEntry+"\',"+
       "\'"+patient.Email+"\',"+ "\'"+patient.Gender+"\');";
}
    }
```

4) RemovePatientService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class RemovePatientService extends JPanel {
    private JLabel jcomp1;
    private JTextField PatientID;
    private JButton PatientRemove;


    public RemovePatientService() {
        //construct components
        jcomp1 = new JLabel ("Enter the Patient ID you want to remove
: ");
        PatientID = new JTextField (5);
        PatientRemove = new JButton ("Remove Patient");


        //adjust size and set layout
        setPreferredSize (new Dimension (377, 245));
        setLayout (null);
        setBackground(Color.CYAN);
        PatientRemove.setBackground(Color.green);
        PatientID.setBackground(Color.green);
```

```
        PatientID.setBorder(new LineBorder(Color.black,2));


        //add components
        add (jcomp1);
        add (PatientID);
        add (PatientRemove);

        //set component bounds (only needed by Absolute Positioning)
        jcomp1.setBounds (10, 35, 240, 30);
        PatientID.setBounds (255, 40, 100, 25);
        PatientRemove.setBounds (115, 85, 140, 40);
    }

    public void  RemovePatient() throws SQLException
    {
       String query = null;
    try
    {
       query = FormulateQuery();
    }
    catch(NumberFormatException exc)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  "Please enter a number not a string in the fields
where a number is required.",
                  "Number format error",
                  JOptionPane.WARNING_MESSAGE);
    }
    DatabaseConnection.GetDatabaseConnection();
    boolean error=false;
    try
    {
    int RowsAffected = DatabaseConnection.Instance.Delete(query);
    if(RowsAffected==0)
       JOptionPane.showMessageDialog(new JDialog(),
                  "Patient ID does not exist",
                  "Failure",
                  JOptionPane.WARNING_MESSAGE);
    }
    catch(SQLException exc)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  exc.getMessage(),
                  "Database error",
                  JOptionPane.WARNING_MESSAGE);
       error=true;
    }

    if(!error)
    {
       JOptionPane.showMessageDialog(new JDialog(),
```

```
                    "Patient  was successfully deleted",
                    "Success",
                    JOptionPane.INFORMATION_MESSAGE);
    }
    }


    //formulate query required for the remove patient service
       private String FormulateQuery() {

              return "Delete FROM Patient WHERE
ID="+Integer.parseInt(PatientID.getText())+";";
       }
}
```

5) PatientSearchService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class PatientSearchService extends JPanel {
    private JTextField FirstName;
    private JTextField LastName;
    private JLabel jcomp3;
    private JLabel jcomp4;
    private JLabel jcomp5;
    private JLabel jcomp6;
    private JLabel jcomp7;
    private JLabel jcomp8;
    private JLabel jcomp9;
    private JLabel jcomp10;
    private JLabel jcomp11;
    private JLabel jcomp12;
    private JLabel jcomp13;
    private JButton Search;
    private JLabel jcomp15;
    private JLabel ID;
    private JLabel PhoneNumber;
    private JLabel Age;
    private JLabel Email;
    private JLabel Gender;
```

```
    private JLabel Profession;
    private JLabel EmergencyContactPhone;
    private JLabel DateOfEntry;
    private JLabel jcomp24;
    private JLabel jcomp25;
    private JLabel Present;
    private JLabel RoomNumber;

    ResultSet result;

    public PatientSearchService() {
        //construct components
        FirstName = new JTextField (5);
        FirstName.setBackground(Color.green);
        FirstName.setBorder(new LineBorder(Color.BLACK, 2));
        LastName = new JTextField (5);
        LastName.setBackground(Color.green);
        LastName.setBorder(new LineBorder(Color.BLACK, 2));
        jcomp3 = new JLabel ("First Name");
        jcomp4 = new JLabel ("Last Name");
        jcomp5 = new JLabel ("Date of Entry :");
        jcomp6 = new JLabel ("ID");
        jcomp7 = new JLabel ("Phone Number : ");
        jcomp8 = new JLabel ("Reason of Entry :");
        jcomp9 = new JLabel ("Email :");
        jcomp10 = new JLabel ("Age :");
        jcomp11 = new JLabel ("Gender :");
        jcomp12 = new JLabel ("Emergency Contact Phone :");
        jcomp13 = new JLabel ("Enter the Patient First Name and
Last Name to Search for");
        Search = new JButton ("Search");
        Search.setBackground(Color.green);
        jcomp15 = new JLabel ("ID :");
        ID = new JLabel ("--------------------");
        PhoneNumber = new JLabel ("--------------------");
        Age = new JLabel ("--------------------");
        Email = new JLabel ("--------------------");
        Gender = new JLabel ("--------------------");
        Profession = new JLabel ("--------------------");
        EmergencyContactPhone = new JLabel
("--------------------");
        DateOfEntry = new JLabel ("--------------------");
        jcomp24 = new JLabel ("Status : ");
        jcomp25 = new JLabel ("Room Number : ");
        Present = new JLabel ("--------------------");
        RoomNumber = new JLabel ("--------------------");
```

```
        //adjust size and set layout
        setPreferredSize (new Dimension (667, 459));
        setLayout (null);
        setBackground(Color.CYAN);
        //add components
        add (FirstName);
        add (LastName);
        add (jcomp3);
        add (jcomp4);
        add (jcomp5);
        add (jcomp6);
        add (jcomp7);
        add (jcomp8);
        add (jcomp9);
        add (jcomp10);
        add (jcomp11);
        add (jcomp12);
        add (jcomp13);
        add (Search);
        add (jcomp15);
        add (ID);
        add (PhoneNumber);
        add (Age);
        add (Email);
        add (Gender);
        add (Profession);
        add (EmergencyContactPhone);
        add (DateOfEntry);
        add (jcomp24);
        add (jcomp25);
        add (Present);
        add (RoomNumber);

        //set component bounds (only needed by Absolute
Positioning)
        FirstName.setBounds (110, 50, 160, 30);
        LastName.setBounds (430, 50, 175, 30);
        jcomp3.setBounds (30, 50, 100, 25);
        jcomp4.setBounds (310, 50, 100, 25);
        jcomp5.setBounds (375, 230, 100, 25);
        jcomp6.setBounds (-20, 105, 100, 25);
        jcomp7.setBounds (25, 290, 100, 25);
        jcomp8.setBounds (375, 400, 100, 25);
        jcomp9.setBounds (30, 400, 50, 25);
        jcomp10.setBounds (30, 345, 45, 25);
        jcomp11.setBounds (375, 345, 55, 30);
```

```
        jcomp12.setBounds (375, 290, 165, 25);
        jcomp13.setBounds (165, 10, 345, 25);
        Search.setBounds (0, 100, 675, 45);
        jcomp15.setBounds (25, 240, 35, 25);
        ID.setBounds (75, 240, 100, 25);
        PhoneNumber.setBounds (135, 290, 100, 25);
        Age.setBounds (80, 345, 100, 25);
        Email.setBounds (80, 400, 100, 25);
        Gender.setBounds (430, 350, 100, 25);
        Profession.setBounds (480, 400, 100, 25);
        EmergencyContactPhone.setBounds (540, 290, 100, 25);
        DateOfEntry.setBounds (465, 230, 100, 25);
        jcomp24.setBounds (20, 185, 60, 25);
        jcomp25.setBounds (375, 180, 100, 25);
        Present.setBounds (85, 185, 100, 25);
        RoomNumber.setBounds (470, 180, 100, 25);

        Search.addActionListener(new ActionListener()
             {

                   @Override
                   public void actionPerformed(ActionEvent arg0)
{
                         try {
                               SearchForaPatient();
                         } catch (SQLException e) {
                               // TODO Auto-generated catch
block
                               e.printStackTrace();
                         }
                   }

             }
);
    }

    public void PresentResults() throws SQLException{
      if(result.next() == false)
      {
            // no patient with the specified name was found
            JOptionPane.showMessageDialog(new JDialog(),
                      "Patient is not in the hospital",
                      "Result",
                      JOptionPane.WARNING_MESSAGE);
      }
      else
      {
```

```
        ID.setText(String.valueOf(result.getInt("ID")));
        PhoneNumber.setText(result.getString("PhoneNumber"));

EmergencyContactPhone.setText(result.getString("EmergencyContact
Phone"));
        DateOfEntry.setText(result.getString("DateofEntry"));
        Age.setText(String.valueOf(result.getInt("Age")));
        Email.setText(result.getString("Email"));

RoomNumber.setText(String.valueOf(result.getInt("RoomNumber")));
        //ReasonOfEntry.setText();
        Gender.setText(result.getString("Gender"));

        }
        }

    public String FormulateQuery()
    {
   return " Select * FROM PATIENT WHERE FIRSTNAME =
"+"\'"+FirstName.getText()+"\'"+" AND LASTNAME =
"+"\'"+LastName.getText()+"\';";
    }



    public void SearchForaPatient() throws SQLException
    {
    String query = FormulateQuery();  //generate the query
required

    DatabaseConnection.GetDatabaseConnection();
    boolean error=false;
    try
    {
    result = DatabaseConnection.Instance.Select(query);
    }
    catch(SQLException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                exc.getMessage(),
                "Database error",
                JOptionPane.WARNING_MESSAGE);
      error=true;
    }
    PresentResults();      //implemented above

    }
```

```
}
```

Subsystem 2: Test Management Subsystem

1) Lab Technician :

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class LabTechnician extends JPanel {
    private static JButton AddTest;
    private JLabel jcomp2;
    private JButton DeleteTest;
    private JButton ViewTestResult;
    private JButton InsertResultForTest;
    private JButton ViewAllTests;


    public LabTechnician() {


        //construct components
       AddTest = new JButton ("Add Test");
        jcomp2 = new JLabel ("Please Choose one of the following
Operations");
        DeleteTest = new JButton ("Delete Test");
        setBackground(Color.CYAN);
        InsertResultForTest = new JButton ("Insert Result For a
Test");
        ViewAllTests = new JButton ("View All Tests results that are
available");
        ViewTestResult = new JButton ("Generate Test Report");

        //adjust size and set layout
        setPreferredSize (new Dimension (667, 366));
        setLayout (null);

        //add components
        add (AddTest);
        add (jcomp2);
        add (DeleteTest);
        add (ViewTestResult);
        add (InsertResultForTest);
        add (ViewAllTests);
```

```
        jcomp2.setFont(new Font("Arial", Font.PLAIN, 18));

        //set component bounds (only needed by Absolute Positioning)
        AddTest.setBounds (0, 105, 225, 55);
        jcomp2.setBounds (150, 30, 390, 35);
        DeleteTest.setBounds (225, 105, 225, 55);
        ViewTestResult.setBounds (450, 105, 225, 55);

        InsertResultForTest.setBounds (0, 160, 340, 55);
        ViewAllTests.setBounds (340, 160, 340, 55);
        AddTest.setBackground(Color.green);
        DeleteTest.setBackground(Color.green);
        ViewTestResult.setBackground(Color.green);
        InsertResultForTest.setBackground(Color.green);
        ViewAllTests.setBackground(Color.green);

        AddTest.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Add Test");
                    d.setSize(700, 350);
                    d.getContentPane().add(new AddTestService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
              });

        DeleteTest.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Delete
Test");
                    d.setSize(700, 350);
                    d.getContentPane().add(new DeleteTestService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
              });

        InsertResultForTest.addActionListener(new ActionListener()
        {
            @Override
```

```
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Insert Test
Result");
                    d.setSize(700, 350);
                    d.getContentPane().add(new
InsertTestResultService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
              });

        ViewTestResult.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Generate
Test Report");
                    d.setSize(700, 350);
                    d.getContentPane().add(new
GenerateTestReportService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
              });

        ViewAllTests.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "View All
Tests");
                    d.setSize(700, 350);
                    d.getContentPane().add(new ViewAllTests());
                    // set visibility of dialog
                    d.setVisible(true);

                }
              });
```

```
        }

    public static void main (String[] args)
    {
       JFrame  frame = new JFrame ("Test Management Sub-System");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add (new LabTechnician());
        frame.pack();
        frame.setVisible (true);

    }
}
```

2) AddTestService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

import org.jdatepicker.impl.JDatePanelImpl;
import org.jdatepicker.impl.JDatePickerImpl;
import org.jdatepicker.impl.UtilDateModel;

public class AddTestService extends JPanel {
    private JLabel jcomp1;
    private JLabel jcomp2;
    private JLabel jcomp3;
    private JLabel jcomp4;

    private JTextField TestID;
    private JTextField PatientID;
    private JTextField TestName;
    private JTextField DateDone;

    private JButton AddTest;

    public Test test;
    public AddTestService() {
        //construct components
```

```
jcomp1 = new JLabel ("Test ID : ");
jcomp2 = new JLabel ("Patient ID :");
jcomp3 = new JLabel ("Test Name :");
jcomp4 = new JLabel ("Date Done : ");


TestID = new JTextField (5);
PatientID = new JTextField (5);
TestName = new JTextField (5);
DateDone = new JTextField (5);


TestID.setBackground(Color.green);
TestID.setBorder(new LineBorder(Color.BLACK, 2));
PatientID = new JTextField (5);
PatientID.setBackground(Color.green);
PatientID.setBorder(new LineBorder(Color.BLACK, 2));
TestName.setBackground(Color.green);
TestName.setBorder(new LineBorder(Color.BLACK, 2));
DateDone.setBackground(Color.green);
DateDone.setBorder(new LineBorder(Color.BLACK, 2));


AddTest = new JButton ("Add Test");
AddTest.setBackground(Color.green);

//adjust size and set layout
setPreferredSize (new Dimension (667, 258));
setLayout (null);
setBackground(Color.CYAN);

//add components
add (jcomp1);
add (jcomp2);
add (jcomp3);
add (jcomp4);
add (TestID);
add (PatientID);
add (TestName);
add (DateDone);
add (AddTest);



jcomp1.setBounds (50, 55, 100, 25);
jcomp2.setBounds (380, 55, 100, 25);
jcomp3.setBounds (50, 125, 100, 25);
```

```
        jcomp4.setBounds (380, 125, 100, 25);
        TestID.setBounds (130, 55, 125, 30);
        PatientID.setBounds (455, 55, 130, 30);
        TestName.setBounds (130, 125, 125, 30);
        DateDone.setBounds (455, 125, 130, 30);
        AddTest.setBounds (260, 180, 145, 45);


        JButton b = new JButton("...");
        b.setBounds(570, 125, 50, 30);
             //p.add(text);
             add(b);
             b.setBackground(Color.green);
             JFrame frame = new JFrame ("Add a new Test");
             b.addActionListener(new ActionListener() {
                   public void actionPerformed(ActionEvent ae) {
                         DateDone.setText(new
DatePicker(frame).setPickedDate());
                   }
             });

             AddTest.addActionListener(new ActionListener()
                         {

                               @Override
                               public void
actionPerformed(ActionEvent arg0)
                               {
                                     try
                                     {
                                           AddTest();
                                     } catch (SQLException e)
                                     {
                                           // TODO
Auto-generated catch block
                                           e.printStackTrace();
                                     }
                               }

                         }
             );

    }

    public void ValidateAndAssignValues()
    {
      test= new Test(Integer.parseInt(TestID.getText()),
```

```
                    Integer.parseInt(PatientID.getText()),
                    TestName.getText(),
                    DateDone.getText(),
                    "");
    }


   public String FormulateQuery()
   {
   return "INSERT INTO TEST VALUES ( "+test.ID+","
+test.PatientID+"," +"\'"+test.Name+"\',"+
"\'"+test.DateDone+"\',"+ "\'No Result Yet\'"+");";
   }


   public String  AddTest() throws SQLException
    {
    try
    {
    ValidateAndAssignValues();
    }

    catch(NumberFormatException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                "Please enter a number not a string in the
fields where a number is required.",
                "Number format error",
                JOptionPane.WARNING_MESSAGE);
    }

    String query = FormulateQuery();
    boolean error=false;
    DatabaseConnection.GetDatabaseConnection();
    try
    {
    DatabaseConnection.Instance.Insert(query);
    }
    catch(SQLException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                exc.getMessage(),
                "Database error",
                JOptionPane.WARNING_MESSAGE);
      error=true;
    }
    if(!error)
```

```
    {
    JOptionPane.showMessageDialog(new JDialog(),
                "Test was successfully added",
                "Success",
                JOptionPane.INFORMATION_MESSAGE);
    }
    return null;
    }
}
```

3) InsertTestResultService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class InsertTestResultService extends JPanel {
    private JButton SubmitTestResult;
    private JLabel jcomp2;
    private JLabel jcomp3;
    private JTextField TestID;
    private JTextField Result;
    private JLabel jcomp6;
    private JTextField DoctorID;

    public InsertTestResultService() {
        //construct components
        SubmitTestResult = new JButton ("Submit");
        jcomp2 = new JLabel ("Test ID :");
        jcomp3 = new JLabel ("Result : ");
        TestID = new JTextField (5);
        Result = new JTextField (5);
        jcomp6 = new JLabel ("Lab technician ID :");
        DoctorID = new JTextField (5);

        //adjust size and set layout
        setPreferredSize (new Dimension (667, 236));
        setLayout (null);
        setBackground(Color.CYAN);


        TestID.setBackground(Color.green);
        TestID.setBorder(new LineBorder(Color.BLACK, 2));
        Result.setBackground(Color.green);
        Result.setBorder(new LineBorder(Color.BLACK, 2));
```

```
        DoctorID.setBackground(Color.green);
        DoctorID.setBorder(new LineBorder(Color.BLACK, 2));
        SubmitTestResult.setBackground(Color.green);


        //add components
        add (SubmitTestResult);
        add (jcomp2);
        add (jcomp3);
        add (TestID);
        add (Result);
        add (jcomp6);
        add (DoctorID);

        //set component bounds (only needed by Absolute Positioning)
        SubmitTestResult.setBounds (215, 150, 215, 50);
        jcomp2.setBounds (30, 15, 55, 25);
        jcomp3.setBounds (30, 100, 55, 25);
        TestID.setBounds (130, 15, 120, 30);
        Result.setBounds (130, 100, 500, 30);
        jcomp6.setBounds (20, 55, 125, 25);
        DoctorID.setBounds (130, 55, 120, 30);

        SubmitTestResult.addActionListener(new ActionListener()
              {

                    @Override
                    public void actionPerformed(ActionEvent arg0) {
                          try {
                                InsertTestResult();
                          } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                          }
                    }

              }
);

}
```

```
    public String  InsertTestResult() throws SQLException
    {
       boolean ErrorHappened=false;
       String query = null;
    try
    {
```

```
            query = FormulateQuery();
            System.out.println(query);
        }

        catch(NumberFormatException exc)
        {
            JOptionPane.showMessageDialog(new JDialog(),
                        "Please enter a number not a string in the fields
where a number is required.",
                        "Number format error",
                        JOptionPane.WARNING_MESSAGE);
        }
        DatabaseConnection.GetDatabaseConnection();
        try
        {
        int RowsAffected=DatabaseConnection.Instance.update(query);
        if(RowsAffected==0)
        {
            ErrorHappened=true;
            JOptionPane.showMessageDialog(new JDialog(),
                        "Test ID does not exist",
                        "Database error",
                        JOptionPane.WARNING_MESSAGE);
        }
        }
        catch(SQLException exc)
        {
            JOptionPane.showMessageDialog(new JDialog(),
                        exc.getMessage(),
                        "Database error",
                        JOptionPane.WARNING_MESSAGE);
            ErrorHappened=true;
        }
        if(!ErrorHappened)
        {
            JOptionPane.showMessageDialog(new JDialog(),
                        "Test Result was successfully added",
                        "Success",
                        JOptionPane.INFORMATION_MESSAGE);

             return "Success";
        }
        return null;
        }



            private String FormulateQuery() {
```

```
            return "UPDATE TEST SET RESULT =
"+"\'"+Result.getText()+"\'"+" WHERE
ID="+Integer.parseInt(TestID.getText())+";";

        }

}
```

4) GenerateTestReportService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class GenerateTestReportService extends JPanel {
    private JLabel jcomp1;
    private JLabel jcomp2;
    private JLabel jcomp3;
    private JLabel jcomp4;
    private JLabel jcomp5;
    private JLabel jcomp6;
    private JTextField ID;
    private JLabel TestID;
    private JLabel TestName;
    private JLabel Result;
    private JLabel DateDone;
    private JLabel PatientID;
    private JButton PrintTestInfo;
    private JButton GenerateReport;



    ResultSet result;
    public Test test;


    public GenerateTestReportService() {
        //construct components
        jcomp1 = new JLabel ("Test ID : ");
        jcomp2 = new JLabel ("Patient ID :");
```

```
        jcomp3 = new JLabel ("Test Name :");
        jcomp4 = new JLabel ("Date Done : ");
        jcomp5 = new JLabel ("Result :");
        jcomp6 = new JLabel ("Enter the Test ID");
        ID = new JTextField (5);
        ID.setBorder(new LineBorder(Color.BLACK, 2));
        ID.setBackground(Color.green);
        TestID = new JLabel ("------------------------");
        TestName = new JLabel ("------------------------");
        Result = new JLabel
("-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-----------------------------------------------------------");
        DateDone = new JLabel ("------------------------");
        PatientID = new JLabel ("------------------------");
        PrintTestInfo = new JButton ("Print Info");
        GenerateReport = new JButton ("Generate Report");


        PrintTestInfo.setBackground(Color.green);
        setBackground(Color.CYAN);
        GenerateReport.setBackground(Color.green);

        //adjust size and set layout
        setPreferredSize (new Dimension (667, 366));
        setLayout (null);

        //add components
        add (jcomp1);
        add (jcomp2);
        add (jcomp3);
        add (jcomp4);
        add (jcomp5);
        add (jcomp6);
        add (ID);
        add (TestID);
        add (TestName);
        add (Result);
        add (DateDone);
        add (PatientID);
        add (PrintTestInfo);
        add (GenerateReport);

        //set component bounds (only needed by Absolute
Positioning)
        jcomp1.setBounds (40, 101, 100, 25);
```

```
        jcomp2.setBounds (360, 100, 100, 25);
        jcomp3.setBounds (40, 190, 100, 25);
        jcomp4.setBounds (355, 200, 100, 25);
        jcomp5.setBounds (40, 275, 50, 25);
        jcomp6.setBounds (145, 25, 125, 40);
        ID.setBounds (270, 30, 170, 30);
        TestID.setBounds (105, 100, 100, 25);
        TestName.setBounds (120, 190, 100, 25);
        Result.setBounds (90, 275, 540, 25);
        DateDone.setBounds (430, 200, 100, 25);
        PatientID.setBounds (425, 100, 100, 25);
        PrintTestInfo.setBounds (255, 310, 185, 35);
        GenerateReport.setBounds(450, 30 , 185, 35);
        PatientID.setText(String.valueOf(123));

        GenerateReport.addActionListener(new ActionListener()
             {

                    @Override
                    public void actionPerformed(ActionEvent arg0)
{

                          try {
                                RetrieveResults();
                          } catch (ParseException e) {
                                // TODO Auto-generated catch
block
                                e.printStackTrace();
                          } catch (SQLException e) {
                                // TODO Auto-generated catch
block
                                e.printStackTrace();
                          }
                    }

             }
);

}
```

```
    public String FormulateQuery() throws ParseException
    {
```

```
    return "SELECT * FROM TEST WHERE ID =
"+Integer.parseInt(ID.getText())+";";
    }


    public void PresentResults() throws SQLException
    {
      if(result.next()==false)
      {
            JOptionPane.showMessageDialog(new JDialog(),
                        "No Test with the specified ID is Found",
                        "Database error",
                        JOptionPane.WARNING_MESSAGE);
      }
      else
      {
            TestID.setText(String.valueOf(result.getInt("ID")));

PatientID.setText(String.valueOf(result.getInt("PatientID")));
          TestName.setText(result.getString("Name"));
          DateDone.setText(result.getString("DateDone"));
          Result.setText(result.getString("Result"));
      }
    }


    public String RetrieveResults() throws ParseException,
SQLException
    {

    String query=null;
    try
      {
     query = FormulateQuery();
      }

    catch(NumberFormatException exc)
      {
            JOptionPane.showMessageDialog(new JDialog(),
                        "Please enter a number not a string in
the fields where a number is required.",
                        "Number format error",
                        JOptionPane.WARNING_MESSAGE);
      }
    boolean error=false;
    DatabaseConnection.GetDatabaseConnection();
    try
```

```
    {
            result = DatabaseConnection.Instance.Select(query);
       }
    catch (SQLException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                exc.getMessage(),
                "Database error",
                JOptionPane.WARNING_MESSAGE);
       }
    PresentResults();
    return "Test information was successfully retrieved";
    }



}
```

5) DeleteTestService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class DeleteTestService extends JPanel {
    private JLabel jcomp1;
    private JTextField TestID;
    private JButton TestDelete;


    public DeleteTestService() {
        //construct components
        jcomp1 = new JLabel ("Enter the Test ID you want to Delete :
");
        TestID = new JTextField (5);
        TestDelete = new JButton ("Delete Test");


        //adjust size and set layout
        setPreferredSize (new Dimension (377, 245));
        setLayout (null);
```

```
        TestID.setBackground(Color.green);
        TestID.setBorder(new LineBorder(Color.black,2));
        TestDelete.setBackground(Color.green);
        setBackground(Color.CYAN);
        //add components
        add (jcomp1);
        add (TestID);
        add (TestDelete);

        //set component bounds (only needed by Absolute Positioning)
        jcomp1.setBounds (10, 35, 240, 30);
        TestID.setBounds (255, 40, 100, 25);
        TestDelete.setBounds (115, 85, 140, 40);

        TestDelete.addActionListener(new ActionListener()
              {

                    @Override
                    public void actionPerformed(ActionEvent arg0)
                    {
                          try
                          {
                                DeleteTest();
                          } catch (SQLException e)
                          {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                          }
                    }

              }
);
    }


    public String  DeleteTest() throws SQLException
    {
       String query = null;
    try
    {
       query = FormulateQuery();
    }

    catch(NumberFormatException exc)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  "Please enter a number not a string in the fields
where a number is required.",
                  "Number format error",
                  JOptionPane.WARNING_MESSAGE);
```

```
    }

    DatabaseConnection.GetDatabaseConnection();

    boolean error = false;

    try{
       int RowsAffected=DatabaseConnection.Instance.Delete(query);
        if(RowsAffected==0)
        {
              error=true;
              JOptionPane.showMessageDialog(new JDialog(),
                         "Test ID does not exist",
                         "Database error",
                         JOptionPane.WARNING_MESSAGE);
        }
    }
    catch(SQLException exc)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  exc.getMessage(),
                  "Database error",
                  JOptionPane.WARNING_MESSAGE);
    }
    if(!error)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  "Test  was successfully deleted",
                  "Success",
                  JOptionPane.INFORMATION_MESSAGE);


    }
    return null;
    }



       private String FormulateQuery() {

              return "Delete FROM TEST WHERE
ID="+Integer.parseInt(TestID.getText())+";";
       }

}
```

6) ViewAllTests

```
import java.awt.Dimension;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class ViewAllTests extends JPanel {



	ResultSet result;
	String data;
	JTextArea table;

	public ViewAllTests()
	{
		setPreferredSize (new Dimension (667, 366));
	    setLayout (null);
	    table = new JTextArea("");

	    //add components
	    add (table);
	    table.setBounds (0, 0, 600, 350);
	    try {
			RetrieveResults();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	    table.setText(data);
	}




 public static void main (String[] args) {
     JFrame frame = new JFrame ("All Tests");
     frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
     frame.getContentPane().add (new ViewAllTests());
     frame.pack();
     frame.setVisible (true);

 }

 public String FormulateQuery()
```

```
 {
 return "SELECT * FROM TEST;";
 }


  public void PresentResults() throws SQLException
       {
         data="";
       while(result.next())
       {
       data = data + "\t" + result.getInt("ID") + "           " +
result.getInt("PatientID") + "           " + result.getString("Name")
+
                     "           " + result.getString("DateDone") + "
" + result.getString("Result") + "\n" ;
       }
       table.setText(data);
       }


  public String RetrieveResults() throws SQLException
  {


   String query = FormulateQuery();


  DatabaseConnection.GetDatabaseConnection();
  try
  {
             result = DatabaseConnection.Instance.Select(query);
  }
  catch (SQLException exc)
  {
       JOptionPane.showMessageDialog(new JDialog(),
                 exc.getMessage(),
                 "Database error",
                 JOptionPane.WARNING_MESSAGE);
       }
  PresentResults();
  return "All Tests information was successfully retrieved";
  }


}
```

7) Test

```
public class Test {
```

```
        public int ID;
        public int PatientID;
        public String Name;
        public String DateDone;
        public String Result;


        public Test(int Id, int patientID, String name, String
dateDone, String result) {
                super();
                ID=Id;
                PatientID = patientID;
                Name = name;
                DateDone = dateDone;
                Result = result;
        }
}
```

Subsystem 3: Inventory Management Subsystem:

1) InsertProductService

```
//Generated by GuiGenie - Copyright (c) 2004 Mario Awad.
//Home Page http://guigenie.cjb.net - Check often for new
versions!

import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;
import java.text.ParseException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class InsertProductService extends JPanel {
    private JLabel jcomp1;
    private JLabel jcomp2;
    private JLabel jcomp3;
    private JLabel jcomp4;
    private JTextField ProductName;
    private JTextField Description;
    private JTextField Amount;
    private JTextField Price;
    private JTextField RoomNumber;
    private JLabel jcomp10;
    private JButton Submit;
    private JLabel jcomp12;
    private JRadioButton Medicine;
    private JRadioButton Equipment;
    private JRadioButton Blood;

    Product product;

    public InsertProductService() {
        //construct components
        jcomp1 = new JLabel ("Name :");
        jcomp2 = new JLabel ("Amount :");
        jcomp3 = new JLabel ("Price :");
        jcomp4 = new JLabel ("Room Number :");
        ProductName = new JTextField (5);
```

```
        Description = new JTextField (5);
        Amount = new JTextField (5);
        Price = new JTextField (5);
        RoomNumber = new JTextField (5);
        jcomp10 = new JLabel ("Description :");
        Submit = new JButton ("Submit");
        jcomp12 = new JLabel ("Please Choose the Product Type");
        Medicine = new JRadioButton ("Medicine");
        Equipment = new JRadioButton ("Equipment");
        Blood = new JRadioButton ("Blood");

        //adjust size and set layout
        setPreferredSize (new Dimension (300, 366));
        setLayout (null);

        //add components
        add (jcomp1);
        add (jcomp2);
        add (jcomp3);
        add (jcomp4);
        add (ProductName);
        add (Description);
        add (Amount);
        add (Price);
        add (RoomNumber);
        add (jcomp10);
        add (Submit);
        add (jcomp12);
        add (Medicine);
        add (Equipment);
        add (Blood);

        setBackground(Color.cyan);
        //set component bounds (only needed by Absolute
Positioning)
        jcomp1.setBounds (25, 120, 110, 30);
        jcomp2.setBounds (10, 200, 100, 25);
        jcomp3.setBounds (10, 240, 95, 25);
        jcomp4.setBounds (0, 285, 95, 25);
        ProductName.setBounds (95, 125, 200, 25);
        Description.setBounds (95, 165, 200, 25);
        Amount.setBounds (95, 205, 200, 25);
        Price.setBounds (95, 245, 200, 25);
        RoomNumber.setBounds (95, 285, 200, 25);
        jcomp10.setBounds (10, 160, 100, 25);
        Submit.setBounds (90, 320, 150, 40);
        jcomp12.setBounds (35, 25, 200, 25);
```

```
        Medicine.setBounds (5, 75, 100, 25);
        Equipment.setBounds (105, 75, 100, 25);
        Blood.setBounds (220, 75, 100, 25);

        ProductName.setBackground(Color.green);
        ProductName.setBorder(new LineBorder(Color.BLACK, 2));
        Description.setBackground(Color.green);
        Description.setBorder(new LineBorder(Color.BLACK, 2));
        Amount.setBackground(Color.green);
        Amount.setBorder(new LineBorder(Color.BLACK, 2));
        Price.setBackground(Color.green);
        Price.setBorder(new LineBorder(Color.BLACK, 2));
        RoomNumber.setBackground(Color.green);
        RoomNumber.setBorder(new LineBorder(Color.BLACK, 2));
        Medicine.setBackground(Color.cyan);
             Equipment.setBackground(Color.cyan);
             Blood.setBackground(Color.cyan);
             Submit.setBackground(Color.green);


             Submit.addActionListener(new ActionListener()
             {

                    @Override
                    public void actionPerformed(ActionEvent arg0)
{
                           try {
                                  AddProduct();
                           } catch (ParseException e) {
                                  // TODO Auto-generated catch
block
                                  e.printStackTrace();
                           }
                    }

             }
);


    }

    public String ValidateAndAssignValues() throws
ParseException
    {
      String type=null;
      if(Medicine.isSelected())
      {
```

```
            type="Medicine";
        }
        else if(Equipment.isSelected())
        {
            type="Equipment";
        }
        else if(Blood.isSelected())
        {
            type="Blood";
        }
        else
        {
            JOptionPane.showMessageDialog(new JDialog(),
                    "Please Select one of the products type",
                    "Type Selection error",
                    JOptionPane.WARNING_MESSAGE);
        }
        product =new Product(
        ProductName.getText(),
        type,
        Description.getText(),
        Integer.parseInt(Amount.getText()),
        Integer.parseInt(Price.getText()),
        Integer.parseInt(RoomNumber.getText()));
         return "Product information was validated and assigned";
    }


    public String FormulateQuery()
    {
return "INSERT INTO Product VALUES ("+"\'" + product.Name +
"\'," +"\'"+product.Type+"\'," +"\'"+product.Description+"\',"+
    +product.Amount+","+ product.Price+","+ product.RoomID+");";
    }


    public String AddProduct() throws ParseException
    {
    try {
            ValidateAndAssignValues();
        } catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog(new JDialog(),
            "Please Enter a number, not a string in the fields
which require a number",
          "Number Format Error",
          JOptionPane.WARNING_MESSAGE);
        }
```

```
    String query = FormulateQuery();
    boolean error=false;
    DatabaseConnection.GetDatabaseConnection();
    try
    {
    DatabaseConnection.Instance.Insert(query);
    }
    catch(SQLException exc)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  exc.getMessage(),
                  "Database error",
                  JOptionPane.WARNING_MESSAGE);
       error=true;
    }
    if(!error)
    {
       JOptionPane.showMessageDialog(new JDialog(),
                  "Product has been successfully added to the
inventory",
                  "Success",
                  JOptionPane.WARNING_MESSAGE);
       return "Product was successfully added";
    }
    return null;
    }
}
```

2) UpdateProductAmountService

```
import java.awt.*;
import java.awt.event.*;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.event.*;

public class UpdateProductAmountService extends JPanel {
    private JLabel jcomp1;
    private JTextField ProductName;
    private JTextField AmountModification;
    private JLabel jcomp4;
    private JButton ShowAmount;
    private JButton UpdateAmount;
```

```
private JRadioButton AddAmount;
private JRadioButton SubtractAmount;
private JLabel jcomp8;
private JLabel NewAmount;
private JLabel jcomp10;
private JLabel CurrentAmount;
private JLabel jcomp12;


public UpdateProductAmountService() {
    //construct components
    jcomp1 = new JLabel ("Product Name :");
    ProductName = new JTextField (5);
    AmountModification = new JTextField (5);
    jcomp4 = new JLabel ("Amount :");
    ShowAmount = new JButton ("Show Amount");
    AddAmount = new JRadioButton ("Add Amount");
    SubtractAmount = new JRadioButton ("Subtract Amount");
    jcomp8 = new JLabel ("New Amount : ");
    NewAmount = new JLabel ("------------------------");
    jcomp10 = new JLabel ("Current Amount :");
    CurrentAmount = new JLabel ("------------------------");
    jcomp12 = new JLabel ("Type : ");
    UpdateAmount = new JButton("Update Amount");

    //adjust size and set layout
    setPreferredSize (new Dimension (344, 366));
    setLayout (null);

    //add components
    add (jcomp1);
    add (ProductName);
    add (AmountModification);
    add (jcomp4);
    add (ShowAmount);
    add (AddAmount);
    add (SubtractAmount);
    add (jcomp8);
    add (NewAmount);
    add (jcomp10);
    add (CurrentAmount);
    add (jcomp12);
    add(UpdateAmount);

    UpdateAmount.setBackground(Color.green);
    setBackground(Color.CYAN);
```

```
        AddAmount.setBackground(Color.green);
        SubtractAmount.setBackground(Color.green);
        ShowAmount.setBackground(Color.green);

        ProductName.setBackground(Color.green);
        ProductName.setBorder(new LineBorder(Color.BLACK, 2));
        AmountModification.setBackground(Color.green);
        AmountModification.setBorder(new LineBorder(Color.BLACK,
2));

        //set component bounds (only needed by Absolute
Positioning)
        jcomp1.setBounds (10, 10, 110, 30);
        ProductName.setBounds (125, 15, 200, 25);
        AmountModification.setBounds (130, 200, 200, 25);
        jcomp4.setBounds (5, 200, 100, 25);
        ShowAmount.setBounds (0, 90, 250, 40);
        AddAmount.setBounds (0, 240, 170, 45);
        SubtractAmount.setBounds (170, 240, 175, 45);
        jcomp8.setBounds (5, 325, 90, 30);
        NewAmount.setBounds (95, 330, 100, 25);
        jcomp10.setBounds (5, 150, 100, 25);
        CurrentAmount.setBounds (115, 150, 100, 25);
        jcomp12.setBounds (10, 50, 100, 25);
        UpdateAmount.setBounds(50, 295, 210, 30);

        ShowAmount.addActionListener(new ActionListener()
             {

                    @Override
                    public void actionPerformed(ActionEvent arg0)
{
                          try {
                                ShowCurrentAmount();
                          } catch (SQLException e) {
                                // TODO Auto-generated catch
block
                                e.printStackTrace();
                          }
                    }

             });
        UpdateAmount.addActionListener(new ActionListener()
             {

                    @Override
```

```
                    public void actionPerformed(ActionEvent arg0)
{
                        try {
                            UpdateAmount();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch
block
                            e.printStackTrace();
                        }
                    }

            }
);


    }


    public static void main (String[] args) {
        JFrame frame = new JFrame ("Update Product Amount");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add (new
UpdateProductAmountService());
        frame.pack();
        frame.setVisible (true);
    }

    public String ShowCurrentAmount() throws SQLException
    {
    ResultSet result = null;
    String query = "SELECT AMOUNT FROM PRODUCT WHERE Name =
"+"\'"+ProductName.getText()+"\';";
    System.out.println(query);
    DatabaseConnection.GetDatabaseConnection();
    try
    {
        result = DatabaseConnection.Instance.Select(query);
    }
    catch(SQLException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                exc.getMessage(),
                "Database error",
                JOptionPane.WARNING_MESSAGE);
    }
    if(result.next()!=false)
```

```
CurrentAmount.setText(String.valueOf(result.getInt("AMOUNT")));
    else
    {

      JOptionPane.showMessageDialog(new JDialog(),
                "Product Name does not exist",
                "Database error",
                JOptionPane.WARNING_MESSAGE);
    }
    return "Product Current Amount was successfully retrieved";
    }


    public String UpdateAmount() throws SQLException
    {
    int AmountToAddOrSubtract =
Integer.parseInt(AmountModification.getText());
    String query=null;
    if(AddAmount.isSelected())
    query = " UPDATE PRODUCT SET AMOUNT = AMOUNT +
"+AmountToAddOrSubtract+" WHERE Name =
"+"\'"+ProductName.getText()+"\';";
    if(SubtractAmount.isSelected())
      query = " UPDATE PRODUCT SET AMOUNT = AMOUNT -
"+AmountToAddOrSubtract+" WHERE Name =
"+"\'"+ProductName.getText()+"\';";

    System.out.println(query);
    DatabaseConnection.GetDatabaseConnection();
    boolean error=false;
    try
    {
    DatabaseConnection.Instance.update(query);
    }
    catch(SQLException exc)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                exc.getMessage(),
                "Database error",
                JOptionPane.WARNING_MESSAGE);
      error=true;
    }
    if(!error)
    {
      JOptionPane.showMessageDialog(new JDialog(),
                "Product Amount was Successfully Updated",
```

```
                "Database error",
                JOptionPane.PLAIN_MESSAGE);
    }
    ShowNewAmount();
    return "Product Amount was successfully updated";
    }


    public String ShowNewAmount() throws SQLException
    {
      ResultSet result = null;
      String query = " SELECT AMOUNT FROM PRODUCT WHERE Name =
"+"\'"+ProductName.getText()+"\';";
      DatabaseConnection.GetDatabaseConnection();
        try
        {
              result = DatabaseConnection.Instance.Select(query);
        }
        catch(SQLException exc)
        {
            JOptionPane.showMessageDialog(new JDialog(),
                    exc.getMessage(),
                    "Database error",
                    JOptionPane.WARNING_MESSAGE);
        }
        if(result.next()!=false)

NewAmount.setText(String.valueOf(result.getInt("AMOUNT")));
        else
        {

            JOptionPane.showMessageDialog(new JDialog(),
                    "Product Name does not exist",
                    "Database error",
                    JOptionPane.WARNING_MESSAGE);
        }
        return "Product new Amount was successfully retrieved";
    }
}
```

3) ViewAllProducts

```
import java.awt.Dimension;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class ViewAllProducts extends JPanel {



        ResultSet result;
        String data;
        JTextArea table;

        public ViewAllProducts() throws SQLException
        {
                setPreferredSize (new Dimension (667, 366));
            setLayout (null);
            table = new JTextArea("");

            //add components
            add (table);
            table.setBounds (0, 0, 600, 350);
            RetrieveResults();
            table.setText(data);
        }




 public String FormulateQuery()
 {
 return "SELECT * FROM Product;";
 }


  public void PresentResults() throws SQLException
        {
          data="";
        while(result.next())
        {
        data = data + "\t" + result.getString("Name") + "\t" +
result.getString("Type") + "\t" + result.getString("Description") +
                        "\t" + result.getInt("Amount") + "\t" +
result.getInt("Price") + "\t" + result.getInt("RoomNumber")+ "\n" ;
        }
        table.setText(data);
```

```
        }

    public String RetrieveResults() throws SQLException
    {

     String query = FormulateQuery();


    DatabaseConnection.GetDatabaseConnection();
    try
    {
                result = DatabaseConnection.Instance.Select(query);
    }
    catch (SQLException exc)
    {
         JOptionPane.showMessageDialog(new JDialog(),
                    exc.getMessage(),
                    "Database error",
                    JOptionPane.WARNING_MESSAGE);
         }
    PresentResults();
    return "All Product information was successfully retrieved";
    }


}
```

4) NotificationsService

```
import java.awt.Dimension;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class NotificationsService extends JPanel {


        ResultSet result;
```

```
        String data;
        JTextArea table;

        public NotificationsService() throws SQLException
        {
               setPreferredSize (new Dimension (667, 366));
            setLayout (null);
            table = new JTextArea("");

            //add components
            add (table);
            table.setBounds (0, 0, 600, 350);
            RetrieveResults();
            table.setText(data);
        }


 public String FormulateQuery()
 {
 return "SELECT * FROM Product WHERE AMOUNT < 10;";
 }


  public void PresentResults() throws SQLException
        {
          data="";
        while(result.next())
        {
        data = data + "\t" + result.getString("Name") + "\t" +
result.getString("Type") + "\t" + result.getString("Description") +
                        "\t" + result.getInt("Amount") + "\t" +
result.getInt("Price") + "\t" + result.getInt("RoomNumber")+ "\n" ;
        }
        table.setText(data);
        }


  public void RetrieveResults() throws SQLException
  {
          String query = FormulateQuery();
          try
          {
               result = DatabaseConnection.Instance.Select(query);
          }
          catch (SQLException exc)
          {
        JOptionPane.showMessageDialog(new JDialog(),
                    exc.getMessage(),
                    "Database error",
                    JOptionPane.WARNING_MESSAGE);
          }
          PresentResults();
```

```
   }

}
```

5) Product

```
public class Product {


        public String Name;
        public String Type;
        public int Amount;
        public int Price;
        public int RoomID;
        public String Description;


        public Product(String name, String type, String desc, int
amount, int price, int roomID) {
                super();
                Name = name;
                Type = type;
                Description=desc;
                Amount = amount;
                Price = price;
                RoomID = roomID;
        }


}
```

6) Pharmacist

```
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.event.*;

public class Pharmacist extends JPanel {
    private static JButton InsertProducts;
    private JLabel jcomp2;
    private JButton UpdateProductAmount;
    private JButton Notifications;
```

```
    private JButton StockAmount;


    public Pharmacist() {


        //construct components
       InsertProducts = new JButton ("Insert Products");
        jcomp2 = new JLabel ("Please Choose one of the following
Operations");
        UpdateProductAmount = new JButton ("Update Product Amount");
        setBackground(Color.CYAN);
        Notifications = new JButton ("Notifications");
        StockAmount = new JButton ("View all products Amounts");
        //adjust size and set layout
        setPreferredSize (new Dimension (667, 366));
        setLayout (null);

        //add components
        add (InsertProducts);
        add (jcomp2);
        add (UpdateProductAmount);
        add (Notifications);
        add (StockAmount);
        jcomp2.setFont(new Font("Arial", Font.PLAIN, 18));
        StockAmount.setBackground(Color.green);
        StockAmount.setBounds(340, 160, 340, 55);

        //set component bounds (only needed by Absolute Positioning)
        InsertProducts.setBounds (0, 105, 340, 55);
        jcomp2.setBounds (150, 30, 390, 35);
        UpdateProductAmount.setBounds (340, 105, 340, 55);
        Notifications.setBounds (0, 160, 340, 55);

        InsertProducts.setBackground(Color.green);
        UpdateProductAmount.setBackground(Color.green);
        Notifications.setBackground(Color.green);


        InsertProducts.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Add a new
Product");
                    d.setSize(700, 350);
                    d.getContentPane().add(new
InsertProductService());
                    // set visibility of dialog
```

```
                    d.setVisible(true);

                }
            });

        UpdateProductAmount.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Update
Product Amount");
                    d.setSize(700, 350);
                    d.getContentPane().add(new
UpdateProductAmountService());
                    // set visibility of dialog
                    d.setVisible(true);

                }
            });

        Notifications.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {

                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "Products
with low amount");
                    d.setSize(700, 350);
                    try {
                                        d.getContentPane().add(new
NotificationsService());
                                    } catch (SQLException e1) {
                                        // TODO Auto-generated catch
block
                                        e1.printStackTrace();
                                    }
                    // set visibility of dialog
                    d.setVisible(true);

                }
            });

        StockAmount.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {
```

```
                    // create a dialog Box
                    JDialog d = new JDialog(new Frame(), "View All
Products");
                    d.setSize(700, 350);
                    try {
                                        d.getContentPane().add(new
ViewAllProducts());
                                } catch (SQLException e1) {

                                        e1.printStackTrace();
                                }
                    // set visibility of dialog
                    d.setVisible(true);

                }
            });


        }



    public static void main (String[] args)
    {
       JFrame  frame = new JFrame ("Pharmacy and Inventory Management
System");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add (new Pharmacist());
        frame.pack();
        frame.setVisible (true);

    }
}
```