

NANYANG
TECHNOLOGICAL
UNIVERSITY

A Portable Medical Monitoring Device

Final Year Report

By

Shi Feng

School of Computer Engineering

2010-2011



A Portable Medical Monitoring Device

Supervisor: Mr Lau Chiew Tong

Examiner: Mr Li Mo

Submitted by: Shi Feng

School of Computer Engineering

2010-2011

Abstract

Diabetes has become increasingly prevalent due to extremely busy lifestyles, family history of diabetes mellitus, and pregnancy in elder women, unhealthy eating habits. Diabetes accounted for the deaths of 2.9 million people in the 2010. The number of patients suffering from diabetes has grown from 9 percent in 2004, increased 11 percent last year, higher than the global average of 6 percent. Moreover, World Health Organization figures indicate that the incidence of diabetes in Singapore is set to go up from the current 11% of the population to 17% by 2030.

Usually the doctors hope to monitor the patients' status and the patients hope to report the status at any time. Unfortunately, except the patient is in hospital or is a resident at a nursing home, the patient is generally remote from the doctor, nurse or other care giver. In this Thesis, we study a portable medical monitoring device.

This project aims to let the doctors can to monitor the patients' status and the patients can to report the status at any time. The patients can store their blood glucose data through their mobile phones. The doctors can monitor blood glucose of their patient via a web based interface. The doctors can manage the health of their patients by providing recommendations through the web interface

Acknowledgement

I would like to express my gratitude to all those who helped me during the writing of this thesis.

My deepest gratitude goes first and foremost to Professor Lau Chiew Tong, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

I also owe my sincere gratitude to my friends who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.

Contents

Abstract	5
Acknowledgement	7
Contents	9
Chapter 1: Introduction	13
1.1 Background	13
1.2 Project Objectives	14
Chapter 2: Web Client	15
2.1 Objective	15
2.2 Requirements	15
2.3 Methodology and Workflow	17
2.3.1 Home page and user register	17
2.3.2 Mistake in login	19
2.3.3 Monitor the health data and save recommendation	20
2.3.4 Add a new patient for the website	21
2.4 Summary	21
Chapter 3: Mobile Client	23
3.1 Objective	23
3.2 Requirements	23
3.3 Technology Overview	24
3.4 Methodology and Workflow	26
3.4.1 Welcome page and login page	27
3.4.2 Login Wait and Failed Screen	29
3.4.3 The Main Menu	31
3.4.4 New Record	32
3.4.5 Log	33
3.4.6 Recommendation	34
3.5 Summary	34
Chapter 4: Server	35
4.1 Objective	35
4.2 Requirements	35
4.3 Technology Overview	38
4.4 Project Structure	39
4.5 Spring MVC framework	41
4.5.1 Configuring web.xml	42
4.5.2 Application container configuration	42
4.5.3 Data source and JDBC configuration	42
4.5.4 Project build tool	43
4.6 Database design	43
References	45

Appendix	47
----------------	----

List of Figures

Figure 1: workflow for the doctor	15
Figure 2: welcome page	17
Figure 3: register page	18
Figure 4: invalid email address or password.....	19
Figure 5: patient list	20
Figure 6: create an account for the new patient	21
Figure 7: patient workflow	23
Figure 8: J2ME structure platform	25
Figure 9: workflow	26
Figure 10: Welcome Splash screen	27
Figure 11: login page	28
Figure 12: Login Wait Screen	29
Figure 13: login failed Splash screen	30
Figure 14: Menu	31
Figure 15: New record input form	32
Figure 16: Log page	33
Figure 17: View last Recommendation page	34
Figure 18: Entity-Relationship Diagram for the project	35
Figure 19: Three-tier architecture	39
Figure 20: the directory structure	40
Figure 21: Tables in Database	44
Figure 22: Table definition for table user	44
Figure 23: Table definition for table record	44
Figure 24: Table definition for table recommendation	44

Chapter 1: Introduction

1.1 Background

Diabetes has become increasingly prevalent due to extremely busy lifestyles, family history of diabetes mellitus, and pregnancy in elder women, unhealthy eating habits. Diabetes accounted for the deaths of 2.9 million people in the 2010. The number of patients suffering from diabetes has grown from 9 percent in 2004, increased 11 percent last year, higher than the global average of 6 percent. Moreover, World Health Organization figures indicate that the incidence of diabetes in Singapore is set to go up from the current 11% of the population to 17% by 2030.

The patient is not in hospital or is not a resident at a nursing home. They will go to see a doctor, but before that, they may appear some abnormal situations. For the time delay, when they arrive at hospital, the doctor may not discover the abnormal situations and affect the treatment. And the patients need to take their health reading 3 to 5 times a day, sometimes even more. As such, this data should be provided to the patients' doctors to enable better management of chronic diseases.

In order to properly treat the patient's condition and to identify the cause of the problem, I try to define a system to help the patients.

This project is an effort at developing a technological solution that prevents health problems from hindering everyday choices, ease suffering and enhance the quality of life. For this research, A Mobile phone application is needed to generate tokens in response to a local device connection over wireless, this should then allow for a separate of a token on a remote server in response to a wireless connection on a separate laptop terminal.

1.2Project Objectives

The objects of this project were as follows:

- Enable the patients to store their blood glucose data through their mobile phones
- Enable the doctors to monitor blood glucose of their patient via a web based interface
- Enable doctors to manage the health of their patients by providing recommendations through the web interface

Chapter 2 : Web Client

2.1 Objective

The web client provides a platform for the doctors to monitor their patient's information and save his recommendation.

I created a website which named iCare for this project.

2.2 Requirements

The following figure describes the workflow within the web application for the doctor.

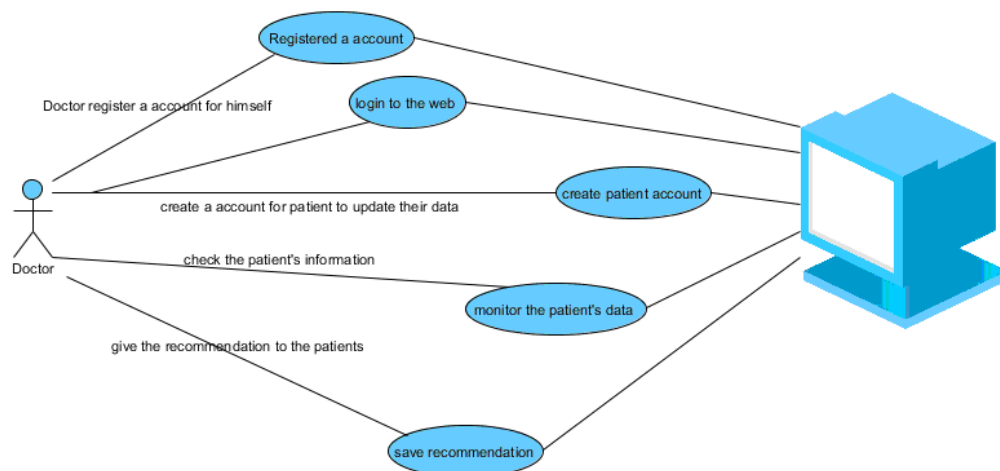


Figure 1: workflow for the doctor

Use case: Doctor Register an account for himself for the iCare website.

When: The doctor wants to register a new Account at iCare

Given: He inputs all the necessary information such as email address and password, full name, phone number, address, gender, age in proper format

Then: When he clicks the “Sign Up” button, the application should create a new

account for him

And: The website navigates to the “Login in” page.

Use case: Doctor Login to iCare website

When: The doctor wants to enter the iCare website

Given: Doctor inputs his email address and password

Then: When he clicks the “Sign In” button, if the email address and password are correct, the doctor enters the website. Otherwise the website will show “invalid email address and password” to let the user login again

And: The website navigates to the “patient list” page

Use case: Doctor creates a new account for his patient

When: The doctor has a new patient and wants to create an account for him

Given: He inputs the patient’s necessary information such as email address and password, full name, phone number, address, gender, age in proper format

Then: he clicks “click here to add new patient” button, the application will create a new account for his patient

And: The website navigates to the “patient list” page and increases the new patient to the list

Use case: Doctor monitors the patient’s health data

When: The doctor wants to monitor the patient’s health data

Then: he clicks the patient who is in the table to monitor the patient’s data

Use case: Doctor writes and save the recommendation for the patient

When: After the doctor reads all the health data from the patient

Given: The doctor write the recommendation for the patient

Then: He clicks “submit” to save the recommendation on the iCare website

2.3 Methodology and Workflow

2.3.1 Home page and user register

Upon navigating to the website, the user is presented with the home page as displayed in the figure below.

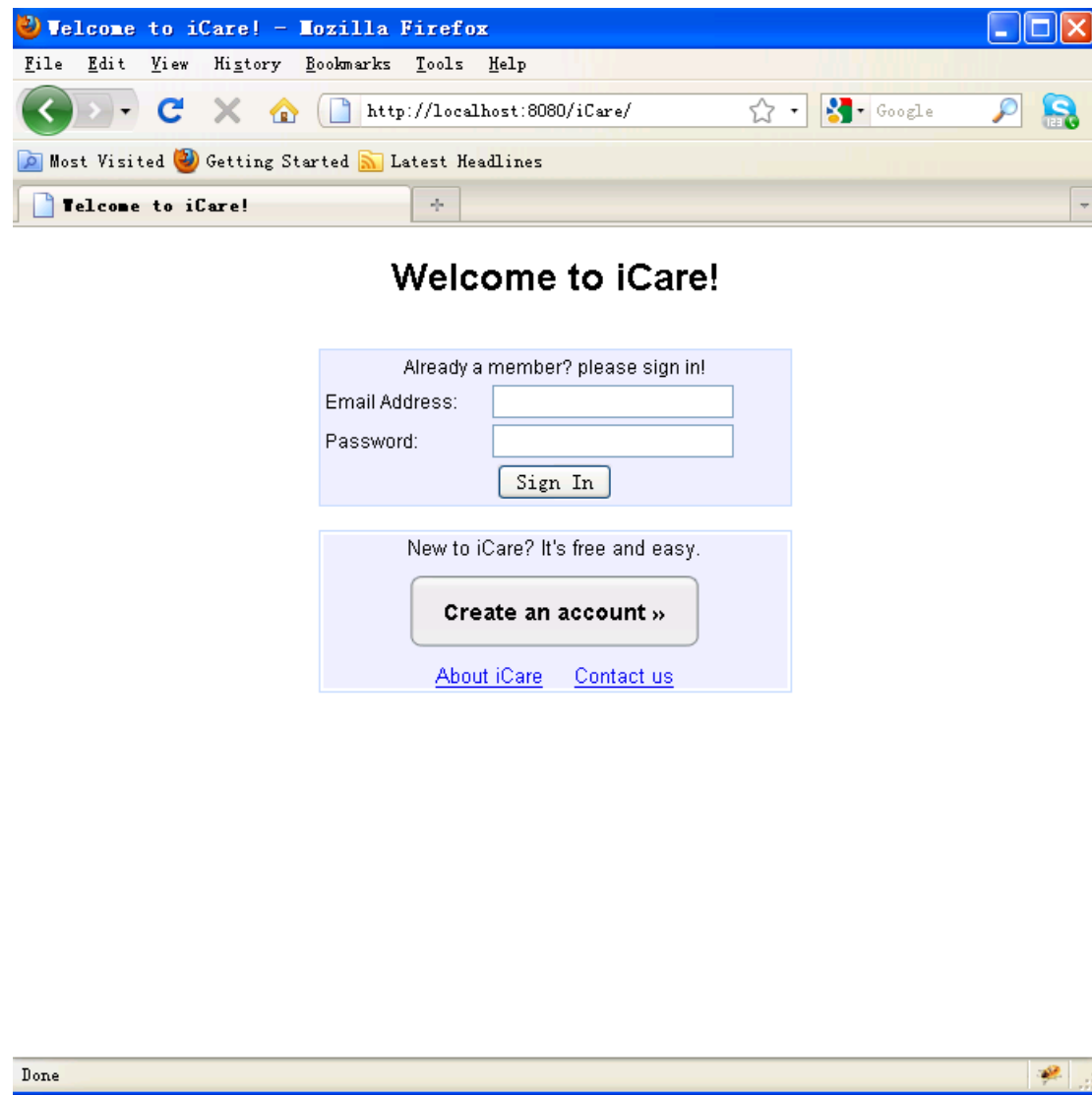


Figure 2: welcome page

The new user needs to register at the first time, after that, just need the email address and password to enter the website.

The screenshot shows a Mozilla Firefox browser window with the title "Welcome to iCare! - Mozilla Firefox". The address bar displays "http://localhost:8080/iCare/web/registr...". The page content includes a "Get started with iCare" section with the following form fields:

- Email Address: shifeng@abcd.com
- Password: (masked with three dots)
- Full Name: (empty text box)
- Phone Number: (empty text box)
- Address: (empty text box)
- Gender: ☒ Male ☐ Female
- Age: (empty text box)

Below the form fields is a button labeled "Add new patient". The browser's status bar at the bottom shows "Done".

Figure 3: register page

2.3.2 Mistake in login

If the user makes a mistake for his email address or password, he could not login the website.

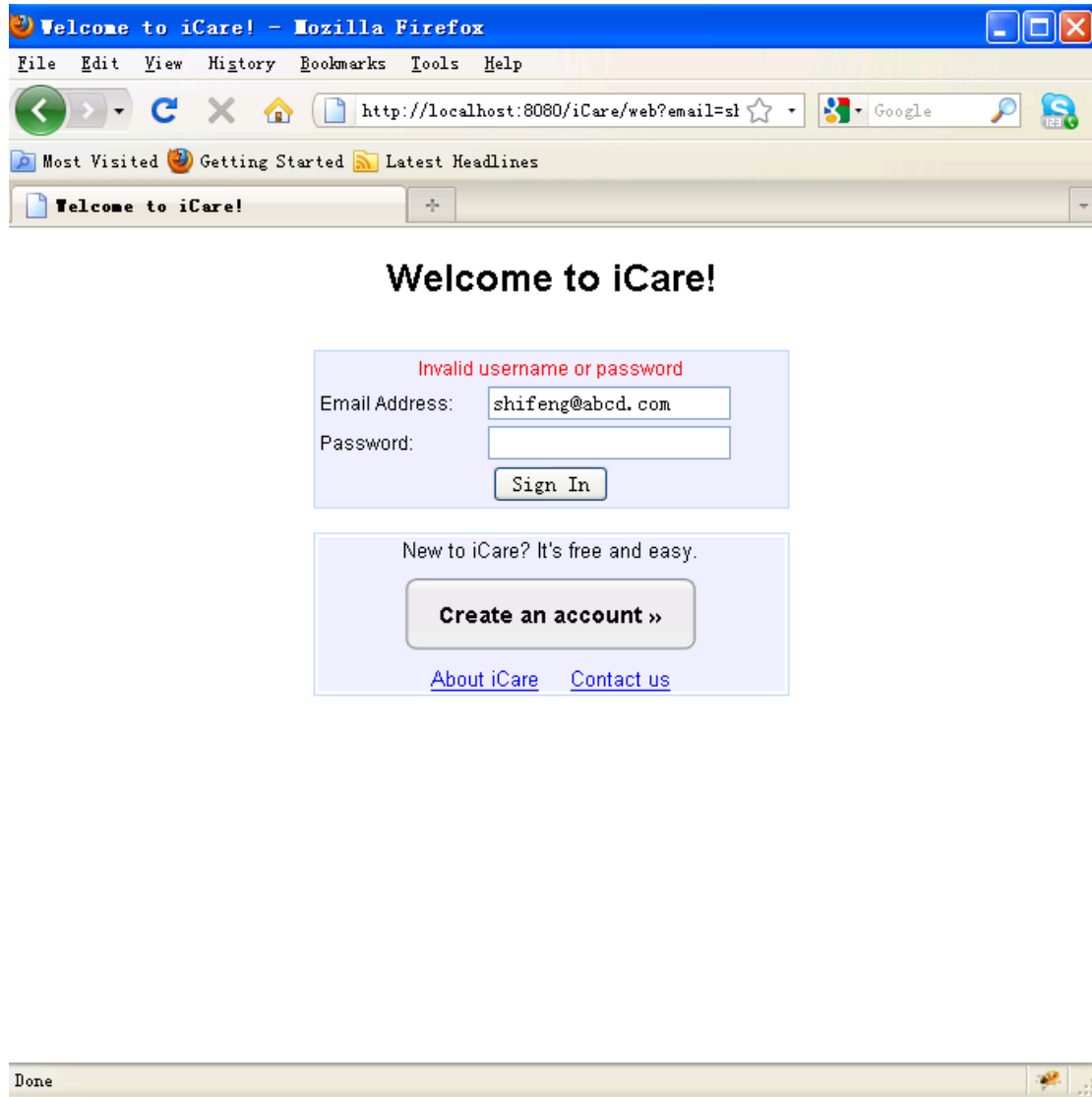


Figure 4: invalid email address or password

2.3.3 Monitor the health data and save recommendation

The following picture displays the doctor detail page. The doctor can through this page to see the patients' health data and give the recommendations.

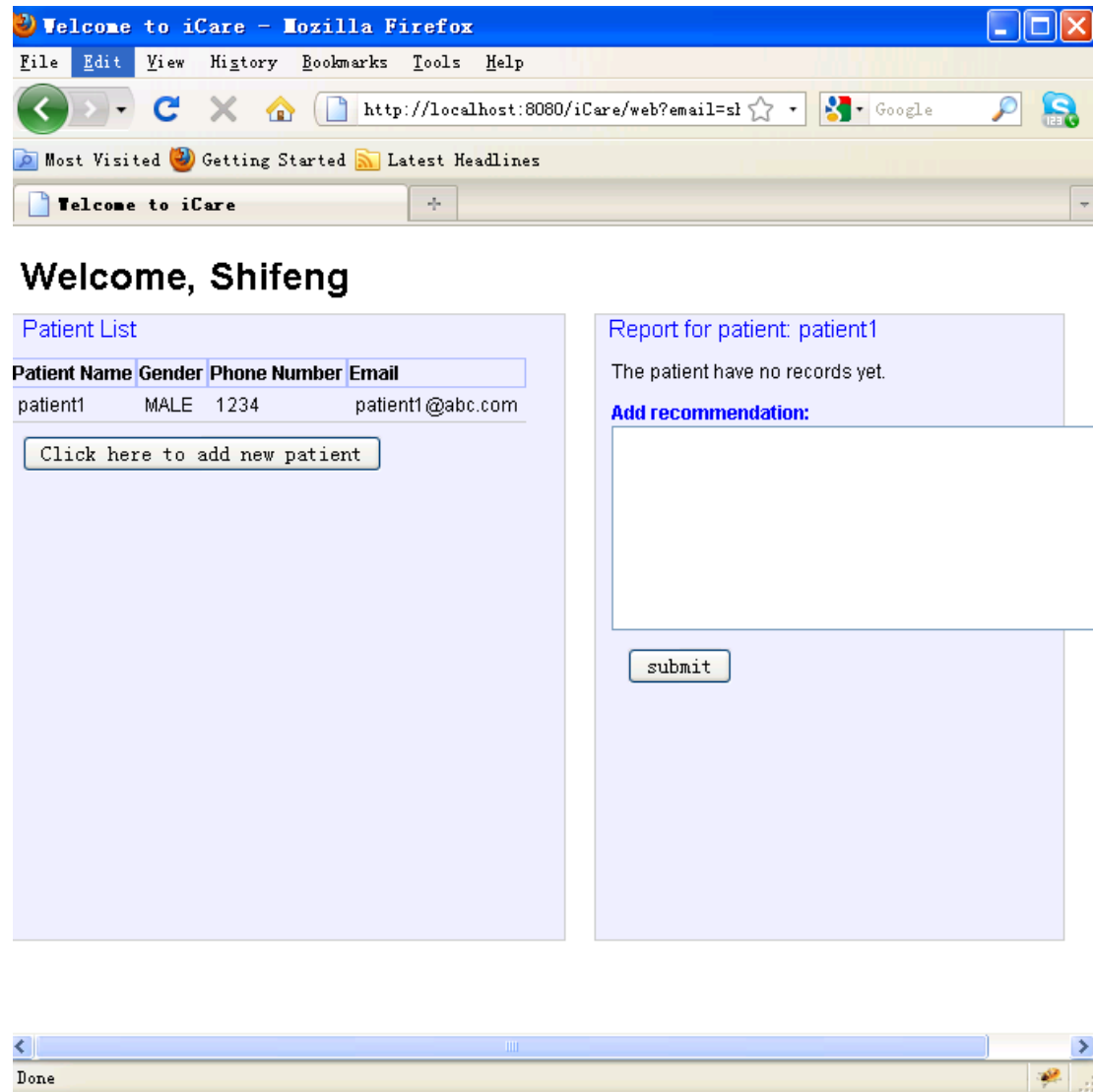


Figure 5: patient list

2.3.4 Add a new patient for the website

If the doctor has a new patient, so he can use the “Click here to add new patient” to create an account for the new patient, then the patient can send his health data to the web by himself at any time.

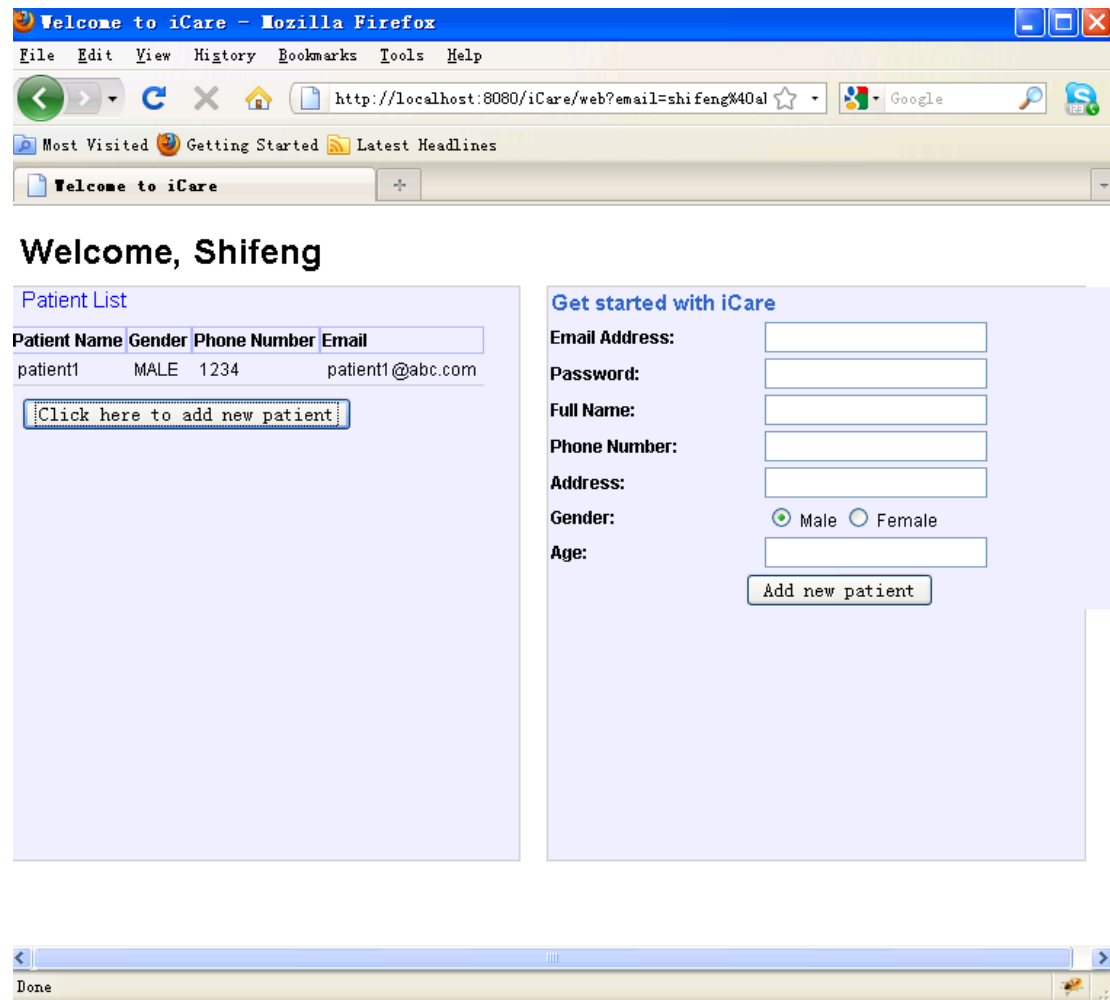


Figure 6: create an account for the new patient

2.4 Summary

This chapter described the website client for this remote health monitoring application. Currently, only the doctor can use the website to monitor the blood glucose. However, with the extension of this system to let more people to monitor a wide range of health parameters, the service should provide greater benefits to both doctors and patients.

Chapter 3 : Mobile Client

3.1 Objective

The mobile client is the first part in this project. It provides the interface through which patients can provide their health data (in this case is blood glucose) to the server (the third part in this project). Then patient store the data on the server.

3.2 Requirements

The mobile client will used by the patient to provide their health data to the server. The following figure describes the workflow within the mobile client for the patient.

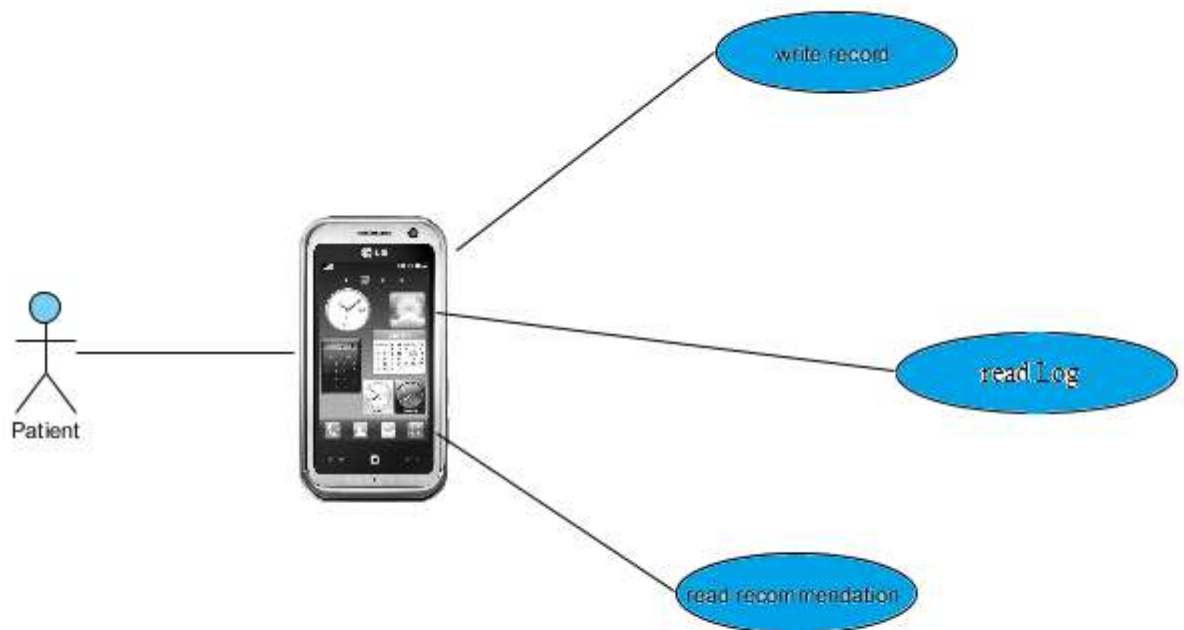


Figure 7: patient workflow

Use case: Patient writes the record on the mobile client

When: If the patient login the iCare system and wants to write his health data, he needs to go in the “New Record” from the Menu

Given: The patient writes the Record Type (for this project is just for the blood glucose), Value and notes.

Then: The patient clicks the “send” button to send them

Use case: Patient read the Log

When: The patient wants to read the Log

Then: The patient goes in the “Log” from the Menu to see his blood glucose level log

Use case: Patient read the recommendation from the doctor

When: The patient wants to see the recommendation from the doctor

Then: The patient goes in the “Recommendation” from the Menu to see the recommendation from his doctor

3.3 Technology Overview

For this part, mobile client, patients can communicate with the server and also can store the health data to the database. We use the Java 2 Micro Edition (Java 2 ME) to develop this system. J2ME is a highly optimized Java runtime environment, the market for a large number of consumer electronics devices, such as Papers, cellular phones, screen-phones, Digital set-top boxes, car navigation systems and so on. J2ME technology platform-independent Java language features and migrates to small electronic devices that allow mobile wireless devices to share applications.

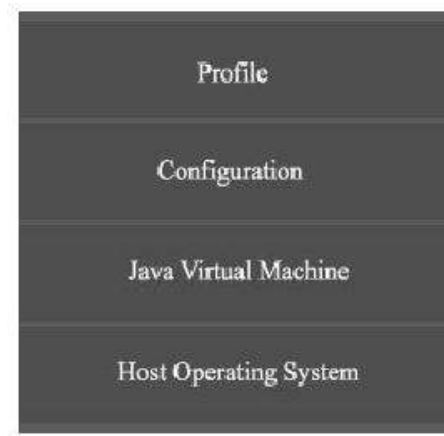


Figure 8: J2ME structure platform

We need to install some software before we use the J2ME. For J2ME, there are two develop tools, one is General development tools and the second one is Special development tools. We use the General development tools ----J2ME Wireless Toolkit. But before we install the J2ME Wireless Toolkit, we must to install the J2SDK (Java 2 Software Development Kit) first, because J2SDK has the JVM (Java Visual Machine) which J2ME Wireless Toolkit need. SDK provides tools for creating connected mobile games for a broad range of mobile devices that comply with Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC). The SDK includes the SNAP Mobile Client API libraries, an Emulation Environment application, a handset and network compatibility test tool, and sample applications and documentation that can be accessed from any Java Integrated Developer Environment (IDE).

3.4 Methodology and Workflow

The following figure describes the workflow within the mobile client for the patient.

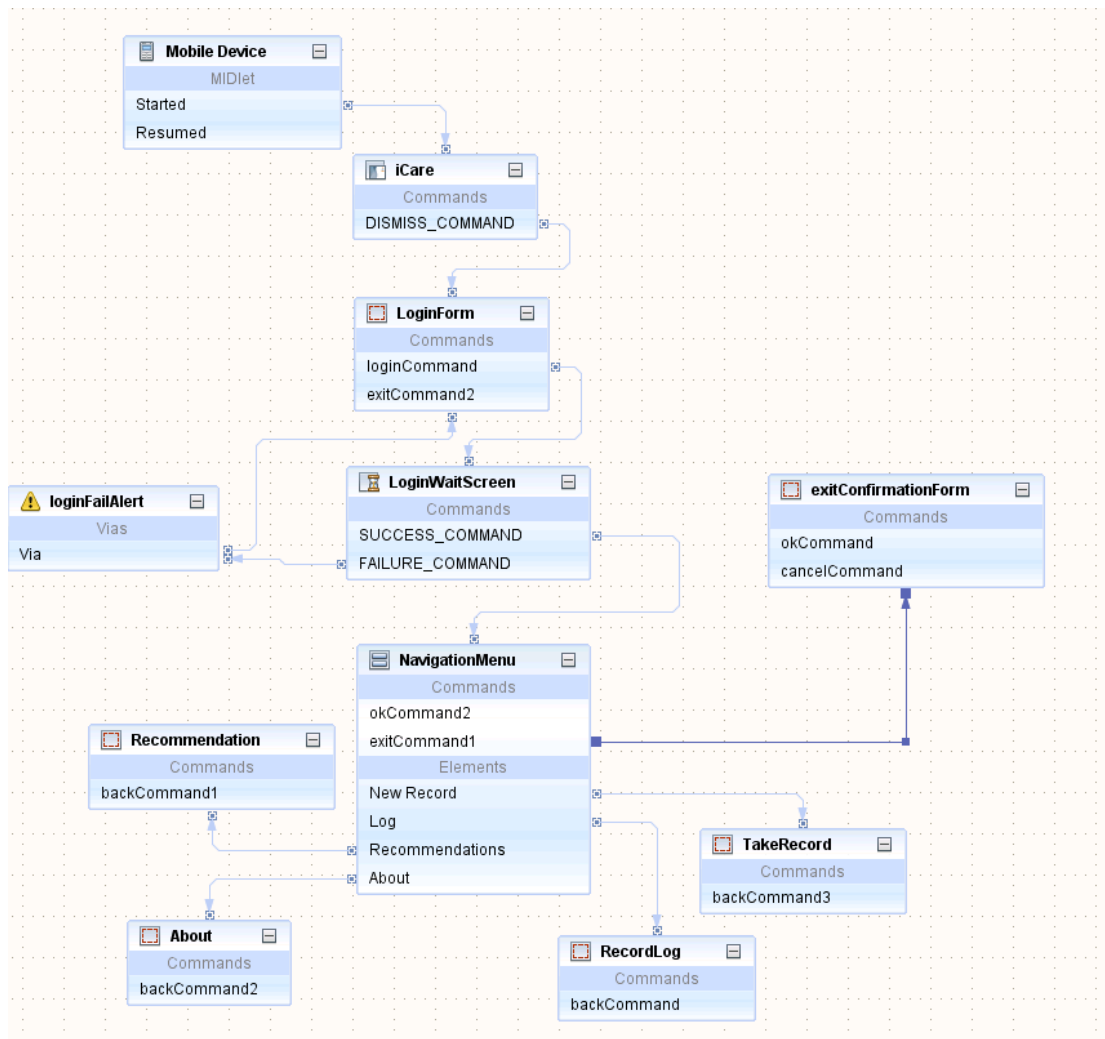


Figure 9: workflow

3.4.1 Welcome page and login page

Upon navigating to the website, the user is presented with the home page as displayed in the figure below.



Figure 10: Welcome Splash screen

For the patient wants login, he needs to enter the correct “Phone Number” and ”Password”.

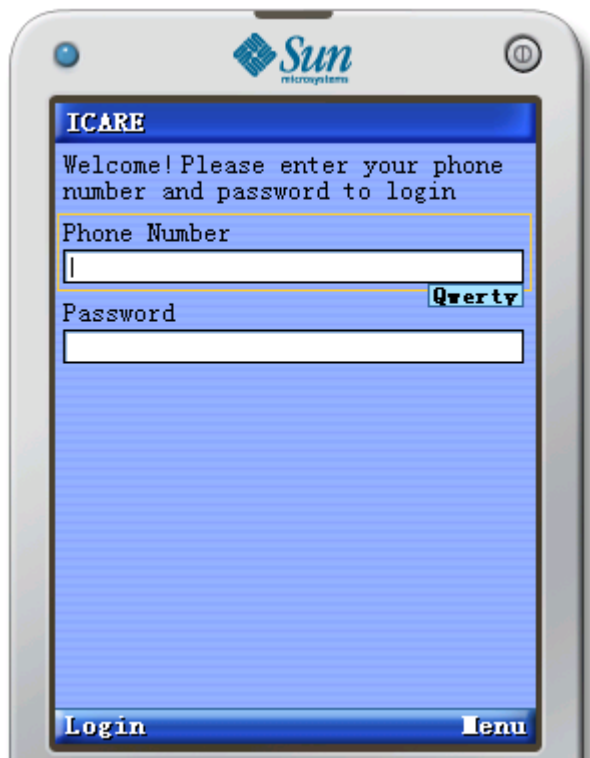


Figure 11: login page

3.4.2 Login Wait and Failed Screen

The patient needs to wait some seconds to enter the system. And if he inputs a incorrect phone number or password, the screed will show “invalid username of password” to let patient know he makes a mistake for the inputs.



Figure 12: Login Wait Screen



Figure 13: login failed Splash screen

3.4.3 The Main Menu

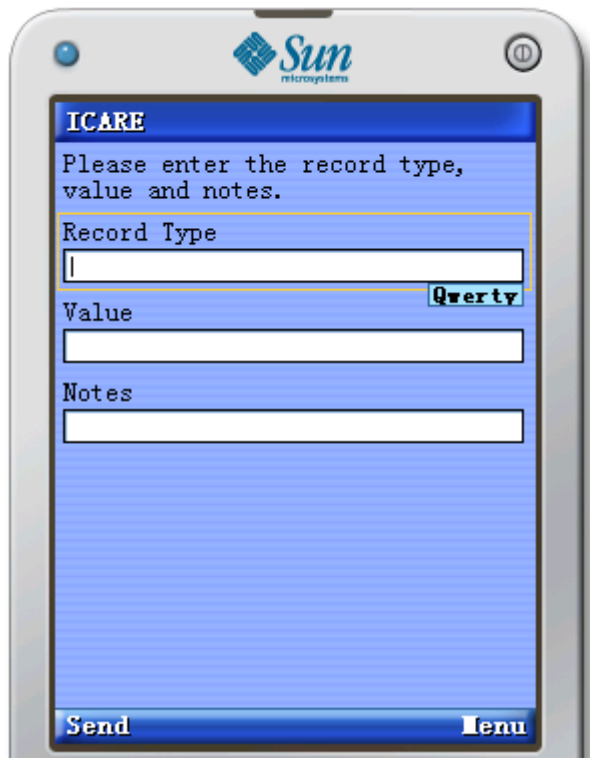
This is the Main Menu for the patient to choose what he wants to do.



Figure 14: Menu

3.4.4 New Record

Patient could write his health data from this site and send them to the database to show them to the doctor.



ICARE

Please enter the record type,
value and notes.

Record Type

Value **Query**

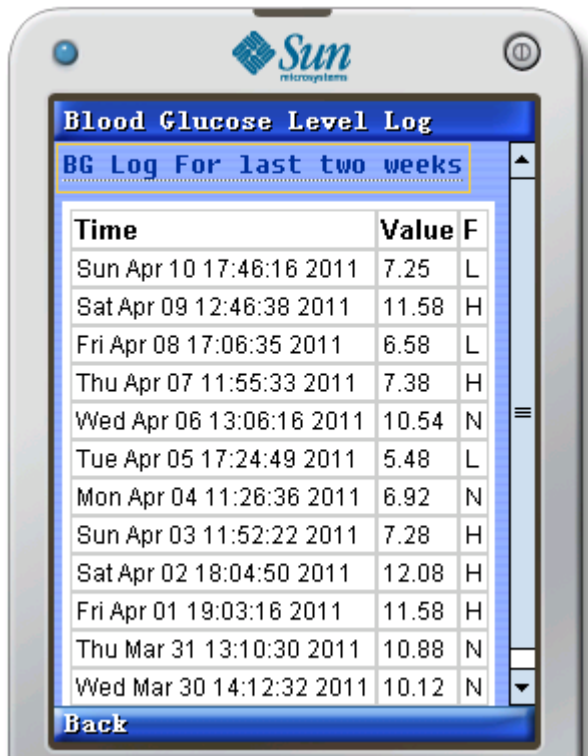
Notes

Send **Menu**

Figure 15: New record input form

3.4.5 Log

Patient can see his blood glucose log from this site.



Blood Glucose Level Log
BG Log For last two weeks

Time	Value	F
Sun Apr 10 17:46:16 2011	7.25	L
Sat Apr 09 12:46:38 2011	11.58	H
Fri Apr 08 17:06:35 2011	6.58	L
Thu Apr 07 11:55:33 2011	7.38	H
Wed Apr 06 13:06:16 2011	10.54	N
Tue Apr 05 17:24:49 2011	5.48	L
Mon Apr 04 11:26:36 2011	6.92	N
Sun Apr 03 11:52:22 2011	7.28	H
Sat Apr 02 18:04:50 2011	12.08	H
Fri Apr 01 19:03:16 2011	11.58	H
Thu Mar 31 13:10:30 2011	10.88	N
Wed Mar 30 14:12:32 2011	10.12	N

Back

Figure 16: Log page

3.4.6 Recommendation

The patients can read the doctor's recommendation from this site.

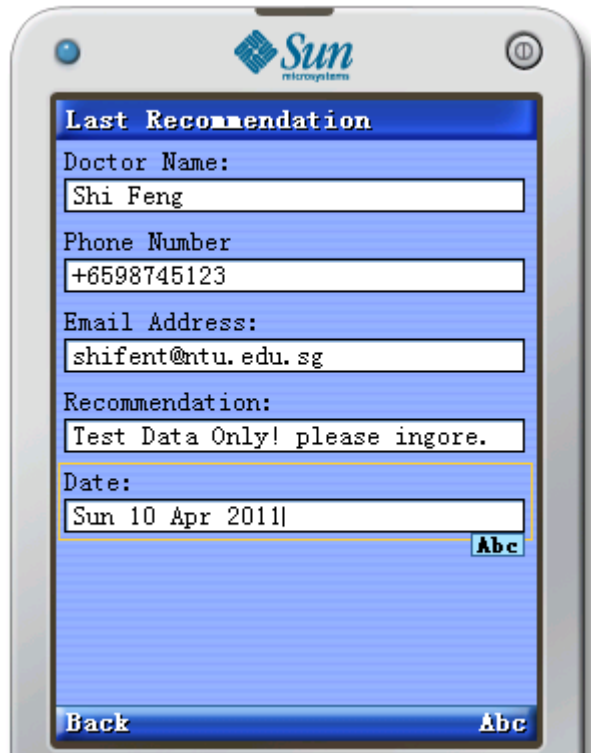


Figure 17: View last Recommendation page

3.5 Summary

This mobile client makes patients more conscious of their health due to more frequent and convenient monitoring. And the mobile client has been designed using the standard high level API's of Java ME, it can be ported across most mobile phones which support JAVA.

Chapter 4: Server

4.1 Objective

This chapter aims at providing a brief description of the implementation of server side, including overview of the technologies applied and design documentations and details.

When in operation, the server provides the services for the mobile client to send medical record, view report and so on. Also it hosts the website for the doctor to view the patient's information and give recommendation.

4.2 Requirements

The following figure illustrates Entity-Relationship Diagram for the project:

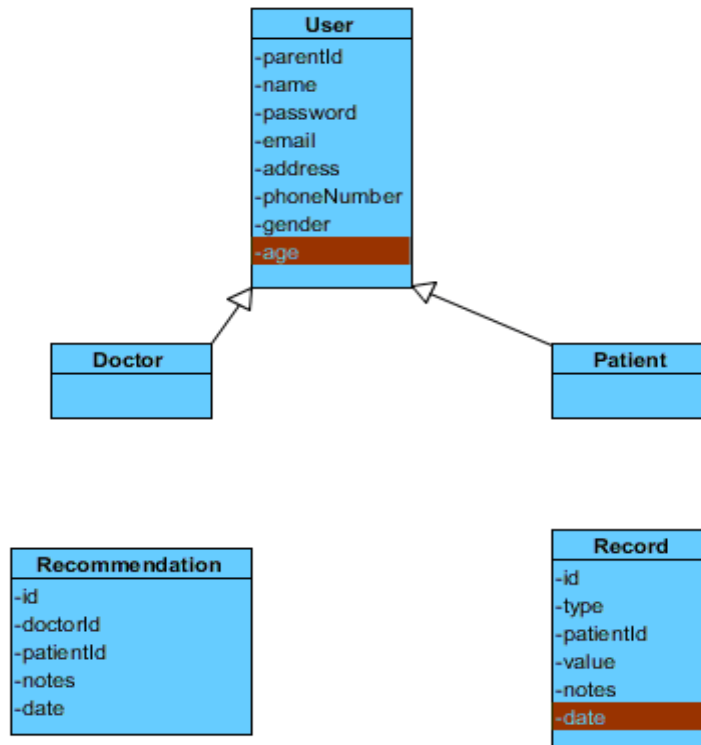


Figure 18:Entity-Relationship Diagram for the project

In this project, we define two type users, one is doctor and another one is patient. The server acts as a communication medium between the web client which the doctors use to monitor patients' health data and the mobile client which the patients use to send the health data.

For the use case in the web client part:

Use case: Doctor Register an account for himself for the iCare website.

When: The doctor wants to register a new Account at iCare

Given: He inputs all the necessary information such as email address and password, full name, phone number, address, gender, age in proper format

Then: When he clicks the "Sign Up" button, the application should create a new account for him

And: The website navigates to the "Login in" page.

Use case: Doctor Login to iCare website

When: The doctor wants to enter the iCare website

Given: Doctor inputs his email address and password

Then: When he clicks the "Sign In" button, if the email address and password are correct, the doctor enters the website. Otherwise the website will show "invalid email address and password" to let the user login again

And: The website navigates to the "patient list" page

Use case: Doctor creates a new account for his patient

When: The doctor has a new patient and wants to create an account for him

Given: He inputs the patient's necessary information such as email address and password, full name, phone number, address, gender, age in proper format

Then: he clicks "click here to add new patient" button, the application will create a new account for his patient

And: The website navigates to the "patient list" page and increases the new

patient to the list

Use case: Doctor monitors the patient's health data

When: The doctor wants to monitor the patient's health data

Then: he clicks the patient who is in the table to monitor the patient's data

Use case: Doctor writes and save the recommendation for the patient

When: After the doctor reads all the health data from the patient

Given: The doctor write the recommendation for the patient

Then: He clicks "submit" to save the recommendation on the iCare website

For the use case in the mobile client part:

Use case: Patient writes the record on the mobile client

When: If the patient login the iCare system and wants to write his health data, he needs to go in the "New Record" from the Menu

Given: The patient writes the Record Type (for this project is just for the blood glucose), Value and notes.

Then: The patient clicks the "send" button to send them

Use case: Patient read the Log

When: The patient wants to read the Log

Then: The patient goes in the "Log" from the Menu to see his blood glucose level log

Use case: Patient read the recommendation from the doctor

When: The patient wants to see the recommendation from the doctor

Then: The patient goes in the "Recommendation" from the Menu to see the recommendation from his doctor

4.3 Technology Overview

GlassFish is used as the web application container. It is an open source application server project led by Sun Microsystems for the Java EE platform. The proprietary version is called Sun GlassFish Enterprise Server. GlassFish is free software, dual-licensed under two free software licenses: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception.

GlassFish is based on source code released by Sun and Oracle Corporation's TopLink persistence system. It uses a derivative of Apache Tomcat as the servlet container for serving Web content, with an added component called Grizzly which uses Java New I/O (NIO) for scalability and speed.

I write the program using NetBeans IDE. NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others.

4.5 Project Structure

The project is constructed following three-tier architecture.

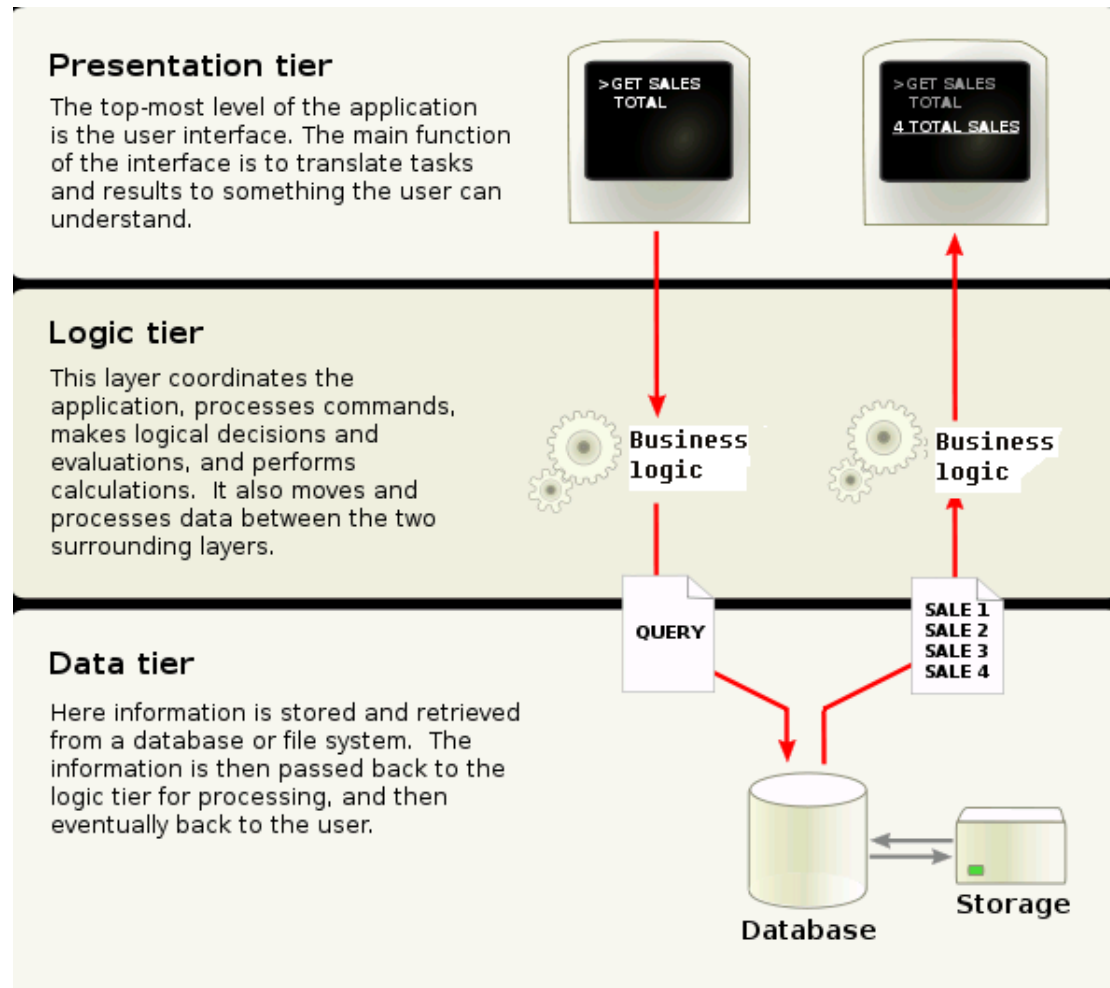


Figure 19: three-tier architecture

As the business logic in the project is not complicated the database and the web server operates on the same host.

Following the conversion of spring framework, the project contains three logical layers: Entity Layer, DAO layer, and the service Layer.

The entity layer defines all the entities in the business domain and the DAP layer hosts the data access object and the service layer focuses on the services provided to the clients.

The directory structure is as follows:

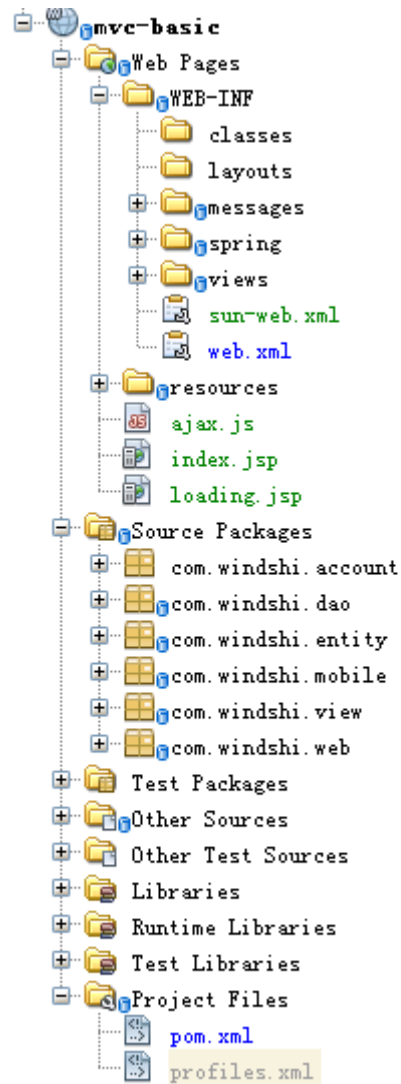


Figure 20: the directory structure

4.5 Spring MVC framework

The project utilizes the Spring Framework to build the web server and web client. For complete spring reference, please refer to <http://www.springsource.com> .

The Spring Framework is an open source application framework for the J2EE platform. Since the first version of this article was published in October, 2003, the Spring Framework has steadily grown in popularity.

Spring acts as a bean factory and a container, which is responsible for managing object lifecycles: creating objects, calling initialization methods, and configuring objects by wiring them together.

The MVC stands for “Model-View-Controller”. Spring MVC Clearly separates business, navigation and presentation logic. It provides a proven mechanism for building a thin, clean web-tier.

The Controller is user created component for handling requests and encapsulates navigation logic.

This project provides two set of controller, one set for the web client and the other for mobile client.

The view is created by the controller and stores the Model data.

The controller of the web service produces view with standard JSP, JSTL (Standard Tag Library) and JavaScript technologies, while the controller for the mobile service produces view of plain text in JSON format, which will be consumed by the mobile client.

4.5.1 Configuring web.xml

The web.xml file defines each servlet and JSP page within a Web Application. It also enumerates enterprise beans referenced in the Web application.

As for the project, the Servlet Dispatcher has a url-pattern of "/", which means all the request that contains the root context in the url will be handled by the dispatcher.

The configuration also includes locale handling for future usage.

The web.xml is configured as **Appendix I**.

4.5.2 Application container configuration

The application context is a bean factor which instantiates, configures, and manages a number of beans. It also defines interceptors, including view-resolver, locale resolver.

The application context is configured as **Appendix II**.

4.5.3 Data source and JDBC configuration

The data source and JDBC configuration is defined separately and imported in the main application context. This arrangement is for separation of concern.

The data source and JDBC templates is configured as **Appendix III**.

4.5.4 Project build tool

This project takes advantages of Apache Maven to manage the project dependencies and facilitate the build, package process.

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. The complete reference can be found at <http://maven.apache.org/>.

The detailed POM file is attached as **Appendix IV**.

4.6 Database design

The database for the project is set up using MySQL tool, which is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. Standard Structured Query Language is used to communicate with the database.

1. Based on the Entity Relationship Diagram, three tables are created : user, record and recommendation.

User table is to store the user information for both doctors and patients. And the record table stores all the record sent by patients form Mobile clients. The recommendation tables stores recommendations given by doctors.

The table definitions for each table are illustrated as below.

```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| recommendation |
| record          |
| user            |
+-----+
```

Figure 21: Tables in Database

```
mysql> describe user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| name       | varchar(40)   | YES  |     | NULL    |                |
| email      | varchar(100)  | YES  |     | NULL    |                |
| address    | varchar(100)  | YES  |     | NULL    |                |
| phoneNumber | varchar(20)   | YES  |     | NULL    |                |
| gender     | varchar(2)    | YES  |     | NULL    |                |
| age        | tinyint(4)    | YES  |     | NULL    |                |
| password   | varchar(20)   | YES  |     | NULL    |                |
| parentId   | int(11)       | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.55 sec)
```

Figure 22: Table definition for table user

```
mysql> describe record;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default          | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL            | auto_increment |
| type       | varchar(10)   | YES  |     | NULL            |                |
| patientId  | int(11)       | YES  |     | NULL            |                |
| timestamp  | timestamp     | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| notes      | varchar(30)   | YES  |     | NULL            |                |
| value      | varchar(10)   | NO   |     | NULL            |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.05 sec)
```

Figure 23: Table definition for table record

```
mysql> describe recommendation;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default          | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL            | auto_increment |
| doctorId   | int(11)       | NO   |     | NULL            |                |
| patientId  | int(11)       | NO   |     | NULL            |                |
| notes      | varchar(100)  | NO   |     | NULL            |                |
| timestamp  | timestamp     | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

Figure 24: Table definition for table recommendation

References

[1] Blood glucose

<http://news.xin.msn.com/zh/singapore/article.aspx?cp-documentid=4670428>

[2] Glassfish

<http://java.sun.com/developer/community/askxpert/2005/j11114.html>

<http://en.wikipedia.org/wiki/GlassFish>

[3] Web Services Architecture

<http://www.w3.org/TR/ws-arch>

[4] Java Micro Edition

<http://java.sun.com/javame/technology/index.jsp>

[5] Connected Limited Device Configuration (CLDC)

<http://java.sun.com/products/cldc/>

[6] NetBeans IDE – Java ME Development

<http://www.netbeans.org/features/javame/index.html>

Appendix

Source Code Listing

Appendix V

Welcome:

```
<% @page contentType="text/html; charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ page session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="{pageContext.request.contextPath}" />
<html>
    <head>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Welcome to iCare!</title>
        <link rel="stylesheet" href="<c:url value="/resources/blueprint/layout.css" />"
type="text/css" media="screen, projection">
        <!--[if lt IE 8]>
        <link rel="stylesheet" href="<c:url value="/resources/blueprint/ie.css" />"
type="text/css" media="screen, projection">
        <![endif]-->
    </head>
    <body>
        <h1>
            <div align="center"><fmt:message key="welcome.title"/></div>
        </h1>
        <br/>
        <div align="center" >
            <form action="{context}/web">
                <table class="form-noindent loginForm" width="300px">
                    <c:if test="{not empty(sucess)}">
                        <tr>
                            <td colspan="99" class="errorMessage">{sucess}</td>
                        </tr>
                    </c:if>
                    <c:if test="{not empty(error)}" var="hasError">
                        <tr>
                            <td colspan="99" class="errorMessage">{error}</td>
                        </tr>
                    </c:if>
                    <c:if test="{!hasError}">
                        <tr>
                            <td colspan="99" style="text-align: center">Already a
```

```
member? please sign in!</td>
    </tr>
</c:if>
<tr>
    <td>
        Email Address:
    </td>
    <td>
        <input type="text" name="email"/>
    </td>
</tr>

<tr>
    <td>
        Password:
    </td>
    <td>
        <input type="password" name="password"/>
    </td>
</tr>
<tr>
    <td colspan="2" style="text-align: center"> <input
type="submit" value="Sign In"/></td>
</tr>
</table>
</form>

<br/>

<table class=form-noindent cellpadding=0 bgcolor=#E8EEFA
width="300px">
    <tr bgcolor=#E8EEFA>
        <td valign=top>
            <div align=center style="margin:10 0">
                <font size="-1">New to iCare? It's free and easy.</font>
                <table cellpadding=0 cellspacing=0 align=center
class="signup_btn"><tr>
                    <td
class="SPRITE_signup_button_grey_l"></td>
                    <td class="SPRITE_signup_button_grey_m"><a
class="signup_btn_link" href="{context}/web/registration">
                        Create an account &#187;
```


[illegible]

My patient:

```
<%@page contentType="text/html; charset=UTF-8"%>
<%@page pageEncoding="UTF-8"%>
<%@ page session="true" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="{pageContext.request.contextPath}" />
<c:set var="doctorId" value="{doctor.id}" scope="session" />
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Welcome to iCare</title>
    <link rel="stylesheet" href="{c:url value="/resources/blueprint/layout.css" />
type="text/css" media="screen, projection">
    <!--[if lt IE 8]>
    <link rel="stylesheet" href="{c:url value="/resources/blueprint/ie.css" />
type="text/css" media="screen, projection">
    <![endif]-->
    <script type="text/javascript" >
      var xmlhttp

      function loadContent(url, callbackFunction)
      {

        xmlhttp=GetXmlHttpRequestObject();

        if (xmlhttp==null)
        {
          alert ("Your browser does not support Ajax HTTP");
          return;
        }

        var url="web/" + url;

        xmlhttp.onreadystatechange=callbackFunction;
        xmlhttp.open("GET",url,true);
        xmlhttp.send(null);
      }

      function onGetPatientList()
      {
        if (xmlhttp.readyState==4)
        {
          document.getElementById("patient-list").innerHTML=xmlhttp.responseText;
        }
      }

      function onGetPatientRecord()
```

```
{
    if (xmlhttp.readyState==4)
    {

document.getElementById("right-page").innerHTML=xmlhttp.responseText;
    }
}

function onAddPatient()
{
    if (xmlhttp.readyState==4)
    {

document.getElementById("right-page").innerHTML=xmlhttp.responseText;
    }
}

function GetXmlHttpRequestObject()
{
    if (window.XMLHttpRequest)
    {
        return new XMLHttpRequest();
    }
    if (window.ActiveXObject)
    {
        return new ActiveXObject("Microsoft.XMLHTTP");
    }
    return null;
}
</script>
</head>
<body>
    <h1>
        <div align="left">Welcome, ${doctor.name}</div>
    </h1>
    <div class="my-patients">
        <div id="patient-list">
            <jsp:include page="patientList.jsp" />
        </div>
        <div id="right-page">
            Click on the patient to view the record statistics.
        </div>
    </div>
</div>
```

```
</body>
</html>
```

Patient list:

```
<% @page contentType="text/html; charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ page session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="${pageContext.request.contextPath}" />
<div>
    <span style="font-size: larger; color: blue">&nbsp; Patient List</span>

    <table style="margin-top: 10px " cellspacing="0" border="0">
        <thead>
            <td class="header">Patient Name</td>
            <td class="header">Gender</td>
            <td class="header">Phone Number</td>
            <td class="header">Email</td>
        </thead>
        <c:if test="${not empty list}" var="hasPatient">

            <c:forEach items="${list}" var="patient">
                <tr onClick="javascript:loadContent('listRecord?patientId=${patient.id}',
onGetPatientRecord)">
                    <td class="item">${patient.name}</td>
                    <td class="item">${patient.gender}</td>
                    <td class="item">${patient.phoneNumber}</td>
                    <td class="item">${patient.email}</td>
                </tr>
            </c:forEach>
        </c:if>
        <c:if test="${!hasPatient}">
            <tr>
                <td colspan="99" class="empty-message"> You have no patients
yet.</td>
            </tr>
        </c:if>
```

```
</table>
<div class ="button-add">
    <button      type="button"      onClick="javascript:loadContent('registration',
onAddPatient)">Click here to add new patient</button>
</div>
</div>
```

Registration Form:

```
<% @page contentType="text/html; charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ page session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="${pageContext.request.contextPath}" />
<html>
    <head>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Welcome to iCare!</title>
        <link rel="stylesheet" href="<c:url value="/resources/blueprint/layout.css" />"
type="text/css" media="screen, projection">
        <!--[if lt IE 8]>
        <link rel="stylesheet" href="<c:url value="/resources/blueprint/ie.css" />"
type="text/css" media="screen, projection">
        <![endif]-->
    </head>
    <body>
        <script type="text/javascript">
        </script>
        <div align="center">
            <form action="${context}/web/registerUser"
                method="post" onsubmit="return(onPreCreateAccountSubmit());">
                <c:if test="${sessionScope.doctorId!= null}" var="doctor">
                    <c:set var="userType" value="doctor" scope="session" />
                    <input type="hidden" name="doctorId" value
                        ="${sessionScope.doctorId}"/>
                    </c:if>
                    <c:if test="${!doctor}">
                        <c:set var="userType" value="patient" scope="session" />
                    </c:if>
```

```

        <table id = "registrationForm" cellpadding="2" bgcolor="#e8eefa"
        cellspacing="0" border="0" width="400px">
            <tr><td colspan="2">
                <span style="color: #3366CC; font-weight: bold; font-size:
larger">Get started with iCare<br/>
                </span>
            </td></tr>
            <tr>
                <td class = "register-label"> Email Address: </td>
                <td> <input type="text" maxlength="30" name="email"/>
            </td>
            </tr>
            <tr>
                <td class = "register-label"> Password: </td>
                <td> <input type="password" maxlength="20"
name="password"/> </td>
            </tr>
            <tr>
                <td class = "register-label"> Full Name: </td>
                <td> <input type="text" name="name"/> </td>
            </tr>
            <tr>
                <td class = "register-label"> Phone Number: </td>
                <td> <input type="text" maxlength="15"
name="phoneNumber"/> </td>
            </tr>
            <tr>
                <td class = "register-label"> Address: </td>
                <td> <input type="text" maxlength="15" name="address"/>
            </td>
            </tr>
            <tr>
                <td class = "register-label"> Gender: </td>
                <td> <input type="radio" name="gender" value="M"
checked > Male
                <input type="radio" name="gender" value="F"> Female
            </td>
            </tr>
            <tr>
                <td class = "register-label"> Age: </td>
                <td> <input type="text" maxlength="2" name="age"/> </td>
            </tr>
            <tr>
                <td colspan="2" style="text-align: center"> <input

```

```
type="submit" value='${userType eq "doctor" ? "Add new patient" : "Sing Up"}'/></td>
</tr>

</table>
</form>

</div>
</body>
</html>
```


Appendix VI

Account:

```
package com.windshi.account;
```

```
import java.math.BigDecimal;
```

```
import java.util.Date;
```

```
import java.util.concurrent.atomic.AtomicLong;
```

```
import javax.validation.constraints.Future;
```

```
import javax.validation.constraints.NotNull;
```

```
import javax.validation.constraints.Size;
```

```
import org.springframework.format.annotation.DateTimeFormat;
```

```
import org.springframework.format.annotation.NumberFormat;
```

```
import org.springframework.format.annotation.NumberFormat.Style;
```

```
public class Account {
```

```
    private Long id;
```

```
    @NotNull
```

```
    @Size(min=1, max=25)
```

```
    private String name;
```

```
    @NotNull
```

```
    @NumberFormat(style=Style.CURRENCY)
```

```
    private BigDecimal balance = new BigDecimal("1000");
```

```
    @NotNull
```

```
    @NumberFormat(style=Style.PERCENT)
```

```
    private BigDecimal equityAllocation = new BigDecimal(".60");
```

```
    @DateTimeFormat(style="S-")
```

```
    @Future
```

```
    private Date renewalDate = new Date(new Date().getTime() + 31536000000L);
```

```
    public Long getId() {
```

```
        return id;
```

```
    }
```

```
    void setId(Long id) {
```

```
        this.id = id;
```

```
    }
```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public BigDecimal getBalance() {
    return balance;
}

public void setBalance(BigDecimal balance) {
    this.balance = balance;
}

public BigDecimal getEquityAllocation() {
    return equityAllocation;
}

public void setEquityAllocation(BigDecimal equityAllocation) {
    this.equityAllocation = equityAllocation;
}

public Date getRenewalDate() {
    return renewalDate;
}

public void setRenewalDate(Date renewalDate) {
    this.renewalDate = renewalDate;
}

Long assignId() {
    this.id = idSequence.incrementAndGet();
    return id;
}

private static final AtomicLong idSequence = new AtomicLong();
}
```

```
Account controller:
package com.windshi.account;

import com.windshi.dao.DBManager;
import com.windshi.view.TextResponse;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping(value = "/account")
public class AccountController {

    private Map<Long, Account> accounts = new ConcurrentHashMap<Long, Account>();
    @Autowired
    private DBManager dao;

    @RequestMapping(method = RequestMethod.GET)
    public String getcreateForm(Model model) {
        model.addAttribute(new Account());
        return "account/createForm";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String create(@Valid Account account, BindingResult result) {
        if (result.hasErrors()) {
            return "account/createForm";
        }
        this.accounts.put(account.assignId(), account);
        return "redirect:/account/" + account.getId();
    }

    @RequestMapping(value = "{id}", method = RequestMethod.GET)
    public String getView(@PathVariable Long id, Model model) {
        Account account = this.accounts.get(id);
```

```
        if (account == null) {  
            throw new ResourceNotFoundException(id);  
        }  
        model.addAttribute(account);  
        return "account/view";  
    }  
}
```

Resource not found exception:

```
package com.windshi.account;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value=HttpStatus.NOT_FOUND)  
public class ResourceNotFoundException extends RuntimeException {  
  
    private Long resourceId;  
  
    public ResourceNotFoundException(Long resourceId) {  
        this.resourceId = resourceId;  
    }  
  
    public Long getResourceId() {  
        return resourceId;  
    }  
  
}
```

Data base management:

```
package com.windshi.dao;  
  
import com.windshi.entity.Patient;  
import com.windshi.entity.Recommendation;  
import com.windshi.entity.Record;  
import com.windshi.entity.User;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;
```

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

public class DBManager {

    @Autowired
    @Qualifier("jdbcTemplate_USP")
    private JdbcTemplate jdbcTemplate_USP;

    public JdbcTemplate getJdbcTemplate_USP() {
        return jdbcTemplate_USP;
    }

    public void addUser(User entity) {
        jdbcTemplate_USP.update("INSERT INTO user VALUES (?, ?, ?, ?, ?, ?, ?, ?) ",
            new Object[]{0, entity.getName(), entity.getEmail(), entity.getAddress(),
                entity.getPhoneNumber(), entity.getGender().code(), entity.getAge(),
                entity.getPassword(), entity.getParentId()});
    }

    public void addRecord(Record entity) {
        //insert into record values(0, 'BG', 1, '2011-04-07 ', 'the first time measure ')
        jdbcTemplate_USP.update("INSERT INTO record VALUES (?, ?, ?, ?, ?) ",
            new Object[]{0, entity.getType(), entity.getPatientId(), entity.getDate(),
                entity.getNotes()});
    }

    public void addRecommendation(Recommendation entity) {
        // insert into recommendation values(0, 1, 1, 'the first time consultation
        //', '2011-04-08');
        jdbcTemplate_USP.update("INSERT INTO recommendation VALUES (?, ?, ?, ?, ?) ",
            new Object[]{0, entity.getDoctorId(), entity.getPatientId(),
                entity.getNotes(),
                entity.getDate()});
    }

    /**
     * list all the patients under the doctor
     * @param doctorId
     * @return
     */
}
```

```
    */
    public List<Patient> listPatient(int doctorId) {
        if (doctorId <= 0) {
            throw new IllegalStateException("invalid doctorId ");
        }
        return jdbcTemplate_USP.query("SELECT * FROM user WHERE parentId = ? ",
new Object[]{doctorId}, Patient.rowMapper);
    }

    /**
     * list record by patientId
     * @param patientId
     * @return
     */
    public List<Record> listRecord(int patientId) {
        return jdbcTemplate_USP.query("SELECT * FROM record WHERE patientId = ? ",
new Object[]{patientId}, Record.rowMapper);
    }

    /**
     * list record by patientId
     * @param patientId
     * @return
     */
    public List<Recommendation> listConsultation(int patientId) {
        return jdbcTemplate_USP.query("SELECT * FROM recommendation WHERE
patientId = ? ", new Object[]{patientId}, Recommendation.rowMapper);
    }

    /**
     * list record by patientId
     * @param patientId
     * @return
     */
    public List<Recommendation> listRecommendation(int doctorId) {
        return jdbcTemplate_USP.query("SELECT * FROM recommendation WHERE
doctorId = ? ", new Object[]{doctorId}, Recommendation.rowMapper);
    }

    public Patient getPatient(int patientId) {
        final List<Patient> list = jdbcTemplate_USP.query("SELECT * FROM user
WHERE id = ? ", new Object[]{patientId}, Patient.rowMapper);
        if (list.isEmpty()){
            return null;
        }
    }
}
```

```
    }  
    return list.get(0);  
  }  
}
```

Doctor:

```
/*  
 * To change this template, choose Tools | Templates  
 * and open the template in the editor.  
 */  
package com.windshi.entity;  
  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import org.springframework.jdbc.core.RowMapper;  
  
/**  
 *  
 * @author Babie  
 */  
public class Doctor extends User {  
  
    public Doctor(String name, String password, String email, String address, String  
phoneNumber, String gender, int age) {  
        super(name, email, address, phoneNumber, gender, age);  
        setPassword(password);  
    }  
  
    public Doctor(int id, String name, String email, String address, String phoneNumber,  
String gender, int age) {  
        super(name, email, address, phoneNumber, gender, age);  
        setId(id);  
    }  
  
    public static final RowMapper<Doctor> rowMapper = new RowMapper<Doctor>() {  
  
        public Doctor mapRow(ResultSet rs, int rowNum) throws SQLException {  
            Doctor record = new Doctor(rs.getInt("id"), rs.getString("name"),  
rs.getString("email"), rs.getString("address"),  
rs.getString("phoneNumber"), rs.getString("gender"),  
rs.getInt("age"));  
            return record;  
        }  
    }  
}
```

```
rs.getInt("age"));
        return record;
    }
};
}
```

Patient:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Patient extends User {
/**
 * for creation
 * @param doctorId
 * @param name
 * @param password
 * @param email
 * @param address
 * @param phoneNumber
 * @param gender
 * @param age
 */
    public Patient(int doctorId, String name, String password, String email, String address,
String phoneNumber, String gender, int age) {
        super(name, email, address, phoneNumber, gender, age);
        setParentId(doctorId);
        setPassword(password);
    }

/**
 * for retrieving;
 * do not return the security information like password
 * @param name
 * @param email
 * @param address
 * @param phoneNumber
 * @param gender
 * @param age
 */
}
```



```
*/
    public Patient(int id, String name, String email, String address, String phoneNumber,
String gender, int age) {
        super(name, email, address, phoneNumber, gender, age);
        setId(id);
    }

    public static final RowMapper<Patient> rowMapper = new RowMapper<Patient>() {

        public Patient mapRow(ResultSet rs, int rowNum) throws SQLException {
            Patient record = new Patient(rs.getInt("id"), rs.getString("name"),
rs.getString("email"), rs.getString("address"),
rs.getString("phoneNumber"), rs.getString("gender"),
rs.getInt("age"));
            return record;
        }
    };
}
```

Recommendation:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Recommendation {

    // private int id; // not used in entity;
    private int doctorId; // the id of the patient who the recommendation is for
    private int patientId; // the id of the doctor who gave the recommendation.
    private String notes; // the content
    private Date date;

    public Recommendation() {
    }

    public Recommendation(int doctorId, int patientId, String notes, Date date) {
        this.doctorId = doctorId;
        this.patientId = patientId;
        this.notes = notes;
    }
}
```

```
        this.date = date;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public int getDoctorId() {
        return doctorId;
    }

    public void setDoctorId(int doctorId) {
        this.doctorId = doctorId;
    }

    public String getNotes() {
        return notes;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }

    public int getPatientId() {
        return patientId;
    }

    public void setPatientId(int patientId) {
        this.patientId = patientId;
    }

    public static final RowMapper<Recommendation> rowMapper = new
    RowMapper<Recommendation>() {

        public Recommendation mapRow(ResultSet rs, int rowNum) throws SQLException
        {
            Recommendation recommendation = new
            Recommendation(rs.getInt("doctorId"), rs.getInt("patientId"), rs.getString("notes"),
            rs.getTimestamp("timestamp"));
            return recommendation; } };
```

Record:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Record {

    //    private int id;
    /**
     * the type maybe blood pressure; glucose;
     */
    private String type;
    /**
     * the id of the patient the record refer to;
     */
    private int patientId;
    private String value; // the value of the measure
    /**
     * store the notes of the record;
     */
    private String notes;
    private Date date;

    public Record() {
    }

    public Record(int patientId, String type, String value, String notes, Date date) {
        this.type = type;
        this.patientId = patientId;
        this.notes = notes;
        this.date = date;
        this.value = value;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }
}
```

```
    }

    public String getNotes() {
        return notes;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }

    public int getPatientId() {
        return patientId;
    }

    public void setPatientId(int patientId) {
        this.patientId = patientId;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public static final RowMapper<Record> rowMapper = new RowMapper<Record>() {

        public Record mapRow(ResultSet rs, int rowNum) throws SQLException {
            Record record = new Record(rs.getInt("patientId"), rs.getString("type"),
rs.getString("value"), rs.getString("notes"), rs.getTimestamp("timestamp"));
            return record;
        }
    };
}
```

User:

```
package com.windshi.entity;
```

```
/**
```

```
 * base class for Doctor and Patient
```

```
 * @author Babie
```

```
*/  
public abstract class User {  
  
    private int id;// key  
    private int parentId;  
    private String name;  
    private String password;  
    private String email;  
    private String address;  
    private String phoneNumber;  
    private Gender gender;  
    private int age;  
  
    public User(String name, String email, String address, String phoneNumber, String  
gender, int age) {  
        this.name = name;  
        this.email = email;  
        this.address = address;  
        this.phoneNumber = phoneNumber;  
        this.gender = Gender.getGender(gender);  
        this.age = age;  
    }  
  
    public int getParentId() {  
        return parentId;  
    }  
  
    protected void setParentId(int parentId) {  
        this.parentId = parentId;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
}
```

```
public void setEmail(String email) {
    this.email = email;
}

public Gender getGender() {
    return gender;
}

public void setGender(Gender gender) {
    this.gender = gender;
}

public int getId() {
    return id;
}

protected void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}
```

```
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String password) {  
    this.password = password;  
}  
}
```

Mobile service façade:

```
package com.windshi.mobile;  
  
import com.windshi.dao.DBManager;  
import com.windshi.entity.Record;  
import com.windshi.entity.User;  
import com.windshi.view.JSONResponse;  
import com.windshi.view.TextResponse;  
import java.util.Date;  
  
import org.springframework.beans.factory.annotation.Autowired;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
/**  
 *  
 * provide all the services for mobilePhone client needed for the simple application;  
 * the response is in JSON format.  
 */  
@Controller  
@RequestMapping(value = "/mobile")  
public class MobileServiceFacade {  
  
    @Autowired  
    private DBManager dao;  
    private static final String OK = "ok";  
  
    @RequestMapping("/listConsultation")  
    public Object listConsultation(int patientId) {  
        return new JSONResponse().put("list", dao.listConsultation(patientId));  
    }  
}
```

```
/**
 * this service used to store new record
 * @param patientId
 * @param type
 * @param notes
 * @param dateInMilliseconds
 * @return
 */
@RequestMapping("/record")
public Object record(int patientId, String type, String value, String notes, long
dateInMilliseconds) {
    dao.addRecord(new Record(patientId, type, value, notes, new
Date(dateInMilliseconds)));
    return new JsonResponse().put("status", OK);
}

/**
 * this service used to retrieve the report according to saved record
 * @param patientId
 * @param type
 * @param notes
 * @param dateInMilliseconds
 * @return
 */
@RequestMapping("/report")
public Object report(int patientId) {
    return new JsonResponse().put("list", dao.listRecord(patientId));
}
}
```

Web service façade:

```
package com.windshi.web;

import com.windshi.dao.DBManager;
import com.windshi.entity.Doctor;
import com.windshi.entity.Patient;
import com.windshi.entity.Recommendation;
import com.windshi.entity.Record;
import com.windshi.view.JsonResponse;
import java.util.Date;
import java.util.List;
```



```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

/**
 *
 * provide all the services for mobilePhone client needed for the simple application;
 * the response is in JSON format.
 */
@Controller
@RequestMapping(value = "/web")
public class WebServiceFacade {

    private final static String error = "error";
    private final static String success = "success";
    @Autowired
    private DBManager dao;
    private static final String OK = "ok";

    @RequestMapping(method = RequestMethod.GET)
    public Object login(Model model, @RequestParam(required = false) String email,
        @RequestParam(required = false) String password) {
        if (email == null) {
            return "welcome";
        }

        List<Doctor> doctor = dao.getJdbcTemplate_USP().query("SELECT * FROM user
where email = ? AND password = ? and parentId = 0 ", new Object[]{email, password},
            Doctor.rowMapper);
        if (doctor.isEmpty()) {
            model.addAttribute(error, "Invalid username or password");
            return "welcome";
        } else {
            List<Patient> patients = dao.listPatient(doctor.get(0).getId());
            model.addAttribute("doctor", doctor.get(0));
            model.addAttribute("list", patients);
            return "main/myPatients";
        }
    }
}
```

```
}

@RequestMapping("/registration")
public Object redirectToRegistrationForm() {
    return "login/registrationForm";
}

@RequestMapping("/registerUser")
public Object register(Model model, @RequestParam(defaultValue = "0") int doctorId,
String name, String password, String email, String address,
    String phoneNumber, String gender, int age) {
    if (doctorId == 0) {
        dao.addUser(new Doctor(name, password, email, address, phoneNumber,
gender, age));
        model.addAttribute(success, "Your account has been created. Please login
in.");
        return "welcome";
    } else {
        dao.addUser(new Patient(doctorId, name, password, email, address,
phoneNumber, gender, age));
        return gotoMainDoctorPage(doctorId, model);
    }
}

private Object gotoMainDoctorPage(int doctorId, Model model) {
    List<Patient> patients = dao.listPatient(doctorId);
    model.addAttribute("list", patients);
    return "main/myPatients";
}

@RequestMapping("/listPatient")
public Object listPatient(Model model, int doctorId) {
    return new JsonResponse().put("list", dao.listPatient(doctorId));
}

@RequestMapping("/listRecommendation")
public Object listRecommendation(Model model, int doctorId) {
    final List<Recommendation> list = dao.listRecommendation(doctorId);
    model.addAttribute("list", list);
    return "main/content";
}

/**
 * this service used to retrieve the report according to saved record for given patient
```

```
* @param patientId
* @param type
* @param notes
* @param dateInMilliseconds
* @return
*/
@RequestMapping("/saveRecommendation")
public Object saveRecommendation(Model model, int doctorId, int patientId, String
notes) {
    Recommendation rec = new Recommendation(patientId, doctorId, notes, new
Date());
    dao.addRecommendation(rec);
    return gotoMainDoctorPage(doctorId, model);
}

/**
 * this service used to retrieve the report according to saved record
 * @param patientId
 * @param type
 * @param notes
 * @param dateInMilliseconds
 * @return
 */
@RequestMapping("/listRecord")
public Object listRecord(Model model, int patientId) {
    final List<Record> listRecord = dao.listRecord(patientId);
    Patient patient = dao.getPatient(patientId);
    model.addAttribute("list", listRecord);
    model.addAttribute("patient", patient);
    return "main/content";
}
}
```

Appendix V

Content:

```
<% @page contentType="text/html;charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="\${pageContext.request.contextPath}" />
<div>
    <span style="font-size: larger; color: blue">&nbsp;   Report for patient:
    ${patient.name}</span>

    <br/>

    <table style="margin-top: 10px " cellspacing="0" border="0">
        <c:if test="\${not empty list}" var="hasRecord">
            <thead>
                <td class="header">Time</td>
                <td class="header">Type</td>
                <td class="header">Value</td>
                <td class="header">Notes</td>
            </thead>

            <c:forEach items="\${list}" var="record">
                <tr>
                    <td class="item">\${record.date}</td>
                    <td class="item">\${record.type}</td>
                    <td class="item">\${record.value}</td>
                    <td class="item">\${record.notes}</td>
                </tr>
            </c:forEach>
        </c:if>
        <c:if test="\${!hasRecord}">
            <tr>
                <td colspan="99" class ="empty-message">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~The patient have no records yet.</td>
            </tr>
        </c:if>
```

```
</table>

<div style="margin:10px">
  <form action="${context}/web/saveRecommendation" method="post" >
    <span style="font-weight: bold;color:blue"> Add
recommendation:</span> <br/>
    <TEXTAREA NAME="notes" COLS=40 ROWS=6></TEXTAREA>
    <input type="hidden" name="doctorId"
value="${sessionScope.doctorId}"/>
    <input type="hidden" name="patientId" value="${patient.id}"/>
    <div style="margin:10px">
      <input type="submit" value="submit"/>
    </div>
  </form>
</div>

</div>
```

Mypatient:

```
<% @page contentType="text/html;charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ page session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="${pageContext.request.contextPath}" />
<c:set var="doctorId" value="${doctor.id}" scope="session" />
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html;charset=UTF-8">
    <title>Welcome to iCare</title>
    <link rel="stylesheet" href="<c:url value="/resources/blueprint/layout.css"
/>" type="text/css" media="screen, projection">
    <!--[if lt IE 8]>
    <link rel="stylesheet" href="<c:url value="/resources/blueprint/ie.css" />"
type="text/css" media="screen, projection">
    <![endif]-->
    <script type="text/javascript" >
      var xmlhttp
```

```
function loadContent(url, callbackFunction)
{

    xmlhttp=GetXmlHttpRequestObject();

    if (xmlhttp==null)
    {
        alert ("Your browser does not support Ajax HTTP");
        return;
    }

    var url="web/" + url;

    xmlhttp.onreadystatechange=callbackFunction;
    xmlhttp.open("GET",url,true);
    xmlhttp.send(null);
}

function onGetPatientList()
{
    if (xmlhttp.readyState==4)
    {

document.getElementById("patient-list").innerHTML=xmlhttp.responseText;
    }
}

function onGetPatientRecord()
{
    if (xmlhttp.readyState==4)
    {

document.getElementById("right-page").innerHTML=xmlhttp.responseText;
    }
}

function onAddPatient()
{
    if (xmlhttp.readyState==4)
    {

document.getElementById("right-page").innerHTML=xmlhttp.responseText;
    }
}
```

```
    }

    function GetXmlHttpRequest()
    {
        if (window.XMLHttpRequest)
        {
            return new XMLHttpRequest();
        }
        if (window.ActiveXObject)
        {
            return new ActiveXObject("Microsoft.XMLHTTP");
        }
        return null;
    }
</script>
</head>
<body>
    <h1>
        <div align="left">Welcome, ${ doctor.name }</div>
    </h1>
    <div class="my-patients">
        <div id="patient-list">
            <jsp:include page="patientList.jsp" />
        </div>
        <div id="right-page">
            Click on the patient to view the record statistics.
        </div>
    </div>
</body>
</html>
```

Patientlist:

```
<% @page contentType="text/html; charset=UTF-8"%>
<% @page pageEncoding="UTF-8"%>
<% @ page session="true" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="context" value="${pageContext.request.contextPath}" />
```

```
<div>
  <span style="font-size: larger; color: blue">&nbsp; Patient List</span>

  <table style="margin-top:10px " cellspacing="0" border="0">
    <thead>
      <td class="header">Patient Name</td>
      <td class="header">Gender</td>
      <td class="header">Phone Number</td>
      <td class="header">Email</td>
    </thead>
    <c:if test="\${ not empty list}" var="hasPatient">

      <c:forEach items="\${ list}" var="patient">
        <tr
onClick="javascript:loadContent('listRecord?patientId=\${ patient.id }',
onGetPatientRecord)">
          <td class="item" >\${ patient.name }</td>
          <td class="item" >\${ patient.gender }</td>
          <td class="item" >\${ patient.phoneNumber }</td>
          <td class="item" >\${ patient.email }</td>
        </tr>
      </c:forEach>
    </c:if>
    <c:if test="\${ !hasPatient}">
      <tr>
        <td colspan="99" class="empty-message"> You have no patients
yet.</td>
      </tr>
    </c:if>
  </table>
  <div class ="button-add">
    <button type="button" onClick="javascript:loadContent('registration',
onAddPatient)">Click here to add new patient</button>
  </div>
</div>
```


Appendix VI

Account:

```
package com.windshi.account;

import java.math.BigDecimal;
import java.util.Date;
import java.util.concurrent.atomic.AtomicLong;

import javax.validation.constraints.Future;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.format.annotation.NumberFormat;
import org.springframework.format.annotation.NumberFormat.Style;

public class Account {

    private Long id;

    @NotNull
    @Size(min=1, max=25)
    private String name;

    @NotNull
    @NumberFormat(style=Style.CURRENCY)
    private BigDecimal balance = new BigDecimal("1000");

    @NotNull
    @NumberFormat(style=Style.PERCENT)
    private BigDecimal equityAllocation = new BigDecimal(".60");

    @DateTimeFormat(style="S-")
    @Future
    private Date renewalDate = new Date(new Date().getTime() + 31536000000L);

    public Long getId() {
        return id;
    }

    void setId(Long id) {
        this.id = id;
    }
}
```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public BigDecimal getBalance() {
    return balance;
}

public void setBalance(BigDecimal balance) {
    this.balance = balance;
}

public BigDecimal getEquityAllocation() {
    return equityAllocation;
}

public void setEquityAllocation(BigDecimal equityAllocation) {
    this.equityAllocation = equityAllocation;
}

public Date getRenewalDate() {
    return renewalDate;
}

public void setRenewalDate(Date renewalDate) {
    this.renewalDate = renewalDate;
}

Long assignId() {
    this.id = idSequence.incrementAndGet();
    return id;
}

private static final AtomicLong idSequence = new AtomicLong();
}
```

Accountcontorller:

```
package com.windshi.account;

import com.windshi.dao.DBManager;
import com.windshi.view.TextResponse;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping(value = "/account")
public class AccountController {

    private Map<Long, Account> accounts = new ConcurrentHashMap<Long,
Account>();
    @Autowired
    private DBManager dao;

    @RequestMapping(method = RequestMethod.GET)
    public String getcreateForm(Model model) {
        model.addAttribute(new Account());
        return "account/createForm";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String create(@Valid Account account, BindingResult result) {
        if (result.hasErrors()) {
            return "account/createForm";
        }
        this.accounts.put(account.assignId(), account);
        return "redirect:/account/" + account.getId();
    }

    @RequestMapping(value = "{id}", method = RequestMethod.GET)
    public String getView(@PathVariable Long id, Model model) {
        Account account = this.accounts.get(id);
```

```
        if (account == null) {  
            throw new ResourceNotFoundException(id);  
        }  
        model.addAttribute(account);  
        return "account/view";  
    }  
}
```

Resource not found exception:

```
package com.windshi.account;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value=HttpStatus.NOT_FOUND)  
public class ResourceNotFoundException extends RuntimeException {  
  
    private Long resourceId;  
  
    public ResourceNotFoundException(Long resourceId) {  
        this.resourceId = resourceId;  
    }  
  
    public Long getResourceId() {  
        return resourceId;  
    }  
  
}
```

Data base management

```
package com.windshi.dao;  
  
import com.windshi.entity.Patient;  
import com.windshi.entity.Recommendation;  
import com.windshi.entity.Record;  
import com.windshi.entity.User;  
import java.sql.ResultSet;  
import java.sql.SQLException;
```

```
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

public class DBManager {

    @Autowired
    @Qualifier("jdbcTemplate_USP")
    private JdbcTemplate jdbcTemplate_USP;

    public JdbcTemplate getJdbcTemplate_USP() {
        return jdbcTemplate_USP;
    }

    public void addUser(User entity) {
        jdbcTemplate_USP.update("INSERT INTO user VALUES (?, ?, ?, ?, ?, ?, ?, ?)",
        ",
            new Object[]{0, entity.getName(), entity.getEmail(),
entity.getAddress(),
            entity.getPhoneNumber(), entity.getGender().code(),
entity.getAge(),
            entity.getPassword(), entity.getParentId()});
    }

    public void addRecord(Record entity) {
        //insert into record values(0, 'BG', 1, '2011-04-07 ', 'the first time measure ')
        jdbcTemplate_USP.update("INSERT INTO record VALUES (?, ?, ?, ?, ?) ",
            new Object[]{0, entity.getType(), entity.getPatientId(),
entity.getDate(),
            entity.getNotes()});
    }

    public void addRecommendation(Recommendation entity) {
        // insert into recommendation values(0, 1, 1, 'the first time consultation
        ', '2011-04-08' );
        jdbcTemplate_USP.update("INSERT INTO recommendation VALUES
        (?, ?, ?, ?, ?) ",
            new Object[]{0, entity.getDoctorId(), entity.getPatientId(),
entity.getNotes(),
            entity.getDate()});
    }
}
```

```
}

/**
 * list all the patients under the doctor
 * @param doctorId
 * @return
 */
public List<Patient> listPatient(int doctorId) {
    if (doctorId <= 0) {
        throw new IllegalStateException("invalid doctorId ");
    }
    return jdbcTemplate_USP.query("SELECT * FROM user WHERE parentId
= ? ", new Object[]{doctorId}, Patient.rowMapper);
}

/**
 * list record by patientId
 * @param patientId
 * @return
 */
public List<Record> listRecord(int patientId) {
    return jdbcTemplate_USP.query("SELECT * FROM record WHERE
patientId = ? ", new Object[]{patientId}, Record.rowMapper);
}

/**
 * list record by patientId
 * @param patientId
 * @return
 */
public List<Recommendation> listConsultation(int patientId) {
    return jdbcTemplate_USP.query("SELECT * FROM recommendation
WHERE patientId = ? ", new Object[]{patientId}, Recommendation.rowMapper);
}

/**
 * list record by patientId
 * @param patientId
 * @return
 */
public List<Recommendation> listRecommendation(int doctorId) {
    return jdbcTemplate_USP.query("SELECT * FROM recommendation
WHERE doctorId = ? ", new Object[]{doctorId}, Recommendation.rowMapper);
}
```

```
public Patient getPatient(int patientId) {
    final List<Patient> list = jdbcTemplate_USP.query("SELECT * FROM user
WHERE id = ? ", new Object[]{patientId}, Patient.rowMapper);
    if (list.isEmpty()){
        return null;
    }
    return list.get(0);
}
```

Doctor:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/**
 *
 * @author Babie
 */
public class Doctor extends User {

    public Doctor(String name, String password, String email, String address, String
phoneNumber, String gender, int age) {
        super(name, email, address, phoneNumber, gender, age);
        setPassword(password);
    }

    public Doctor(int id, String name, String email, String address, String
phoneNumber, String gender, int age) {
        super(name, email, address, phoneNumber, gender, age);
        setId(id);
    }
}
```

```
public static final RowMapper<Doctor> rowMapper = new
RowMapper<Doctor>() {

    public Doctor mapRow(ResultSet rs, int rowNum) throws SQLException {
        Doctor record = new Doctor(rs.getInt("id"), rs.getString("name"),
rs.getString("email"), rs.getString("address"),
rs.getString("phoneNumber"), rs.getString("gender"),
rs.getInt("age"));
        return record;
    }
};
}
```

Patient:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Patient extends User {
/**
 * for creation
 * @param doctorId
 * @param name
 * @param password
 * @param email
 * @param address
 * @param phoneNumber
 * @param gender
 * @param age
 */
    public Patient(int doctorId, String name, String password, String email, String
address, String phoneNumber, String gender, int age) {
        super(name, email, address, phoneNumber, gender, age);
        setParentId(doctorId);
        setPassword(password);
    }

/**
```



```
* for retrieving;
* do not return the security information like password
* @param name
* @param email
* @param address
* @param phoneNumber
* @param gender
* @param age
*/
public Patient(int id, String name, String email, String address, String
phoneNumber, String gender, int age) {
    super(name, email, address, phoneNumber, gender, age);
    setId(id);
}

public static final RowMapper<Patient> rowMapper = new
RowMapper<Patient>() {

    public Patient mapRow(ResultSet rs, int rowNum) throws SQLException {
        Patient record = new Patient(rs.getInt("id"), rs.getString("name"),
rs.getString("email"), rs.getString("address"),
rs.getString("phoneNumber"), rs.getString("gender"),
rs.getInt("age"));
        return record;
    }
};
}
```

User:

```
package com.windshi.entity;

/**
 * base class for Doctor and Patient
 * @author Babie
 */
public abstract class User {

    private int id;// key
    private int parentId;
    private String name;
    private String password;
```

```
private String email;
private String address;
private String phoneNumber;
private Gender gender;
private int age;

public User(String name, String email, String address, String phoneNumber,
String gender, int age) {
    this.name = name;
    this.email = email;
    this.address = address;
    this.phoneNumber = phoneNumber;
    this.gender = Gender.getGender(gender);
    this.age = age;
}

public int getParentId() {
    return parentId;
}

protected void setParentId(int parentId) {
    this.parentId = parentId;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Gender getGender() {
    return gender;
}
```

```
public void setGender(Gender gender) {
    this.gender = gender;
}

public int getId() {
    return id;
}

protected void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
```

```
}
```

Record:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Record {

    //    private int id;
    /**
     * the type maybe blood pressure; glucose;
     */
    private String type;
    /**
     * the id of the patient the record refer to;
     */
    private int patientId;
    private String value; // the value of the measure
    /**
     * store the notes of the record;
     */
    private String notes;
    private Date date;

    public Record() {
    }

    public Record(int patientId, String type, String value, String notes, Date date) {
        this.type = type;
        this.patientId = patientId;
        this.notes = notes;
        this.date = date;
        this.value = value;
    }

    public Date getDate() {
        return date;
    }
}
```

```
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public String getNotes() {
        return notes;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }

    public int getPatientId() {
        return patientId;
    }

    public void setPatientId(int patientId) {
        this.patientId = patientId;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public static final RowMapper<Record> rowMapper = new
    RowMapper<Record>() {

        public Record mapRow(ResultSet rs, int rowNum) throws SQLException {
            Record record = new Record(rs.getInt("patientId"), rs.getString("type"),
            rs.getString("value"), rs.getString("notes"), rs.getTimestamp("timestamp"));
            return record;
        }
    };
}
```

Recommendation:

```
package com.windshi.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import org.springframework.jdbc.core.RowMapper;

public class Recommendation {

    //    private int id;    // not used in entity;
    private int doctorId; // the id of the patient who the recommendation is for
    private int patientId; // the id of the doctor who gave the recommendation.
    private String notes; // the content
    private Date date;

    public Recommendation() {
    }

    public Recommendation(int doctorId, int patientId, String notes, Date date) {
        this.doctorId = doctorId;
        this.patientId = patientId;
        this.notes = notes;
        this.date = date;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public int getDoctorId() {
        return doctorId;
    }

    public void setDoctorId(int doctorId) {
        this.doctorId = doctorId;
    }

    public String getNotes() {
```

```
        return notes;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }

    public int getPatientId() {
        return patientId;
    }

    public void setPatientId(int patientId) {
        this.patientId = patientId;
    }

    public static final RowMapper<Recommendation> rowMapper = new
    RowMapper<Recommendation>() {

        public Recommendation mapRow(ResultSet rs, int rowNum) throws
        SQLException {
            Recommendation recommendation = new
            Recommendation(rs.getInt("doctorId"), rs.getInt("patientId"), rs.getString("notes"),
            rs.getTimestamp("timestamp"));
            return recommendation;
        }
    };
}
```

Mobile service facade:

```
package com.windshi.mobile;

import com.windshi.dao.DBManager;
import com.windshi.entity.Record;
import com.windshi.entity.User;
import com.windshi.view.JSONResponse;
import com.windshi.view.TextResponse;
import java.util.Date;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.RequestMapping;

/**
 *
 * provide all the services for mobilePhone client needed for the simple application;
 * the response is in JSON format.
 */
@Controller
@RequestMapping(value = "/mobile")
public class MobileServiceFacade {

    @Autowired
    private DBManager dao;
    private static final String OK = "ok";

    @RequestMapping("/listConsultation")
    public Object listConsultation(int patientId) {
        return new JsonResponse().put("list", dao.listConsultation(patientId));
    }

    /**
     * this service used to store new record
     * @param patientId
     * @param type
     * @param notes
     * @param dateInMilliseconds
     * @return
     */
    @RequestMapping("/record")
    public Object record(int patientId, String type, String value, String notes, long
dateInMilliseconds) {
        dao.addRecord(new Record(patientId, type, value, notes, new
Date(dateInMilliseconds)));
        return new JsonResponse().put("status", OK);
    }

    /**
     * this service used to retrieve the report according to saved record
     * @param patientId
     * @param type
     * @param notes
     * @param dateInMilliseconds
     * @return
     */
}
```



```
@RequestMapping("/report")
public Object report(int patientId) {
    return new JsonResponse().put("list", dao.listRecord(patientId));
}
}
```

Web service facade:

```
package com.windshi.web;
```

```
import com.windshi.dao.DBManager;
import com.windshi.entity.Doctor;
import com.windshi.entity.Patient;
import com.windshi.entity.Recommendation;
import com.windshi.entity.Record;
import com.windshi.view.JsonResponse;
import java.util.Date;
import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
```

```
/**
 *
 * provide all the services for mobilePhone client needed for the simple application;
 * the response is in JSON format.
 */
```

```
@Controller
```

```
@RequestMapping(value = "/web")
```

```
public class WebServiceFacade {
```

```
    private final static String error = "error";
    private final static String success = "success";
    @Autowired
    private DBManager dao;
    private static final String OK = "ok";
```

```
@RequestMapping(method = RequestMethod.GET)
public Object login(Model model, @RequestParam(required = false) String
email, @RequestParam(required = false) String password) {
    if (email == null) {
        return "welcome";
    }

    List<Doctor> doctor = dao.getJdbcTemplate_USP().query("SELECT *
FROM user where email = ? AND password = ? and parentId = 0 ", new
Object[]{email, password},
        Doctor.rowMapper);
    if (doctor.isEmpty()) {
        model.addAttribute(error, "Invalid username or password");
        return "welcome";
    } else {
        List<Patient> patients = dao.listPatient(doctor.get(0).getId());
        model.addAttribute("doctor", doctor.get(0));
        model.addAttribute("list", patients);
        return "main/myPatients";
    }
}

@RequestMapping("/registration")
public Object redirectToRegistrationForm() {
    return "login/registrationForm";
}

@RequestMapping("/registerUser")
public Object register(Model model, @RequestParam(defaultValue = "0") int
doctorId, String name, String password, String email, String address,
        String phoneNumber, String gender, int age) {
    if (doctorId == 0) {
        dao.addUser(new Doctor(name, password, email, address,
phoneNumber, gender, age));
        model.addAttribute(success, "Your account has been created. Please
login in.");
        return "welcome";
    } else {
        dao.addUser(new Patient(doctorId, name, password, email, address,
phoneNumber, gender, age));
        return gotoMainDoctorPage(doctorId, model);
    }
}
```

```
private Object gotoMainDoctorPage(int doctorId, Model model) {
    List<Patient> patients = dao.listPatient(doctorId);
    model.addAttribute("list", patients);
    return "main/myPatients";
}

@RequestMapping("/listPatient")
public Object listPatient(Model model, int doctorId) {
    return new JsonResponse().put("list", dao.listPatient(doctorId));
}

@RequestMapping("/listRecommendation")
public Object listRecommendation(Model model, int doctorId) {
    final List<Recommendation> list = dao.listRecommendation(doctorId);
    model.addAttribute("list", list);
    return "main/content";
}

/**
 * this service used to retrieve the report according to saved record for given
patient
 * @param patientId
 * @param type
 * @param notes
 * @param dateInMilliseconds
 * @return
 */
@RequestMapping("/saveRecommendation")
public Object saveRecommendation(Model model, int doctorId, int patientId,
String notes) {
    Recommendation rec = new Recommendation(patientId, doctorId, notes,
new Date());
    dao.addRecommendation(rec);
    return gotoMainDoctorPage(doctorId, model);
}

/**
 * this service used to retrieve the report according to saved record
 * @param patientId
 * @param type
 * @param notes
 * @param dateInMilliseconds
 * @return
```

```
*/  
@RequestMapping("/listRecord")  
public Object listRecord(Model model, int patientId) {  
    final List<Record> listRecord = dao.listRecord(patientId);  
    Patient patient = dao.getPatient(patientId);  
    model.addAttribute("list", listRecord);  
    model.addAttribute("patient", patient);  
    return "main/content";  
}  
}
```