School of Informatics and Applied Mathematics
Pusat Pengajian Informatik dan Matematik Gunaan

Lab Manual 8
Build an MVC App with Servlets, JSP, JavaBeans and JDBC

Instructions:

1. This lab manual is designed for the student of School Informatics and Applied Mathematics, Universiti Malaysia Terengganu. It is prohibited to print and distribute the manual without the author's consent.
2. For each task, the student is required to follow step-by-step instructions as stated in the manual.

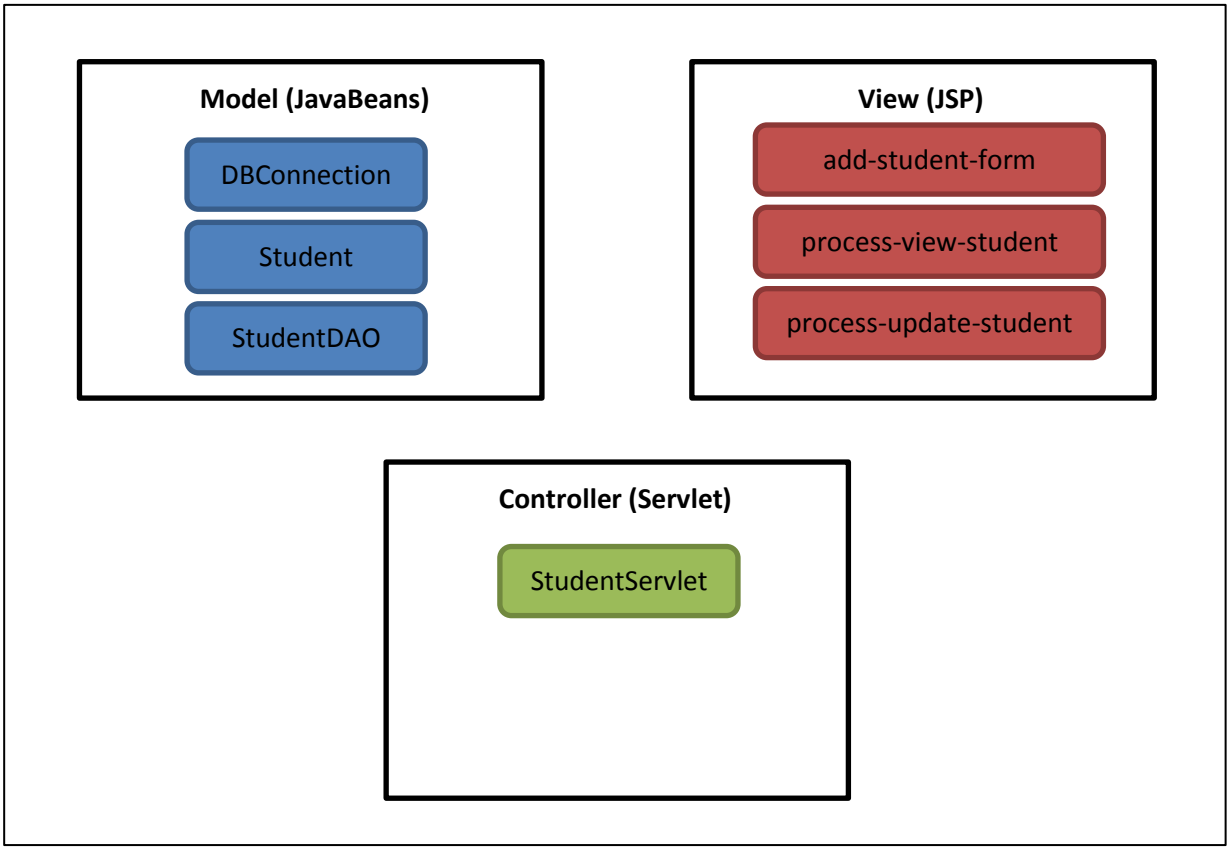
Learning Outcomes

At the end of this lab, the student should be able to:

1. Applying the concept of MVC framework in developing a web-based application.

Task 2: Using MVC framework for manipulating records

Objective	:	Use MVC framework to access and manipulate records to mySql database.
Problem Description	:	Create a simple student registration system that should be able to: • connect to the database. • add a new student to the list. • update the selected student in the list. • delete the selected student from the list. • display the list of all student's (their name and matrics number using Student object).
Estimated time	:	45 minutes



Overall Structure of MVC for Student Registration System

Step-by-Step Instructions:

Database Setup

1. Setup MySQL database environment.

   1.1 Open phpmyadmin

   1.2 Create a database and name it as *universitystudent.sql*.

   1.3 Create one table and name it as *student* with the following data dictionary

Name	Type	Collation	Attributes	Null	Default	Extra
stud_id	int(11)			No	*None*	AUTO_INCREMENT
stud_matric	varchar(10)			No	*None*	
stud_name	varchar(50)			No	*None*	

Model (M)

2. Create a new Java class, *DBConnection* inside package *com.lab8.task1*. This class is used to handle the connection to the database.

   2.1 Go to *Source Packages*, right click -> New then choose *Java Package*.

   2.2 Name the package as *com.lab8.task1*.

   2.3 Create a JavaBeans/Java class, right click package *com.lab6.task2* -> *New* -> *Java Class*.

   2.4 Name the class as *DBConnection*, choose Package as *com.lab8.task1* and click *Finish* button.

   2.5 Write the following Java code inside *DBConnection.java*

```java
package com.lab8.task1;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection implements java.io.Serializable{

    private static Connection connection;

    public static Connection getConnection() {
        try {
            //Step 1: Load the JDBC diver
            Class.forName("com.mysql.jdbc.Driver");
            //Step 2: Establish a connection to mySql database
            String myUrl = "jdbc:mysql://localhost:3308/universitystudent";
            connection = DriverManager.getConnection(myUrl, "webstudent", "webstudent");
        } catch (ClassNotFoundException | SQLException e) {
            e.getMessage();
        }
        return connection;
    }

    public void closeConnection()
    {
        try{
            connection.close();
        }
        catch(SQLException e){
            e.getMessage();
        }
    }
}
```

3. Create another Java class, *Student* inside package *com.lab8.task1*.

   3.1 Right click package *com.lab8.task1* -> *New* -> *Java Class*.

   3.2 Name the class as *Student,* choose Package as *com.lab8.task1* and click *Finish* button.

   3.3 Write the following Java code inside *Student.java*

```java
package com.lab8.task1;

public class Student {
    private int id;
    private String name;
    private String matric;

    public String getName() {
        return name;
    }

    public String getMatric() {
        return matric;
    }

    public int getId() {
        return id;
    }

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

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

    public void setMatric(String matric) {
        this.matric = matric;
    }
}
```

4. Create another Java class, *StudentDAO* inside package *com.lab8.task1*. This class is used to perform the CRUD operations:

   - Create operation - add record to the database via *addStudent()*
   - Retrieve operation - view all the records in the database via *retrieveAllStudent()* and view specific record by student id via *retrieveOnseStudent()*
   - Update operation - update the selected record in the database via *updateStudent()*
   - Delete operation - delete the selected record in the database via *deleteStudent()*

   4.1 Right click package *com.lab8.task1* -> *New* -> *Java Class*.

   4.2 Name the class as *StudentDAO,* choose Package as *com.lab8.task1* and click *Finish* button.

   4.3 Write the following Java code inside *StudentDAO.java*.

```java
package com.lab8.task1;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class StudentDAO {

    private final Connection connection;
    private int result;

    public StudentDAO() {
        connection = DBConnection.getConnection();
    }

    public int addStudent(Student student) {
        try {
            String mySqlQuery = "insert into student "
                    + "(stud_matric, stud_name) "
                    + "values (?, ?)";
            PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
            myPs.setString(1, student.getMatric());
            myPs.setString(2, student.getName());
            result = myPs.executeUpdate();

        } catch (Exception e) {
            System.out.println("Exception is ;" + e);
        }
        return result;
    }
```

```java
    public List<Student> retrieveAllStudent() {
        List<Student> studentAll = new ArrayList<Student>();
        Student student;
        try {
            Statement myStatement = connection.createStatement();
            String myQuery = "select * from student";
            ResultSet myRs = myStatement.executeQuery(myQuery);
            while (myRs.next()) {
                student = new Student();
                student.setId(myRs.getInt(1));
                student.setMatric(myRs.getString(2));
                student.setName(myRs.getString(3));
                studentAll.add(student);
            }
        } catch (Exception e) {
            System.out.println("Exception is ;" + e);
        }
        return studentAll;
    }
```

```java
    public Student retrieveOneStudent(int studentId) {
        Student student = new Student();
        try {
            Statement myStatement = connection.createStatement();
            String myQuery = "select * from student "
                    + "where stud_id=" + studentId;
            ResultSet myRs = myStatement.executeQuery(myQuery);
            while (myRs.next()) {
                student.setId(myRs.getInt(1));
                student.setMatric(myRs.getString(2));
                student.setName(myRs.getString(3));
            }
        } catch (Exception e) {
            System.out.println("Exception is ;" + e);
        }
        return student;
    }
```

```java
    public int updateStudent(Student student) {
        try {

            String mySqlQuery = "update student "
                    + "set stud_matric=?, stud_name=? "
                    + "where stud_id=?";
            PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
            myPs.setString(1, student.getMatric());
            myPs.setString(2, student.getName());
            myPs.setInt(3, student.getId());
            result = myPs.executeUpdate();
        } catch (Exception e) {
            System.out.println("Exception is ;" + e);
        }
        return result;
    }

    public int deleteStudent(int studentId) {
        try {

            String mySqlQuery = "delete from student where stud_id=?";
            PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
            myPs.setInt(1, studentId);
            result = myPs.executeUpdate();
        } catch (Exception e) {
            System.out.println("Exception is ;" + e);
        }
        return result;
    }
}
```

Controller (C)

5. Create a Servlet, *StudentServlet* inside package *com.lab8.task1*. The servlet should be able to manage the flow of the system.
6. Write the following code in the *StudentServlet*.

```java
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        try {
            // read the "command" parameter
            String theCommand = request.getParameter("command");
            // if the command is missing, then default to listing students
            if (theCommand == null) {
                theCommand = "LIST";
            }
            // route to the appropriate method
            switch (theCommand) {
                case "LIST":
                    listStudent(request, response);
                    break;
                case "ADD":
                    addStudent(request, response);
                    break;
                case "LOAD":
                    loadStudent(request, response);
                    break;
                case "UPDATE":
                    updateStudent(request, response);
                    break;
                case "DELETE":
                    deleteStudent(request, response);
                    break;
                default:
                    listStudent(request, response);
            }
        } catch (Exception ex) {
            Logger.getLogger(StudentServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```java
    private void addStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
        // read student info from form data
        String name = request.getParameter("name");
        String matric = request.getParameter("matric");

        Student student = new Student();
        student.setMatric(matric);
        student.setName(name);
        StudentDAO studentDao = new StudentDAO();
        int result = studentDao.addStudent(student);
        if (result > 0) {
            request.setAttribute("theMessage", "Success Add Record");
            listStudent(request, response);
        }
    }

    private void listStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {

        //Step 1: Get student data from studentDao
        StudentDAO studentDao = new StudentDAO();
        List<Student> allStudent = studentDao.retrieveAllStudent();

        //Step 2: Add student to the request
        request.setAttribute("theStudents", allStudent);

        //Step 3: Send to JSP page view
        RequestDispatcher dispatcher = request.getRequestDispatcher("/Task1/process-view-student.jsp");

        //Step 4: Forward to JSP
        dispatcher.forward(request, response);

    }
```

```java
    private void loadStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
        // read student id from form data
        int theStudentId = Integer.parseInt(request.getParameter("id"));

        // get student from database (db util)
        StudentDAO studentDao = new StudentDAO();
        Student theStudent = studentDao.retrieveOneStudent(theStudentId);

        // place student in the request attribute
        request.setAttribute("theStudent", theStudent);

        // send to jsp page: process-update-student.jsp
        RequestDispatcher dispatcher = request.getRequestDispatcher("/Task1/process-update-student.jsp");
        dispatcher.forward(request, response);
    }
```

```java
    private void updateStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {

        // read student info from form data
        int studentId = Integer.parseInt(request.getParameter("hidid"));
        String name = request.getParameter("name");
        String matric = request.getParameter("matric");

        // create a new student object
        StudentDAO studentDao = new StudentDAO();
        Student student = new Student();

        student.setId(studentId);
        student.setName(name);
        student.setMatric(matric);

        // perform update on database
        int result = studentDao.updateStudent(student);
        if (result > 0) {
            request.setAttribute("theMessage", "Success Update Record");
            listStudent(request, response);
        }
    }

    private void deleteStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
        int studentId = Integer.parseInt(request.getParameter("id"));
        StudentDAO studentDao = new StudentDAO();
        int result = studentDao.deleteStudent(studentId);
        if (result > 0) {
            request.setAttribute("theMessage", "Success Delete Record");
            listStudent(request, response);
        }
    }
}
```

View (V)

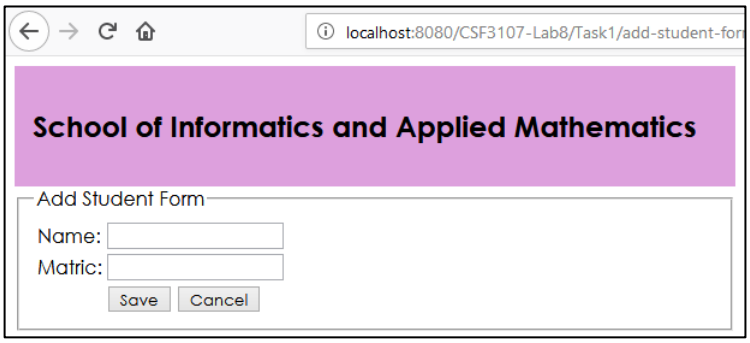
7. Create a JSP page, *add-student-form.jsp* inside *Task1*.

   7.1 Create a new folder inside *Web-Pages* and name it as *Task1*.

   7.2 Right click *Task2* -> *New* -> *JSP* and name the JSP page as *welcome.jsp* and click the *Finish* button.

   7.3 Write an HTML's markup to produce HTML's form in *add-student-form.jsp*. The form should be as depicted below, in which it allow to user to enter name and matric number.

   7.4 This page should be able to pass the data to the Servlet file called as *ServletStudent*



8. Create another JSP page, *process-view-student.jsp* inside *Task1*.

9. Write the following code inside *process-view-student.jsp*

```
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="com.lab8.task1.Student"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Task 1: Retrieve Student</title>
    </head>
    <body style="font-family: century gothic; max-width: 600px">
        <div style="font-family: century gothic; padding: 15px ; background-color: plum;">
            <h1>Display Student Data from Database</h1></div>
        <div>
            <input type=button value="Add Student" style="font-family: century gothic;"
                   onclick="window.location.href = 'Task1/add-student-form.jsp';
                            return false;"/>
        </div>
        <table border='1'>
            <tr>
                <th>ID</th>
                <th>MATRIC</th>
                <th>NAME</th>
                <th>ACTIONS</th>
            </tr>

            <%
                if (request.getAttribute("theMessage") != null) {
                    String message = (String) request.getAttribute("theMessage");
                    out.println("<script type=\"text/javascript\">");
                    out.println("alert(\"" + message + "\")");
                    out.println("</script >");
                }
                List<Student> allStudent = (List<Student>) request.getAttribute("theStudents");
```

```
41                  for (int i = 0; i < allStudent.size(); i++) {
42                      out.println("<tr>");
43                      out.println("<td>" + allStudent.get(i).getId() + "</td>");
44                      out.println("<td>" + allStudent.get(i).getMatric() + "</td>");
45                      out.println("<td>" + allStudent.get(i).getName() + "</td>");
46                      out.println("<td><a href=" + request.getContextPath() + "/StudentServlet?command=LOAD&id="
47                              + allStudent.get(i).getId() + ">Update</a>"
48                              + "|" + "<a href=" + request.getContextPath() + "/StudentServlet?command=DELETE&id="
49                              + allStudent.get(i).getId() + " onclick=\"return confirm"
50                              + "('Are you sure you want to delete?')\">Delete</a>"
51                              + "</td>");
52                      out.println("</tr>");
53                  }
54              %>
55          </table>
56
57      </body>
58  </html>
```

10. Create another JSP page, *process-update-student.jsp* inside *Task1*.
11. Write the following code inside *process-update-student.jsp*.

```
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <%@page import="com.lab8.task1.Student"%>
9   <!DOCTYPE html>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>Task 1: Update Student</title>
14      </head>
15      <%
16          Student student = (Student) request.getAttribute("theStudent");
17      %>
18      <body style="font-family: century gothic; max-width: 600px;">
19          <div style="background-color: yellowgreen; padding: 10px; text-align: center">
20              <h2>School of Informatics & Applied Mathematics</h2>
21          </div>
22          <div>
23              <fieldset><legend><b>Update Student</b></legend>
24                  <form action="<%=request.getContextPath()%>/StudentServlet" method="get">
25                      <input type="hidden" name="hidid" value="<%=student.getId()%>"/>
26                      <table>
27                          <tbody>
28                              <tr>
29                                  <td><label>Name:</label></td>
30                                  <td><input type="text" name="name" value="<%=student.getName()%>"/></td>
31                              </tr>
32                              <tr>
33                                  <td><label>Matric:</label></td>
34                                  <td><input type="text" name="matric" value="<%=student.getMatric()%>"/></td>
35                              </tr>
36                              <tr>
37                                  <td><label></label></td>
38                                  <td><input type="submit" value="Update" style="font-family: century gothic;"/>
39                                      <input type="button" value="Cancel" style="font-family: century gothic;"
40                                              onclick="window.location.href = '<%=request.getContextPath()%>/StudentServlet?command=LIST';
41                                                  return false;"/>
42                                      <input type="hidden" name="command" value="UPDATE"/>
43                                  </td>
```

```
44                              </tr>
45                          </tbody>
46                      </table>
47                  </form>
48              </fieldset>
49          </div>
50      </body>
51  </html>
```

12. Save and run all the files.

Reflections

I. What you have learnt from this exercise?

Exercise: Using MVC framework for implementing CRUD operations

Objective : Applying the JDBC API to access a database by implementing MVC framework for manipulating of records.

Problem Description : Create a simple web-based for Insurance Management System. The system concists of three modules; Customer Authentication, Vehicle Registration and Insurance Quotation. The details requirements for the realization of these modules are as follow:

i. Customer Authentication module should be able to perform
   - customer's registration for information such as customer IC number, name, email address and password.
   - customer's login
   - update customer's information

ii. Vehicle Registration module should be able to perform CRUD for vehicle's information such as vehicle plat number, vehicle type (Car or Motorcyle), vehicle brand, vehicle market price.

iii. Insurance Quotation module should be able to allow customer to request for insurance quotation for a specific vehicle. Final insurance amount must be added with 6% SST. The information required to perform this module is below:
   - Vehichle's information
   - Coverage type:
     - Comprehensive (value as "1")
     - Third Party (value as "2")
   - No Claim Discount (NCD)
     - 10% (value as "10")
     - 25% (value as "25")
     - 35% (value as "35")
     - 55% (value as "55")

   The calculation formula for insurance comprehensive
   - NCD = 55%, 1.8% x market price
   - NCD = 35%, 2.4% x market price
   - NCD = 25%, 3.0% x market price
   - NCD = 10%, 3.8% x market price

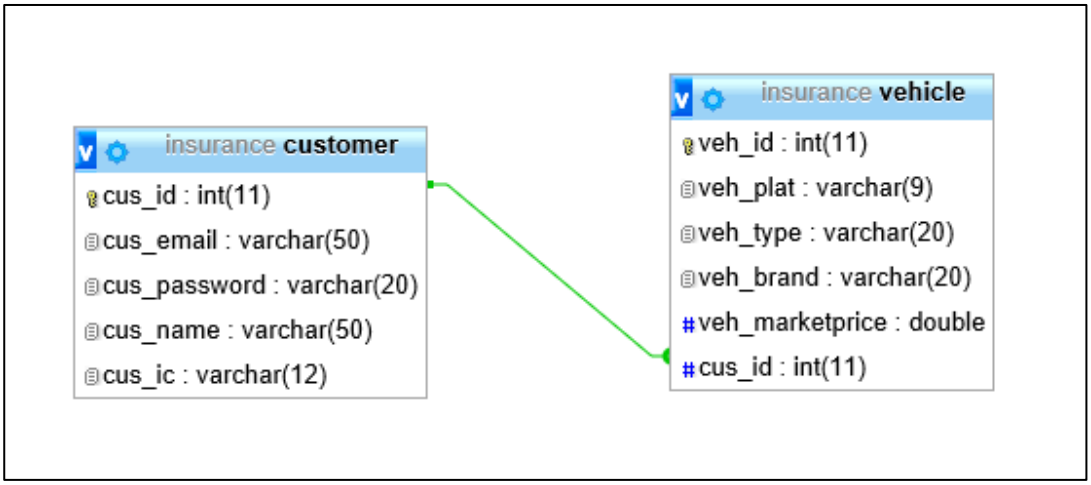
   The calculation formula for third party
   - NCD = 55%, 1.2% x market price
   - NCD = 35%, 1.8% x market price
   - NCD = 25%, 2.5% x market price
   - NCD = 10%, 3.3% x marketprice

i. Each customer may register for one or more vehicles.

ii. Each customer may request for one or more insurance quotation.

Step-by-Step Instructions:

1. Modify your code that have been completed in Lab 6 (Exercise) by implementing MVC framework.
2. Setup MySQL database environment.
   2.1 Create a new database and name it as insurance.sql.
   2.2 Create two tables and name them as customer and vehicle respectively. Figure below shows the Entity-Relationship diagram of insurance database.



   2.3 Details dictionary for both tables are shown below.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	cus_id	int(11)			No	None	AUTO_INCREMENT
2	cus_email	varchar(50)			No	None	
3	cus_password	varchar(20)			No	None	
4	cus_name	varchar(50)			No	None	
5	cus_ic	varchar(12)			No	None	

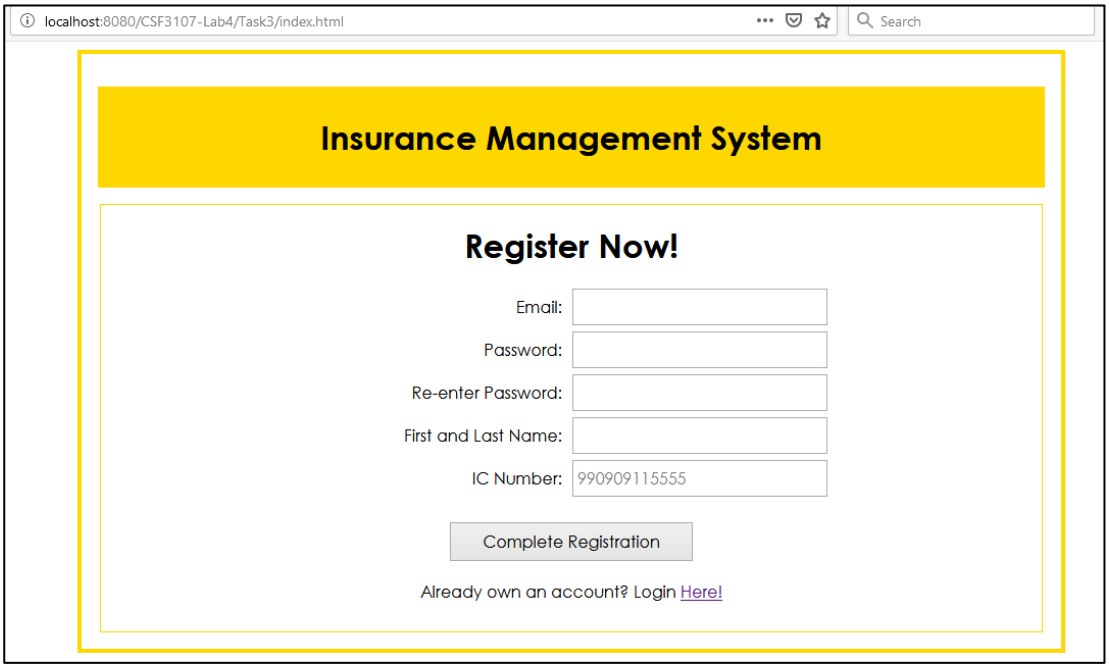
Customer Table

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	veh_id	int(11)			No	None	AUTO_INCREMENT
2	veh_plat	varchar(9)			No	None	
3	veh_type	varchar(20)			No	None	
4	veh_brand	varchar(20)			No	None	
5	veh_marketprice	double			No	None	
6	cus_id	int(11)			No	None	

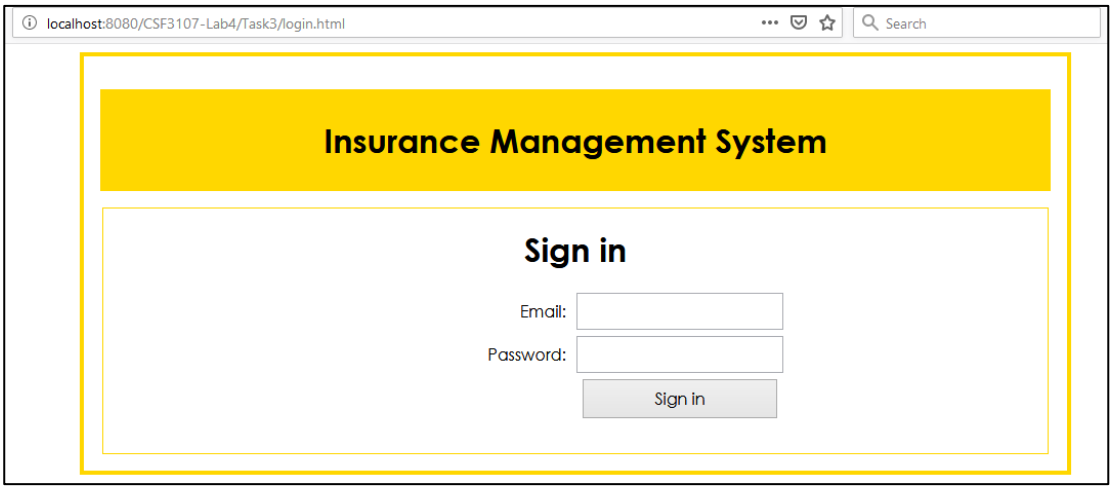
Vehicle Table

3. Create a new folder inside *Web-Pages* and name it as *Exercise*.
4. Produce the interfaces for Customer Authentication, Vehicle Registration and Insurance Quotation modules. The examples of interface design for each modules are given below. **You may design your own interface according to your preferences.**

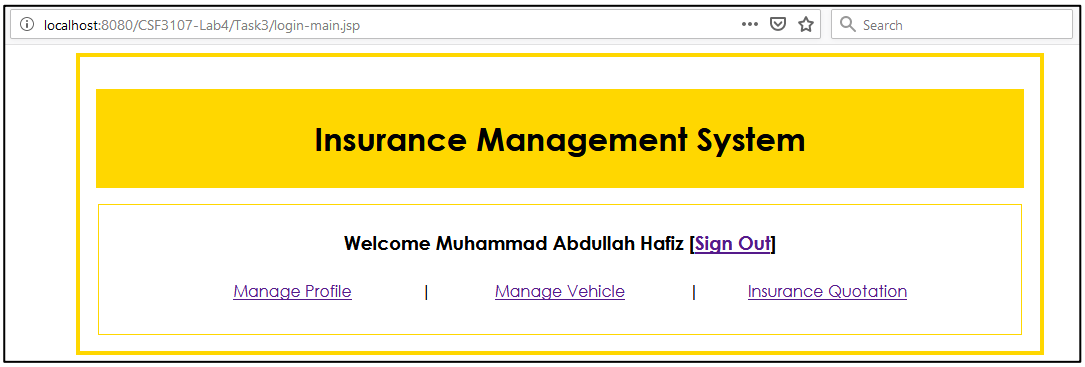
4.1 Customer Authentication Module



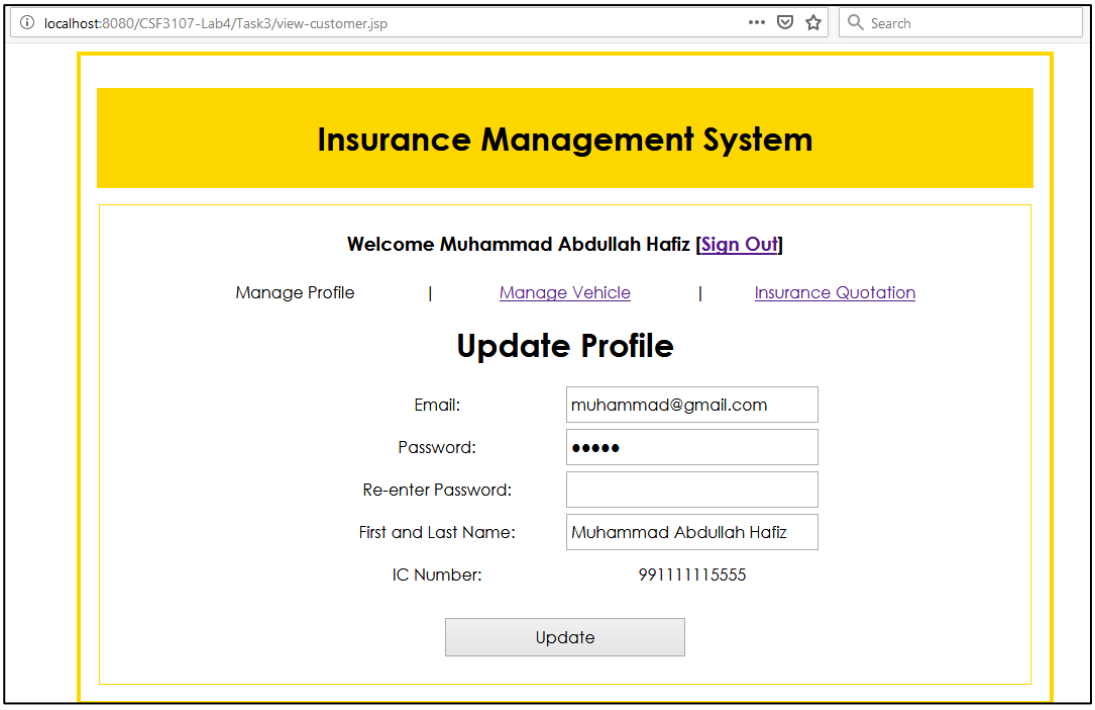
Interface of Customer Registration



Interface of Customer Login

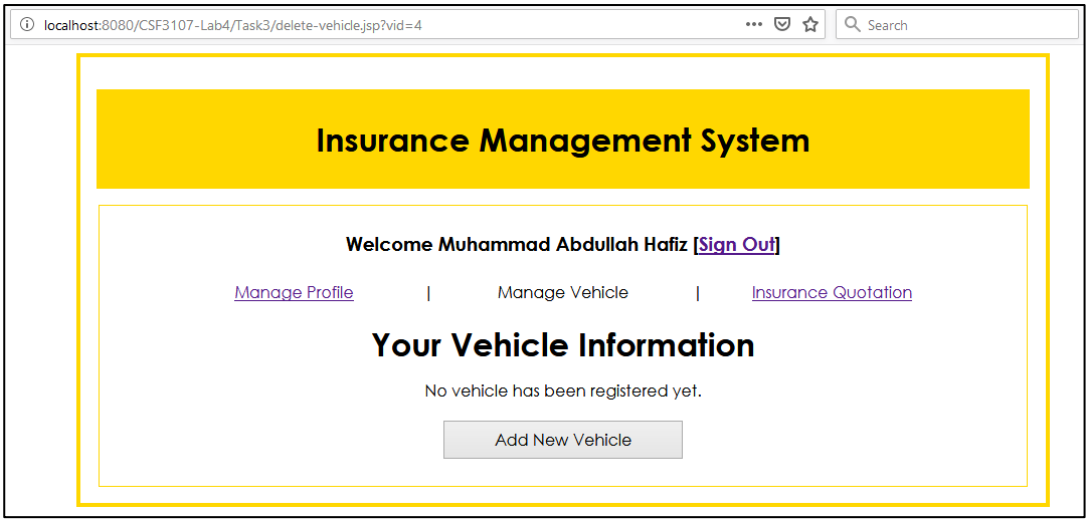


Interface of After Successful Login

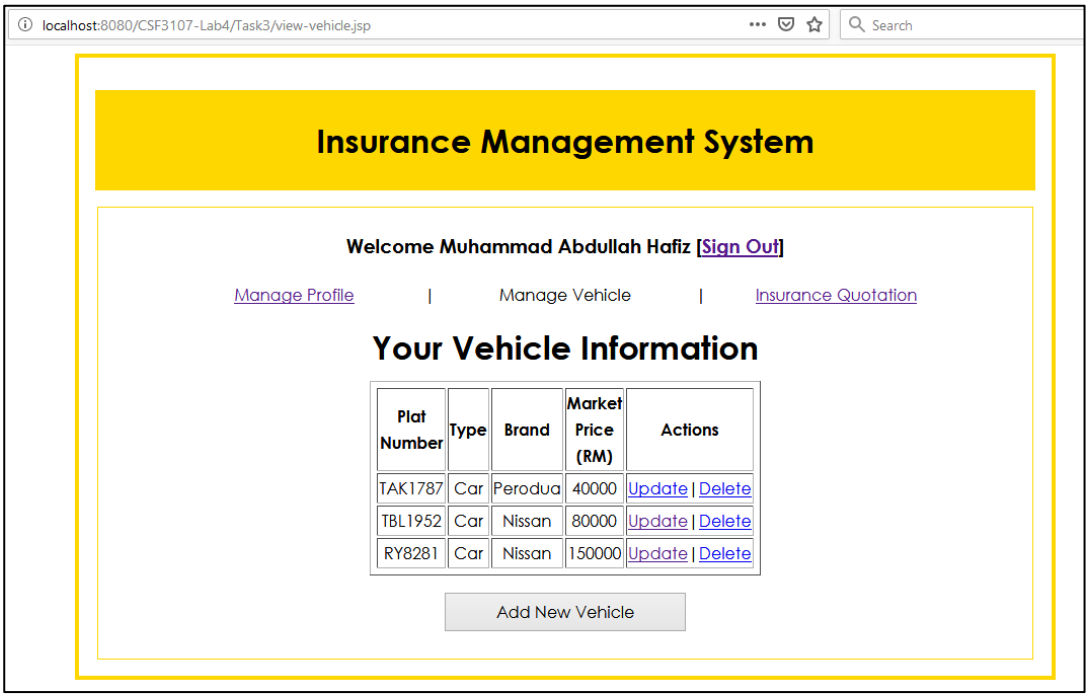


Interface of Update Profile

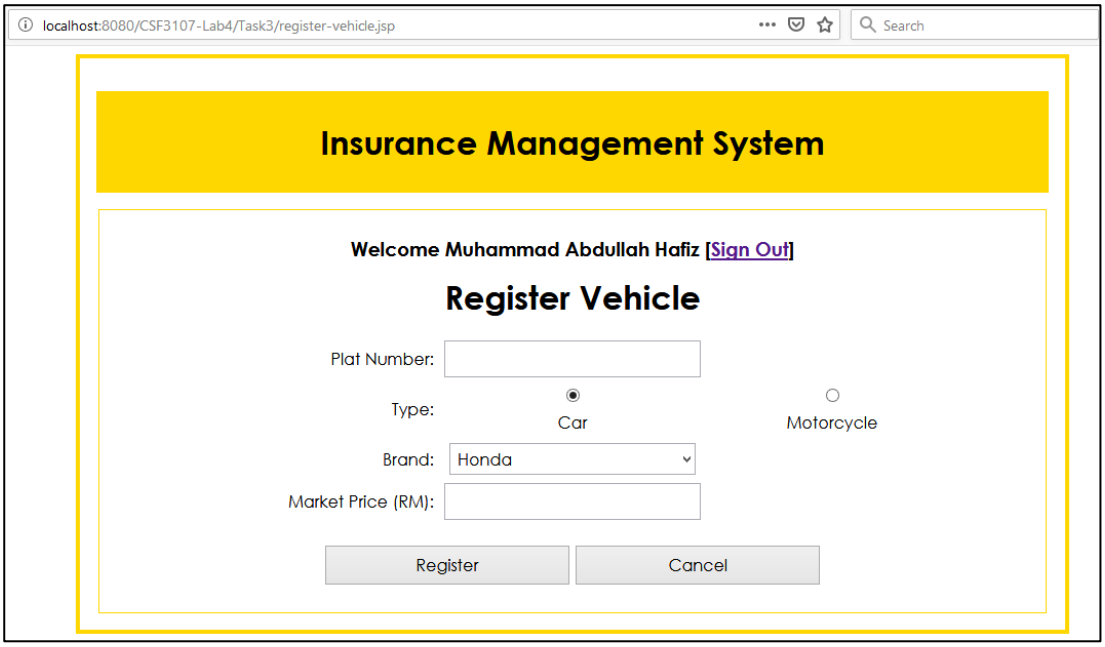
4.2 Vehicle Registration Module



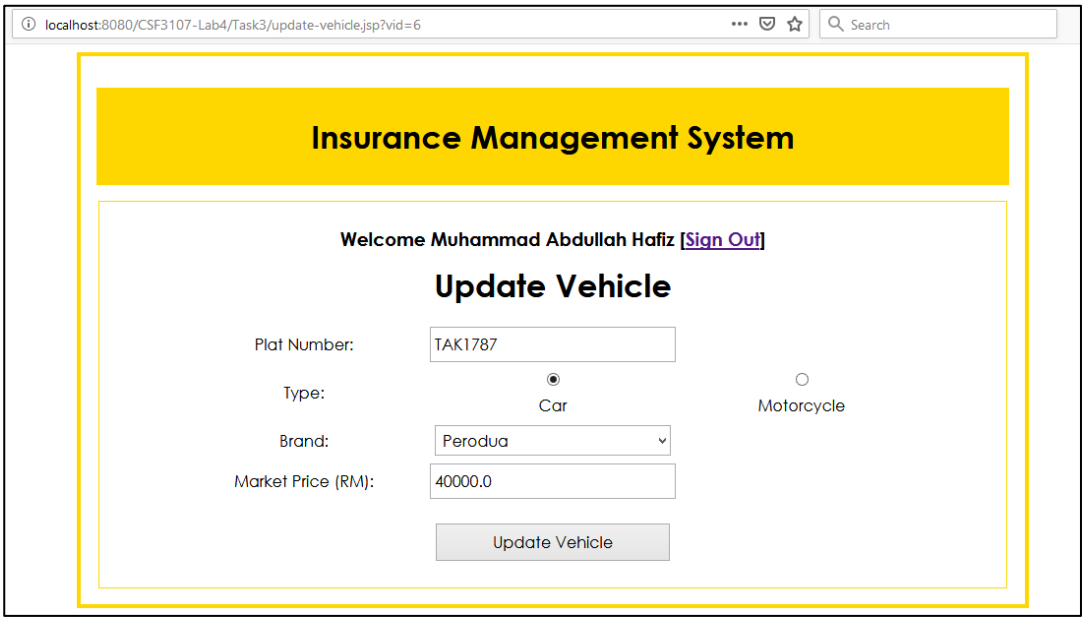
Interface of View Vehicle (Retrieve) – if there is no vehicle yet



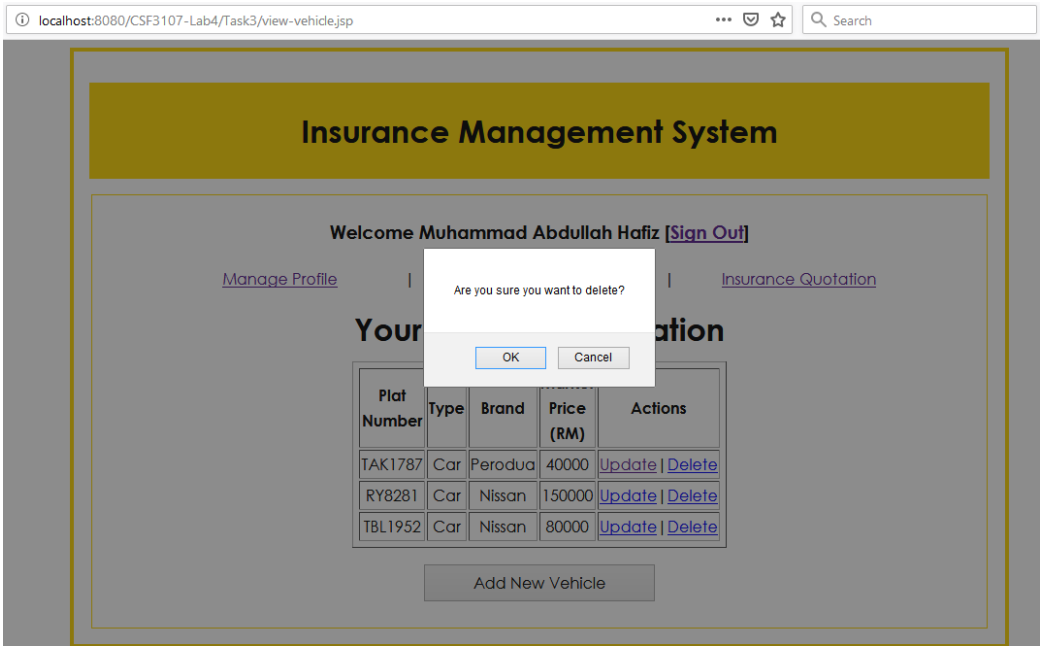
Interface of View Vehicle (Retrieve) – if there is vehicle



Interface of Register Vehicle (Create)

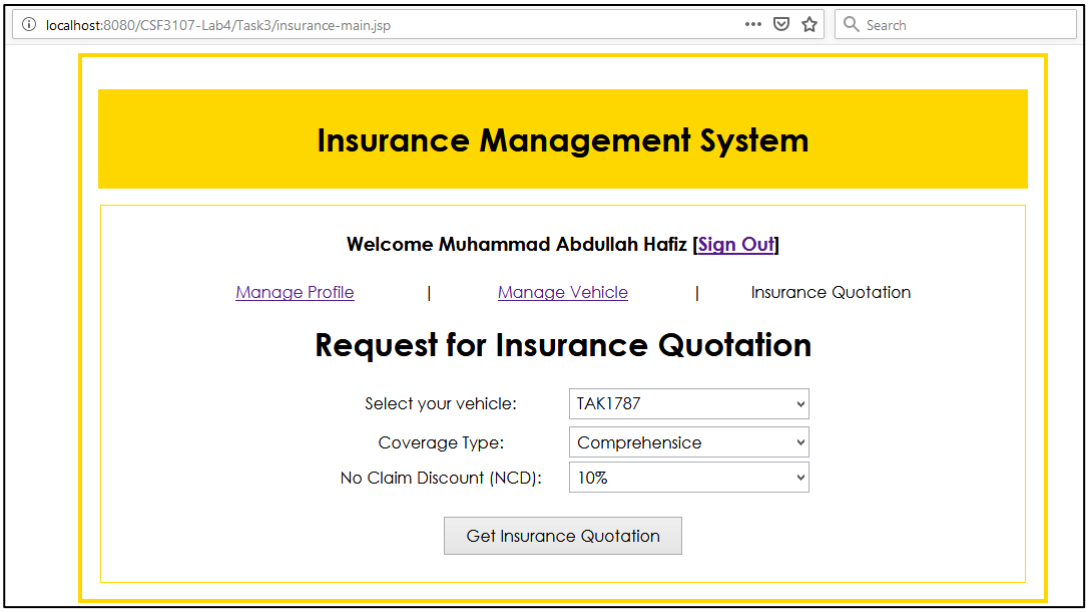


Interface of Update Vehicle

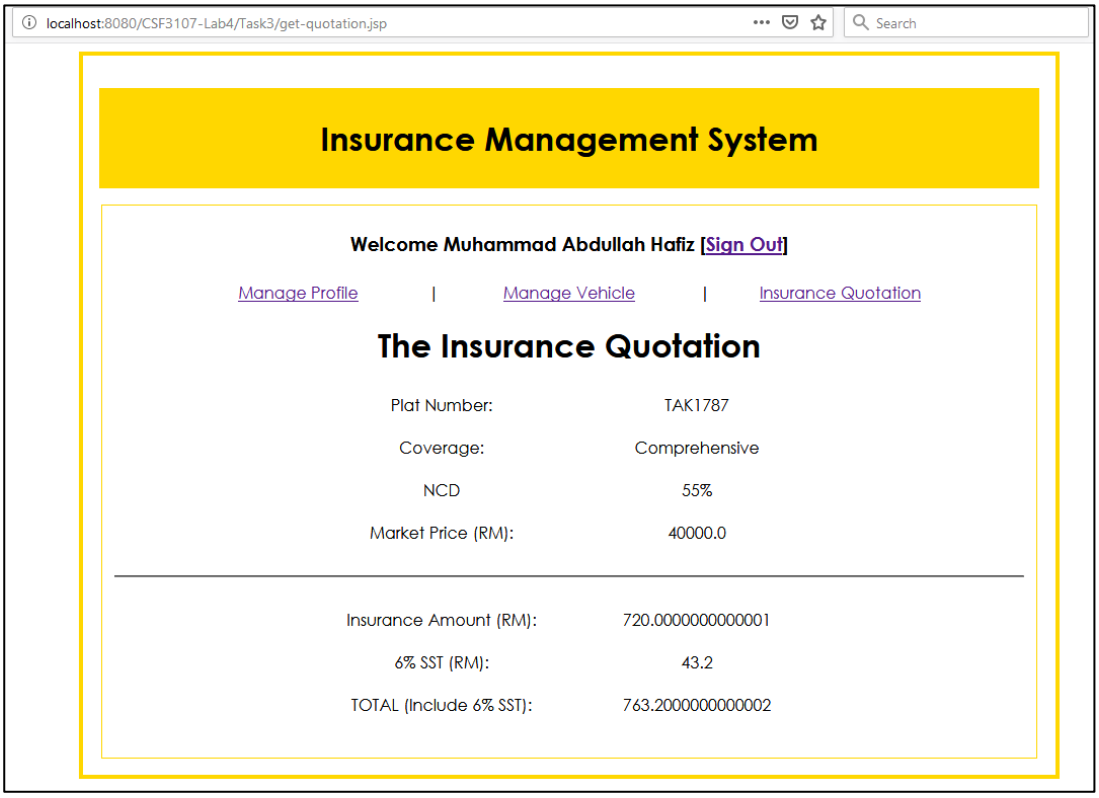


Interface of Deleting Vehicle

4.3 Insurance Quotation Module



Interface of Request for Insurance Quotation



Interface of Display Insurance Quotation