

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.

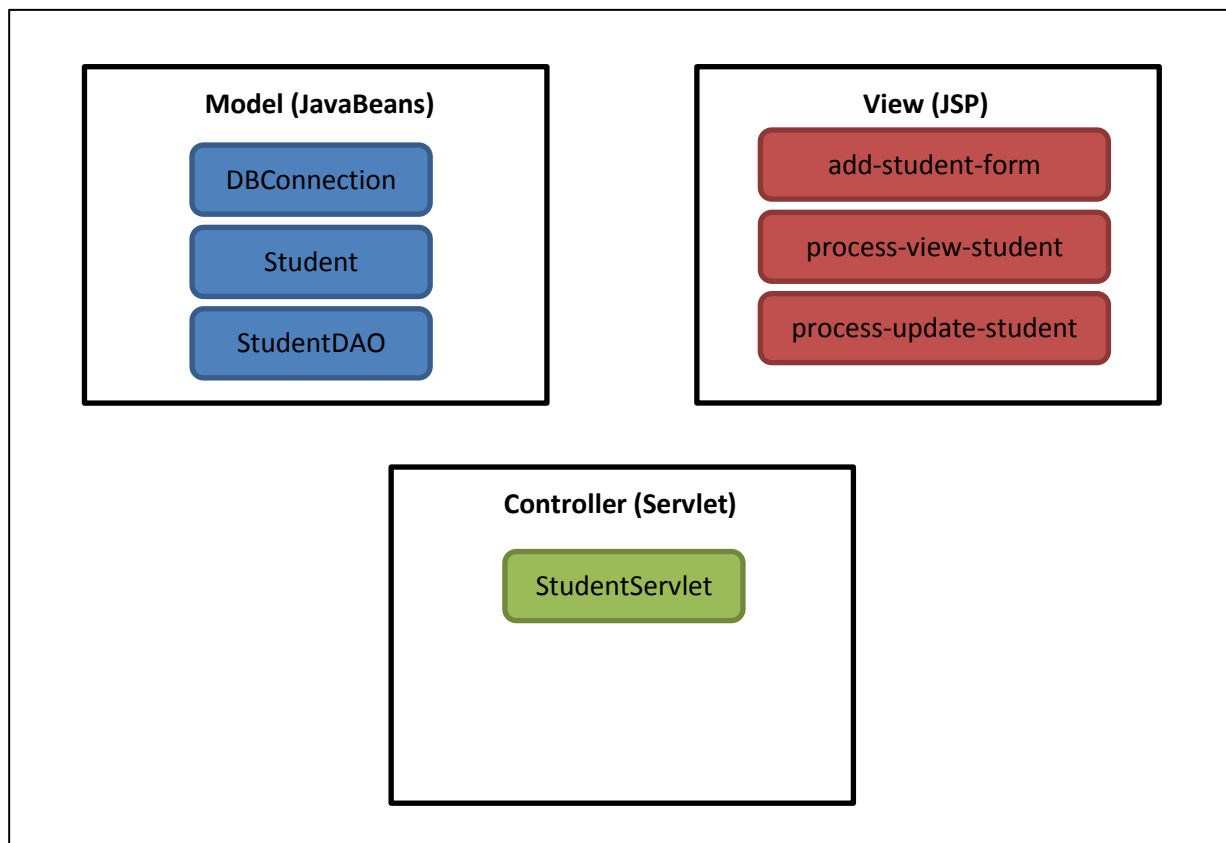
Task 2: Using MVC framework for manipulating records

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*

```
4 package com.lab8.task1;
5
6 import java.sql.Connection;
7 import java.sql.DriverManager;
8 import java.sql.SQLException;
9
10 public class DBConnection implements java.io.Serializable{
11
12     private static Connection connection;
13
14     public static Connection getConnection() {
15         try {
16             //Step 1: Load the JDBC driver
17             Class.forName("com.mysql.jdbc.Driver");
18             //Step 2: Establish a connection to mySql database
19             String myUrl = "jdbc:mysql://localhost:3308/universitystudent";
20             connection = DriverManager.getConnection(myUrl, "webstudent", "webstudent");
21         } catch (ClassNotFoundException | SQLException e) {
22             e.getMessage();
23         }
24         return connection;
25     }
26
27     public void closeConnection()
28     {
29         try{
30             connection.close();
31         }
32         catch(SQLException e){
33             e.getMessage();
34         }
35     }
36 }
```

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*

```
4 package com.lab8.task1;
5
6 public class Student {
7     private int id;
8     private String name;
9     private String matric;
10
11     public String getName() {
12         return name;
13     }
14
15     public String getMatric() {
16         return matric;
17     }
18
19     public int getId() {
20         return id;
21     }
22
23     public void setId(int id) {
24         this.id = id;
25     }
26
27     public void setName(String name) {
28         this.name = name;
29     }
30
31     public void setMatric(String matric) {
32         this.matric = matric;
33     }
34 }
```

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*.

```

6 package com.lab8.task1;
7
8 import java.sql.*;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 public class StudentDAO {
13
14     private final Connection connection;
15     private int result;
16
17     public StudentDAO() {
18         connection = DBConnection.getConnection();
19     }
20
21     public int addStudent(Student student) {
22         try {
23             String mySqlQuery = "insert into student "
24                 + "(stud_matric, stud_name) "
25                 + "values (?, ?)";
26             PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
27             myPs.setString(1, student.getMatric());
28             myPs.setString(2, student.getName());
29             result = myPs.executeUpdate();
30
31         } catch (Exception e) {
32             System.out.println("Exception is ;" + e);
33         }
34         return result;
35     }
36 }

```

```

37 public List<Student> retrieveAllStudent() {
38     List<Student> studentAll = new ArrayList<Student>();
39     Student student;
40     try {
41         Statement myStatement = connection.createStatement();
42         String myQuery = "select * from student";
43         ResultSet myRs = myStatement.executeQuery(myQuery);
44         while (myRs.next()) {
45             student = new Student();
46             student.setId(myRs.getInt(1));
47             student.setMatric(myRs.getString(2));
48             student.setName(myRs.getString(3));
49             studentAll.add(student);
50         }
51     } catch (Exception e) {
52         System.out.println("Exception is ;" + e);
53     }
54     return studentAll;
55 }

```

```

56
57 public Student retrieveOneStudent(int studentId) {
58     Student student = new Student();
59     try {
60         Statement myStatement = connection.createStatement();
61         String myQuery = "select * from student "
62             + "where stud_id=" + studentId;
63         ResultSet myRs = myStatement.executeQuery(myQuery);
64         while (myRs.next()) {
65             student.setId(myRs.getInt(1));
66             student.setMatric(myRs.getString(2));
67             student.setName(myRs.getString(3));
68         }
69     } catch (Exception e) {
70         System.out.println("Exception is ;" + e);
71     }
72     return student;
73 }

```

```

75 public int updateStudent(Student student) {
76     try {
77         String mySqlQuery = "update student "
78             + "set stud_matric=?, stud_name=? "
79             + "where stud_id=?";
80         PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
81         myPs.setString(1, student.getMatric());
82         myPs.setString(2, student.getName());
83         myPs.setInt(3, student.getId());
84         result = myPs.executeUpdate();
85     } catch (Exception e) {
86         System.out.println("Exception is ;" + e);
87     }
88     return result;
89 }
90
91 public int deleteStudent(int studentId) {
92     try {
93         String mySqlQuery = "delete from student where stud_id=?";
94         PreparedStatement myPs = connection.prepareStatement(mySqlQuery);
95         myPs.setInt(1, studentId);
96         result = myPs.executeUpdate();
97     } catch (Exception e) {
98         System.out.println("Exception is ;" + e);
99     }
100     return result;
101 }
102 }
103 }
104 }

```

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*.

```

25 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
26     throws ServletException, IOException {
27
28     try {
29         // read the "command" parameter
30         String theCommand = request.getParameter("command");
31         // if the command is missing, then default to listing students
32         if (theCommand == null) {
33             theCommand = "LIST";
34         }
35         // route to the appropriate method
36         switch (theCommand) {
37             case "LIST":
38                 listStudent(request, response);
39                 break;
40             case "ADD":
41                 addStudent(request, response);
42                 break;
43             case "LOAD":
44                 loadStudent(request, response);
45                 break;
46             case "UPDATE":
47                 updateStudent(request, response);
48                 break;
49             case "DELETE":
50                 deleteStudent(request, response);
51                 break;
52             default:
53                 listStudent(request, response);
54         }
55     } catch (Exception ex) {
56         Logger.getLogger(StudentServlet.class.getName()).log(Level.SEVERE, null, ex);
57     }
58 }

```

```

99 private void addStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
100     // read student info from form data
101     String name = request.getParameter("name");
102     String matric = request.getParameter("matric");
103
104     Student student = new Student();
105     student.setMatric(matric);
106     student.setName(name);
107     StudentDAO studentDao = new StudentDAO();
108     int result = studentDao.addStudent(student);
109     if (result > 0) {
110         request.setAttribute("theMessage", "Success Add Record");
111         listStudent(request, response);
112     }
113 }
114
115 private void listStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
116
117     //Step 1: Get student data from studentDao
118     StudentDAO studentDao = new StudentDAO();
119     List<Student> allStudent = studentDao.retrieveAllStudent();
120
121     //Step 2: Add student to the request
122     request.setAttribute("theStudents", allStudent);
123
124     //Step 3: Send to JSP page view
125     RequestDispatcher dispatcher = request.getRequestDispatcher("/Task1/process-view-student.jsp");
126
127     //Step 4: Forward to JSP
128     dispatcher.forward(request, response);
129
130 }

```

```

132 private void loadStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
133     // read student id from form data
134     int theStudentId = Integer.parseInt(request.getParameter("id"));
135
136     // get student from database (db util)
137     StudentDAO studentDao = new StudentDAO();
138     Student theStudent = studentDao.retrieveOneStudent(theStudentId);
139
140     // place student in the request attribute
141     request.setAttribute("theStudent", theStudent);
142
143     // send to jsp page: process-update-student.jsp
144     RequestDispatcher dispatcher = request.getRequestDispatcher("/Task1/process-update-student.jsp");
145     dispatcher.forward(request, response);
146 }

```

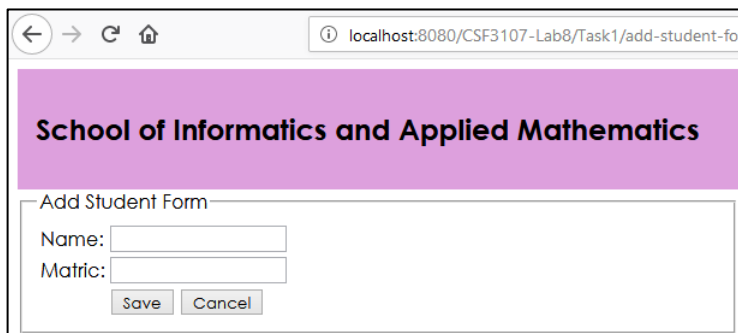
```

148 private void updateStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
149
150     // read student info from form data
151     int studentId = Integer.parseInt(request.getParameter("hidid"));
152     String name = request.getParameter("name");
153     String matric = request.getParameter("matric");
154
155     // create a new student object
156     StudentDAO studentDao = new StudentDAO();
157     Student student = new Student();
158
159     student.setId(studentId);
160     student.setName(name);
161     student.setMatric(matric);
162
163     // perform update on database
164     int result = studentDao.updateStudent(student);
165     if (result > 0) {
166         request.setAttribute("theMessage", "Success Update Record");
167         listStudent(request, response);
168     }
169 }
170
171 private void deleteStudent(HttpServletRequest request, HttpServletResponse response) throws Exception {
172     int studentId = Integer.parseInt(request.getParameter("id"));
173     StudentDAO studentDao = new StudentDAO();
174     int result = studentDao.deleteStudent(studentId);
175     if (result > 0) {
176         request.setAttribute("theMessage", "Success Delete Record");
177         listStudent(request, response);
178     }
179 }
180 }

```


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*

```
7  <%@page import="java.util.ArrayList"%>
8  <%@page import="java.util.List"%>
9  <%@page contentType="text/html" pageEncoding="UTF-8"%>
10 <%@page import="com.lab8.task1.Student"%>
11 <!DOCTYPE html>
12 <html>
13 <head>
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15 <title>Task 1: Retrieve Student</title>
16 </head>
17 <body style="font-family: century gothic; max-width: 600px">
18 <div style="font-family: century gothic; padding: 15px ; background-color: plum;">
19 <h1>Display Student Data from Database</h1></div>
20 <div>
21 <input type="button" value="Add Student" style="font-family: century gothic;"
22 <onclick="window.location.href = 'Task1/add-student-form.jsp';"
23 <return false;"/>
24 </div>
25 <table border="1">
26 <tr>
27 <th>ID</th>
28 <th>MATRIC</th>
29 <th>NAME</th>
30 <th>ACTIONS</th>
31 </tr>
32
33 <%
34 if (request.getAttribute("theMessage") != null) {
35 String message = (String) request.getAttribute("theMessage");
36 out.println("<script type='text/javascript'>");
37 out.println("alert(\"" + message + "\");");
38 out.println("</script >");
39 }
40 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 }
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 consists 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 Motorcycle), 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:
 - Vehicle'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

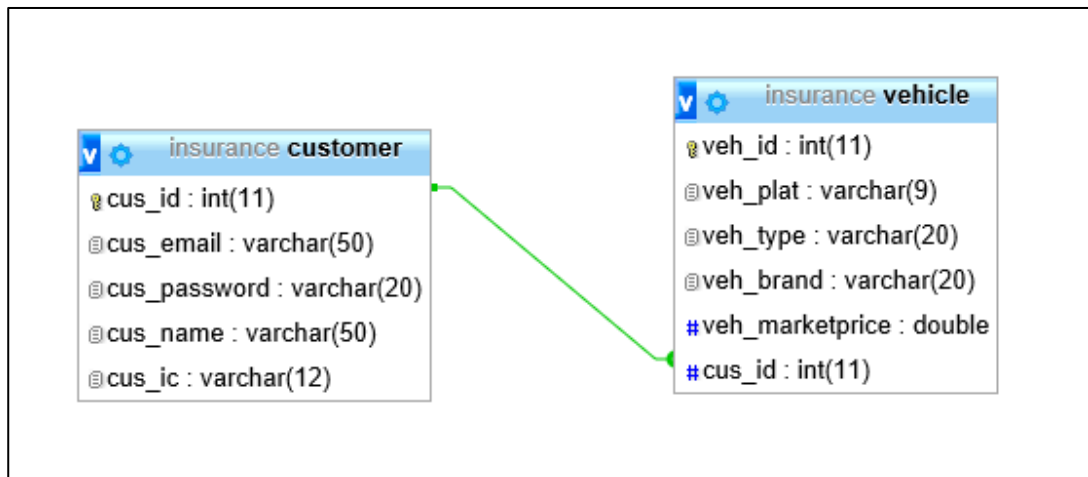
The calculation formula for third party

- 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 market price

- 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
<input type="checkbox"/> 1	cus_id	int(11)			No	None	AUTO_INCREMENT
<input type="checkbox"/> 2	cus_email	varchar(50)			No	None	
<input type="checkbox"/> 3	cus_password	varchar(20)			No	None	
<input type="checkbox"/> 4	cus_name	varchar(50)			No	None	
<input type="checkbox"/> 5	cus_ic	varchar(12)			No	None	

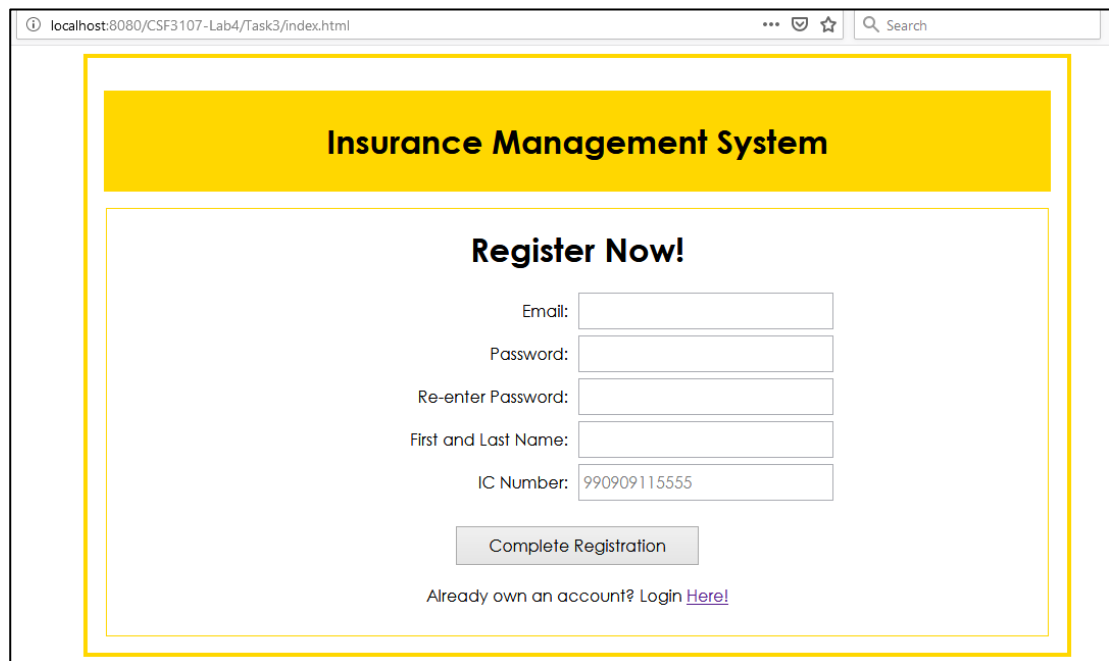
Customer Table

#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/> 1	veh_id	int(11)			No	None	AUTO_INCREMENT
<input type="checkbox"/> 2	veh_plat	varchar(9)			No	None	
<input type="checkbox"/> 3	veh_type	varchar(20)			No	None	
<input type="checkbox"/> 4	veh_brand	varchar(20)			No	None	
<input type="checkbox"/> 5	veh_marketprice	double			No	None	
<input type="checkbox"/> 6	cus_id	int(11)			No	None	

Vehicle Table

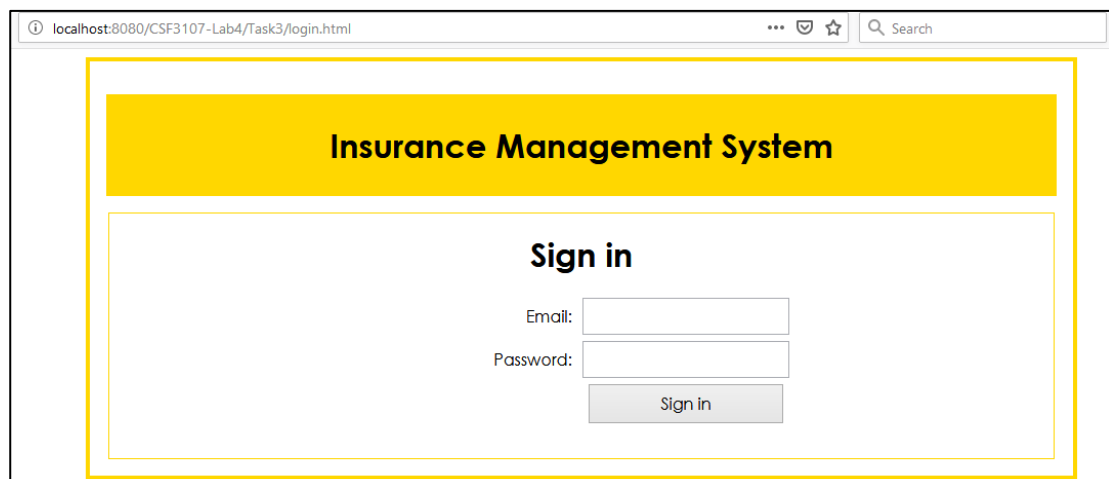
3. Create a new folder inside *Web-Pages* and name it as *Exercise*.
4. Produce the interfaces for Customer Authentication, Vehicle Registration and Insurance Quotation modules. The examples of interface design for each modules are given below. **You may design your own interface according to your preferences.**

4.1 Customer Authentication Module



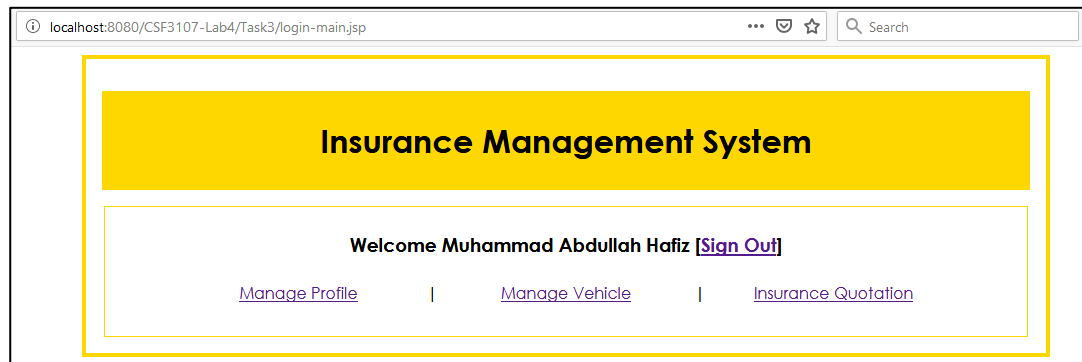
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/CSF3107-Lab4/Task3/index.html'. The page features a yellow header with the text 'Insurance Management System'. Below the header, the title 'Register Now!' is centered. The registration form includes five input fields: 'Email:', 'Password:', 'Re-enter Password:', 'First and Last Name:', and 'IC Number:'. The 'IC Number' field contains the value '990909115555'. A 'Complete Registration' button is positioned below the fields. At the bottom, a link reads 'Already own an account? Login [Here!](#)'.

Interface of Customer Registration

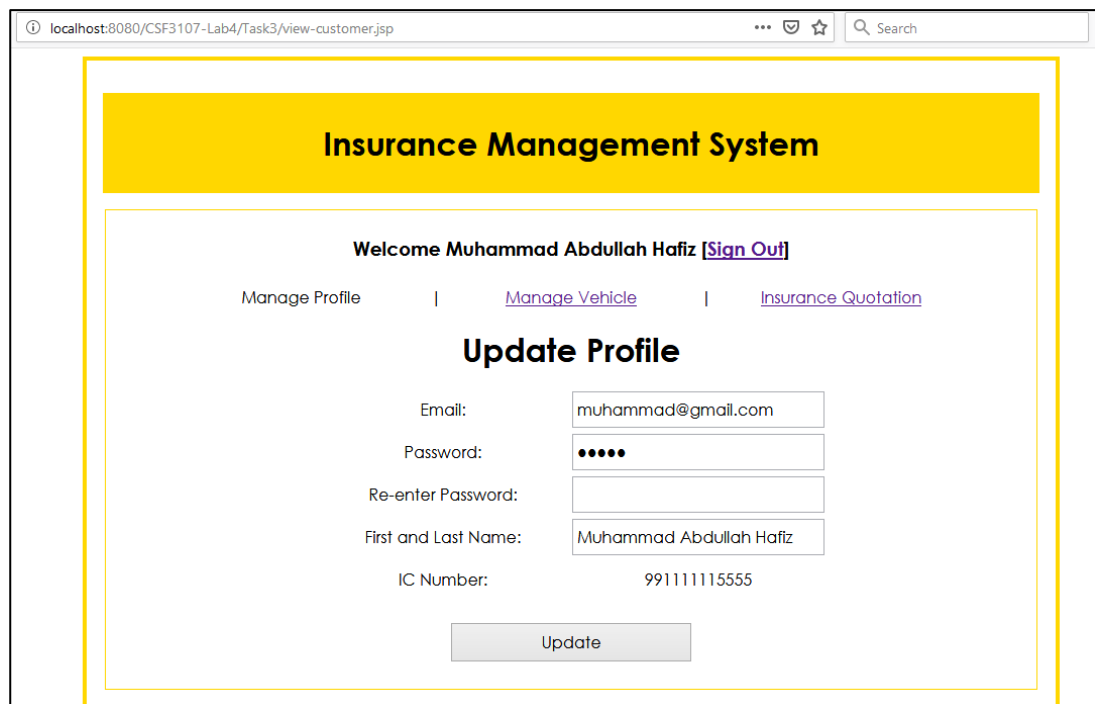


The screenshot shows a web browser window with the address bar displaying 'localhost:8080/CSF3107-Lab4/Task3/login.html'. The page features a yellow header with the text 'Insurance Management System'. Below the header, the title 'Sign in' is centered. The login form includes two input fields: 'Email:' and 'Password:'. A 'Sign in' button is positioned below the fields.

Interface of Customer Login

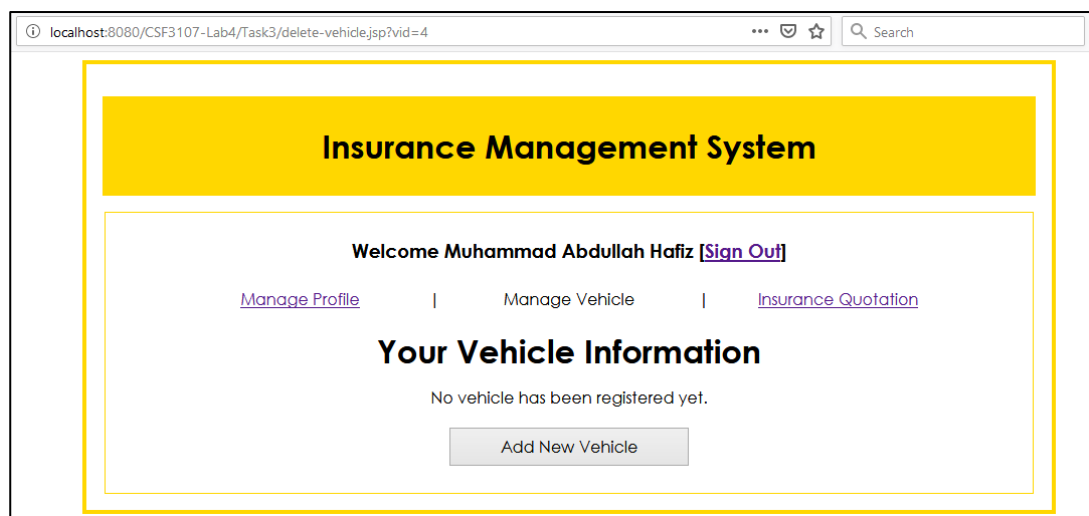


Interface of After Successful Login



Interface of Update Profile

4.2 Vehicle Registration Module



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

localhost:8080/CSF3107-Lab4/Task3/view-vehicle.jsp

...

🔍 Search

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

[Manage Profile](#)
|
 [Manage Vehicle](#)
|
 [Insurance Quotation](#)

Your Vehicle Information

Plat Number	Type	Brand	Market Price (RM)	Actions
TAK1787	Car	Perodua	40000	Update Delete
TBL1952	Car	Nissan	80000	Update Delete
RY8281	Car	Nissan	150000	Update Delete

Add New Vehicle

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

localhost:8080/CSF3107-Lab4/Task3/register-vehicle.jsp

...

🔍 Search

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

Register Vehicle

Plat Number:

Type:

☒ Car
 ☐ Motorcycle

Brand:

Honda

Market Price (RM):

Register

Cancel

Interface of Register Vehicle (Create)

localhost:8080/CSF3107-Lab4/Task3/update-vehide.jsp?vid=6

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

Update Vehicle

Plat Number:

Type: ☒ Car ☐ Motorcycle

Brand:

Market Price (RM):

Interface of Update Vehicle

localhost:8080/CSF3107-Lab4/Task3/view-vehicle.jsp

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

[Manage Profile](#) | [Insurance Quotation](#)

Are you sure you want to delete?

Your Vehicle

Plat Number	Type	Brand	Price (RM)	Actions
TAK1787	Car	Perodua	40000	Update Delete
RY8281	Car	Nissan	150000	Update Delete
TBL1952	Car	Nissan	80000	Update Delete

Interface of Deleting Vehicle

4.3 Insurance Quotation Module

The screenshot shows a web browser window with the URL `localhost:8080/CSF3107-Lab4/Task3/insurance-main.jsp`. The page has a yellow header with the text "Insurance Management System". Below the header, a welcome message reads "Welcome Muhammad Abdullah Hafiz [Sign Out]". A navigation bar contains links for "Manage Profile", "Manage Vehicle", and "Insurance Quotation". The main heading is "Request for Insurance Quotation". There are three dropdown menus: "Select your vehicle:" with the value "TAK1787", "Coverage Type:" with the value "Comprehensive", and "No Claim Discount (NCD):" with the value "10%". A "Get Insurance Quotation" button is at the bottom.

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

[Manage Profile](#) | [Manage Vehicle](#) | Insurance Quotation

Request for Insurance Quotation

Select your vehicle:

Coverage Type:

No Claim Discount (NCD):

Interface of Request for Insurance Quotation

The screenshot shows a web browser window with the URL `localhost:8080/CSF3107-Lab4/Task3/get-quotation.jsp`. The page has a yellow header with the text "Insurance Management System". Below the header, a welcome message reads "Welcome Muhammad Abdullah Hafiz [Sign Out]". A navigation bar contains links for "Manage Profile", "Manage Vehicle", and "Insurance Quotation". The main heading is "The Insurance Quotation". The quotation details are displayed in a table format.

Insurance Management System

Welcome Muhammad Abdullah Hafiz [\[Sign Out\]](#)

[Manage Profile](#) | [Manage Vehicle](#) | [Insurance Quotation](#)

The Insurance Quotation

Plat Number:	TAK1787
Coverage:	Comprehensive
NCD	55%
Market Price (RM):	40000.0

Insurance Amount (RM):	720.00000000000001
6% SST (RM):	43.2
TOTAL (Include 6% SST):	763.20000000000002

Interface of Display Insurance Quotation