Section 1: Scenario and design

Case Study Scenario: A simple system, which maintains an inventory of items. The scenario is designed to demonstrate all possible utilization of the characteristics of the BAW-MVC style, with a minimal set of features for easy referencing and understanding.

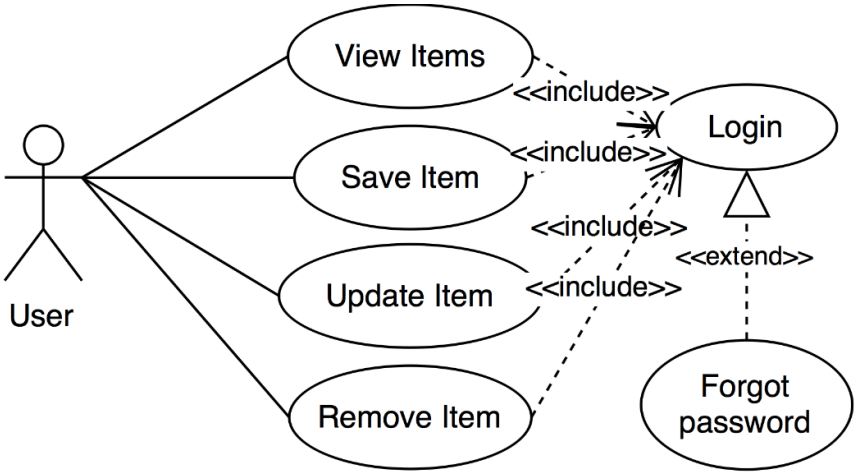


Figure1: Use case diagram

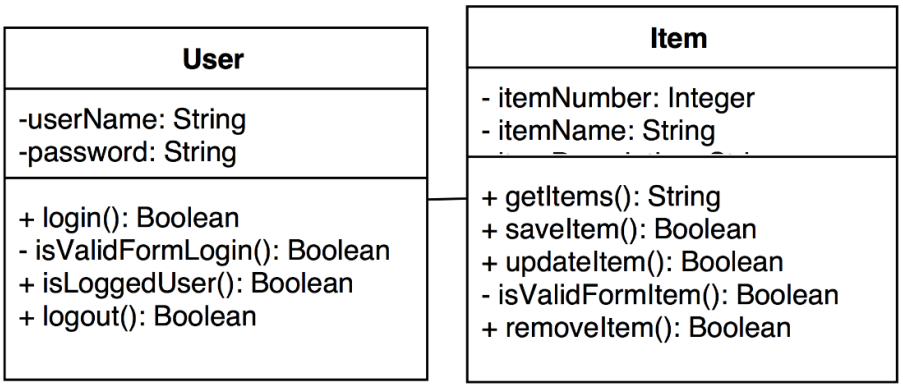


Figure 2: Class diagram

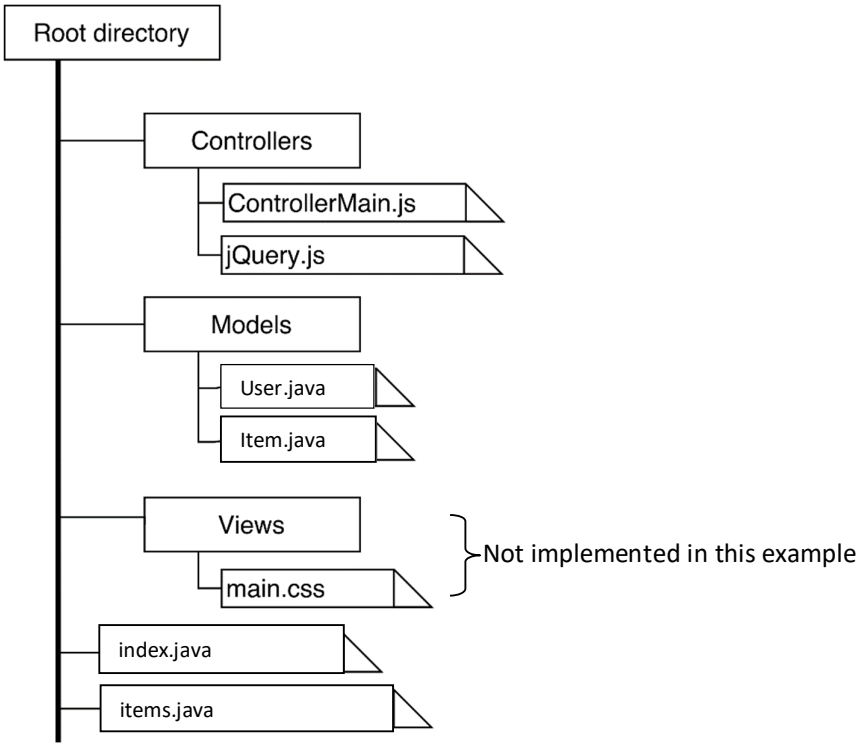


Figure 3: Directory structure of the system

Figure 4: Loging GUI

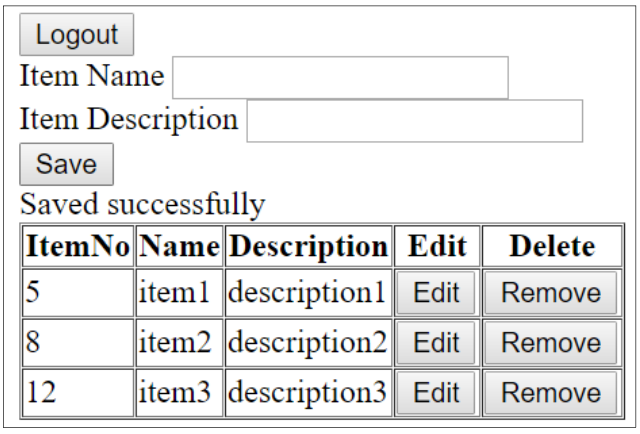
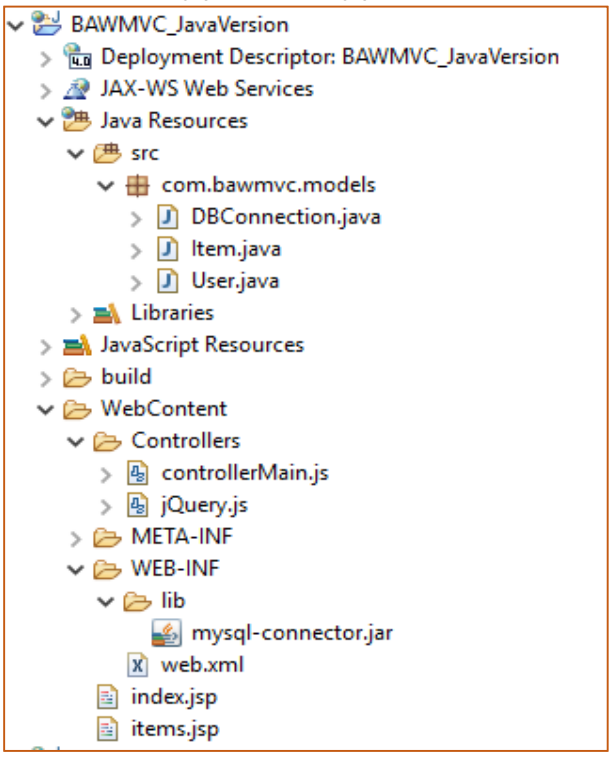


Figure 4: Form and Grid UI pattern for the item page

Section 2: Create and configure the project

Step 1: Create a web project

- Create a new Dynamic Web Project in Eclipse
- Add a package called "models" to include java files of the Server-Model
    - Create the server-model files **DBConnection.java, Item.java**, and **User.java**
- Add a directory called "**Controllers**" to the "**WebContent**" directory.
    - Download the *.js* file of the latest jQuery version to this directory
    - Create a new file and name it as *controllerMain.js*. This file will contain the code of the Controller.
    - **NOTE**: Since the system is small, the same *controllerMain.js* file will contain the code of the Client-Model as well.
    - Create the Views **index.jsp** and **items.jsp** inside the **WebContent** directory



Step 2: Configure for the MySQL database connection

In this application we use *JDBC driver from MySQL* to connect to the database. You can find the latest from URL: http://dev.mysql.com/downloads/connector/j

1. **JDBC Driver:** Unzip downloaded compressed *mysql-connector.jar* file and copy it to the **lib** directory of **WEB-INF** which is in your **WebContent** directory.
2. **Connection URL:** The connection URL for the MySQL database is:
   jdbc:mysql://localhost:3306/studentdb where **jdbc** is the API, **mysql** is the database server, **localhost** is the server name on which MySQL is running, we may also use IP address, **3306** is the port number and **studentdb** is the database name. You may use any database, in such case; you need to replace the **studentdb** with your database name.
3. **Username:** The default username for the MySQL database is **root**.
4. **Password:** It is the password given by the user at the time of installing MySQL.

Section 3: Server-side development

Since this practical is focusing on the client-side development, the server-side development code is given to you.

NOTE: You need to know how to generate appropriate HTML content in server-side, needed by the Views.

Step 1: Implement Server-Model: DBConnection

DBConnection.java file. Call this "createConnection" method of "DBConnection" class when you need to create a connection to the database with the configurations specified inside the method

```java
package com.bawmvc.models;

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

public class DBConnection {

    public static Connection createConnection() {

        Connection con = null;
        String url = "jdbc:mysql://localhost:3306/studentdb"; //MySQL URL and fol
        String username = "root"; //MySQL username
        String password = "root"; //MySQL password

        try {
            try {
                Class.forName("com.mysql.jdbc.Driver"); //loading mysql driver
            }
            catch (ClassNotFoundException e) {
                e.printStackTrace();
            }

            con = DriverManager.getConnection(url, username, password); //attempt
            System.out.println("Printing connection object "+con);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return con;
    }
}
```

Step 2: Implement Server-Model: User

```
1  package com.bawmvc.models;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7
8  public class User {
9
10     public String login(String userN, String passW) {
11
12         Connection con = null;
13         Statement statement = null;
14         ResultSet resultSet = null;
15
16         String userNameDB = "";
17         String passwordDB = "";
18
19         try {
20             con = DBConnection.createConnection(); //establishing connection
21             statement = con.createStatement(); //Statement is used to write c
22             resultSet = statement.executeQuery("select nameUsers, passUsers f
23
24             while(resultSet.next()) {  //Until next row is present otherwise
25                 userNameDB = resultSet.getString("nameUsers"); //fetch the va
26                 passwordDB = resultSet.getString("passUsers");
27
28                 if(userN.equals(userNameDB) && passW.equals(passwordDB)) {
29                     return "SUCCESS"; //If the user entered values are alread
30                 }
31             }
        }
        catch (SQLException e) {
            e.printStackTrace();
        }

        return "Invalid user credentials..."; // Just returning appropriate messa
    }
}
```

Step 3: Implement Server-Model: Item

NOTE: Continue this class by adding methods to update and delete an item.

```
package com.bawmvc.models;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Item {

	public String getItems() {

		String itemsGrid = null;
		Connection con = null;
		Statement st = null;
		ResultSet rs = null;

	try {
	con = DBConnection.createConnection(); //establishing connection
	st = con.createStatement(); //Statement is used to write queries.
Read more about it.
	rs = st.executeQuery("select * from items"); //Here table name is
users and nameUsesrs, passUsers are columns. fetching all the records and
storing in a resultSet.

	if (rs.first())
	{

itemsGrid = "<table border='1' cellspacing='1'
cellpadding='1'><tr><th>No</th><th>Name</th><th>Description</th><th>Edit</t
h><th>Delete</th></tr>";

rs.beforeFirst();

while(rs.next())
{
  itemsGrid = itemsGrid + "<tr><td>" + rs.getString(1) + "</td>"
	+ "<td>" + rs.getString(2) + "</td>"
	+ "<td>" + rs.getString(3) + "</td>"
	+ "<td><input id=\"btnEdit\" type=\"button\" name=\"btnEdit\"
 param=\"" + rs.getString(1) + "\" value=\"Edit\"</td>"
	+ "<td>" + "<input id=\"btnRemove\" type=\"button\"
name=\"btnRemove\" param=\"" + rs.getString(1) + "\"
value=\"Remove\"</td></tr>";
			}
}
		else
			itemsGrid = "There are no items...";
		itemsGrid = itemsGrid + "</table></br>";
		}
		catch (SQLException e) {
			e.printStackTrace();
		}

		return itemsGrid;
	}


	public String saveAnItem(String itmName, String itmDesc) {
```

```
            String res = null;
            String sql = null;
            Connection con = null;
            Statement st = null;

            try {
                  con = DBConnection.createConnection(); //establishing
connection
                  st = con.createStatement(); //Statement is used to write
queries. Read more about it.
                  sql = "insert into items (nameitems, descitems) values
('" + itmName + "', '" + itmDesc + "')";
                  st.executeUpdate(sql);
                  res = "Successfully inserted...";
            }
            catch (SQLException e) {
                  e.printStackTrace();
            }

            return res;
      }
```

Section 3: Client-side development

Step 1: Implement View: Login (index.asp)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Login</title>
<script type="text/javascript" src=Controllers/jQuery.js></script>
<script type="text/javascript" src=Controllers/controllerMain.js></script>

</head>
<body>

    <form id="formLogin" action="index.jsp" method="post">
        UserName <input id="txtUserName" name="txtUserName" type="text"> <br>
        Password <input id="txtPassword" name="txtPassword" type="password"> <br>
        <input id="btnLogin" name="btnLogin" type="button" value="Login"> <br>
        <div id="divStsMsgLogin">
            <% out.println(loginMsg); %>
        </div>
    </form>

</body>
</html>
```

Q: There is java code, trying to display the value of a variable named **loginMsg**. What is it?
NOTE: To test the application before completing, remove/comment the java code

Step 2: Implement View: Items (items.jsp)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Items</title>
<script type="text/javascript" src=Controllers/jQuery.js></script>
<script type="text/javascript" src=Controllers/controllerMain.js></script>

</head>
<body>
    <form id="formItems" action="items.jsp" method="post">
        <input id="hidMode" name="hidMode" type="hidden" value="save">
        <input id="hidID" name="hidID" type="hidden" value="0">
        Item Name: <input id="txtItemName" type="text" name="txtItemName"> <br>
        Item Description: <input id="txtItemDesc" type="text" name="txtItemDesc"> <br>
        <input id="btnSave" type="button" name="btnSave" value="Save"><br><br>
        <div id="divStsMsgItem"><% out.println(rudFeedback); %></div>
        <% out.println(itemsGrid); %>
    </form><br>
</body>
</html>
```

Q: There is java code, trying to display the value of a variable named **rudFeedback** and **itemsGrid**. What are they?
NOTE: To test the application before completing, remove/comment the java code

Q: There are no table elements in the html code. But in items GUI you can see a table. How does this items table appear on the View?

Step 3: Implement Controller: Login

Open the *controllerMain.js* file and try the following code.

```
1  //==CONTROLLER=================================================
2  //--User---------------------------------------------------
3  //--Login--
4  $(document).on("click","#btnLogin",function()
5  {
6      var result = isValidFormLogin();//use client-Model
7      if(result=="true")
8      {   $("#formLogin").submit();   }
9      else
10     {   $("#divStsMsgLogin").html(result);  }
11 });
12
```

Q: What is this "isValidFormLogin" function?

Step 4: Implement Client-Model: User

```
45 //==CLIENT-MODEL=============================================
46 //--User-----------------------
47 function isValidFormLogin()
48 {
49     if($.trim($("#txtUserName").val())=="")
50     {   return "Enter Username";    }
51
52     if($.trim($("#txtPassword").val())=="")
53     {   return "Enter Password";    }
54
55     return "true";
56 }
```

This function is to validate the login form. If any empty field it will display a proper message. Otherwise return true as a valid attempt. If and only if this validation is pass at the event of login button click the form will submit. Otherwise the validation error message will be shown through the html element "**divStsMsgLogin**".

Write a similar jQuery function to validate the GUI of items. Return "**true**" if both **Item Name** & **Item Description** is not empty. Otherwise return a proper message.

```
function isValidFormItem()
{
        if ($.trim($("#txtItemName").val()) == "") {
                return "Enter Item Name";
.
.       //Your code to complete the function...
.       //Your code to complete the function...
.
        return "true/false";
}
```

Step 5: Implement Controller: Items

Understand the following event handlers for "**Save**", "**Edit**" and "**Delete**" button clicks

```
//--items-----------------------------------------------------
//--Save/Update--
$(document).on("click","#btnSave", function()
{
    var result = isValidFormItem();//use client-Model
    if(result=="true")
    {   $("#formItems").submit();   }
    else
    {   $("#divStsMsgItem").html(result);   }
});

//--Edit--
$(document).on("click", "#btnEdit", function()
{
    $("#hidMode").val("update");
    $("#hidID").val($(this).attr("param"));
    $("#txtItemName").val($(this).closest("tr").find('td:eq(1)').text());
    $("#txtItemDesc").val($(this).closest("tr").find('td:eq(2)').text());
});

//--Remove--
$(document).on("click", "#btnRemove", function()
{
    $("#hidMode").val("remove");
    $("#hidID").val($(this).attr("param"));
    var res = confirm("Are you sure?");
      if (res == true) {
          $("#formItems").submit();
      }
});
```

Purpose of this Line?

What does this line Do?

Why the value of hidMode is changing to "update"? What was the initial value???

What does these 2 lines Do?

Why we need this line?
What is hidID???

What is the purpose of variable "res"???

NOTE: Try to understand the highlighted code lines of above jQuery event handling functions.

Q: What are the hidden fields?
Q: Why we use hidden fields?

Step 6: Implement server-side code for the Views

For this application to work properly, you may need some code in the Views to be executed in the server, before loading to the browser.

Add the following code to the top of the correct Views. Try to understand the need for them and their purpose.

```
<%@ page import="com.bawmvc.models.User" %>
<%@ page import ="javax.sql.*" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<%
    User user = new User();
    String loginMsg = "Please login to continue...";

    if (request.getMethod().equalsIgnoreCase("post"))
    {
        if (user.login(request.getParameter("txtUserName"), request.getParameter("txtPassword")).equals("SUCCESS"))
        {
            //RequestDispatcher is used to send the control to the invoked page
            request.getRequestDispatcher("/items.jsp").forward(request, response);
        }
        else
        {
            loginMsg = "Invalid credentials...";
        }
    }
%>
```

```
<%@ page import="com.bawmvc.models.Item" %>
<%@ page import ="javax.sql.*" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<%
    Item item = new Item();
    String itemsGrid = "";
    String rudFeedback = "";

    if (request.getParameter("hidMode") != null && request.getParameter("hidMode").equalsIgnoreCase("save"))
    {
        rudFeedback = item.saveAnItem(request.getParameter("txtItemName"), request.getParameter("txtItemDesc"));
    }
    if (request.getParameter("hidMode") != null && request.getParameter("hidMode").equalsIgnoreCase("update"))
    {
        rudFeedback = item.updateAnItem(request.getParameter("hidID"), request.getParameter("txtItemName"),
                request.getParameter("txtItemDesc"));
    }
    if (request.getParameter("hidMode") != null && request.getParameter("hidMode").equalsIgnoreCase("remove"))
    {
        rudFeedback = item.deleteAnItem(request.getParameter("hidID"));
    }

    itemsGrid = item.getItems();
%>
```