FALLARM PROJECT REPORT

Capstone Project Report

In partial fulfillment for the degree of

Master of Science in Computer Science

By

Pratyusha Dommata (11091)
Shruti Rajan (10904)
Uttam Polkampally (12211)
Xianghui Ma (9957)
Sravani Anumula (12433)

Prepared under the guidance of Dr. Henry Chang

School of Engineering

Northwestern Polytechnic University

47671 Westinghouse Dr., Fremont, CA 94539

April 2015

Acknowledgments

TBC(To be completed)

Abstract

The United Nations predicts that by the year 2050, 22% of the global population will be aged 60 years or over, from current 11.2% (2011). This ageing population has brought about multiple challenges and opportunities to the healthcare industry. One of these challenges is fall and injury prevention of the aged in hospitals, care facilities, as well as home settings.

According to the health report published by Centers for Disease Control and Prevention (CSC, 2009), nearly one-third of people over age of 65 suffer from severe falling injuries. Therefore, a remote monitoring system, which detects patient activities and sends alert directly to caregivers' receiver upon detection of an adverse event is essential.

The project, "Fall-Arm", is a full and complete application that covers information collection, network communication, and data analysis. The application is compatible with a number of device sensors such as Gyro Sensor and Accelerator Sensor to provide location information. The device application enables the sensors to transfer measurement data automatically to server via network. Server tables the collected data and would send alert to emergency health-care immediately in case of a falling accident happens.

CONTENTS

1.0 INTRODUCTION

1.1 The Concept

FallArm is a remote patient monitoring strategy that enables care providers for remote management of patient's risk of falling 24 hours a day, 7 days a week. Device continuously sense acceleration and orientation patterns and send this data to server which analyze it and classify risk. This risk class is conveyed back to patient as a feedback. Also the system reports any adverse events and alert is send directly to caretaker. Staff can login to website to get full details about the incident or simply to analyze patient data. This closed loop of information exchange increases patient awareness of risk and provides measures to prevent adverse events (QIRIS: Quality Innovation Research Instruction Safety).

This project focuses on the software part, which in turn can be broadly categorized into three areas:

A) Internet programming – to develop a client-server model where the sensor will directly communicate with the server to provide the device data and the client interface will be used by the hospital staff to view, track patient's information, and process the data.

B) Network programming – socket programming to enable communication between the client and the server.

C) Database – to hold the data related to the system at the server. This will include information about the user profile of the clients, the information received from the device.

D) QA – Testing the whole project with Junit and Selenium.

1.2 Background

Fallarm is a project originated by QIRIS (Quality Innovation Research Instruction Safety), a non-profit association whose objective is to foster scientific research and technological innovation for personcentered health-care. QIRIS has developed fallarm, a technological solution for fall prevention. Fallarm is based on a wearable device that informs the patient in situations where the risk is higher, promoting users' awareness on their responsible mobility (QIRIS: Quality Innovation Research Instruction Safety).

Fallarm is approved as capstone project and is customized by using Android as sensor device . This project will be researched, designed and developed, to be submitted towards fulfillment of the degree requirements.

1.3 Objectives

To design and develop Fallarm patient monitoring system including programming on device side to record activity pattern and supporting software on server side to process data.

1.4 Environment

1. Host operating system platform: Server:

    Windows 7/8, Mac OS 64-bit Client: Android OS

2. Database:

    MySql 5.6

3. Virtual Machine:

    VirtualBox 4.3.12 (Genymotion Virtual Device)

4. Programing Languages:

    Java, HTML 5, CSS 3, and Java Script

5.Design Pattern:

    MVC

6. Environment:

    Selenium IDE 2.8.0, Eclipse 4.4.1, NetBeans 8.0, Android Studio (beta)

2.0 Design

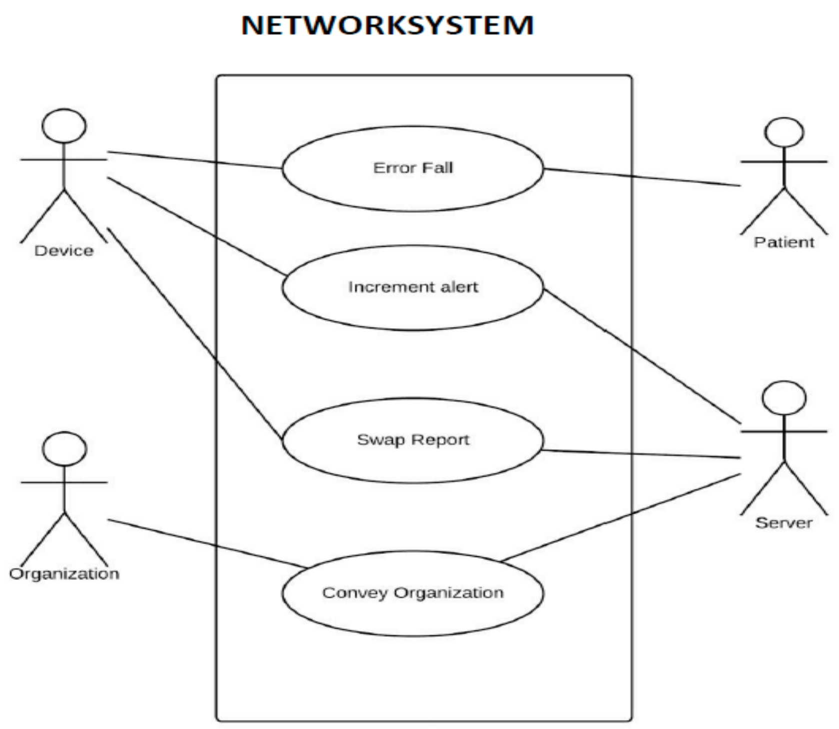
2.1 Introduction

Unified Modeling Language (UML) is modeling framework used in object oriented software engineering. Class Diagram is a Structure diagram which describes structure of a system. Behavior diagrams include Use Case Diagram which describes the functionality provided by a system in terms of actors and Activity diagram which describes the operational step-by-step work-flows in a system. Sequence diagram is an Interaction diagram which shows how objects communicate with each other

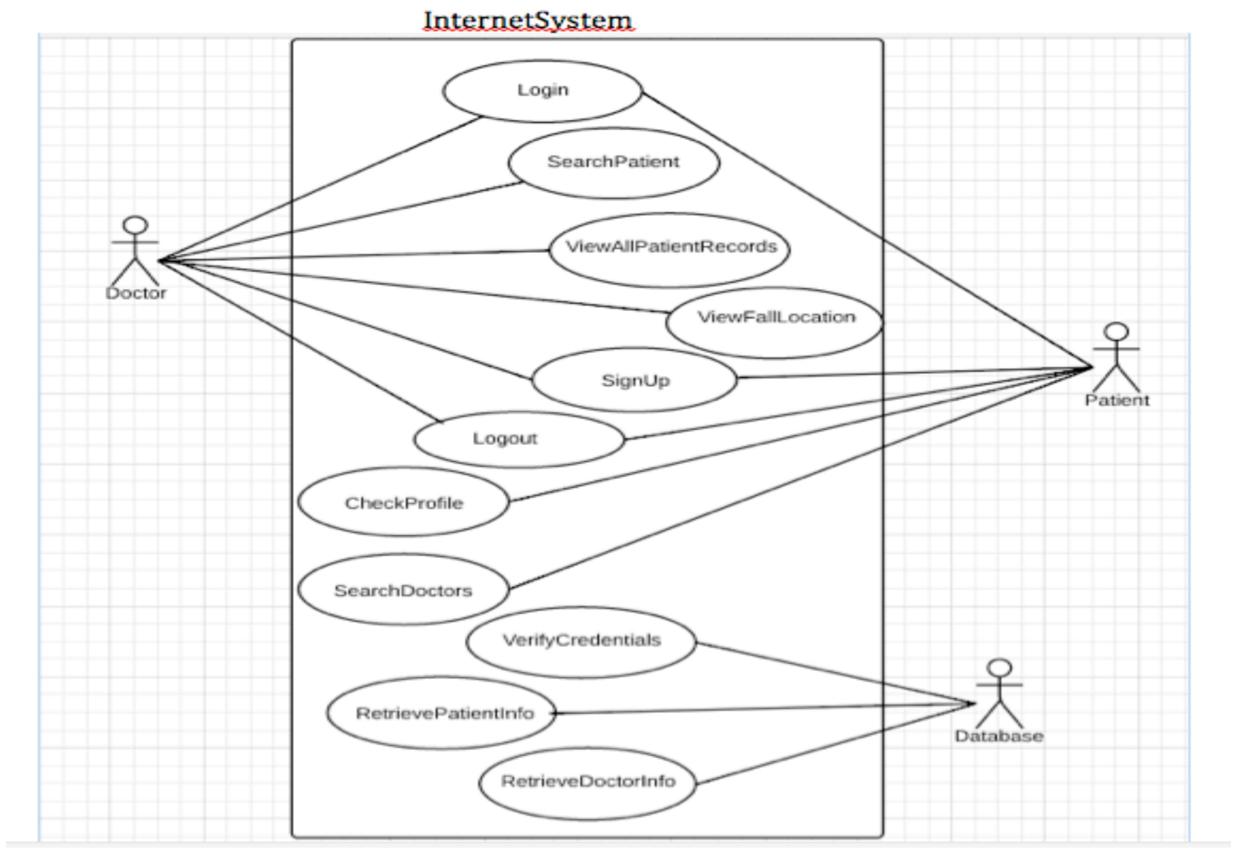
2.2 Use Case Diagram

To achieve a goal there must be some level of interaction between role and system and these interactions are represented in the form of use case diagram. The role can be an external actor or a computer system, which is capable of making decisions. Also a person can be represented as different actors based on the role he plays.

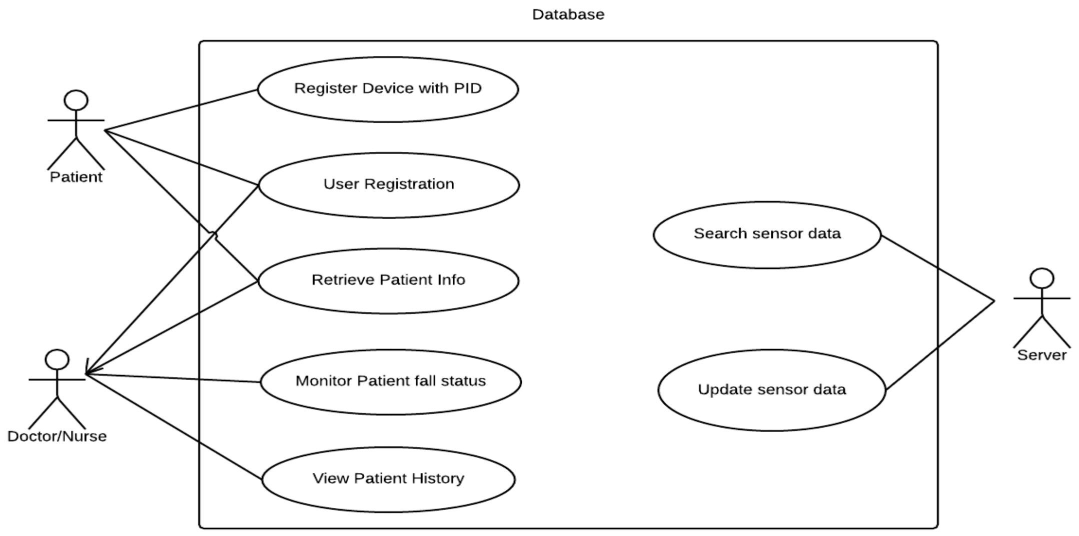
2.2.1 Use Case Diagram of Networking - *Sravani Anumula (12433)*



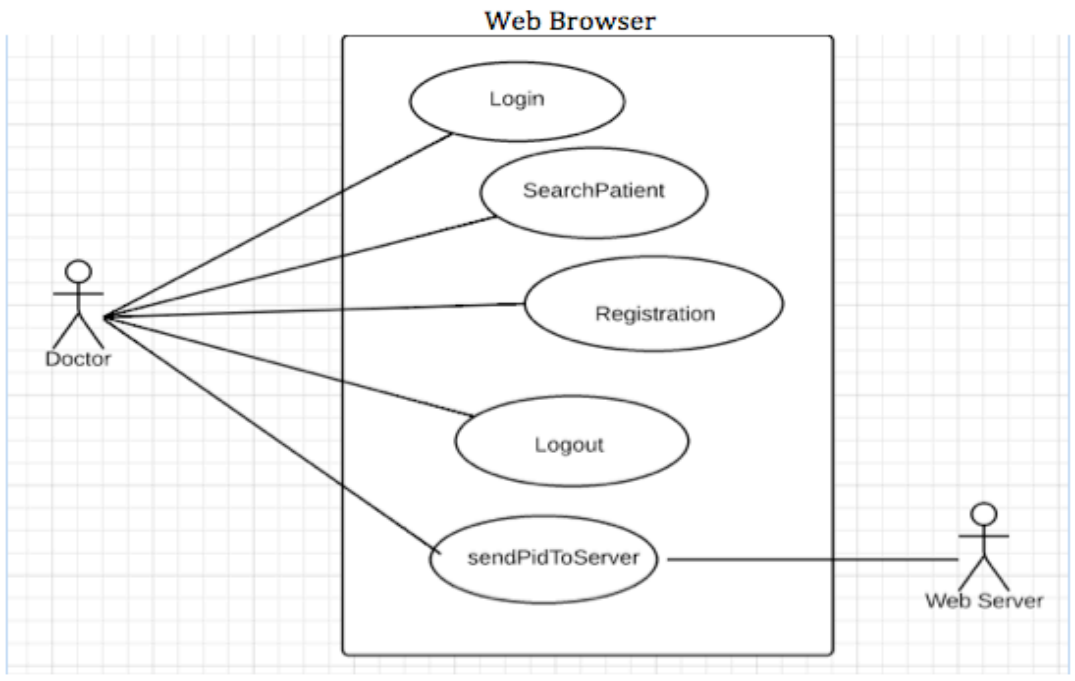
2.2.2 Use Case Diagram of Internet - *Pratyusha Dommata (11091)*



2.2.3 Use Case Diagram of Database - *Shruti Rajan (10904)*



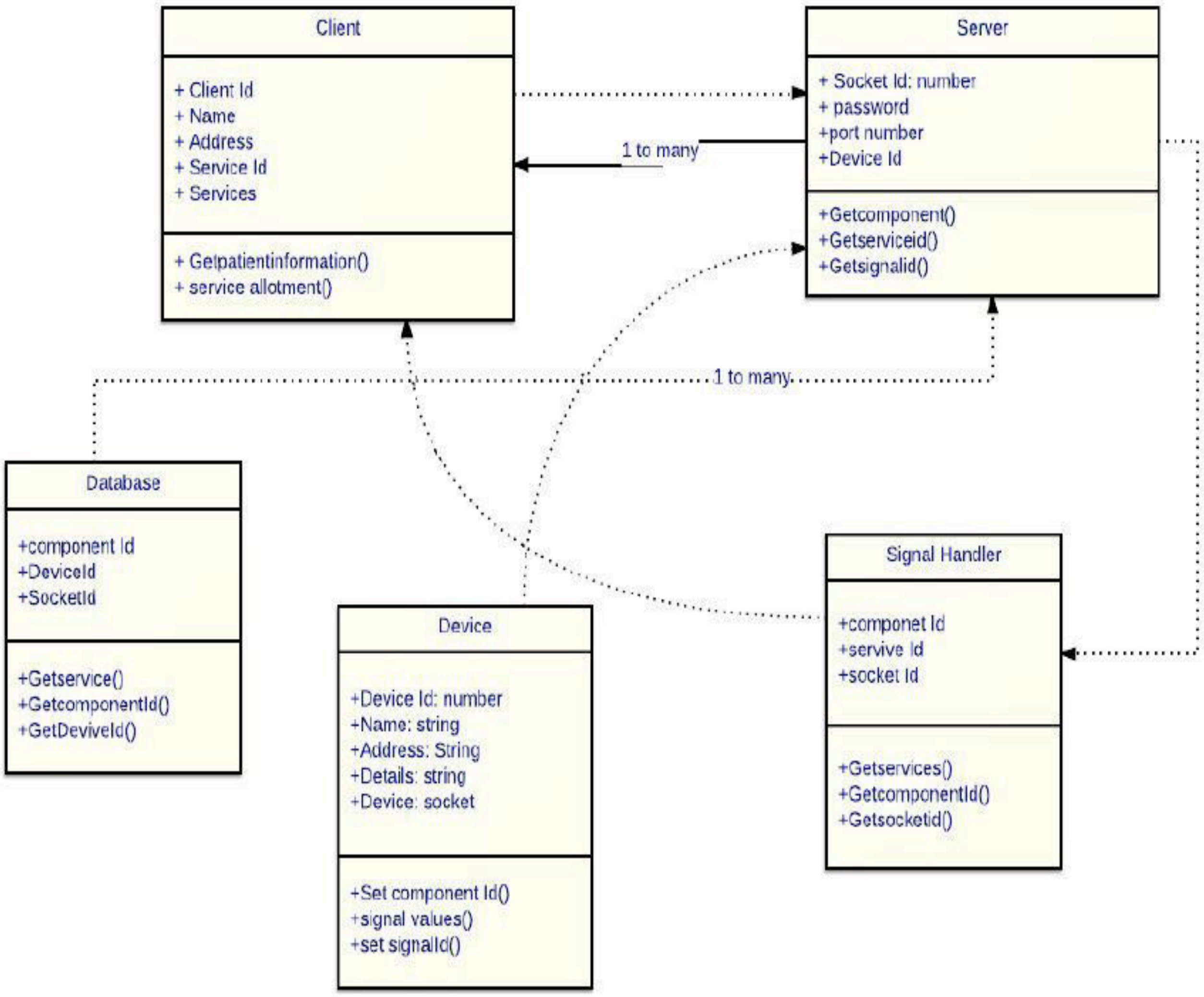
2.2.4 Use Case Diagram of Front End - *Uttam Polkampally (12211), Xianghui Ma (9957)*



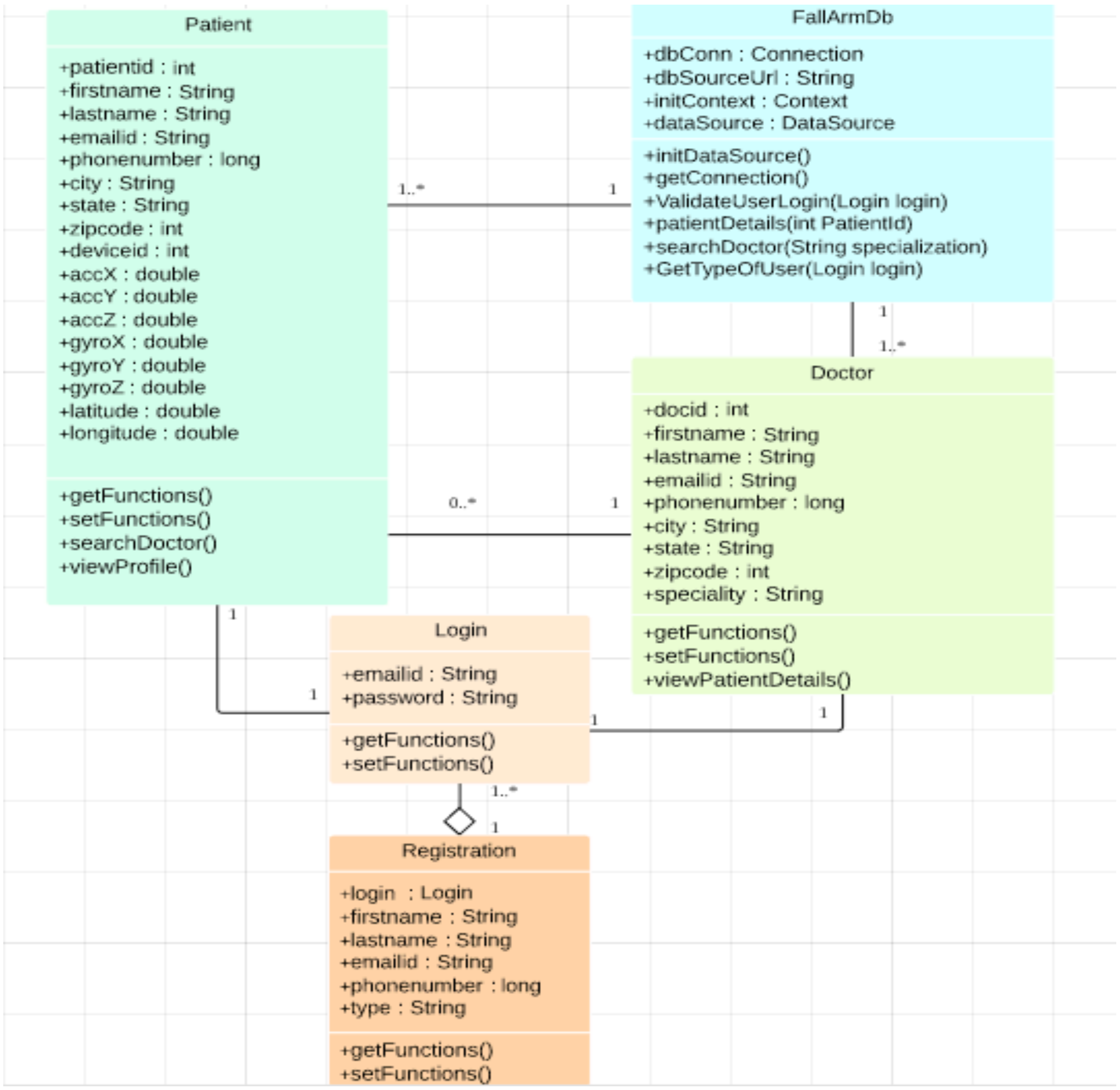
2.3 Class Diagram

Class diagram is a static diagram that shows structure of a system and contains classes, attributes and methods. Additionally it includes relationships between classes. It can also be used for data modeling to define and analyze data needed to support processes.

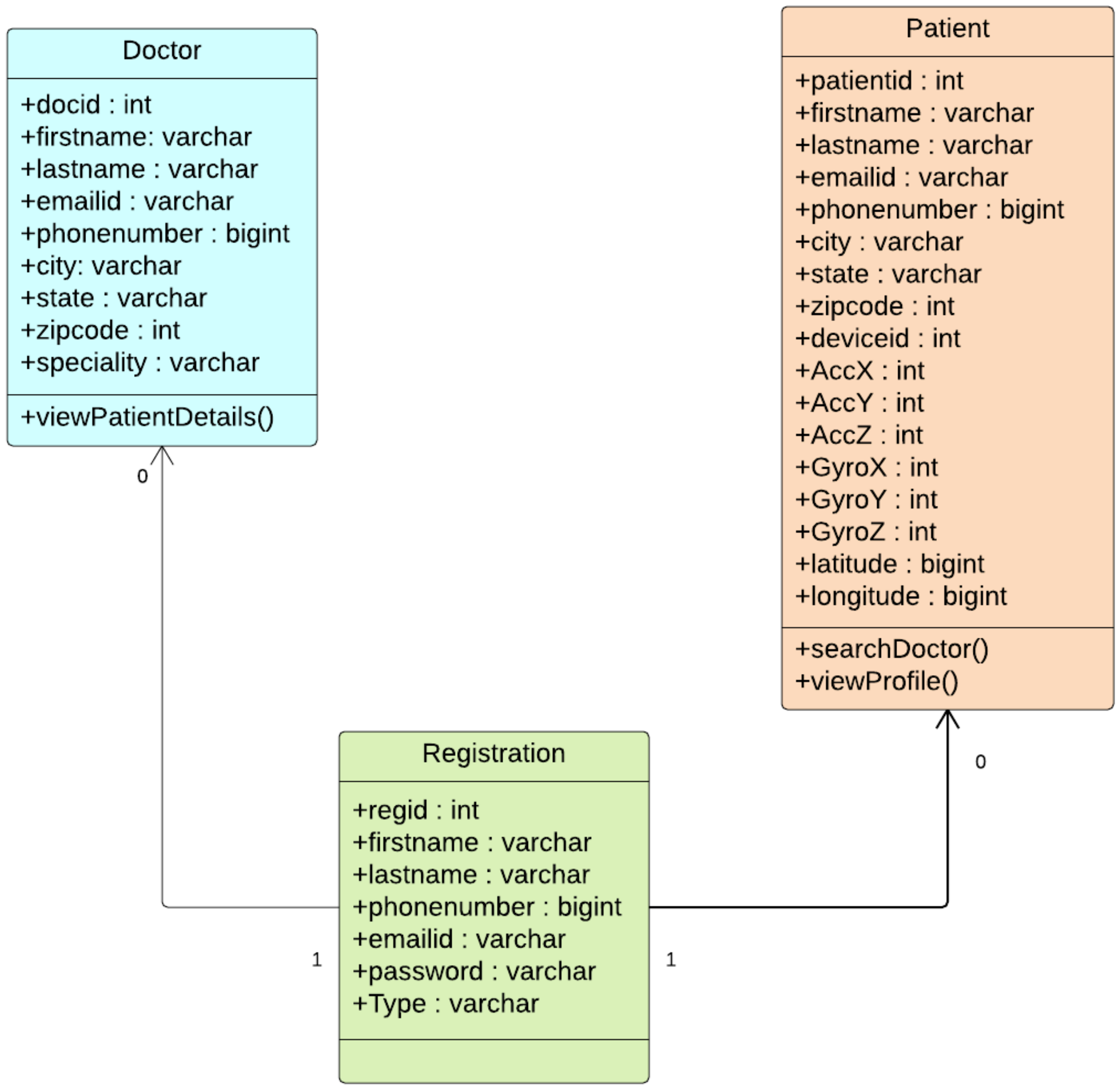
2.3.1 Class Diagram of Device and Networking - *Sravani Anumula (12433)*



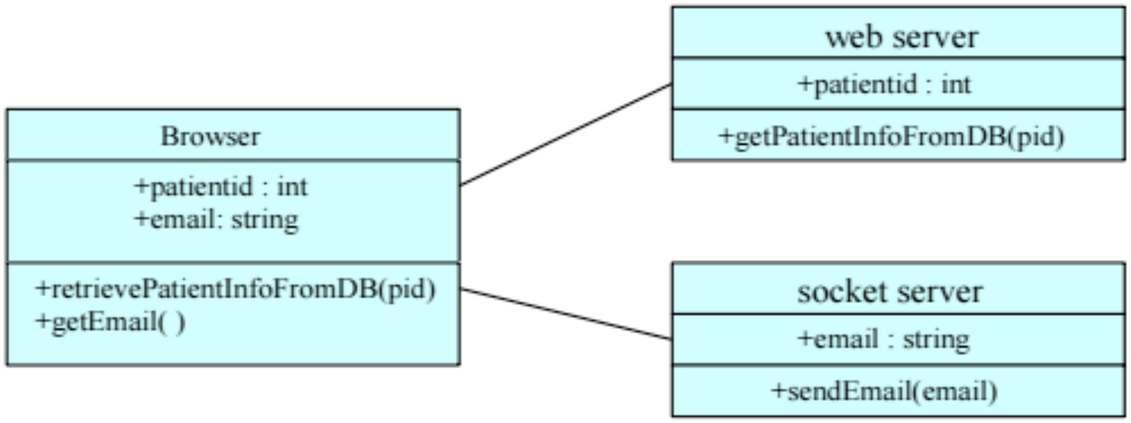
2.3.2 Class Diagram of Internet - *Pratyusha Dommata (11091)*



2.3.3 Class Diagram of Database - *Shruti Rajan (10904)*



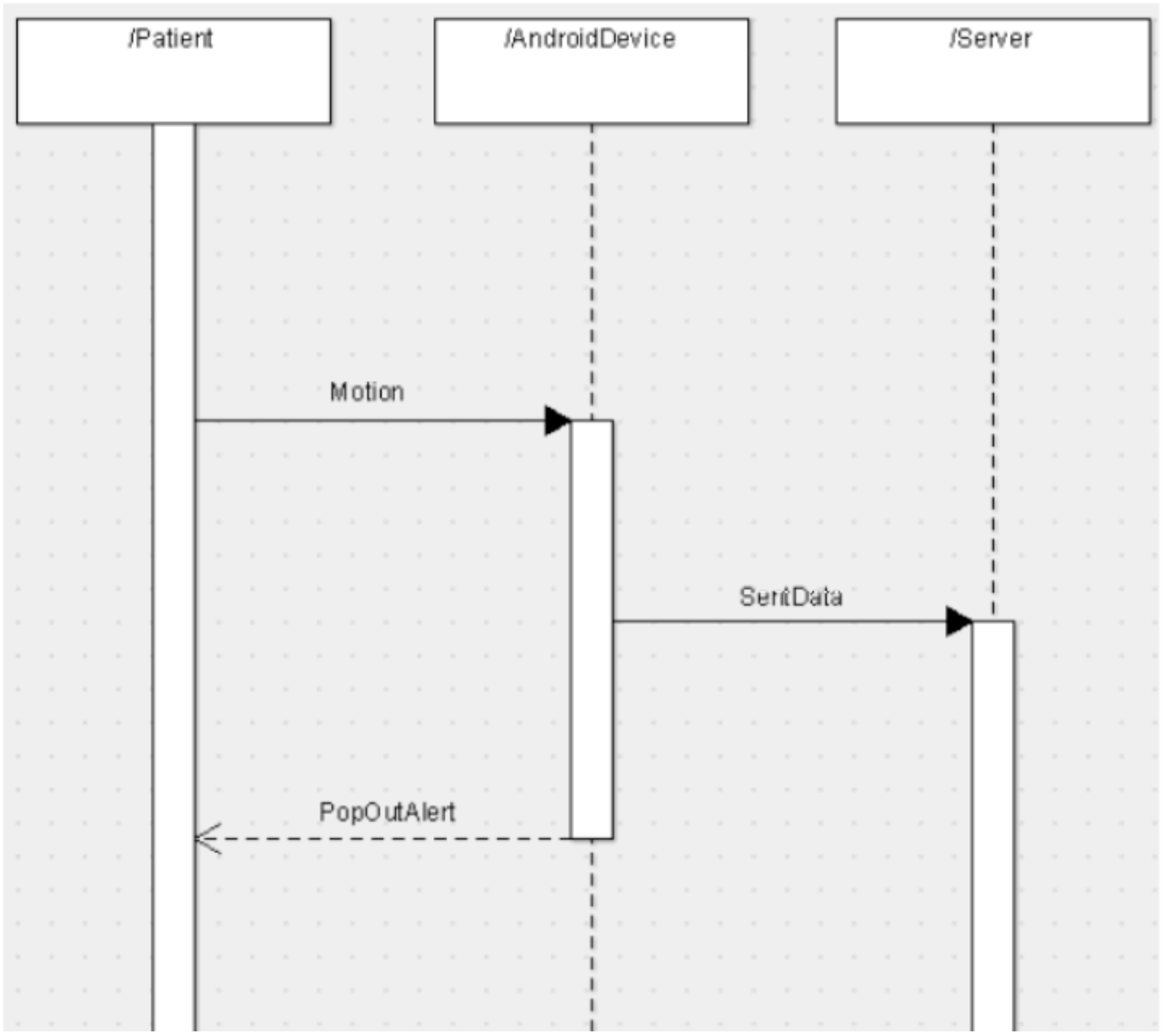
2.3.4 Class Diagram of Front End - *Uttam Polkampally (12211), Xianghui Ma (9957)*



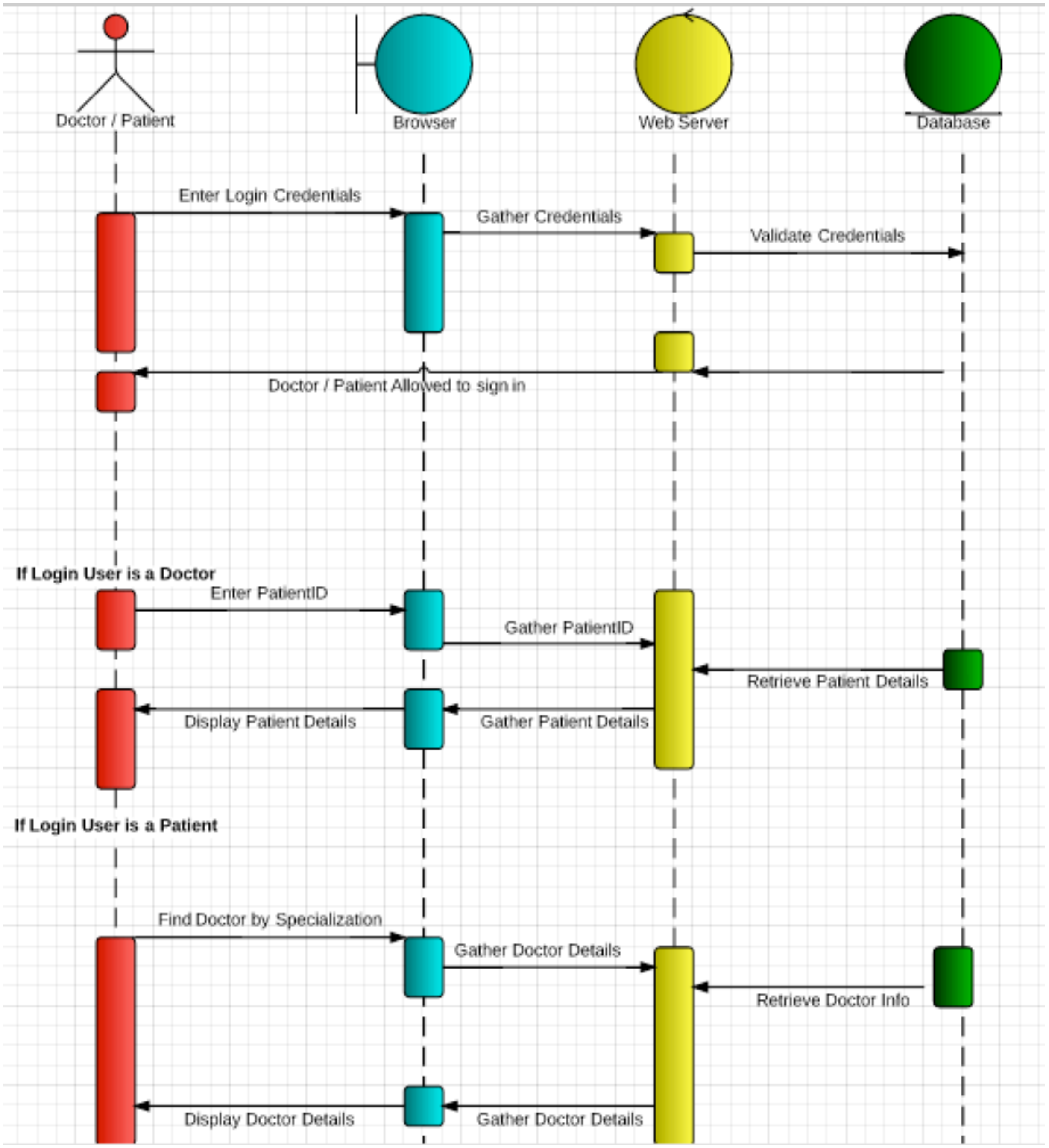
2.4 Sequence Diagram

A sequence diagram, in the context of UML, represents object collaboration and is used to define event sequences between objects for a certain outcome. A sequence diagram is an essential component used in processes related to analysis, design and documentation. A sequence diagram is also known as a timing diagram, event diagram and event scenario

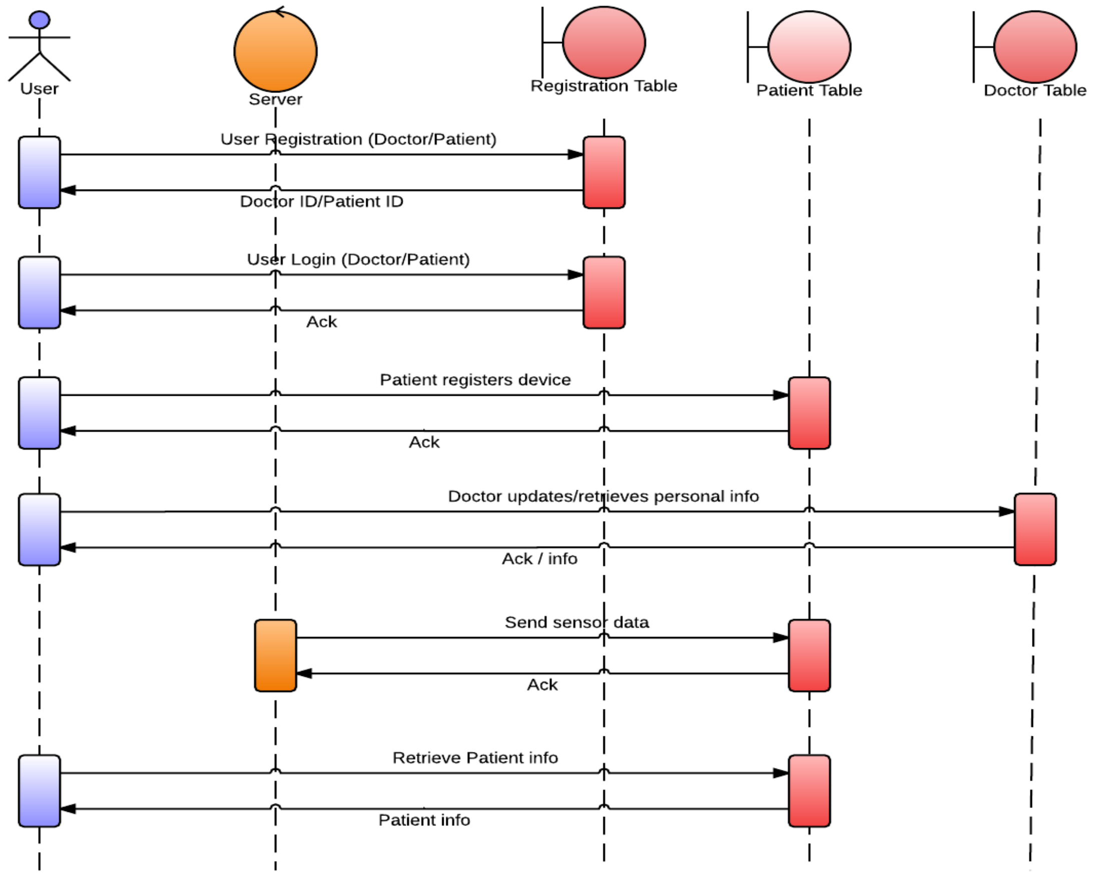
2.4.1 Sequence Diagram of Networking - *Sravani Anumula (12433)*



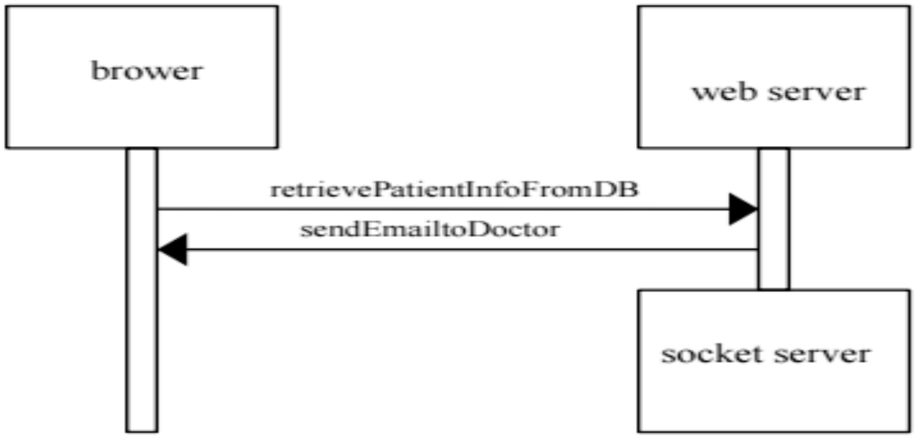
2.4.2 Sequence Diagram of Internet - *Pratyusha Dommata (11091)*



2.4.3 Sequence Diagram of Database - *Shruti Rajan (10904)*



2.4.4 Sequence Diagram of Front End - *Uttam Polkampally (12211), Xianghui Ma (9957)*



3.0 Implementation Details

3.1 Introduction

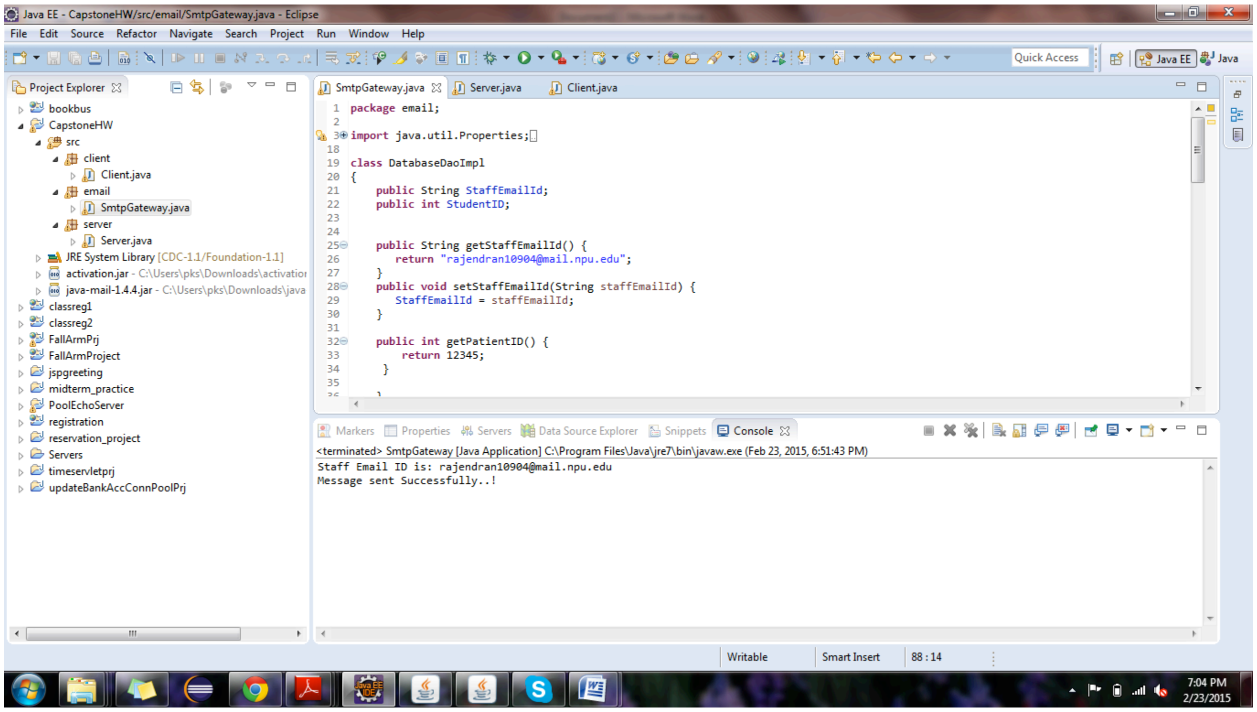
3.2 Environment Setup

3.3 Internet programming – Java

3.4 Database – MySQL

3.5 Network Programming – Java

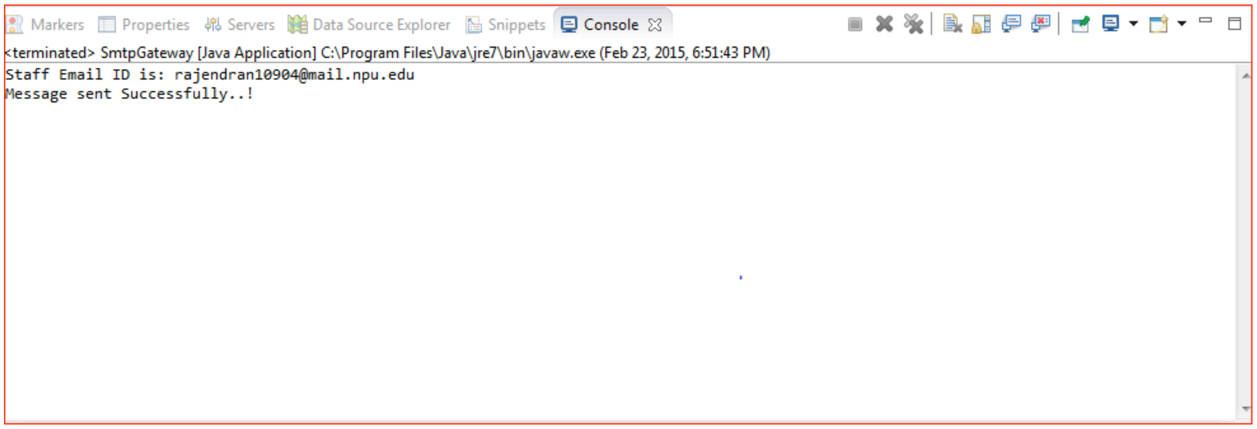
<u>Send email/SMS from the socket server to the nurse's device</u>



```java
package email;

import java.util.Properties;

class DatabaseDaoImpl
{
    public String StaffEmailId;
    public int StudentID;


    public String getStaffEmailId() {
       return "rajendran10904@mail.npu.edu";
    }
    public void setStaffEmailId(String staffEmailId) {
       StaffEmailId = staffEmailId;
    }

    public int getPatientID() {
        return 12345;
     }

```

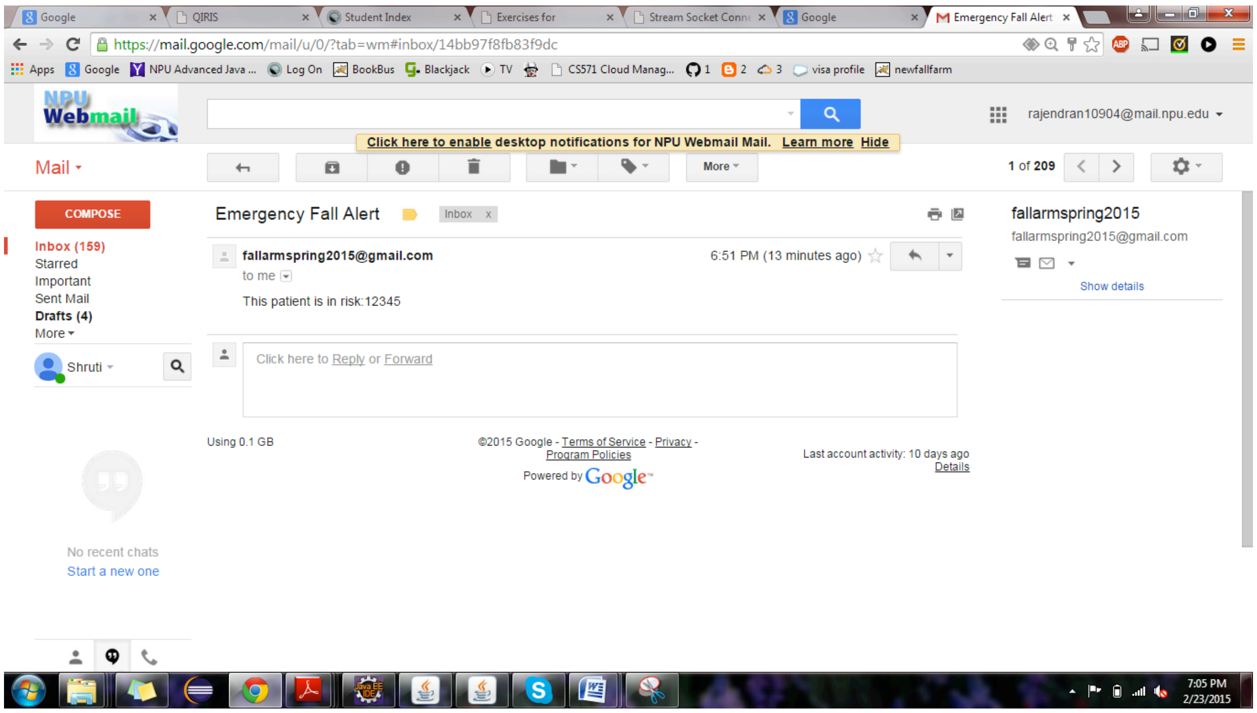
```
<terminated> SmtpGateway [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 23, 2015, 6:51:43 PM)
Staff Email ID is: rajendran10904@mail.npu.edu
Message sent Successfully..!
```

Message successfully sent to the staff/nurse



```
<terminated> SmtpGateway [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 23, 2015, 6:51:43 PM)
Staff Email ID is: rajendran10904@mail.npu.edu
Message sent Successfully..!
```

An alert message sent to via mail to the nurse/staff. The message displays the PID number of the patient who is in risk


https://mail.google.com/mail/u/0/?tab=wm#inbox/14bb97f8fb83f9dc
rajendran10904@mail.npu.edu
Click here to enable desktop notifications for NPU Webmail Mail. Learn more Hide
Mail
COMPOSE
Inbox (159)
Starred
Important
Sent Mail
Drafts (4)
More
1 of 209
fallarmspring2015
fallarmspring2015@gmail.com
Show details

Emergency Fall Alert Inbox

fallarmspring2015@gmail.com 6:51 PM (13 minutes ago)
to me

This patient is in risk:12345

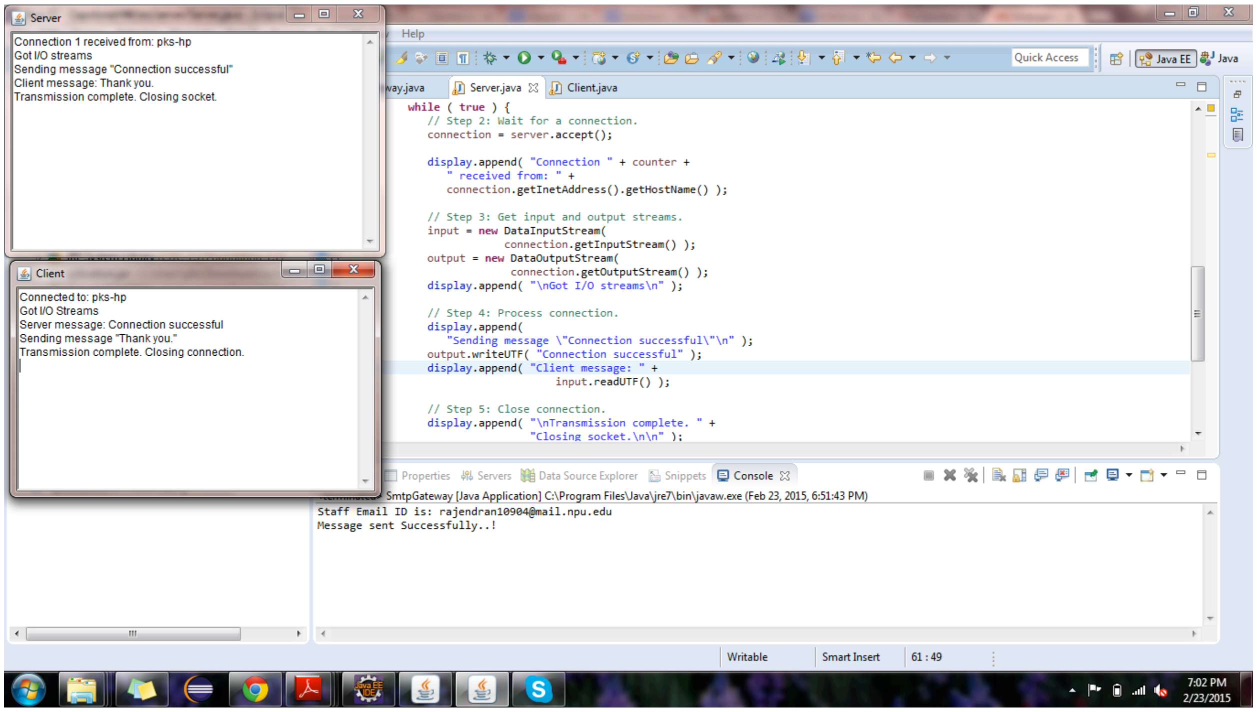
Click here to Reply or Forward

SmtpGateway.java

```
package email;

import java.util.Properties;

import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

class DatabaseDaoImpl
{
       public String StaffEmailId;
    public int StudentID;


    public String getStaffEmailId() {
       return "rajendran10904@mail.npu.edu";
    }
    public void setStaffEmailId(String staffEmailId) {
       StaffEmailId = staffEmailId;
    }

    public int getPatientID() {
        return 12345;
     }

       }
public class SmtpGateway {

          public static void main(String args[]) {
                 DatabaseDaoImpl dbService = new DatabaseDaoImpl();
             String staffEmail = dbService.getStaffEmailId();
             System.out.println("Staff Email ID is: "+staffEmail);
             SmtpGateway.sendEmail(staffEmail);
          }

          /**
          * Send Email via Gmail SMTP
          * Need library mail.jar
          * Use Authentication: Yes
          * Port for SSL: 465
          * @param emailId Email Id of assigned staff to send alert
          */
          public static void sendEmail(String to) {

                 DatabaseDaoImpl dataObj = new DatabaseDaoImpl();
               //Get the session object
                 Properties props = new Properties();
                 props.put("mail.smtp.host", "smtp.gmail.com");
                 props.put("mail.smtp.socketFactory.port", "465");
                 props.put("mail.smtp.socketFactory.class",
                           "javax.net.ssl.SSLSocketFactory");
                 props.put("mail.smtp.auth", "true");
```

```
                props.put("mail.smtp.port", "465");

                Session session = Session.getDefaultInstance(props,
                                  new javax.mail.Authenticator() {
                 protected PasswordAuthentication getPasswordAuthentication() {
                 return new
      PasswordAuthentication("fallarmspring2015@gmail.com","2015springfallarm");
                 }
                });

              //compose message
                try {
                 MimeMessage message = new MimeMessage(session);
                 message.setFrom(new
                       InternetAddress("fallarmspring2015@gmail.com"));//From
                 message.addRecipient(Message.RecipientType.TO,new
                       InternetAddress(to));//staff emailid
                 message.setSubject("Emergency Fall Alert");
                 //String htmlText = ("<div style=\"color:red;\">This patient is in
                        risk: "+pid+"</div>");
                 message.setText("This patient is in risk:"+dataObj.getPatientID());

                 //send message
                 Transport.send(message);

                 System.out.println("Message sent Successfully..!");

                } catch (MessagingException e) {throw new RuntimeException(e);}



         }
}
```

Send message from the socket client to the socket server



```
Connection 1 received from: pks-hp
Got I/O streams
Sending message "Connection successful"
Client message: Thank you.
Transmission complete. Closing socket.
```

```
Connected to: pks-hp
Got I/O Streams
Server message: Connection successful
Sending message "Thank you."
Transmission complete. Closing connection.
```

```java
while ( true ) {
   // Step 2: Wait for a connection.
   connection = server.accept();

   display.append( "Connection " + counter +
      " received from: " +
      connection.getInetAddress().getHostName() );

   // Step 3: Get input and output streams.
   input = new DataInputStream(
               connection.getInputStream() );
   output = new DataOutputStream(
                connection.getOutputStream() );
   display.append( "\nGot I/O streams\n" );

   // Step 4: Process connection.
   display.append(
      "Sending message \"Connection successful\"\n" );
   output.writeUTF( "Connection successful" );
   display.append( "Client message: " +
                   input.readUTF() );

   // Step 5: Close connection.
   display.append( "\nTransmission complete. " +
                   "Closing socket.\n\n" );
```

```
SmtpGateway [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 23, 2015, 6:51:43 PM)
Staff Email ID is: rajendran10904@mail.npu.edu
Message sent Successfully..!
```

Client server communication.

Client makes connection to the server, which is acknowledged by the server with "connection successful" message. On receiving that client sends a "thank you" message.

server.java

```java
package server;
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;

class CloseWindowAndExit extends WindowAdapter {
   public void windowClosing( WindowEvent e )
   {
      System.exit( 0 );
   }
}

public class Server extends Frame {
   private TextArea display;

   public Server()
   {
      super( "Server" );
      display = new TextArea( "", 0, 0,
                    TextArea.SCROLLBARS_VERTICAL_ONLY );
      add( display, BorderLayout.CENTER );
      setSize( 300, 150 );
      setVisible( true );
   }

   public void runServer()
   {
      ServerSocket server;
      Socket connection;
      DataOutputStream output;
      DataInputStream input;
      int counter = 1;

      try {
         // Step 1: Create a ServerSocket.
         // Change the port number from 5000 to other number
         // if you encounter JVM_bind error
         // when running this program.
         server = new ServerSocket( 5000, 100 );

         while ( true ) {
            // Step 2: Wait for a connection.
            connection = server.accept();

            display.append( "Connection " + counter +
               " received from: " +
               connection.getInetAddress().getHostName() );

            // Step 3: Get input and output streams.
            input = new DataInputStream(
                       connection.getInputStream() );
            output = new DataOutputStream(
                        connection.getOutputStream() );
            display.append( "\nGot I/O streams\n" );

            // Step 4: Process connection.
            display.append(
               "Sending message \"Connection successful\"\n" );
            output.writeUTF( "Connection successful" );
            display.append( "Client message: " +
                                input.readUTF() );

            // Step 5: Close connection.
            display.append( "\nTransmission complete. " +
                            "Closing socket.\n\n" );
            connection.close();
            ++counter;
         }
      }
      catch ( IOException e ) {
         e.printStackTrace();
      }
   }
   public static void main( String args[] )
   {
      Server s = new Server();

      s.addWindowListener( new CloseWindowAndExit() );
      s.runServer();
   }
}
```

client.java

```
package client;
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;


class CloseWindowAndExit extends WindowAdapter {
   public void windowClosing( WindowEvent e )
   {
      System.exit( 0 );
   }
}


public class Client extends Frame {
   private TextArea display;

   public Client()
   {
      super( "Client" );
      display = new TextArea( "", 0, 0,
                     TextArea.SCROLLBARS_VERTICAL_ONLY );
      add( display, BorderLayout.CENTER );
      setSize( 300, 150 );
      setVisible( true );
   }

   public void runClient()
   {
      Socket client;
      DataInputStream input;
      DataOutputStream output;

      try {
         // Step 1: Create a Socket to make connection.
         client = new Socket( InetAddress.getLocalHost(),
                              5000 );

         display.append( "Connected to: " +
            client.getInetAddress().getHostName() );

         // Step 2: Get the input and output streams.
         input = new DataInputStream(
                     client.getInputStream() );
         output = new DataOutputStream(
                      client.getOutputStream() );
         display.append( "\nGot I/O Streams\n" );

         // Step 3: Process connection.
         display.append( "Server message: " +
            input.readUTF() );
         display.append(
            "\nSending message \"Thank you.\"\n" );
         output.writeUTF( "Thank you." );

         // Step 4: Close connection.
         display.append( "Transmission complete. " +
            "Closing connection.\n" );
         client.close();
      }
      catch ( IOException e ) {
         e.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      Client c = new Client();

      c.addWindowListener( new CloseWindowAndExit() );
```

```
        c.runClient();
    }
}
```

4.0 Test Result

4.1 Introduction

4.1 Test Environment

4.3 Test Result Summary

7.0 Future Enhancements

8.0 Conclusion

Bibliography

Appendix A
Appendix B