

# My Expense Tracker

Abstract	2
MVC Implementation	2
Database	18
Tables	19
Execution	21
Url's	21

- 17081 Sruthi Mavuleti

# Abstract

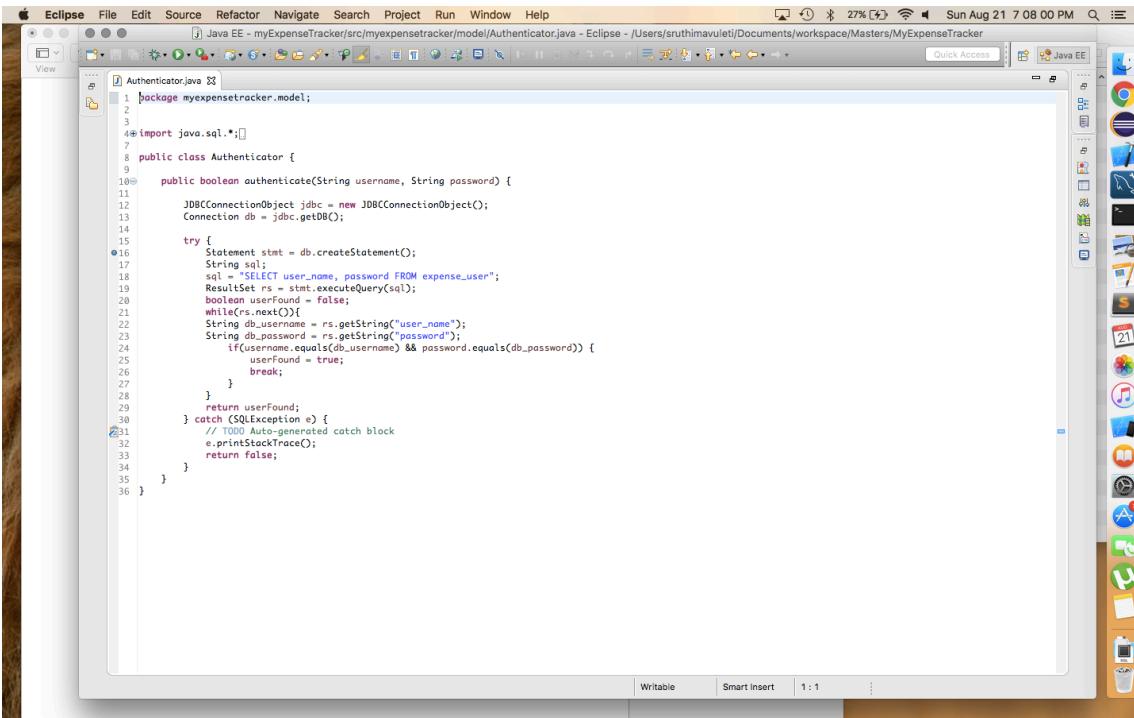
My Expense Tracker is a web-based application for tracking all my expenses. In our daily life we want to note all our expenses and calculate how much we earn and how much we spent. This application allows users to track their expenses very easily. Users just need to sign up with their names and passwords and login to add their expense data. Initially there will be an admin, who can add users after login and later, a user can add many other users to enter and save their expenses data.

## MVC Implementation

This is a web-based application written using MVC architecture of Java Servlets. And application is deployed on Tomcat 7. Java Server Pages, Java Standard Tag Library, Hyper Text Markup Language.

Model:

Myexpensetracker.model



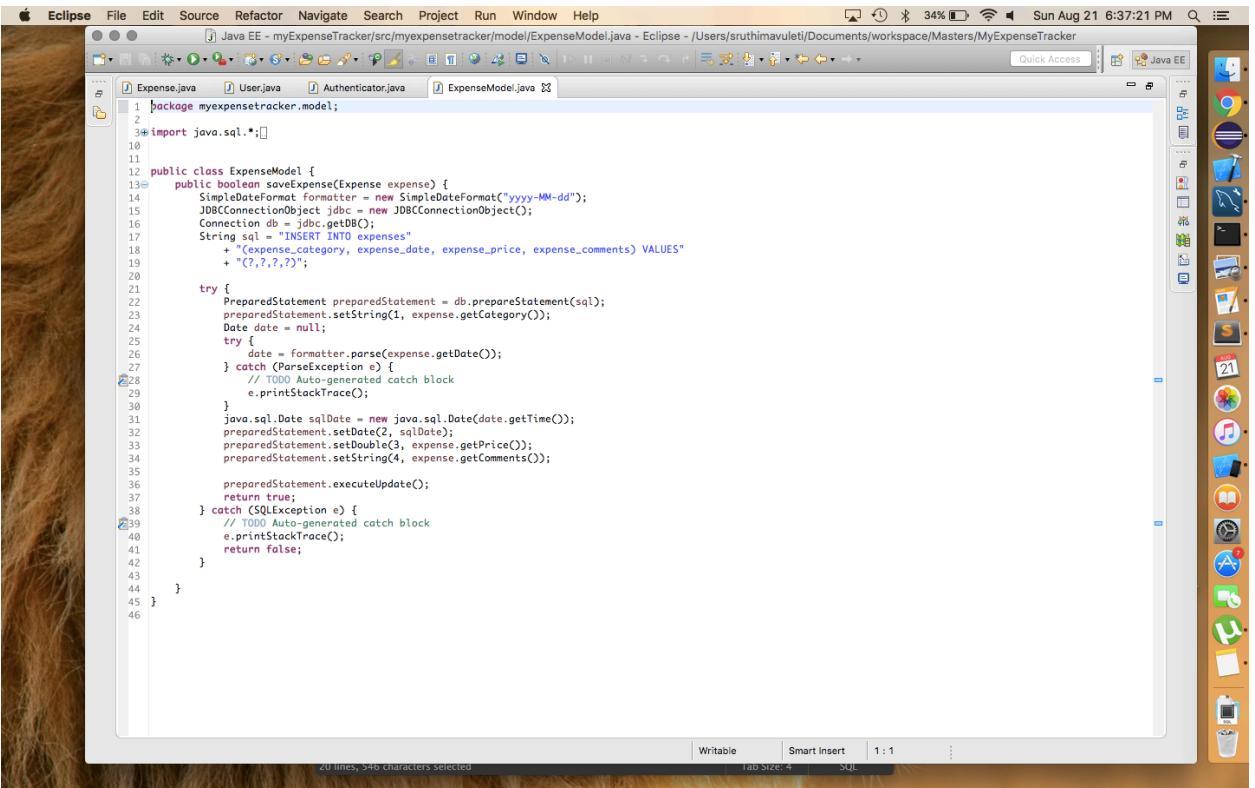
The screenshot shows the Eclipse IDE interface with the code for `Authenticator.java` in the center editor window. The code implements a class named `Authenticator` that checks if a given username and password match those stored in a database table named `expense_user`. The code uses JDBC to execute a query and check the results. The Eclipse toolbar, menu bar, and various tool windows are visible around the code editor.

```
1 package myexpensetracker.model;
2
3
4 import java.sql.*;
5
6 public class Authenticator {
7
8     public boolean authenticate(String username, String password) {
9
10        JDBCConnectionObject jdbc = new JDBCConnectionObject();
11        Connection db = jdbc.getDB();
12
13        try {
14            Statement stmt = db.createStatement();
15            String sql;
16            sql = "SELECT user_name, password FROM expense_user";
17            ResultSet rs = stmt.executeQuery(sql);
18            boolean userFound = false;
19            while(rs.next()) {
20                if(rs.getString("user_name").equals(username) && rs.getString("password").equals(password)) {
21                    userFound = true;
22                    break;
23                }
24            }
25        } catch (SQLException e) {
26            // TODO Auto-generated catch block
27            e.printStackTrace();
28        }
29        return userFound;
30    }
31
32 }
```

## Authenticator.java

This is the model for login. It will authenticates from the database.

## ExpenseModel.java

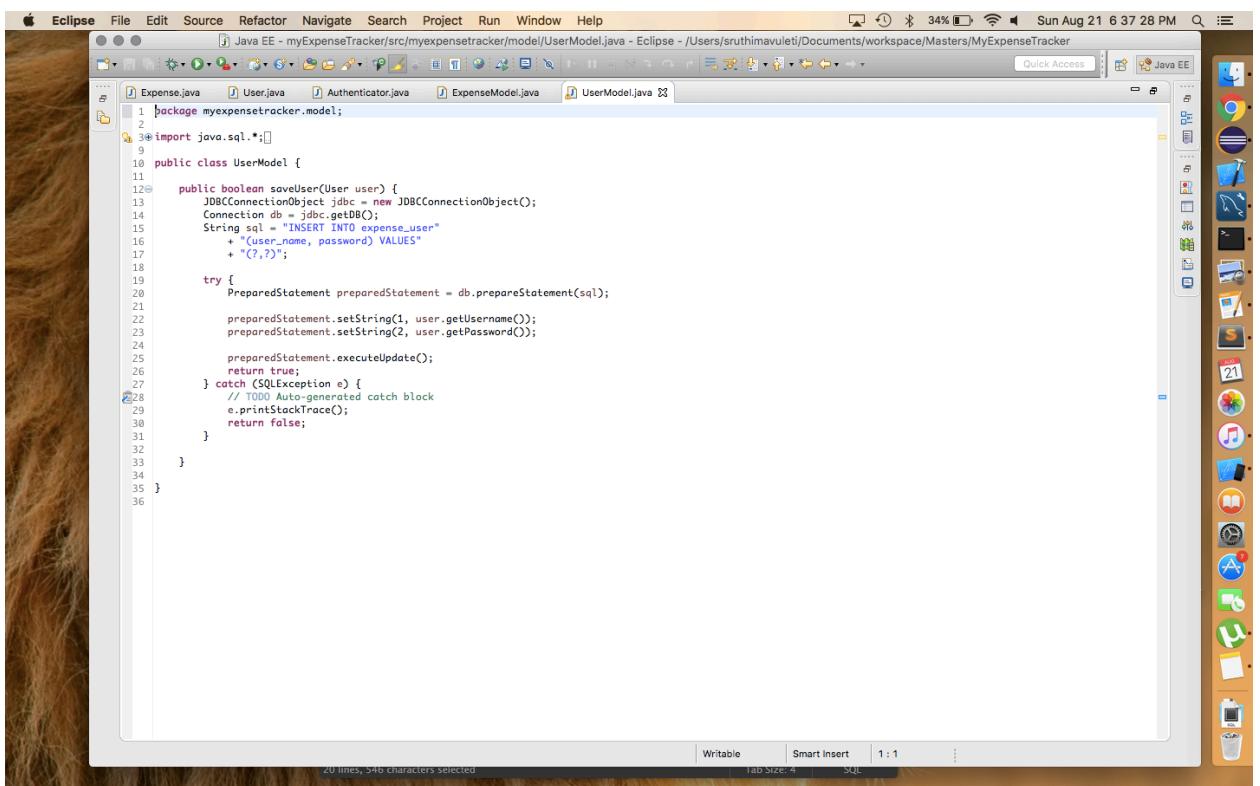
A screenshot of the Eclipse IDE interface. The title bar reads "Java EE - myExpenseTracker/src/myexpensetracker/model/ExpenseModel.java - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main window shows the Java code for "ExpenseModel.java". The code implements a class "ExpenseModel" with a static method "saveExpense" that inserts data into a database table named "expenses".

```
1 package myexpensetracker.model;
2
3 import java.sql.*;
4
5 public class ExpenseModel {
6     public boolean saveExpense(Expense expense) {
7         SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
8         JDBCConnectionObject jdbc = new JDBCConnectionObject();
9         Connection db = jdbc.getDB();
10        String sql = "INSERT INTO expenses"
11                  + "(expense_category, expense_date, expense_price, expense_comments) VALUES"
12                  + "(?, ?, ?, ?)";
13
14        try {
15            PreparedStatement preparedStatement = db.prepareStatement(sql);
16            preparedStatement.setString(1, expense.getCategory());
17            Date date = null;
18            try {
19                date = formatter.parse(expense.getDate());
20            } catch (ParseException e) {
21                // TODO Auto-generated catch block
22                e.printStackTrace();
23            }
24            java.sql.Date sqlDate = new java.sql.Date(date.getTime());
25            preparedStatement.setDate(2, sqlDate);
26            preparedStatement.setDouble(3, expense.getPrice());
27            preparedStatement.setString(4, expense.getComments());
28
29            preparedStatement.executeUpdate();
30            return true;
31        } catch (SQLException e) {
32            // TODO Auto-generated catch block
33            e.printStackTrace();
34            return false;
35        }
36    }
37}
38
39}
```

The status bar at the bottom indicates "20 lines, 546 characters selected".

This  
is  
the model for adding expense by connecting to the database.

## UserModel.java

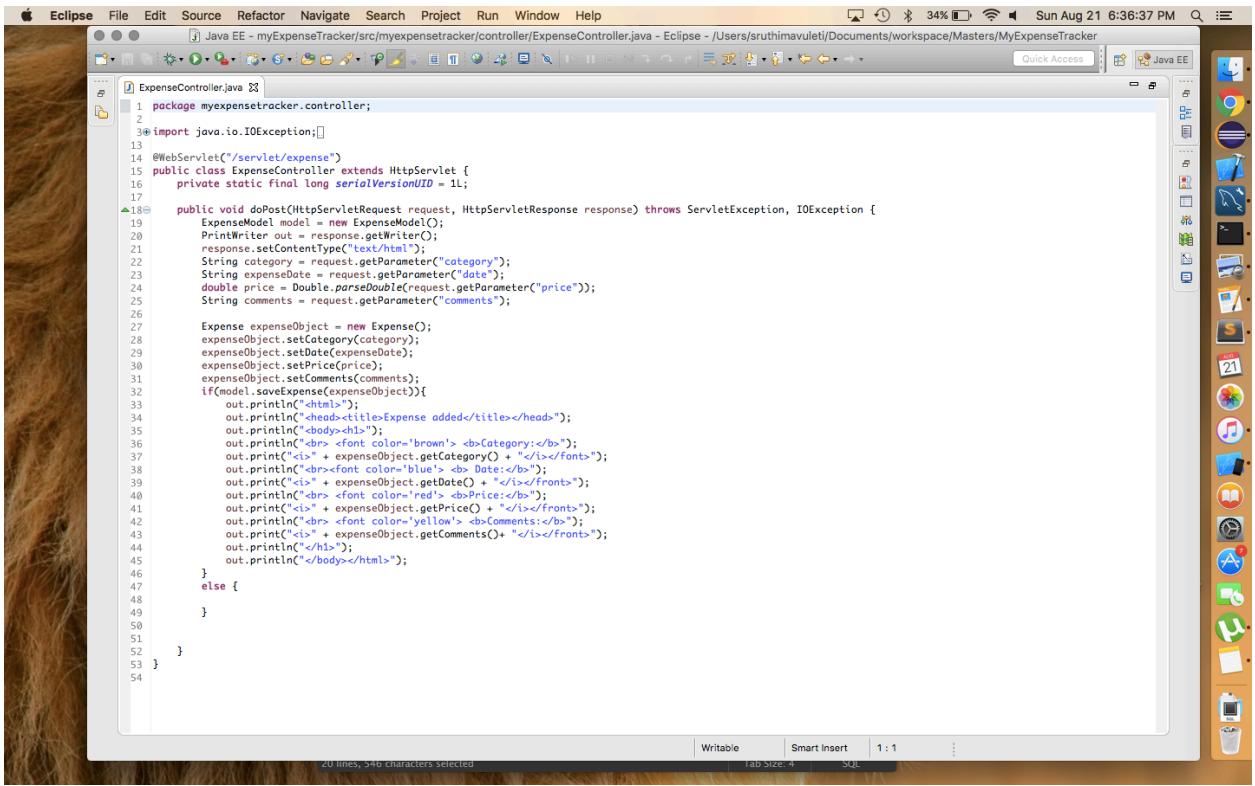


The screenshot shows the Eclipse IDE interface with the User Model Java code open in the editor. The code is for a class named UserModel that implements a saveUser method using JDBC to insert a new user into a database table.

```
1 package myexpensetracker.model;
2
3 import java.sql.*;
4
5 public class UserModel {
6
7     public boolean saveUser(User user) {
8         JDBCConnectionObject jdbc = new JDBCConnectionObject();
9         Connection db = jdbc.getDB();
10        String sql = "INSERT INTO expense_user"
11            + "(user_name, password) VALUES"
12            + "(?,?)";
13
14        try {
15            PreparedStatement preparedStatement = db.prepareStatement(sql);
16            preparedStatement.setString(1, user.getUsername());
17            preparedStatement.setString(2, user.getPassword());
18            preparedStatement.executeUpdate();
19            return true;
20        } catch (SQLException e) {
21            // TODO Auto-generated catch block
22            e.printStackTrace();
23            return false;
24        }
25    }
26
27 }
28
29
30 }
```

This  
is the model to add the user by connecting to the database.

**Controller:**  
**Myexpensetracker.controller**  
**ExpenseController.java**



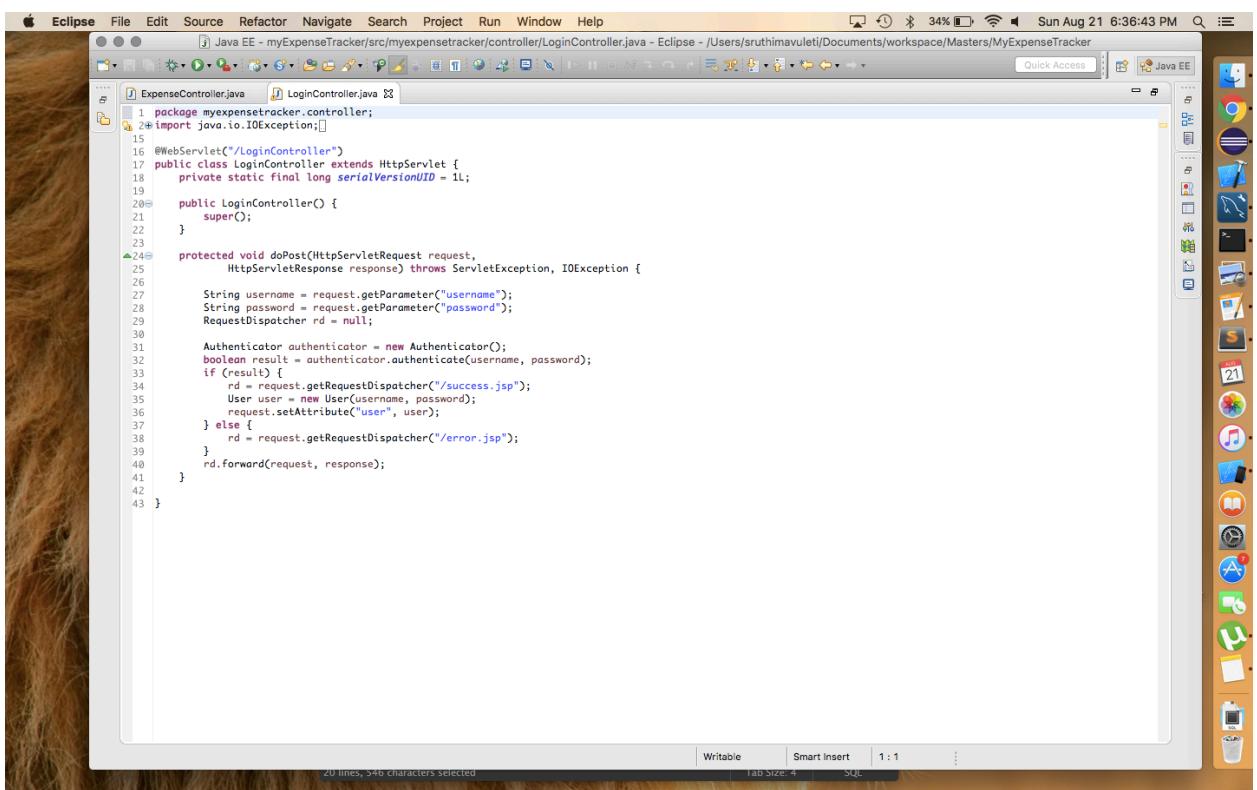
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java EE - myExpenseTracker/src/myexpensetracker/controller/ExpenseController.java - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and help.
- Left Sidebar:** Project Explorer showing the file structure.
- Right Sidebar:** Quick Access bar with various application icons.
- Code Editor:** The main window displays the `ExpenseController.java` source code. The code is a servlet that handles POST requests to save expense data to a model and then prints an HTML response back to the client. It uses Java's `PrintWriter` to output HTML with colored text (brown, blue, red, yellow) to represent different parts of the expense object.
- Status Bar:** Shows "Writable", "Smart Insert", "1:1", and "20 lines, 546 Characters selected".

```
package myexpensetracker.controller;
import java.io.IOException;
@WebServlet("/servlet/expense")
public class ExpenseController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        ExpenseModel model = new ExpenseModel();
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        String category = request.getParameter("category");
        String expenseDate = request.getParameter("date");
        double price = Double.parseDouble(request.getParameter("price"));
        String comments = request.getParameter("comments");
        Expense expenseObject = new Expense();
        expenseObject.setCategory(category);
        expenseObject.setDate(expenseDate);
        expenseObject.setPrice(price);
        expenseObject.setComments(comments);
        if(model.saveExpense(expenseObject)){
            out.println("<html>");
            out.println("<head><title>Expense added</title></head>");
            out.println("<body><h1>");
            out.println("<br> <font color='brown'> <b>Category:</b> ");
            out.print("<i>" + expenseObject.getCategory() + "</i></font> ");
            out.println("<br> <font color='blue'> <b>Date:</b> ");
            out.print("<i>" + expenseObject.getDate() + "</i></font> ");
            out.println("<br> <font color='red'> <b>Price:</b> ");
            out.print("<i>" + expenseObject.getPrice() + "</i></font> ");
            out.println("<br> <font color='yellow'> <b>Comments:</b> ");
            out.print("<i>" + expenseObject.getComments() + "</i></font> ");
            out.println("</h1>");
            out.println("</body></html>");
        }
    }
}
```

this  
controller is for interacting with myExpenseTrackerMain.html page .

## LoginController.java



The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar reads "Java EE - myExpenseTracker/src/myexpensetracker/controller/LoginController.java - Eclipse - /Users/sruhimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main window displays the Java code for LoginController.java:

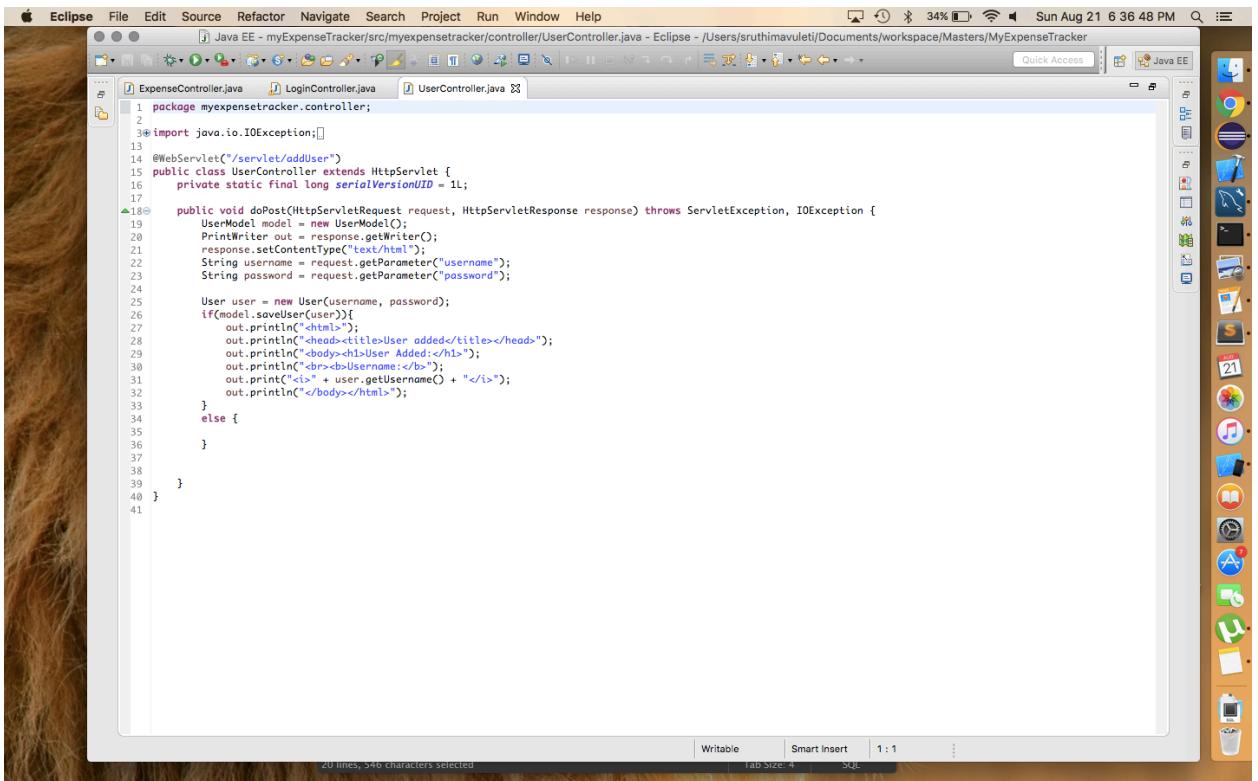
```
1 package myexpensetracker.controller;
2 import java.io.IOException;
3
4 @WebServlet("/")
5 public class LoginController extends HttpServlet {
6     private static final long serialVersionUID = 1L;
7
8     public LoginController() {
9         super();
10    }
11
12    protected void doPost(HttpServletRequest request,
13                          HttpServletResponse response) throws ServletException, IOException {
14
15        String username = request.getParameter("username");
16        String password = request.getParameter("password");
17        RequestDispatcher rd = null;
18
19        Authenticator authenticator = new Authenticator();
20        boolean result = authenticator.authenticate(username, password);
21        if (result) {
22            rd = request.getRequestDispatcher("/success.jsp");
23            User user = new User(username, password);
24            request.setAttribute("user", user);
25        } else {
26            rd = request.getRequestDispatcher("/error.jsp");
27        }
28        rd.forward(request, response);
29    }
30
31 }
```

The code implements a HttpServlet that handles POST requests. It retrieves the username and password from the request parameters. It then uses an Authenticator object to authenticate the user. If successful, it forwards the request to "/success.jsp"; otherwise, it forwards it to "/error.jsp". The code is annotated with Javadoc-style comments.

this

controller interacts with the login.jsp for the username and password with the url: / LoginController. Here, if the login is success it renders /success.jsp if not it goes to error.jsp page.

## UserController.java



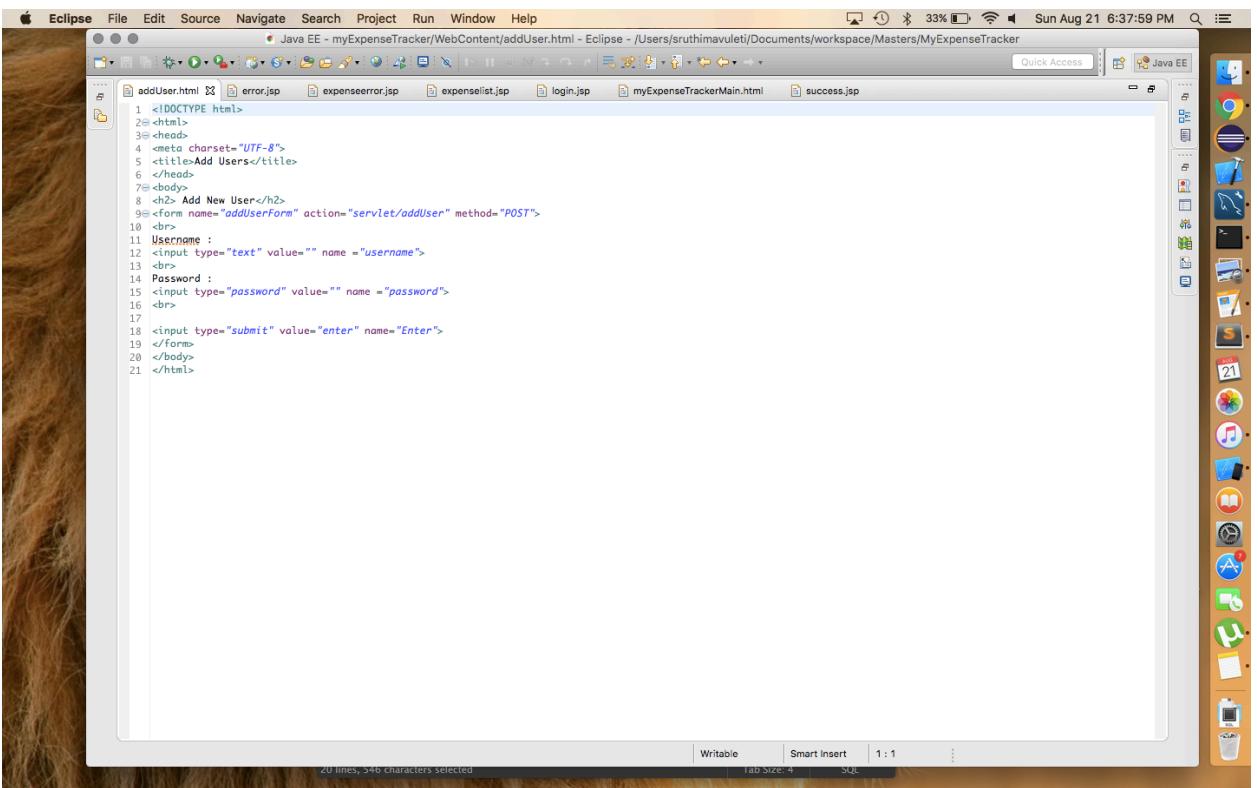
The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar reads "Java EE - myExpenseTracker/src/myexpensetracker/controller/UserController.java - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main editor window displays the following Java code:

```
1 package myexpensetracker.controller;
2
3 import java.io.IOException;
4
5 @WebServlet("/addUser")
6 public class UserController extends HttpServlet {
7     private static final long serialVersionUID = 1L;
8
9     public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
10         UserModel model = new UserModel();
11         PrintWriter out = response.getWriter();
12         response.setContentType("text/html");
13         String username = request.getParameter("username");
14         String password = request.getParameter("password");
15
16         User user = new User(username, password);
17         if(model.saveUser(user)){
18             out.println("<head><title>User added</title></head>");
19             out.println("<body><h1>User Added:</h1>");
20             out.println("<br><b>Username:</b>");
21             out.print("<i>" + user.getUsername() + "</i>");
22             out.println("</body></html>");
23         }
24     }
25
26 }
27
28 }
```

The status bar at the bottom indicates "20 lines, 546 characters selected". The Eclipse toolbar and menu bar are visible at the top, and the Mac OS X dock is visible on the right side of the screen.

This controller is for adding user,it displays the added user name.

View:  
addUser.html

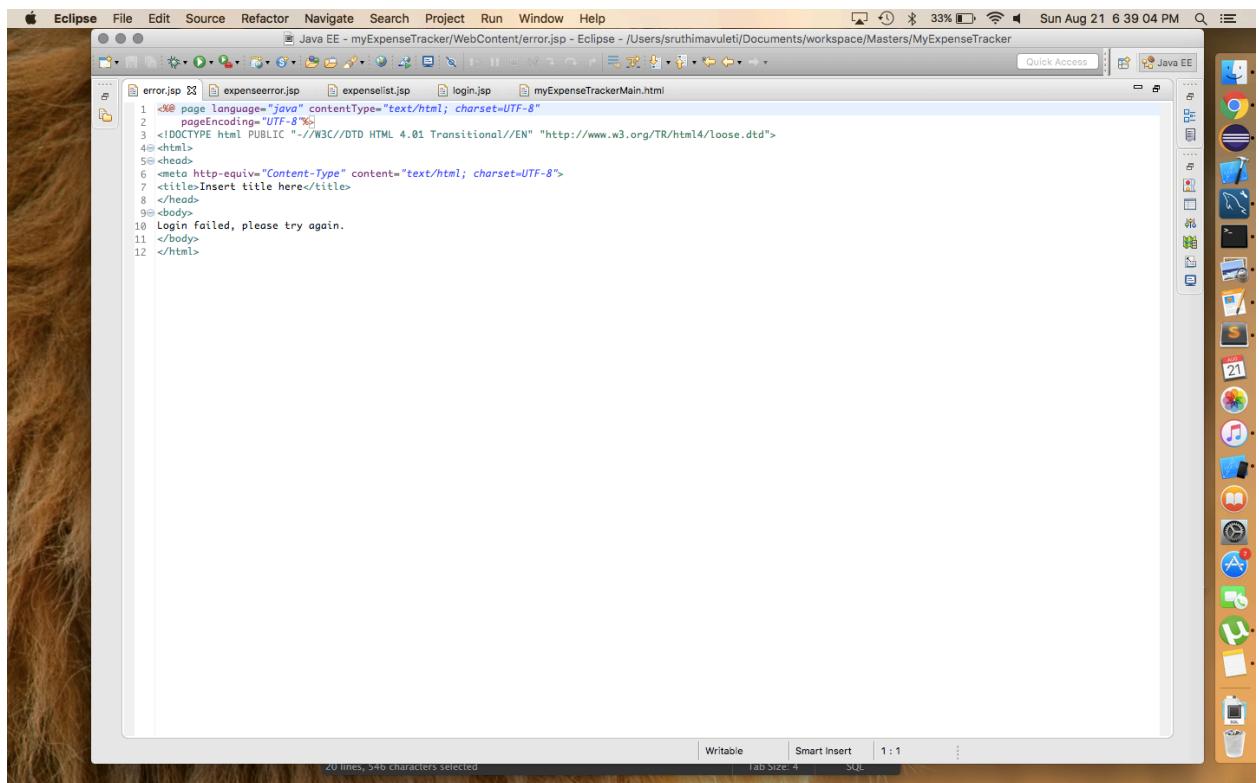


The screenshot shows the Eclipse IDE interface with the title bar "Java EE - myExpenseTracker/WebContent/addUser.html - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main window displays the HTML code for "addUser.html". The code includes a DOCTYPE declaration, an HTML head section with charset="UTF-8", a title "Add Users", and a body section containing a form for adding a new user. The form has fields for Username and Password, and a submit button labeled "Enter". The code is color-coded, and the status bar at the bottom indicates "20 lines, 546 characters selected".

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Add Users</title>
</head>
<body>
<h2> Add New User</h2>
<form name="addUserForm" action="servlet/addUser" method="POST">
<br>
11 <label> Username :</label>
12 <input type="text" value="" name="username">
13 <br>
14 <label> Password :</label>
15 <input type="password" value="" name="password">
16 <br>
17 <input type="submit" value="enter" name="Enter">
18 </form>
20 </body>
21 </html>
```

This  
html page is consists of Username, Password input fields and Submit button. Here the form is connected to servlet/addUser (UserController.java).

## error.jsp



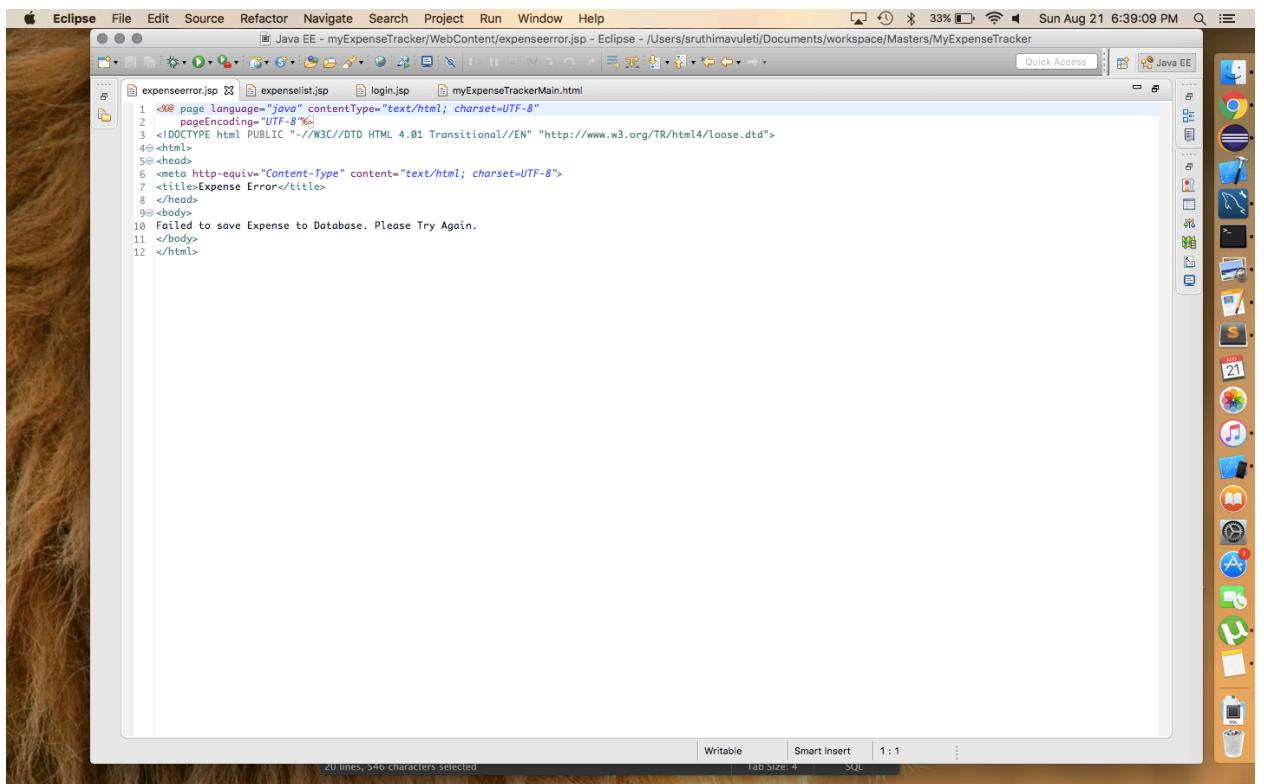
The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar reads "Java EE - myExpenseTracker/WebContent/error.jsp - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The menu bar includes Eclipse, File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, etc. The left sidebar shows project files: error.jsp, expenseerror.jsp, expenselist.jsp, login.jsp, and myExpenseTrackerMain.html. The main editor window displays the following JSP code:

```
error.jsp
1 <%@ page language="Java" contentType="text/html; charset=UTF-8"
2 %>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Insert title here</title>
8 </head>
9 <body>
10 Login failed, please try again.
11 </body>
12 </html>
```

The status bar at the bottom indicates "20 lines, 546 characters selected". The right side of the screen shows the Dock with various application icons.

This  
is the error page displayed when the username or password is wrong.

## expenseerror.jsp



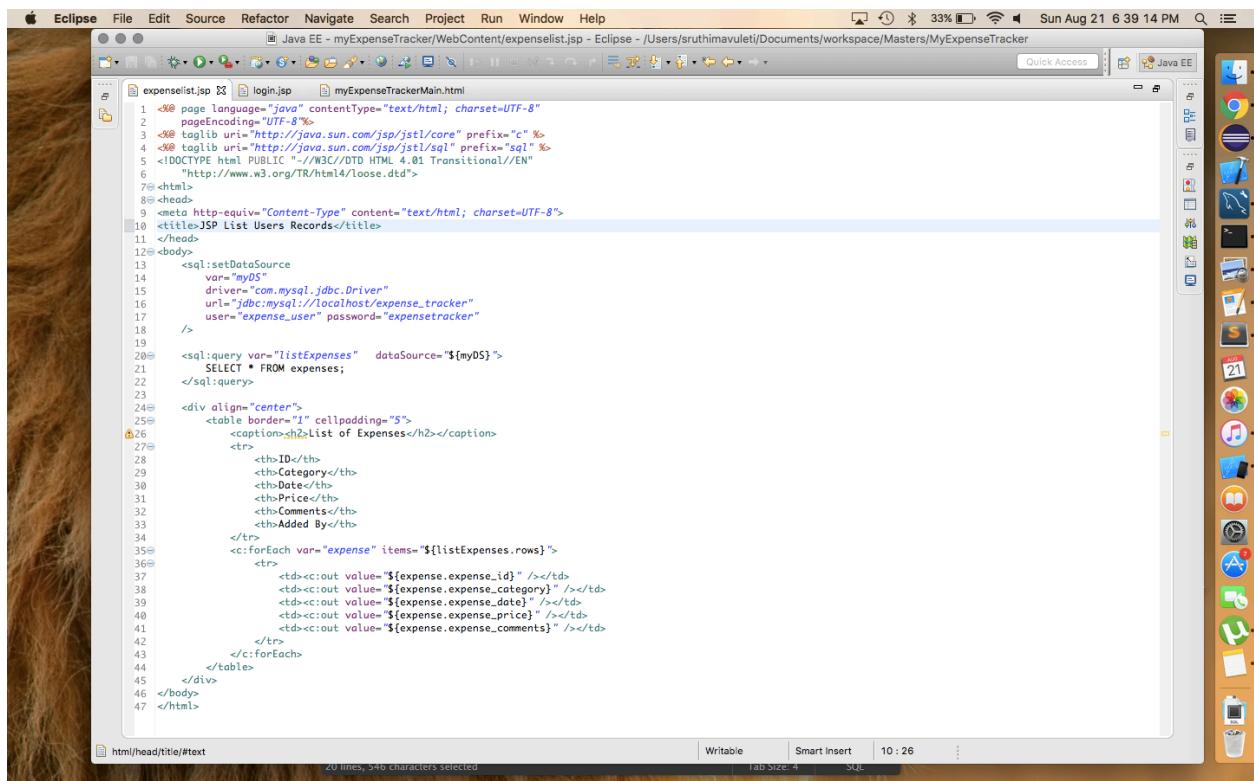
The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar indicates the project is "Java EE - myExpenseTracker/WebContent/expenseerror.jsp - Eclipse". The status bar at the bottom shows the date and time as "Sun Aug 21 6:39:09 PM". The central editor window displays the JSP code for "expenseerror.jsp". The code includes standard HTML headers and a single line of body content:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ pageEncoding="UTF-8"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Expense Error</title>
</head>
<body>
Failed to save Expense to Database. Please Try Again.
</body>
</html>
```

The status bar at the bottom of the editor shows "20 lines, 546 characters selected". The Eclipse toolbar and various icons are visible along the top and right edges of the interface.

This page is displayed there is error in database connection while saving the expense data.

## expenselist.jsp

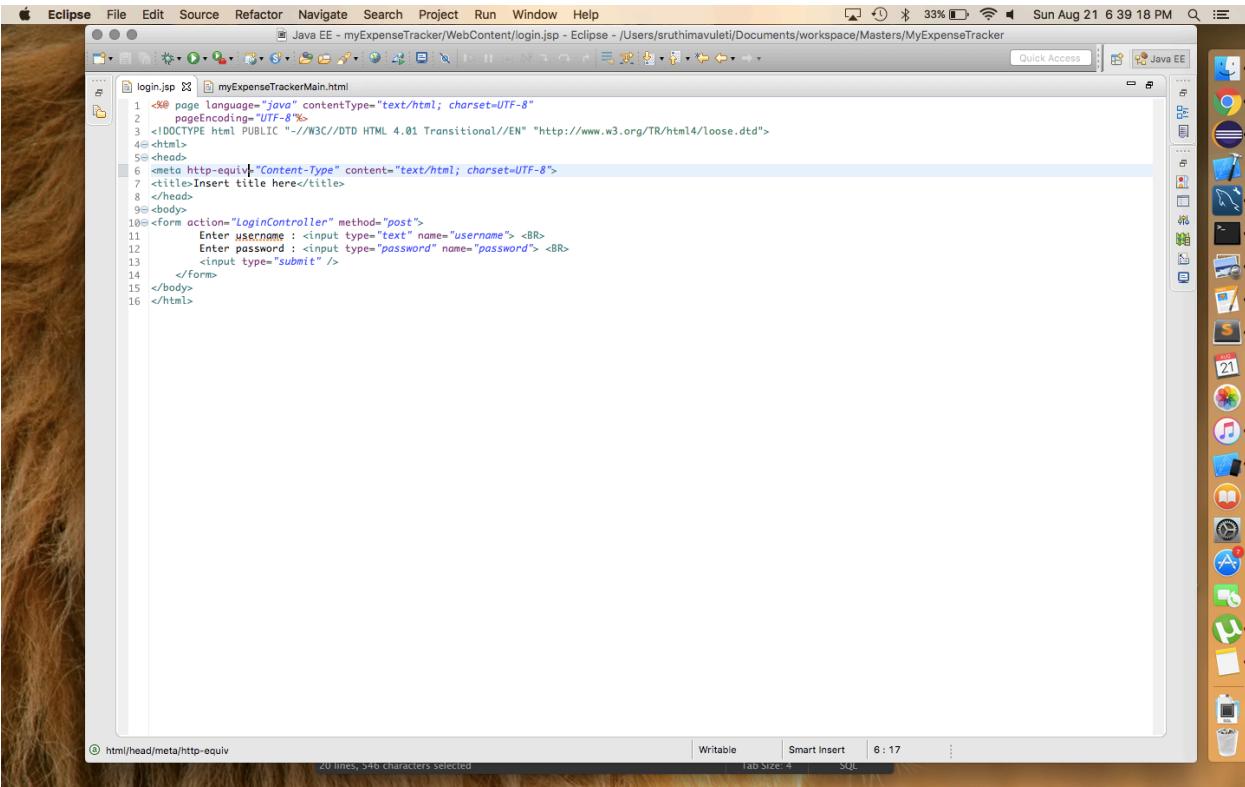


The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The central workspace displays the JSP file 'expenselist.jsp'. The code uses JSTL and SQL tags to connect to a MySQL database named 'expense\_tracker' and list expense records. The code includes imports for page, sql:(dataSource), and sql:query. It defines a dataSource variable and a query named 'listExpenses' that selects all columns from the 'expenses' table. The JSP then iterates over the results using a c:forEach loop to generate a table with columns for ID, Category, Date, Price, Comments, and Added By.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
6 "http://www.w3.org/TR/html4/loose.dtd">
7<html>
8<head>
9<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10<title>JSP List Users Records</title>
11</head>
12<body>
13    <sql:setDataSource
14        var="myDS"
15        driver="com.mysql.jdbc.Driver"
16        url="jdbc:mysql://localhost/expense_tracker"
17        user="expense_user" password="expensetracker"
18    />
19
20    <sql:query var="listExpenses" dataSource="${myDS}">
21        SELECT * FROM expenses;
22    </sql:query>
23
24    <div align="center">
25        <table border="1" cellpadding="5">
26            <caption><h2>List of Expenses</h2></caption>
27            <tr>
28                <th>ID</th>
29                <th>Category</th>
30                <th>Date</th>
31                <th>Price</th>
32                <th>Comments</th>
33                <th>Added By</th>
34            </tr>
35            <c:forEach var="expense" items="${listExpenses.rows}">
36                <tr>
37                    <td><c:out value="${expense.expense_id}" /></td>
38                    <td><c:out value="${expense.expense_category}" /></td>
39                    <td><c:out value="${expense.expense_date}" /></td>
40                    <td><c:out value="${expense.expense_price}" /></td>
41                    <td><c:out value="${expense.expense_comments}" /></td>
42                </tr>
43            </c:forEach>
44        </table>
45    </div>
46</body>
47</html>
```

This page is to display the list of the expenses added. Here sql tag is used to connect with the database.

## login.jsp

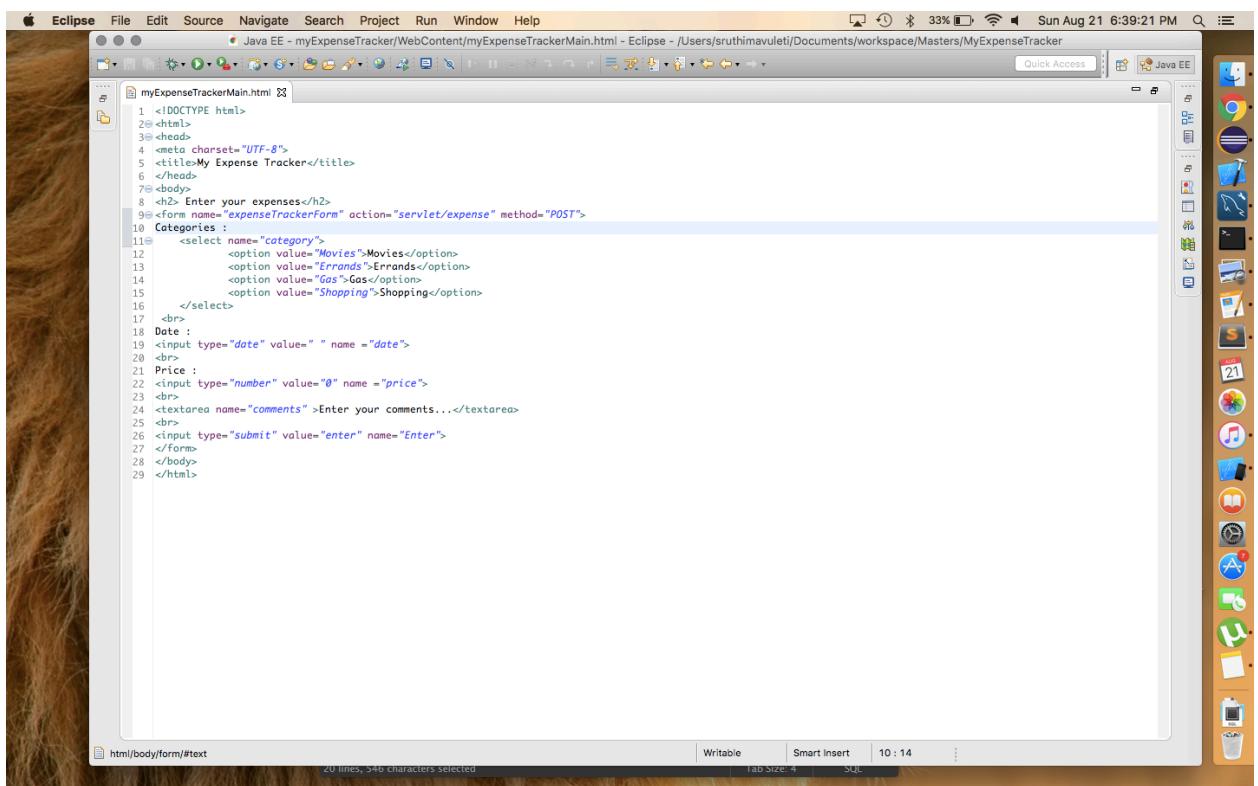


The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar reads "Java EE - myExpenseTracker/WebContent/login.jsp - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main editor window displays the JSP code for "login.jsp". The code includes standard HTML tags like head, body, and form, along with JSP syntax such as scriptlets and directives. The status bar at the bottom indicates "html/head/meta/http-equiv" and "20 lines, 546 characters selected". The Eclipse toolbar and various perspectives are visible along the top and right edges of the interface.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="LoginController" method="post">
    Enter username : <input type="text" name="username"> <br>
    Enter password : <input type="password" name="password"> <br>
    <input type="submit" />
</form>
</body>
</html>
```

This page is to display the login page where user can enter the user name and password. This page is connected to LoginController for the data.

### myExpenseTrackerMain.html



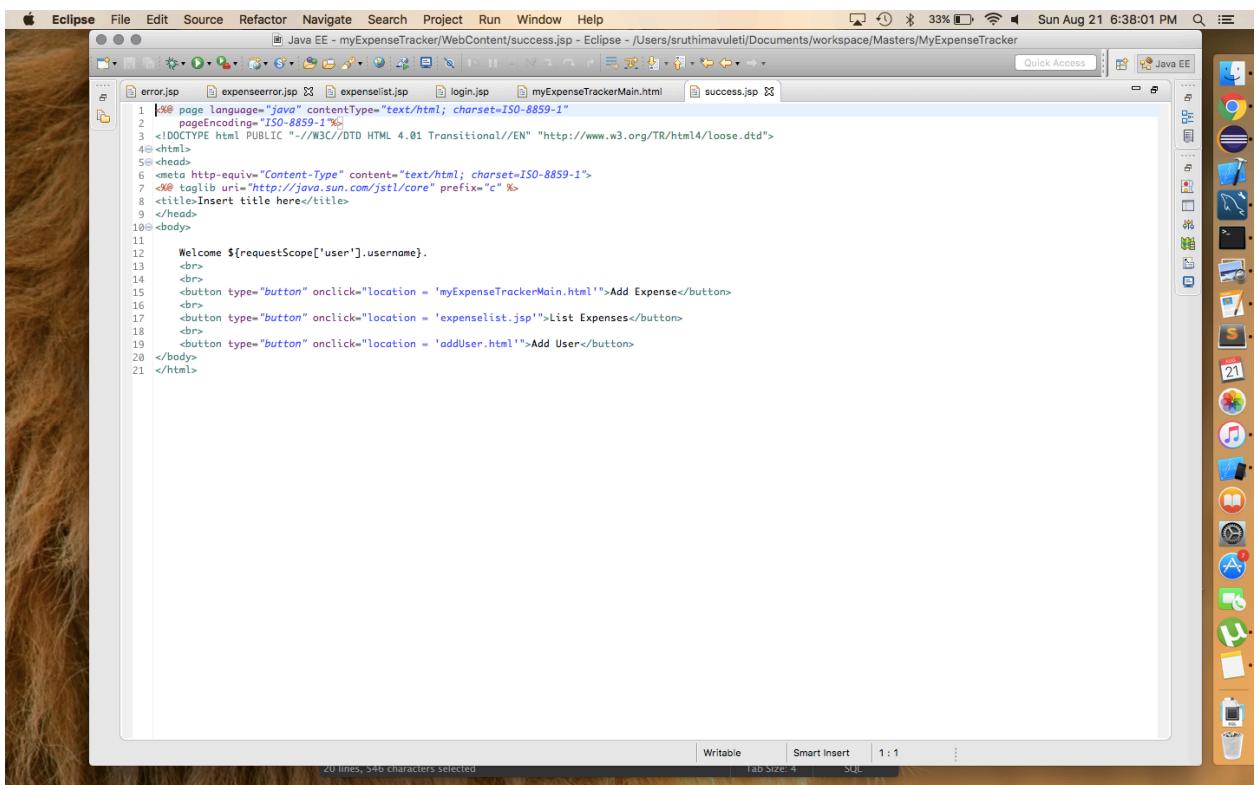
The screenshot shows the Eclipse IDE interface with the title bar "Java EE - myExpenseTracker/WebContent/myExpenseTrackerMain.html - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The main window displays the source code of the HTML file:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>My Expense Tracker</title>
</head>
<body>
<h2> Enter your expenses</h2>
<form name="expenseTrackerForm" action="servlet/expense" method="POST">
    Categories :
    <select name="category">
        <option value="Movies">Movies</option>
        <option value="Errands">Errands</option>
        <option value="Gas">Gas</option>
        <option value="Shopping">Shopping</option>
    </select>
    <br>
    Date :
    <input type="date" value="" name="date">
    <br>
    Price :
    <input type="number" value="0" name="price">
    <br>
    <textarea name="comments" >Enter your comments...</textarea>
    <br>
    <input type="submit" value="enter" name="Enter">
</form>
</body>
</html>
```

The status bar at the bottom indicates "20 lines, 546 characters selected".

This is the main page where we can add the expenses by giving: category, date, price, and comments.

### success.jsp



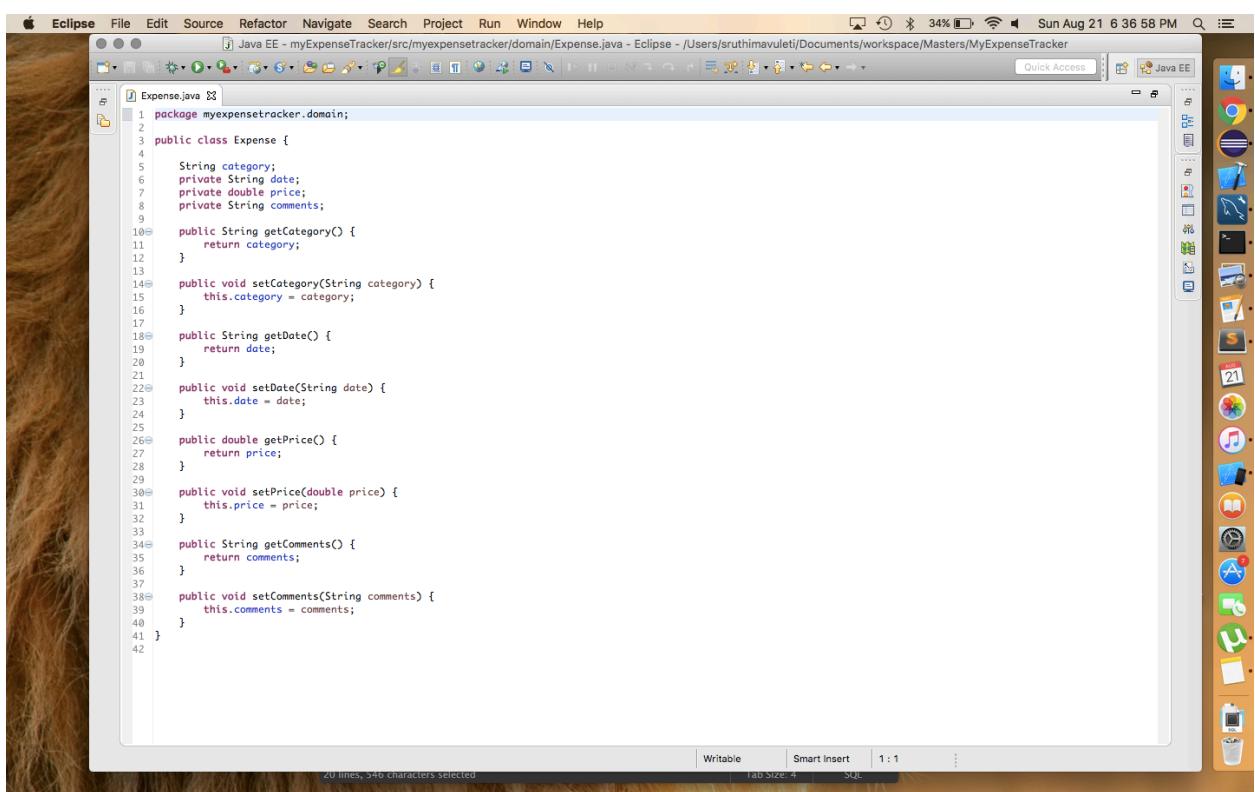
The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar indicates the project is "Java EE - myExpenseTracker" and the file being edited is "success.jsp". The code editor displays the following JSP code:

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4<html>
5<head>
6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7 <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
8 <title>Insert title here</title>
9 </head>
10<body>
11   Welcome ${requestScope['user'].username}.
12   <br>
13   <br>
14   <button type="button" onclick="location = 'myExpenseTrackerMain.html'">Add Expense</button>
15   <br>
16   <button type="button" onclick="location = 'expenselist.jsp'">List Expenses</button>
17   <br>
18   <button type="button" onclick="location = 'addUser.html'">Add User</button>
19 </body>
20 </html>
```

The Eclipse interface includes a toolbar, a menu bar, and a right-hand docked window showing various application icons. The status bar at the bottom shows "20 lines, 546 characters selected" and "Tab Size: 4".

This page is the success page displays after the login is success. Here we'll display all the three options : Add expenses, List expenses, add user.

Helper :  
Myexpensetracker.domain  
Expense.java



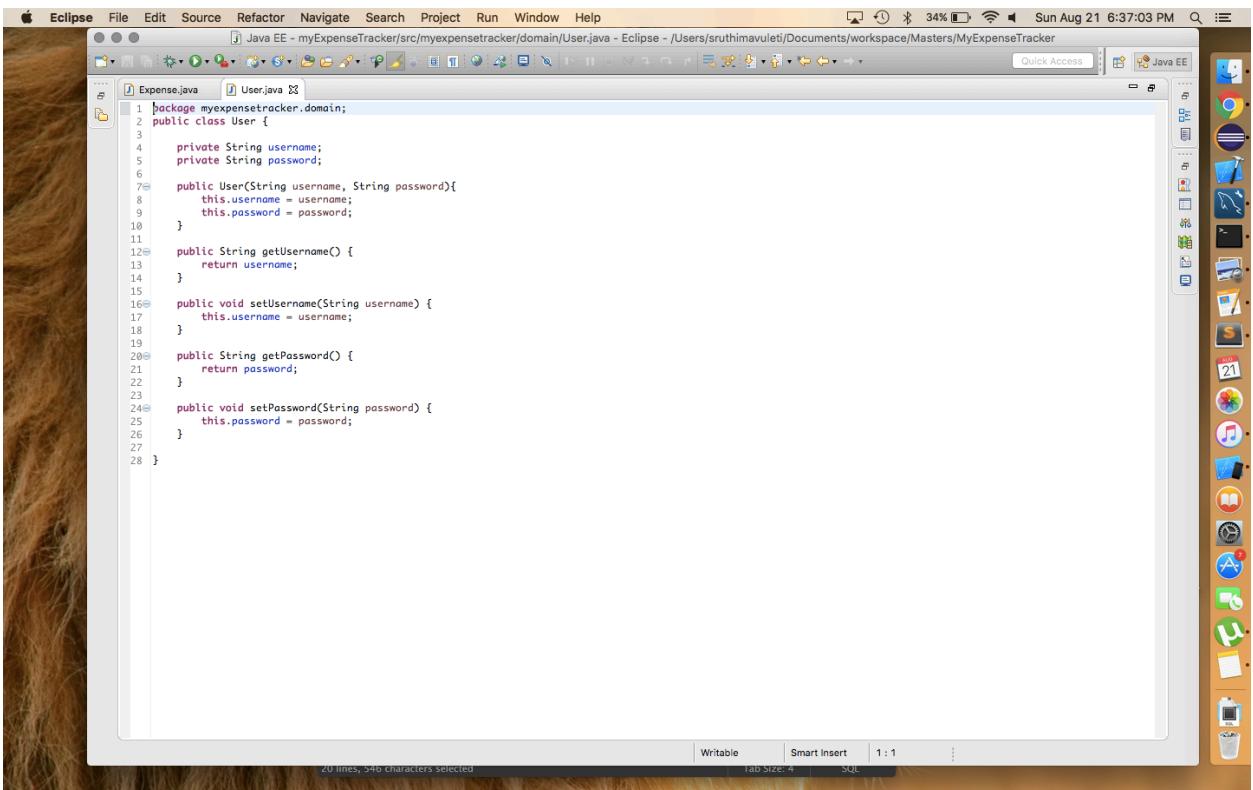
The screenshot shows the Eclipse IDE interface with the following details:

- Menu Bar:** Eclipse, File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Title Bar:** Java EE - myExpenseTracker/src/myexpensetracker/domain/Expense.java - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker. The date and time are Sun Aug 21 6:36:58 PM.
- Toolbar:** Standard Eclipse toolbar icons.
- Left Sidebar:** Shows the package structure: myexpensetracker.domain.
- Code Editor:** The file "Expense.java" is open. The code defines a class "Expense" with private attributes category, date, price, and comments, and corresponding public getters and setters.

```
1 package myexpensetracker.domain;
2
3 public class Expense {
4     String category;
5     private String date;
6     private double price;
7     private String comments;
8
9     public String getCategory() {
10        return category;
11    }
12
13    public void setCategory(String category) {
14        this.category = category;
15    }
16
17    public String getDate() {
18        return date;
19    }
20
21    public void setDate(String date) {
22        this.date = date;
23    }
24
25    public double getPrice() {
26        return price;
27    }
28
29    public void setPrice(double price) {
30        this.price = price;
31    }
32
33    public String getComments() {
34        return comments;
35    }
36
37    public void setComments(String comments) {
38        this.comments = comments;
39    }
40
41 }
```

- Bottom Status Bar:** Writable, Smart Insert, Tab Size: 4, 1:1, 20 lines, 546 characters selected.

## User.java

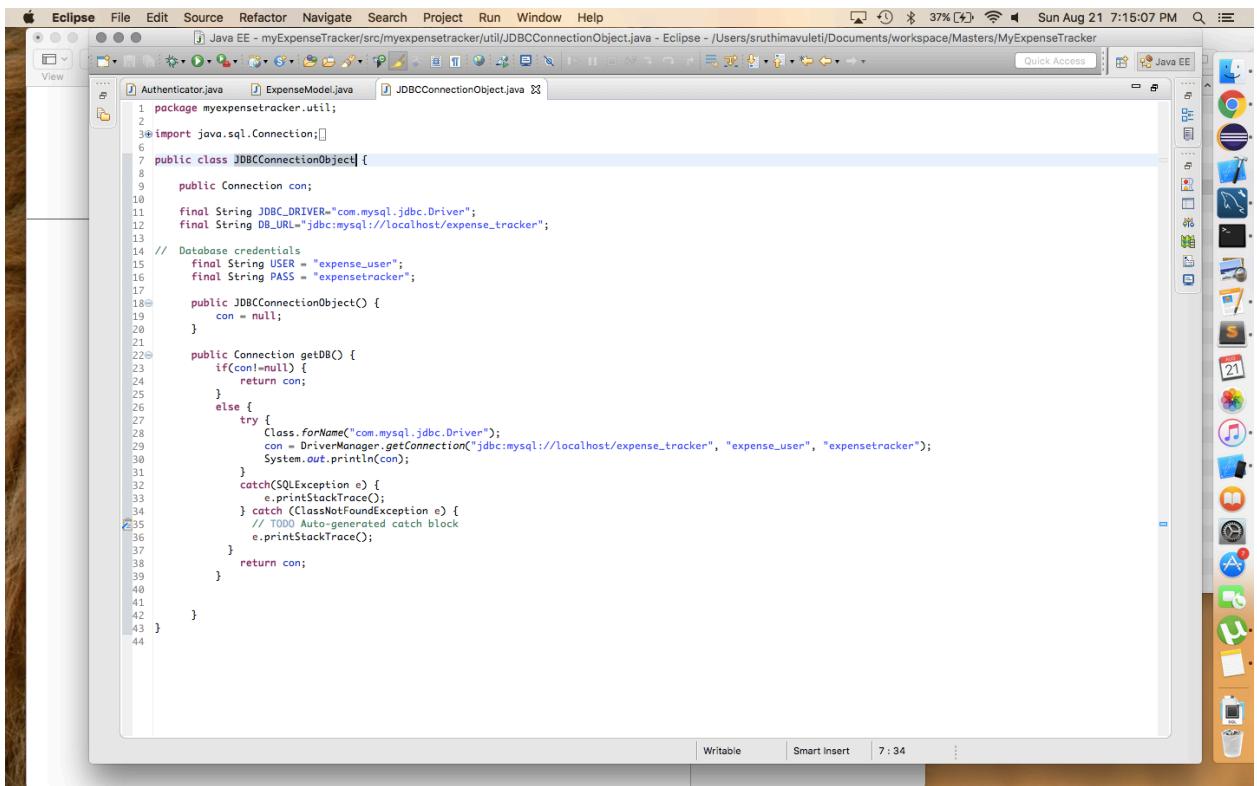


The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar indicates the project is "Java EE - myExpenseTracker/src/myexpensetracker/domain/User.java - Eclipse - /Users/sruthimavuleti/Documents/workspace/Masters/MyExpenseTracker". The status bar at the bottom shows "Sun Aug 21 6:37:03 PM" and battery level at 34%. The code editor displays the following Java code:

```
1 package myexpensetracker.domain;
2 public class User {
3     private String username;
4     private String password;
5
6     public User(String username, String password){
7         this.username = username;
8         this.password = password;
9     }
10
11     public String getUsername() {
12         return username;
13     }
14
15     public void setUsername(String username) {
16         this.username = username;
17     }
18
19     public String getPassword() {
20         return password;
21     }
22
23     public void setPassword(String password) {
24         this.password = password;
25     }
26
27 }
```

The code editor has tabs for "Expense.java" and "User.java". The status bar at the bottom right shows "Writable", "Smart Insert", "Tab Size: 4", and "SQL". The right side of the screen features the Dock with various application icons.

## Myexpensetraker.util JDBCConnectionObject.java

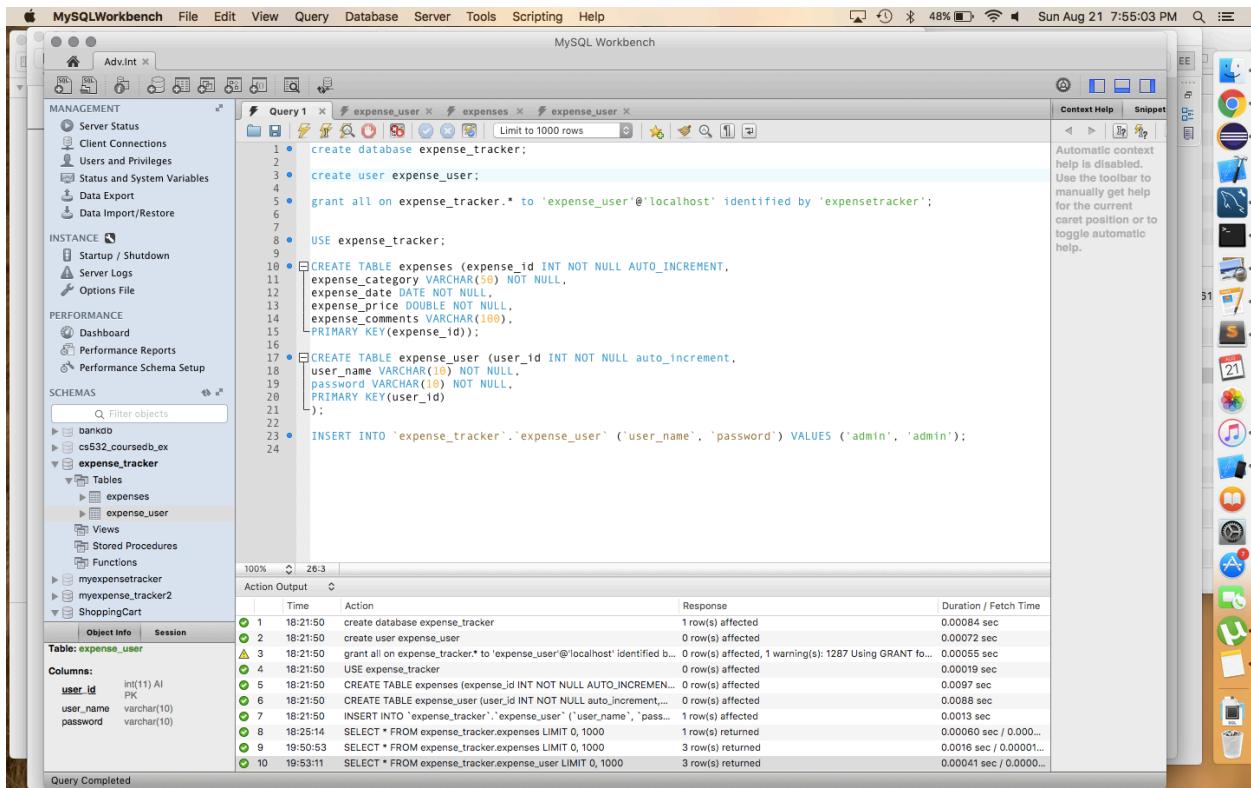


The screenshot shows the Eclipse IDE interface on a Mac OS X desktop. The title bar indicates the project is "Java EE - myExpenseTracker" and the file being edited is "JDBCConnectionObject.java". The code editor displays the following Java code:

```
1 package myexpensetraker.util;
2
3 import java.sql.Connection;
4
5 public class JDBCConnectionObject {
6
7     public Connection con;
8
9     final String JDBC_DRIVER="com.mysql.jdbc.Driver";
10    final String DB_URL="jdbc:mysql://localhost/expense_tracker";
11
12    // Database credentials
13    final String USER = "expense_user";
14    final String PASS = "expensetracker";
15
16    public JDBCConnectionObject() {
17        con = null;
18    }
19
20    public Connection getDB() {
21        if(con==null) {
22            return con;
23        }
24        else {
25            try {
26                Class.forName("com.mysql.jdbc.Driver");
27                con = DriverManager.getConnection("jdbc:mysql://localhost/expense_tracker", "expense_user", "expensetracker");
28                System.out.println(con);
29            }
30            catch(SQLException e) {
31                e.printStackTrace();
32            }
33            catch (ClassNotFoundException e) {
34                // TODO Auto-generated catch block
35                e.printStackTrace();
36            }
37        }
38        return con;
39    }
40
41    }
42}
43}
44}
```

# Database

This application mainly needs two tables: expenses table and expense\_user table.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database structure with the 'expense\_tracker' schema selected. The central pane shows a query editor with the following SQL code:

```
1 • create database expense_tracker;
2 • create user expense_user;
3 • grant all on expense_tracker.* to 'expense_user'@'localhost' identified by 'expensetracker';
4 •
5 • USE expense_tracker;
6 •
7 •
8 •
9 •
10 • CREATE TABLE expenses (expense_id INT NOT NULL AUTO_INCREMENT,
11 •     expense_category VARCHAR(50) NOT NULL,
12 •     expense_date DATE NOT NULL,
13 •     expense_price DOUBLE NOT NULL,
14 •     expense_comments VARCHAR(100),
15 •     PRIMARY KEY(expense_id));
16 •
17 • CREATE TABLE expense_user (user_id INT NOT NULL auto_increment,
18 •     user_name VARCHAR(10) NOT NULL,
19 •     password VARCHAR(10) NOT NULL,
20 •     PRIMARY KEY(user_id));
21 •
22 •
23 • INSERT INTO `expense_tracker`.`expense_user` ('user_name', 'password') VALUES ('admin', 'admin');
24 •
```

The bottom pane shows the execution log with the following entries:

Action	Time	Output	Response	Duration / Fetch Time
1	18:21:50	create database expense_tracker	1 row(s) affected	0.00084 sec
2	18:21:50	create user expense_user	0 row(s) affected	0.00072 sec
3	18:21:50	grant all on expense_tracker.* to 'expense_user'@'localhost' identified by 'expensetracker';	0 row(s) affected, 1 warning(s): 1287 Using GRANT fo...	0.00065 sec
4	18:21:50	USE expense_tracker;	0 row(s) affected	0.00019 sec
5	18:21:50	CREATE TABLE expenses (expense_id INT NOT NULL AUTO_INCREMENT,...	0 row(s) affected	0.0097 sec
6	18:21:50	CREATE TABLE expense_user (user_id INT NOT NULL auto_increment,...	0 row(s) affected	0.0088 sec
7	18:21:50	INSERT INTO `expense_tracker`.`expense_user` ('user_name', 'pass...	1 row(s) affected	0.0013 sec
8	18:25:14	SELECT * FROM expense_tracker.expenses LIMIT 0, 1000	1 row(s) returned	0.00060 sec / 0.000...
9	19:50:53	SELECT * FROM expense_tracker.expenses LIMIT 0, 1000	3 row(s) returned	0.0016 sec / 0.00001...
10	19:53:11	SELECT * FROM expense_tracker.expense_user LIMIT 0, 1000	3 row(s) returned	0.00041 sec / 0.000...

# Tables

expenses table consists of:

expense\_id: auto generated and unique id number for each expense.

expense\_category: Such as Errands, Movies, Gas, Shopping, etc.

expense\_date: on which date that category item was spent.

expense\_price: and what was the item price.

expense\_comments: can enter any comments.

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Management):** Shows the MySQL instance status, performance dashboard, and schema browser for the `expense_tracker` database, which contains the `expenses` table.
- Center Panel (Query Editor):** A query window titled "Query 1" displays the SQL command: `SELECT * FROM expense_tracker.expenses;`. Below the command is the result grid showing the following data:

expense_id	expense_category	expense_date	expense_price	expense_comments
1	Movies	0878-07-08	0	Enter your comments...
2	Gas	2016-03-09	50	GAS COMMENTS
3	Errands	2016-02-08	80	Shopped at Target

- Bottom Panel (Action Output):** A log of database actions with their times, descriptions, responses, and durations. The log includes:

Action	Time	Description	Response	Duration / Fetch Time
grant all on expense_tracker.* to 'expense_user'@'localhost' identified by 'password';	18:21:50	0 row(s) affected, 1 warning(s): 1287 Using GRANT fo...	0 rows(s) affected, 1 warning(s): 1287 Using GRANT fo...	0.00055 sec
USE expense_tracker;	18:21:50	0 row(s) affected	0 row(s) affected	0.00019 sec
CREATE TABLE expenses (expense_id INT NOT NULL AUTO_INCREMENT, expense_category VARCHAR(50), expense_date DATE, expense_price DOUBLE, expense_comments VARCHAR(100)) ENGINE=InnoDB;	18:21:50	0 row(s) affected	0 row(s) affected	0.0097 sec
INSERT INTO `expense_tracker`.`expenses`(`user_name`, `pass...`)	18:21:50	1 row(s) affected	1 row(s) affected	0.0088 sec
SELECT * FROM expense_tracker.expenses LIMIT 0, 1000;	18:25:14	1 row(s) returned	1 row(s) returned	0.0013 sec
SELECT * FROM expense_tracker.expenses LIMIT 0, 1000;	19:50:53	3 row(s) returned	3 row(s) returned	0.0016 sec / 0.0001...
SELECT * FROM expense_tracker.expense_user LIMIT 0, 1000;	19:53:11	3 row(s) returned	3 row(s) returned	0.00041 sec / 0.000...
SELECT * FROM myexpense_tracker2.expense_user LIMIT 0, 1000;	19:55:11	1 row(s) returned	1 row(s) returned	0.00070 sec / 0.000...
SELECT * FROM expense_tracker.expenses LIMIT 0, 1000;	19:55:24	3 row(s) returned	3 row(s) returned	0.00034 sec / 0.000...

- Status Bar:** Shows the date and time as Sun Aug 21 7 55 26 PM.

`expense_user` table consists of:

- `user_id`: auto generated and unique id number for every user.
- `user_name`: name of the user, and
- `user_password`: password of the user.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is open, showing the `expense_tracker` schema selected. Under `Tables`, the `expenses` and `expense_user` tables are listed. The `expense_user` table is currently selected. The main pane displays the data for the `expense_user` table in a grid format:

	user_id	user_name	password
1	1	admin	admin
2	2	sruthi	sruthi
3	3	17081	17081

Below the grid, the 'Action Output' section shows the query history:

Action	Time	Action	Response	Duration / Fetch Time
3	18:21:50	grant all on expense_tracker.* to 'expense_user'@'localhost' identified by ''	0 row(s) affected, 1 warning(s): 1287 Using GRANT fo...	0.00055 sec
4	18:21:50	USE expense_tracker	0 row(s) affected	0.00019 sec
5	18:21:50	CREATE TABLE expenses (expense_id INT NOT NULL AUTO_INCREMENT,...	0 row(s) affected	0.0097 sec
6	18:21:50	CREATE TABLE expense_user (user_id INT NOT NULL auto_increment,...	0 row(s) affected	0.0088 sec
7	18:21:50	INSERT INTO `expense_tracker`..`expense_user` ('user_name', 'pass...	1 row(s) affected	0.0013 sec
8	18:26:14	SELECT * FROM expense_tracker.expenses LIMIT 0, 1000	1 row(s) returned	0.00060 sec / 0.000...
9	19:50:53	SELECT * FROM expense_tracker.expenses LIMIT 0, 1000	3 row(s) returned	0.0016 sec / 0.00001...
10	19:53:11	SELECT * FROM expense_tracker.expense_user LIMIT 0, 1000	3 row(s) returned	0.00041 sec / 0.0000...
11	19:55:11	SELECT * FROM myexpense_tracker2.expense_user LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...
12	19:55:24	SELECT * FROM expense_tracker.expenses LIMIT 0, 1000	3 row(s) returned	0.00034 sec / 0.000...

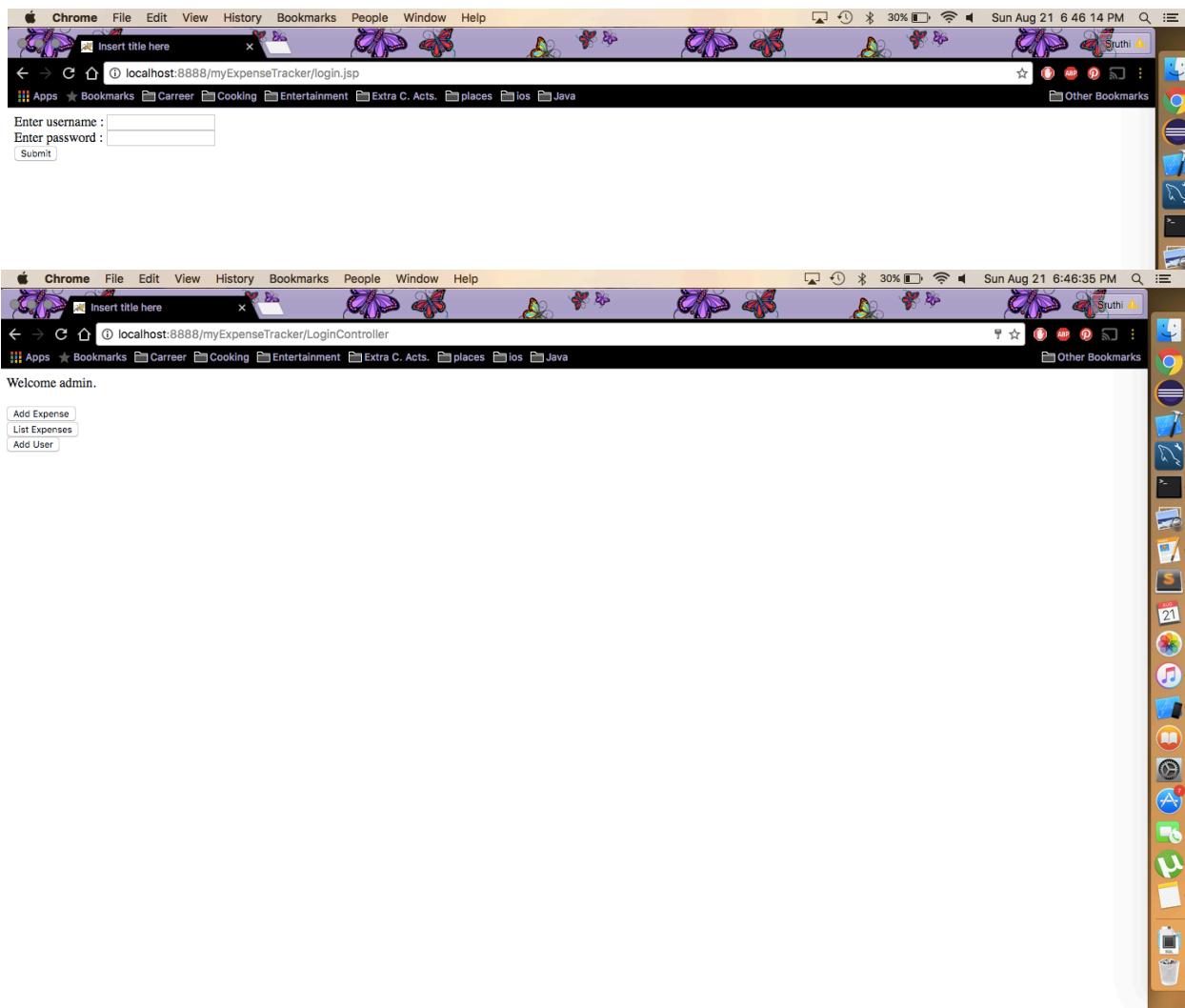
# Execution Url's

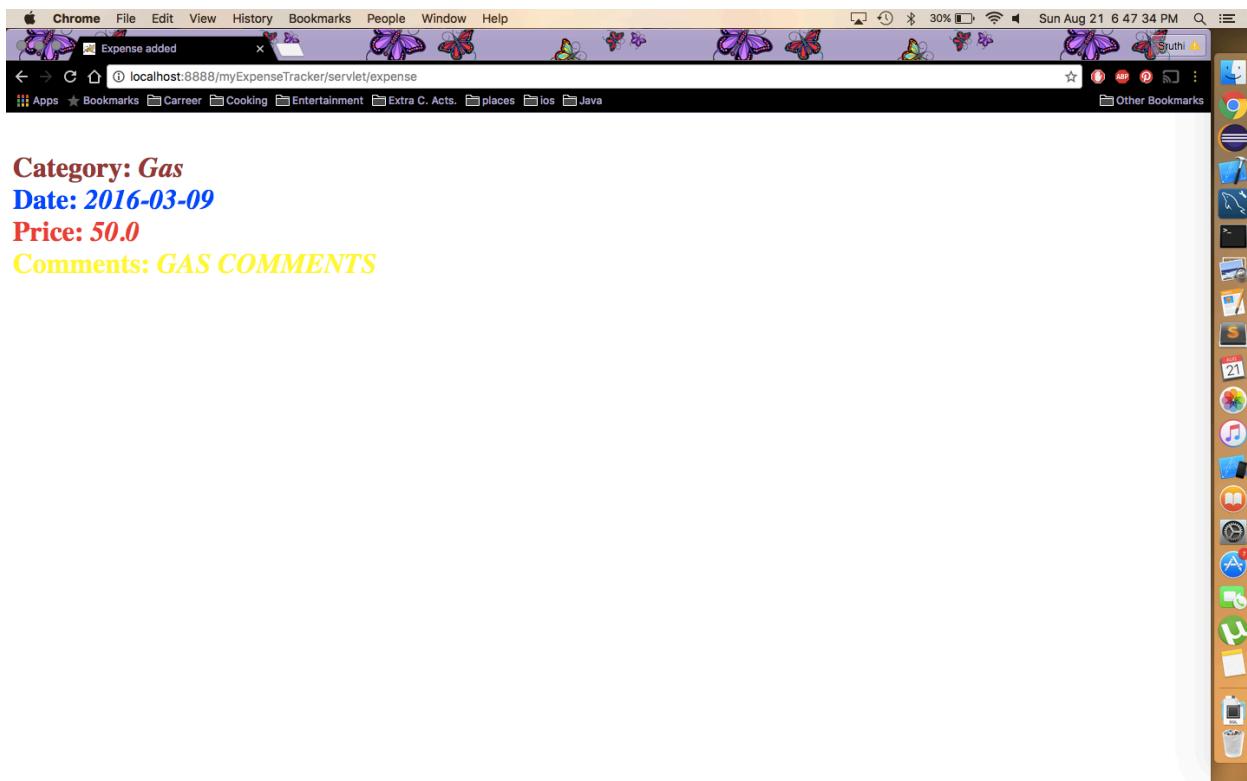
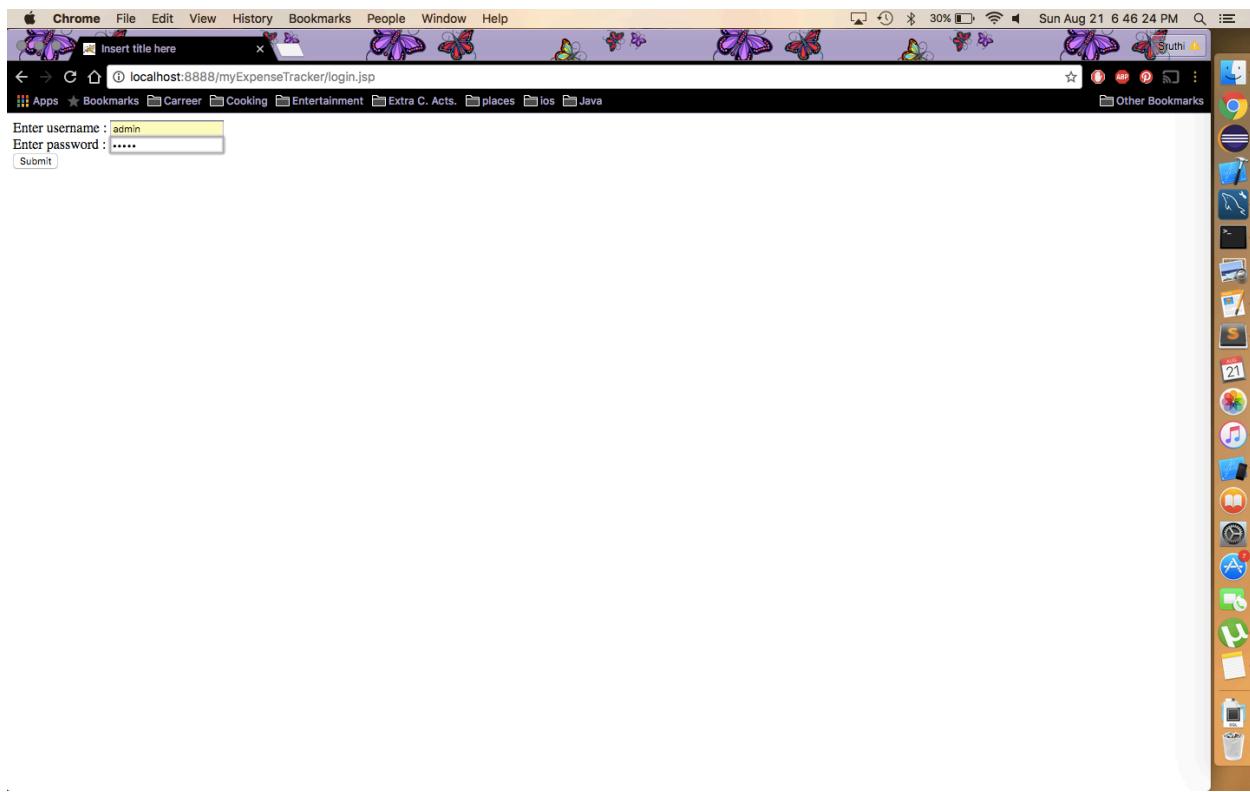
Login page: <http://localhost:8888/myExpenseTracker/login.jsp>

To add Expense: <http://localhost:8888/myExpenseTracker/myExpenseTrackerMain.html>

To View the List of expenses: <http://localhost:8888/myExpenseTracker/expenselist.jsp>

To add user: <http://localhost:8888/myExpenseTracker/addUser.html>





Chrome File Edit View History Bookmarks People Window Help

localhost:8888/myExpenseTracker/expenselist.jsp

Apps Bookmarks Carrer Cooking Entertainment Extra C. Acts. places iOS Java Other Bookmarks

Sruthi

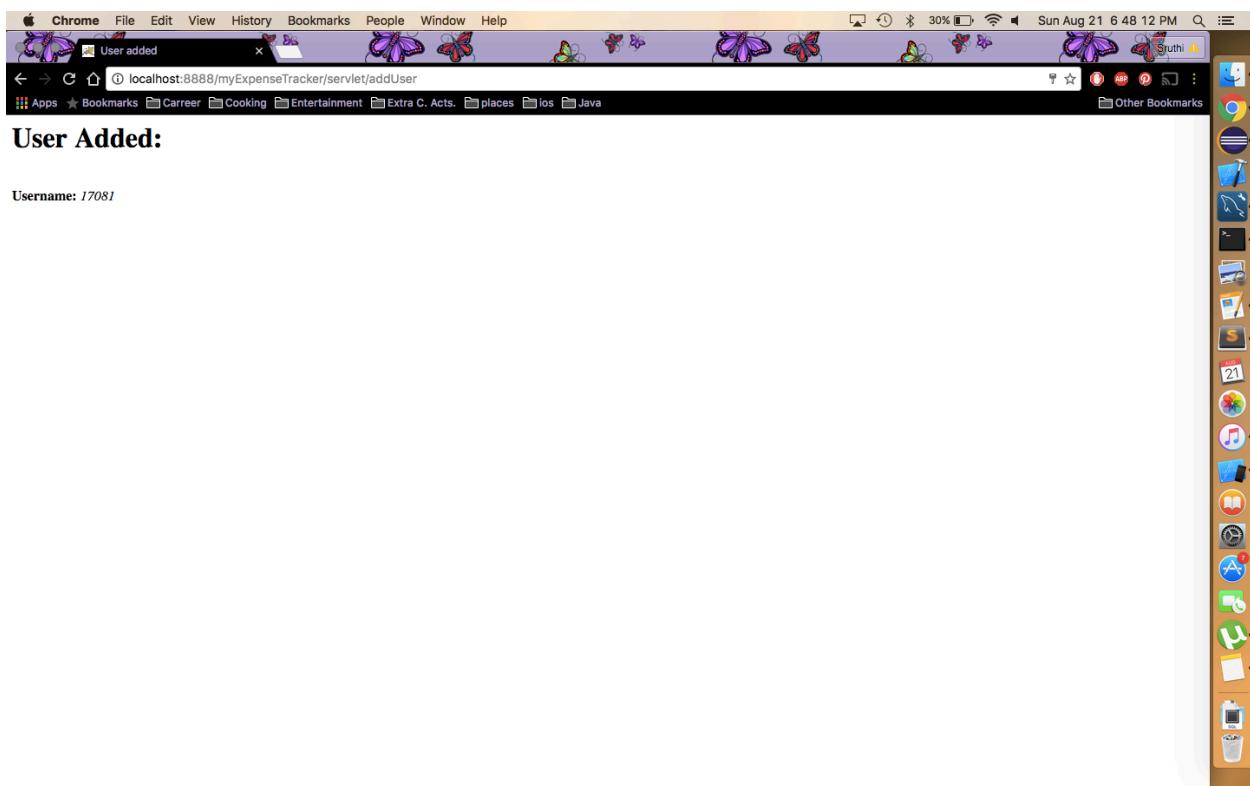
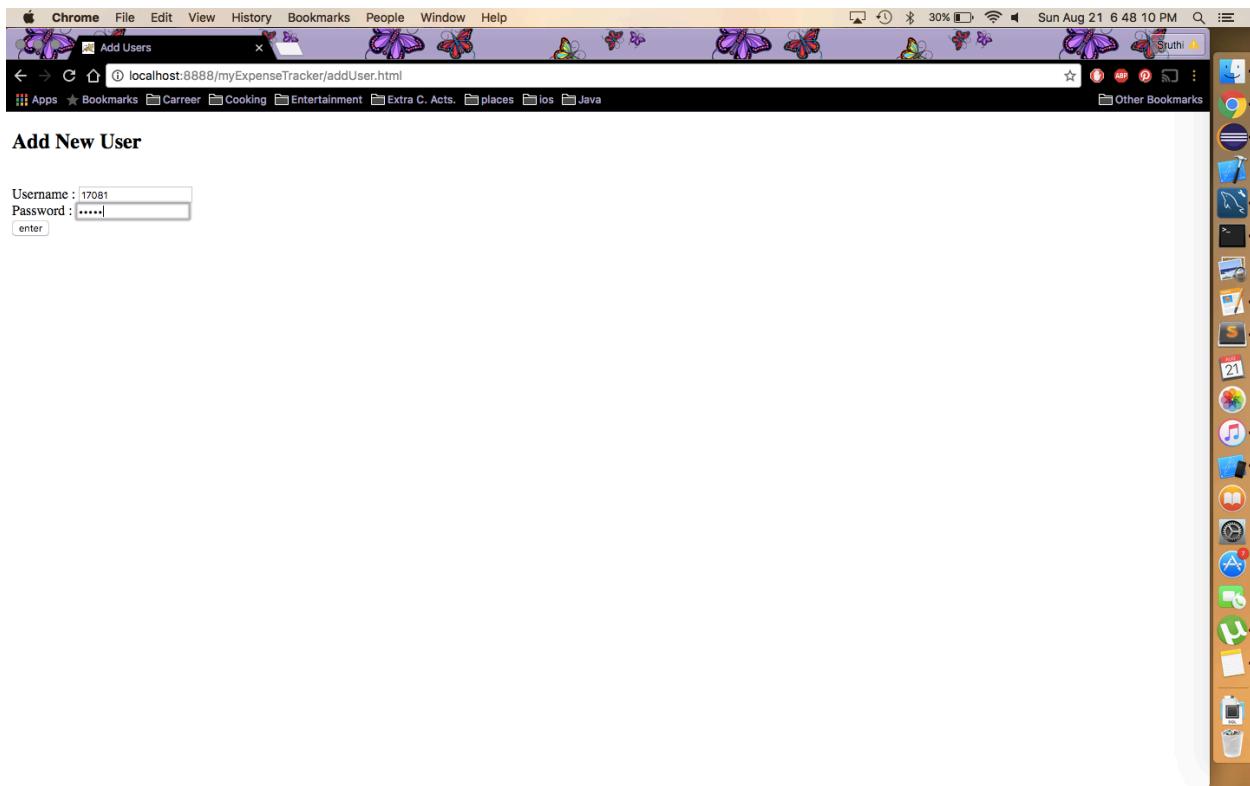
List of Expenses

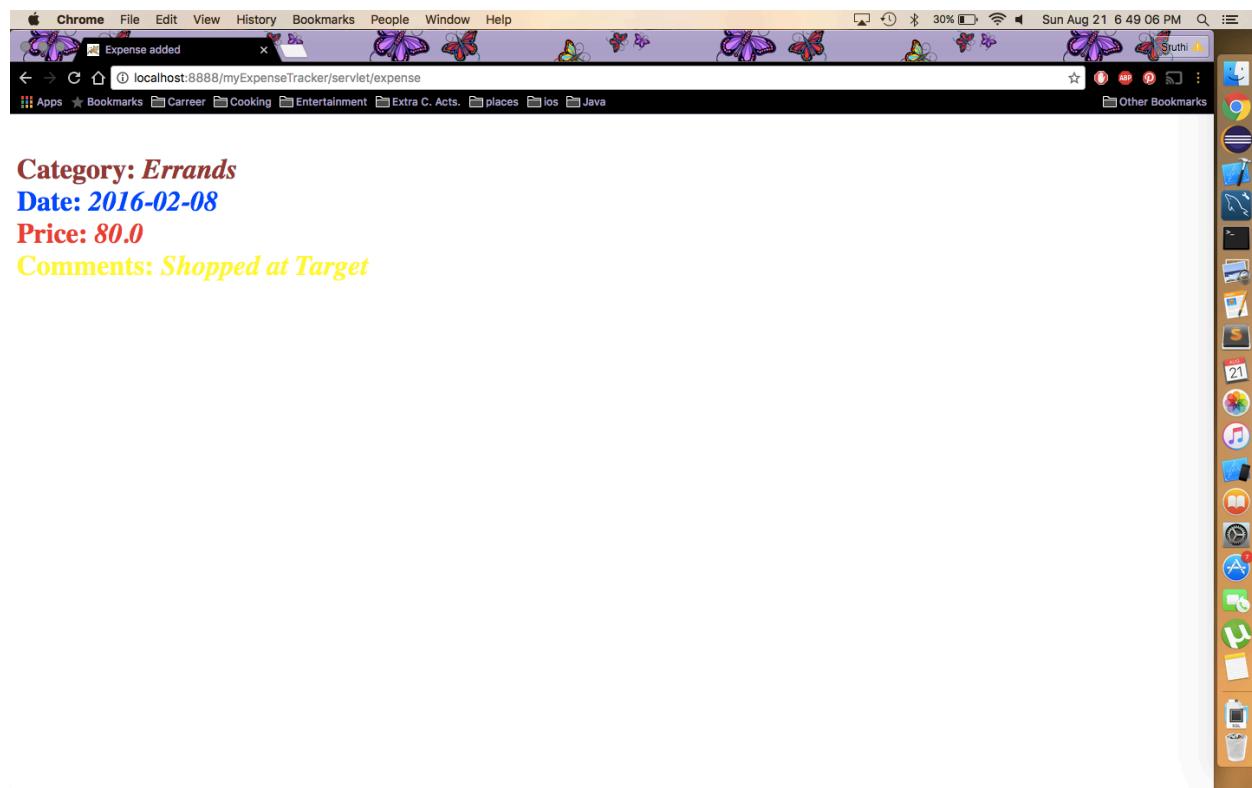
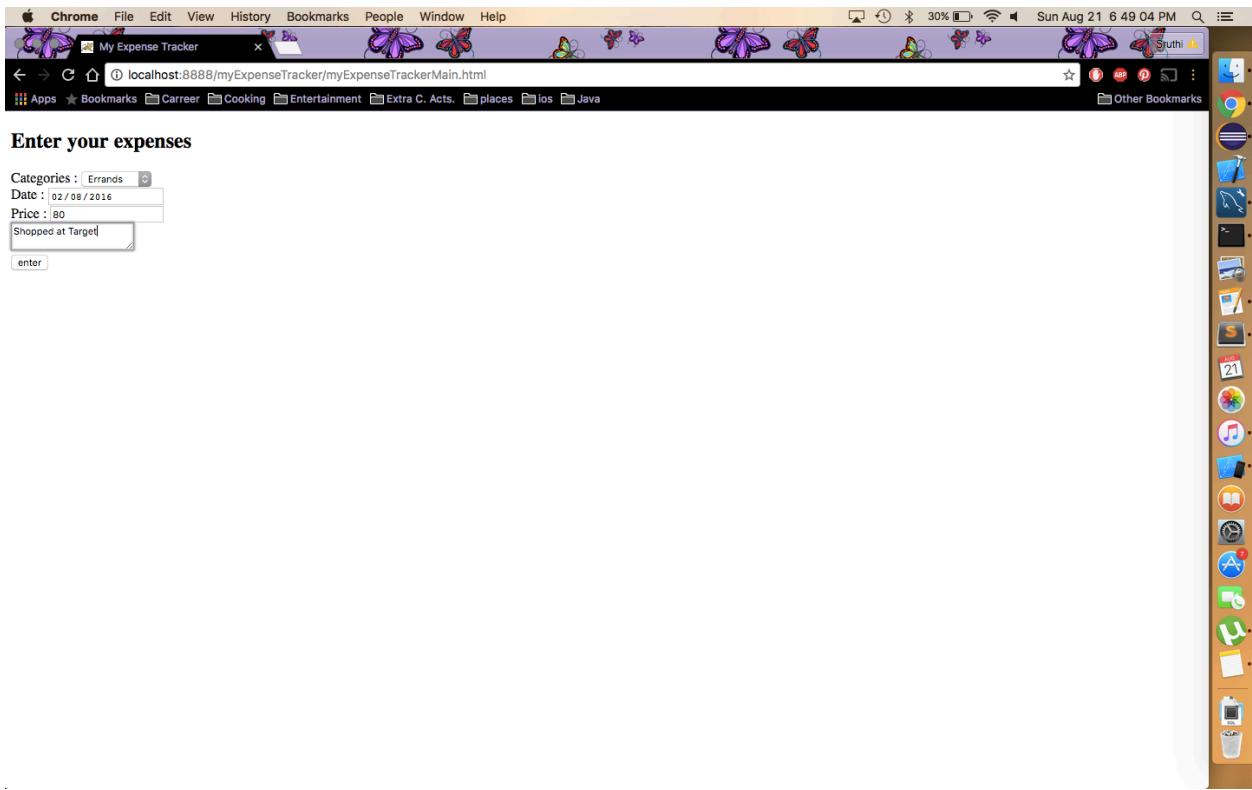
ID	Category	Date	Price	Comments
2	Gas	2016-03-09	\$0.0	GAS COMMENTS

The image shows a Mac OS X desktop environment. A Chrome browser window is open, displaying a JSP page titled "List of Expenses". The page contains a single record in a table:

ID	Category	Date	Price	Comments
2	Gas	2016-03-09	\$0.0	GAS COMMENTS

The desktop background is a wood-grain pattern. A Dock on the right side of the screen contains icons for various Mac applications, including Finder, Mail, Safari, and others. A vertical stack of open application windows is visible on the right edge of the screen.





Chrome File Edit View History Bookmarks People Window Help

localhost:8888/myExpenseTracker/expenselist.jsp

Bookmarks Career Cooking Entertainment Extra C. Acts places ios Java Other Bookmarks

List of Expenses

ID	Category	Date	Price	Comments
2	Gas	2016-03-09	50.0	GAS COMMENTS
6	Errands	2016-02-08	80.0	Shopped at Target



