Cooking Recipes: Final Report

Team 7: Mina Lee, Gel Dalit, Sushil Karki
Spring 2020: CS-157A Sec 80
August 3 , 2020

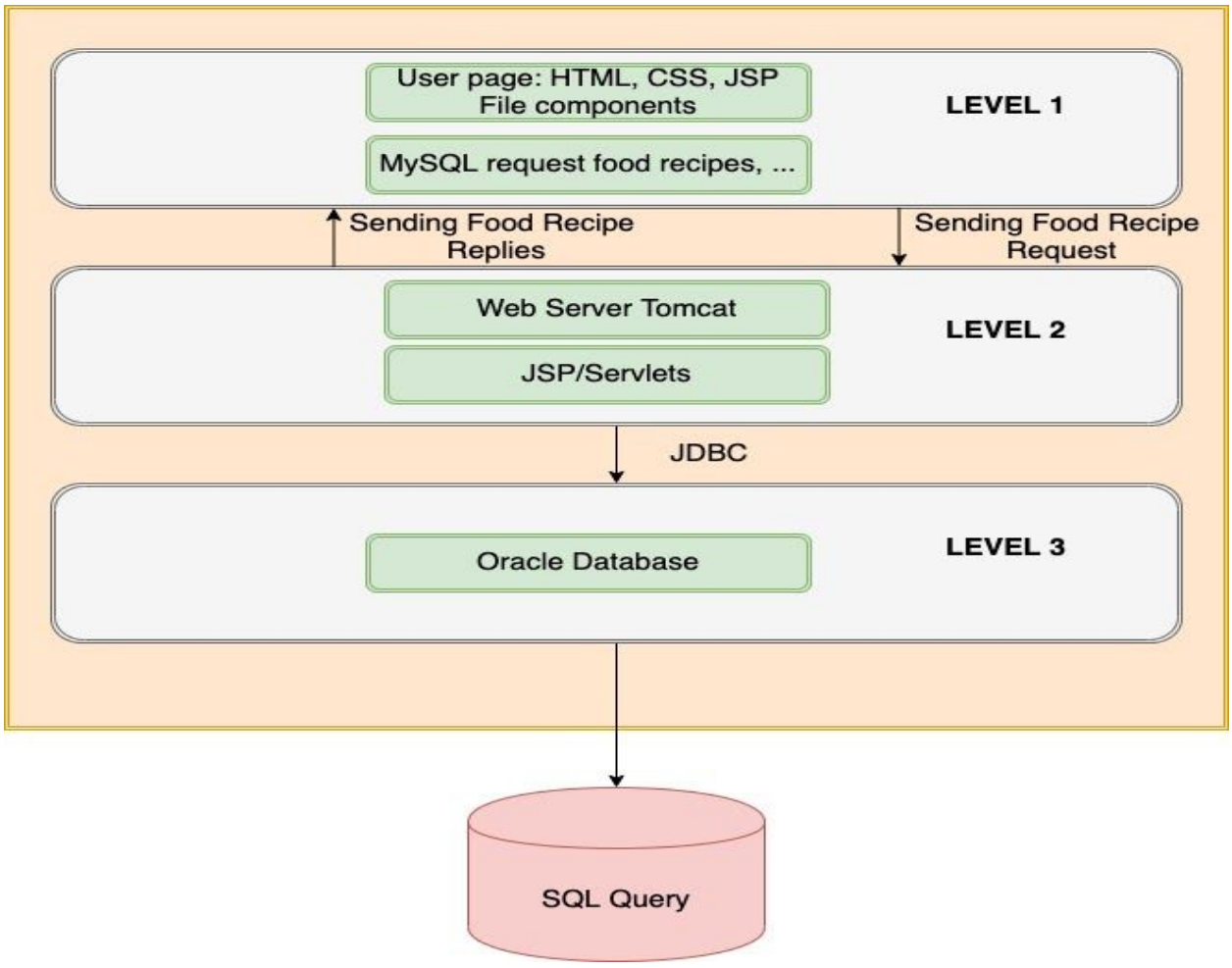
Continue Next Page...

I. Project Overview

This project will be a database that collects the users' recipes. The purpose of this project is to provide an efficient environment for people to share their cooking recipes if they want to, and save their recipes for their future needs.

II. System Environment

Structure of the system: *Figure 1: Figure below is an example of an illustration of the Three-Tier Architecture for our "Cooking Recipes" project.*



Hardware and software used:

- IntelliJ IDEA Ultimate Edition Platform (Coding)
- Git & Github (Version-Control)
- Apache Tomcat (Web-Server)
- MySql Workbench (Database)
- Mysql-connector-java-5.1.49-bin.jar
- Commons-fileupload-1.4.jar
    - http://commons.apache.org/proper/commons-fileupload/download_fileupload.cgi
- Commons-io-2.7.jar
    - https://commons.apache.org/proper/commons-io/download_io.cgi

RDMBS

- MySql Workbench

Application languages

Java(JDBC), MySql (SQL), CSS, HTML, XML, JavaScript

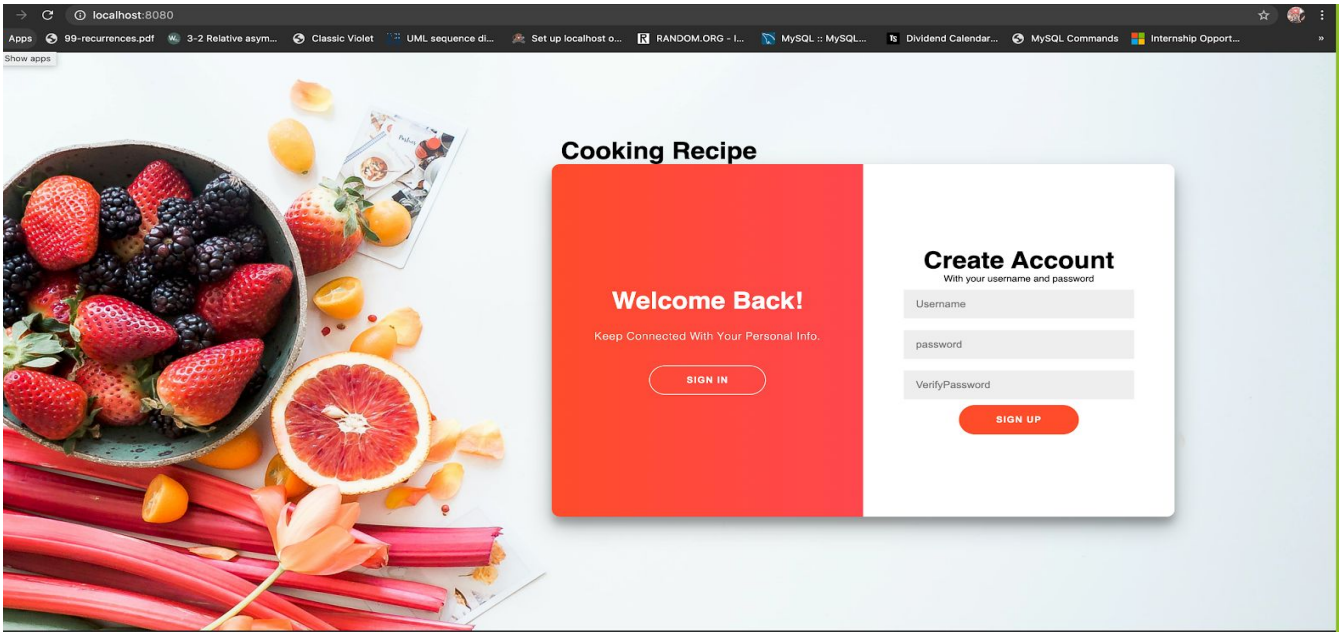
III. Revised Functional Requirements

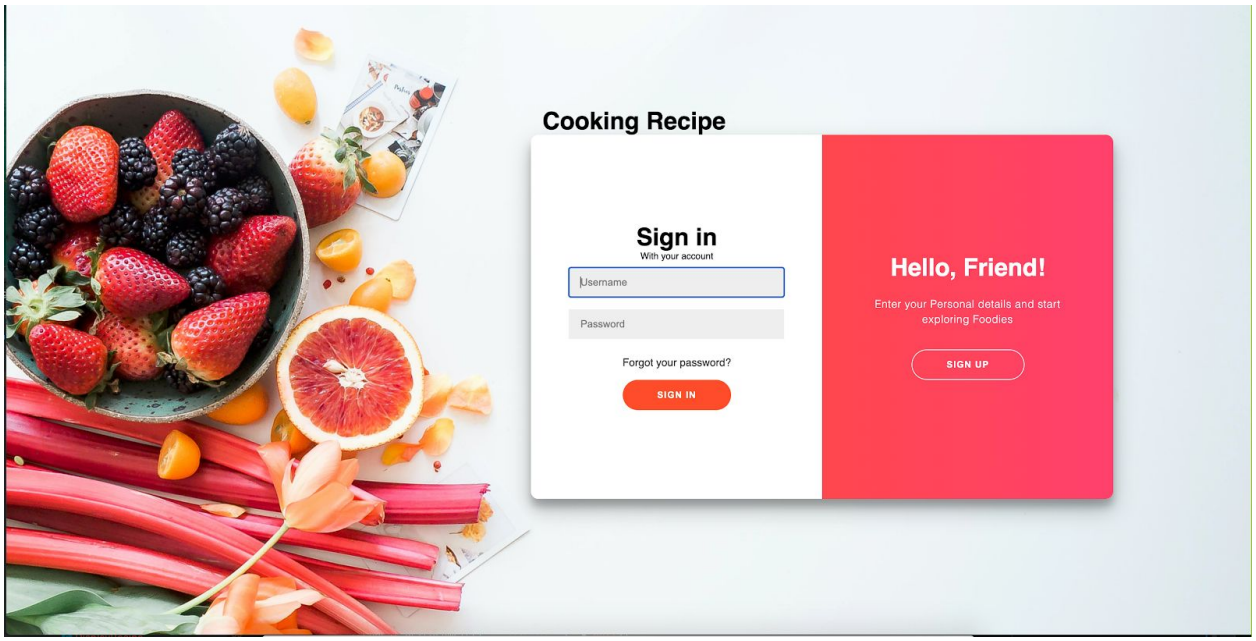
The overview of our project is an example of front-end and back-end architecture. HTML/CSS will be used to design front-end environments where we create a register and login to access inside the main page. Each user has to create an account with First name, Last name and password with the securely stored inside MySQL database. When the user tries to log in inside the system, the login credentials entered by the user will be matched with the login details in the database in order to authenticate the user. If the entered credentials are matched with database credentials, the user is granted access to the main page of the project.

Our backend-end architecture is handled by primarily Java and MySql combinations.

a. Detail Functions

- **SignUp & SingIn Page** (Frontend Implementation)





The above screenshot shows the single webpage which handles the SignUp and SignIn Simultaneously.The idea behind making a single page for handling both SignUp and SignIn is make UI experience better. The webpage is designed with HTML5, CSS and JavaScript to handle the user input. The toggle design of the webpage makes smooth transitions for SignUp and SignIn for UserInterface. When the SignUp is successfully done user data is stored in the database table it redirects to the SignIn div of the webpage Users are redirected to the main page when the login credential is authenticated is successful.

(Front end code)

```
    </div>
    <div class="form-container sign-in-container">
        <!-- Sign In form code goes here -->
        <form action="/login" method="post">
            <h1>Sign in</h1>
            <span>With your account</span>
            <input type="username" placeholder="Username" name="username" />
            <input type="password" placeholder="Password" name="password"/>
            <a href="#">Forgot your password?</a>
            <button type="submit" name="login">Sign In</button>
        </form>


    </div>
    <div class="overlay-container">
        <!-- The overlay code goes here -->
        <div class="overlay">
            <div class="overlay-panel overlay-left">
                <h1>Welcome Back!</h1>
                <p>Keep Connected With Your Personal Info.</p>
                <button class="ghost" id="signIn">Sign In</button>
            </div>
            <div class="overlay-panel overlay-right">
                <h1>Hello, Friend!</h1>
                <p>Enter your Personal details and start exploring Foodies</p>
                <button class="ghost" id="signUp">Sign Up</button>
            </div>
        </div>
    </div>
</div>
```

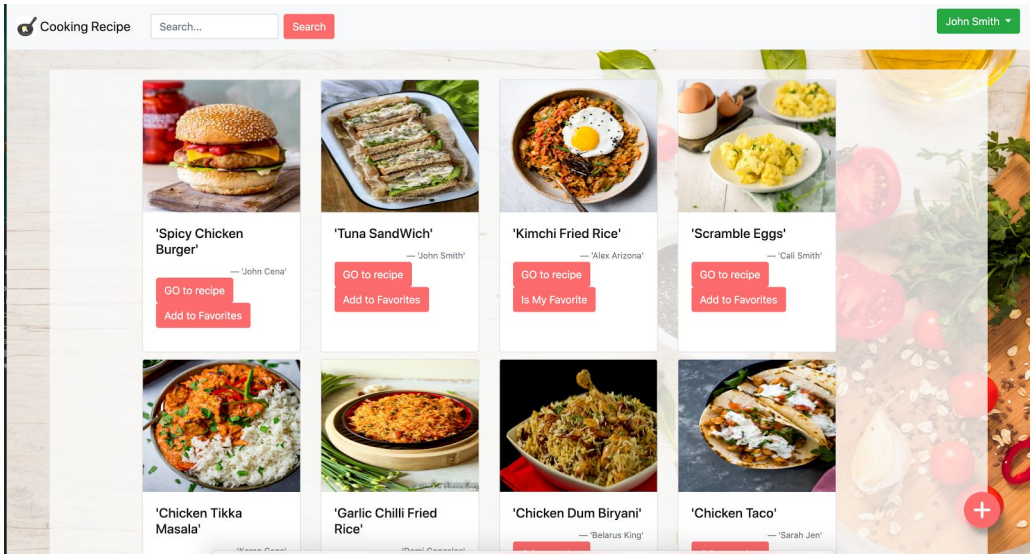
(Back end login code)

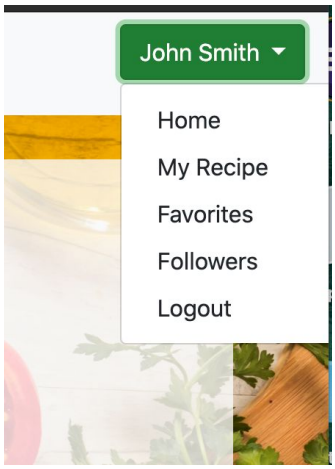
login.jsp Login.java

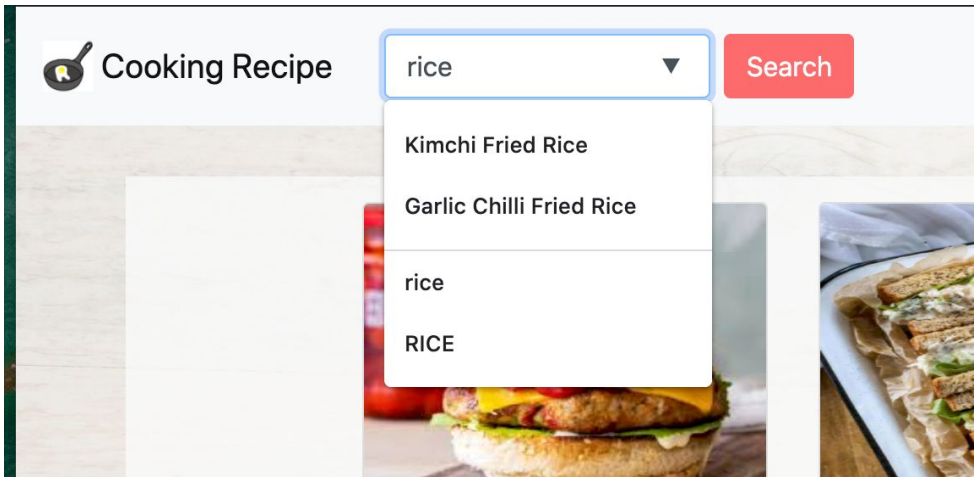
```java
package webapp;

import ...

@WebServlet(name="login")
public class Login extends HttpServlet {

    WebHandler webHandler = new WebHandler();

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        if (request.getParameter( s: "login") != null){
            String username = request.getParameter( s: "username");
            String password = request.getParameter( s: "password");
            try{
                if (webHandler.isDBReady() == true){
                    if (request.getParameter( s: "register") != null){
                        response.sendRedirect( s: "/register.jsp");
                    }
                    if (request.getParameter( s: "login") != null){
                        System.out.println("Try here");
                        if (webHandler.validateUsername(username) == true && webHandler.validatePassword(password) == true){
                            HttpSession sess = request.getSession();
                            int id = webHandler.getAccountID(username, password);
                            sess.setAttribute( s: "currentUserID", id);
                            sess.setAttribute( s: "currentUserName", username);

                            response.sendRedirect( s: "/mainpage.jsp");
                            System.out.println("should be here");
                        }
                        else{
                            // Wrong password notification
                            String errorMessage = "errorMessage";
                            String error = "Invalid Credentials";
                            request.setAttribute(errorMessage, error);
                            String errorMessage = "errorMessage";
                            String error = "Invalid Credentials";
                            request.setAttribute(errorMessage, error);
                            request.getRequestDispatcher( s: "/login.jsp").forward(request, response);
                        }
                    }
                }
                else{
                    System.out.println("Database failed to access");
                }
            }catch (Exception e){
                e.printStackTrace();
            }
        }

    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    }

}
```

```
    }
    public boolean validatePassword(String password)  {

        boolean isValidated = false;

        try{
            Connection connection = dbHandler.startConnection();
            String selectSql = "SELECT * FROM password WHERE account_id='"+accountID+"'";
            Statement statement = connection.createStatement();
            ResultSet rs_password = statement.executeQuery(selectSql);

            if (rs_password.isBeforeFirst() == false) {
                System.out.println("No password found in the database");
            }
            else{
                rs_password.next();
                password_DB = rs_password.getString( s: "password");
                if (password.compareTo(password_DB) == 0){
                    isValidated = true;
                }
                statement.close();
                rs_password.close();
            }
            dbHandler.closeConnection();
        }
        catch (Exception e){
            e.printStackTrace();
        }

        return isValidated;

    }
public boolean validateUsername(String username){
    boolean isValidated = false;
    try{
        Connection connection = dbHandler.startConnection();
        String selectSql = "SELECT * FROM account WHERE username='"+username+"'";
        Statement statement = connection.createStatement();
        ResultSet rs_account = statement.executeQuery(selectSql);
        if (rs_account.isBeforeFirst() == false) {
            System.out.println("No username found in the database");
            System.out.println("In here");
        }
        else{
            rs_account.next();

            username_DB = rs_account.getString( s: "username");
            accountID = rs_account.getInt( s: "account_id");
            if (username.compareTo(username_DB) == 0){
                isValidated = true;
            }
            statement.close();
            rs_account.close();
        }
        dbHandler.closeConnection();
    }
    catch (Exception e){
        e.printStackTrace();
    }
    return isValidated;
}
```

- **MainPage** (Code Implementation)



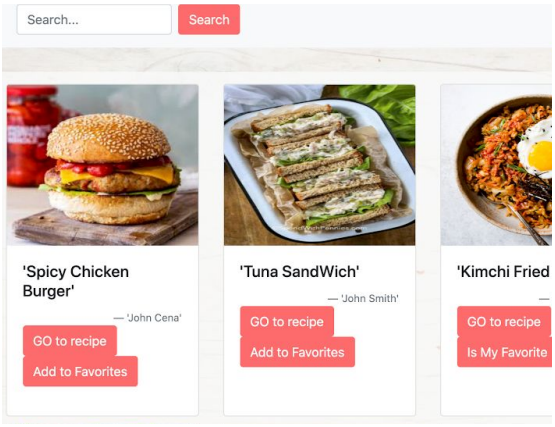


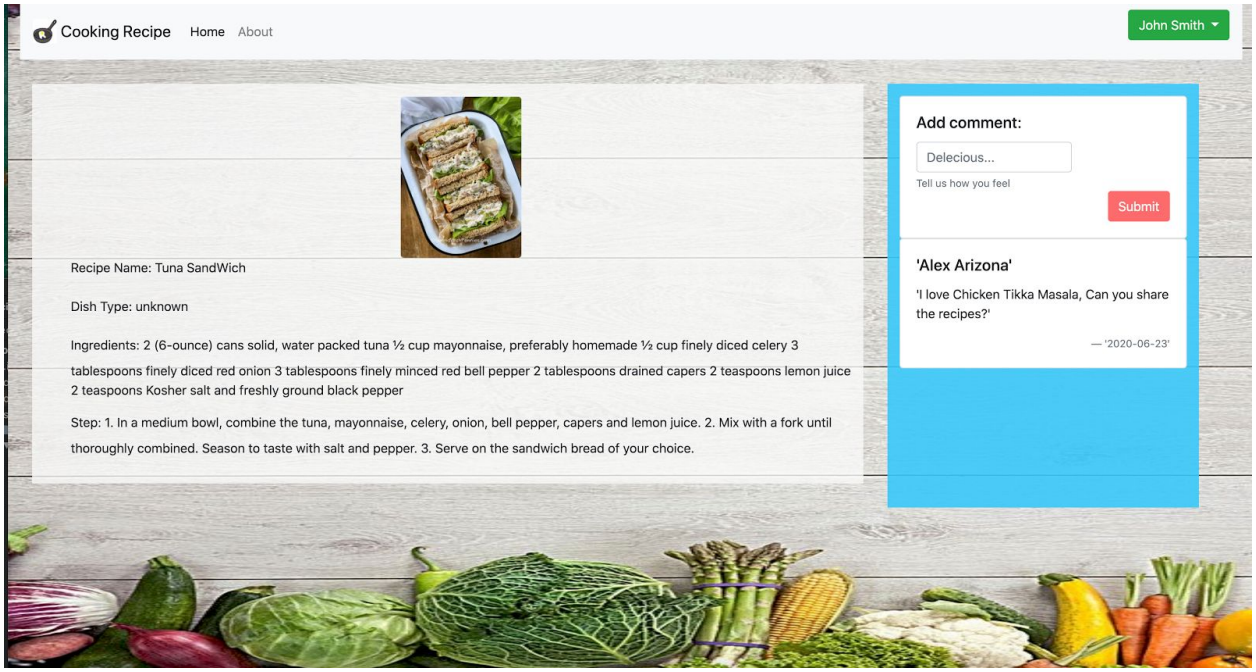


The above screenshot shows the main page of our website where users have many options to explore the website. Users can see many food recipe options and search for the recipes. Users can click the go to recipe button in each recipe which takes them to the recipe page where they can learn about recipe ingredients and steps . The add recipe is a major functionality in our web page where users can upload their own recipes. The design of our main page is a dashboard for each user where users can track their own activities and add, delete and update their existing models which are stored in the database.

```
while (rs.next()) {
    recipeID = rs.getInt("recipe_id");
    recipeName = rs.getString("recipe_name");
    imagePath = webHandler.getImage(recipeID);
    if ((index % 4) == 0 || index == 0){
        out.println( "<div class=\"row\">\n");
    }
    out.println( "<div class=\"col d-flex\" name ='"+recipeID+"'" + ">\n" +
            "        <div class=\"card\" style=\"width: 15rem;\">\n" +
            "             <img src='"+imagePath+"'" +" class=\"card-img-top\" alt=\"...\" width=\"150\" height=\"200\" >\n" +
            "                 <div class=\"card-body\">\n" +
            "                     <h5 class=\"card-title\">'"+recipeName+"'</h5>\n" + "");
    for (int i = 0; i < uploadedRecipes.size(); i++){
        if (uploadedRecipes.get(i) == recipeID){
            out.println("<footer class=\"blockquote-footer text-right\">'"+OwnersUsernames.get(i)+"'</footer>\n");
            break;
        }
    }
    gotoRecipe = "gotoRecipe"+recipeID;
    addToFavorites = "addToFavorites"+recipeID;
    unAddToFavorites = "unAddToFavorites"+recipeID;

    out.println("<form action=\"/mainpage\" method=\"post\">\n" +
            "        <button class=\"btn btn-primary\" type=\"submit\" name='"+gotoRecipe+"'" + ">GO to recipe\n" + "  ");
    if (myFavoriteList.contains(recipeID)){
        //out.println("<img onclick=\"\" class=\"fav-icon\"  name='"+unAddToFavorites+"'\" title=\"Add to favorite\" src=\"./Images/
        out.println("<button class=\"btn btn-primary\" type=\"submit\" name='"+unAddToFavorites+"'" + ">Is My Favorite\n");
    }else{
        //out.println("<img onclick=\"\" class=\"fav-icon\" title=\"Add to favorite\" name='"+addToFavorites+"'\" src=\"./Images/hea
        out.println("<button class=\"btn btn-primary\" type=\"submit\" name='"+addToFavorites+"'" + ">Add to Favorites\n");
    }
    out.println("</button>\n" +
```

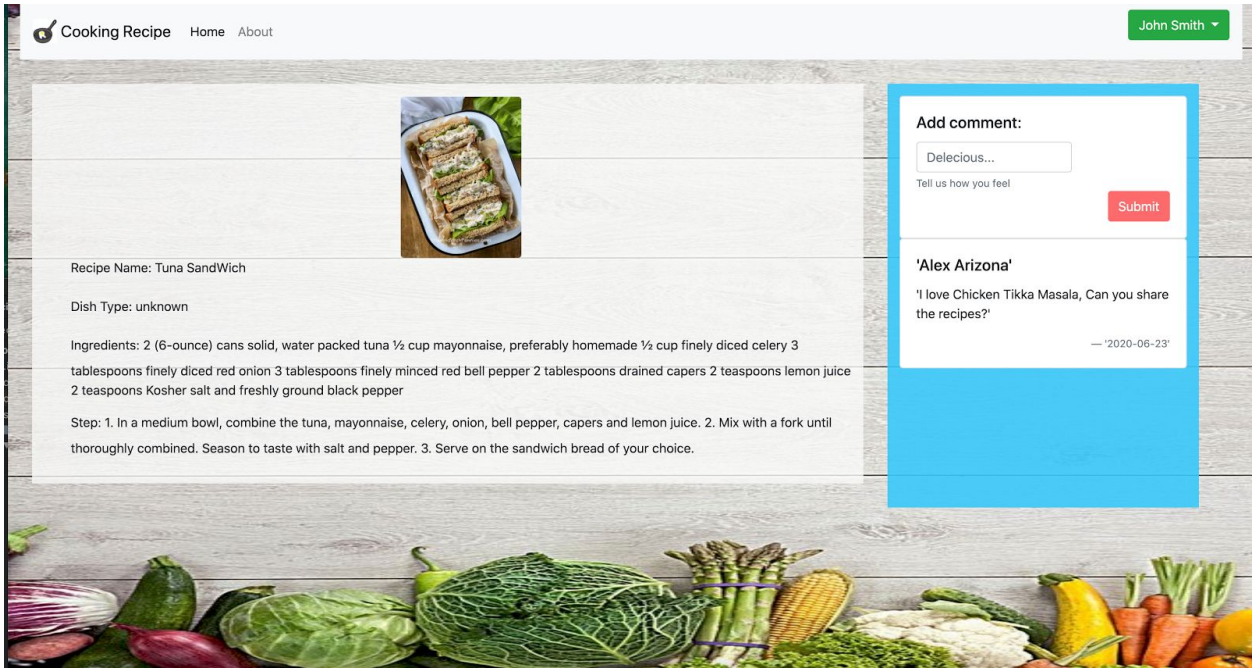
- **Go to recipe**





We have ***Go to recipe*** button in each recipe title on the main page which takes the user to the actual recipe page. The recipe page has the details information about the recipe like recipe title, picture, description. People can post comments about recipes and comments will be posted with the current date.

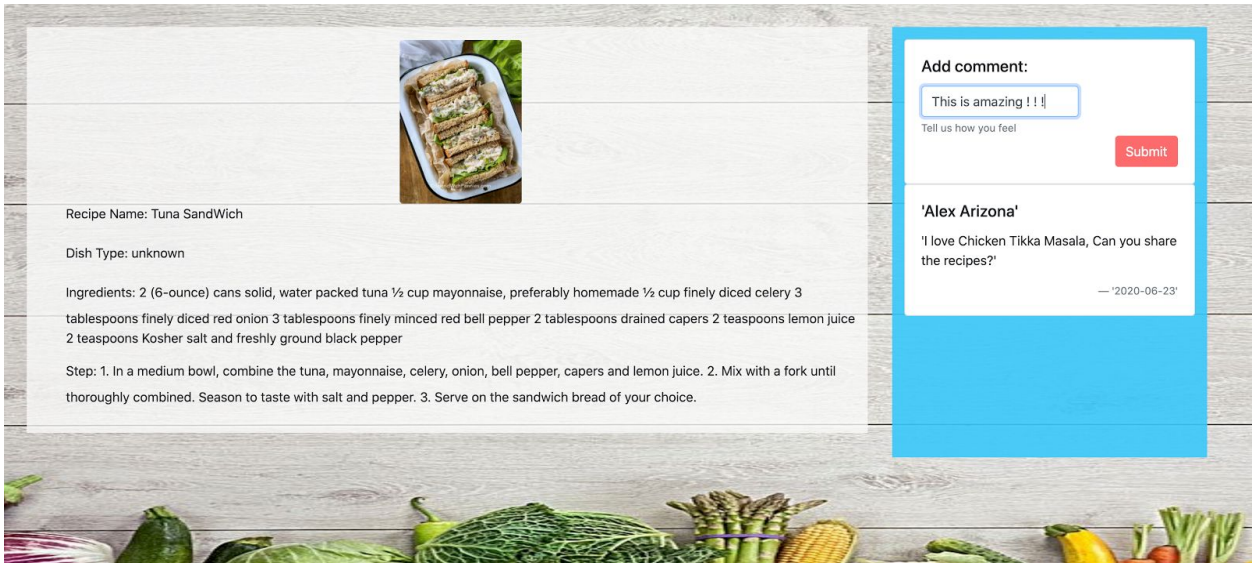
- **Display Recipe**

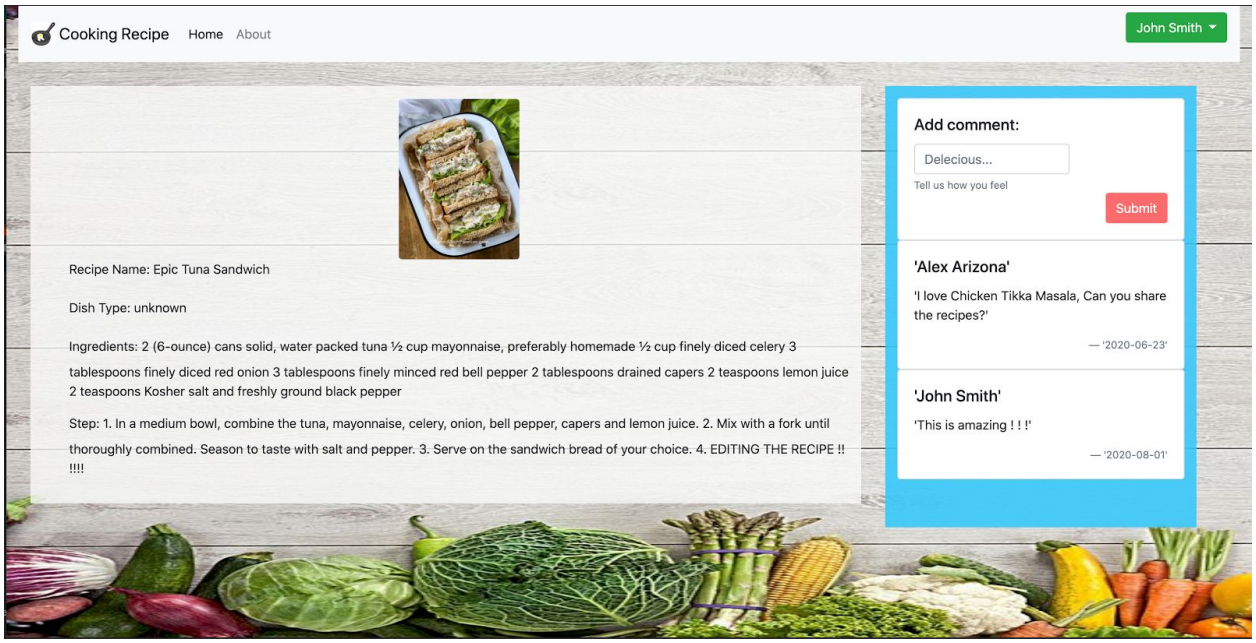


The above screenshot shows the ***Go to recipe*** page where users can go explore the different recipes which they want where they can learn detailed recipes of each food. Users have a comment option on the page where they can comment about how they like about food.

- **View/Submit Comment**

The comment box is on the right side of the recipe page, you will be able to see it after clicking the go to recipe button on the main page. The user will see other people's comments and he/she can also write out the new comment. The comment will display the username and date it posted.



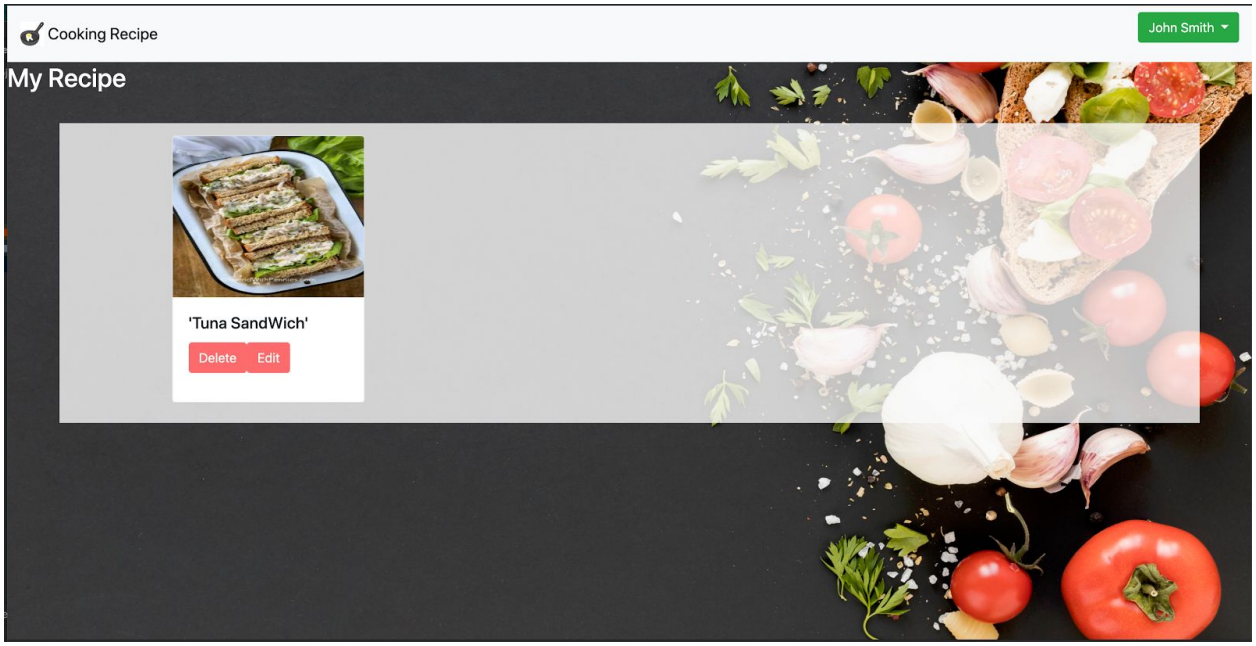



```
</div>
<div class="col-md-3 col-xl-3 comment-background">
    <div class="card w-100">
        <form action="displayRecipe.jsp" method="post" class="form-inline my-2 my-lg-0">
            <div class="card-body">
                <h5 class="card-title">Add comment:</h5>
                <input input type="text" name ="addingComment" class="form-control" placeholder="Delecious...">
                <small class="form-text text-muted">Tell us how you feel</small>
                <div class="text-right">
                    <button type="submit pull-right" class="btn btn-primary" name="addComment">Submit</button>
                </div>
            </div>
    </div>
```

```
/**************************************************************************
 * Printing Recipe's Comments with Username and the Date Posted
 **************************************************************************/
for (int index = 0 ; index < commentList.size(); index++ ){

    out.println("<div class=\"card w-100\">\n" +
            "       <div class=\"card-body\">\n" +
            "           <h5 class=\"card-title\">'"+usernameList.get(index)+"'</h5>\n" +
            "           <p class=\"card-text\">'"+commentList.get(index)+"'</p>\n" +
            "           <footer class=\"blockquote-footer text-right\">'"+datePostedList.get(index)+"'</footer>\n" +
            "       </div>\n" +
            "    </div>");
}
/**************************************************************************
```

```
/**************************************************************************
 * Adding Comment
 **************************************************************************/
if (request.getParameter("addComment") != null){
    WebHandler webHandler = new WebHandler();
    sess = request.getSession(false); //use false to use the existing session
    int currentUserID = (int) sess.getAttribute("currentUserID");//this will return username anytime in the session
    String currentUserName = (String) sess.getAttribute("currentUserName");
    String comment = request.getParameter("addingComment");
    Date date = webHandler.getDateToday();

    try{
        Connection connection = dbHandler.startConnection();
        dbHandler.insertCommentToDB(currentUserID, recipeID, currentUserName, comment, String.valueOf(date));
        connection.close();
        response.sendRedirect("/displayRecipe.jsp");
    }
    catch (Exception e){
```

- **My Recipe** (Frontend Implementation)



The above screenshot shows the web page of my recipe. It is the website where user added recipes are displayed. In the mainpage user can navigate to the + button where they can add their own recipe which is stored in my recipe webpage.

addRecipe.jsp

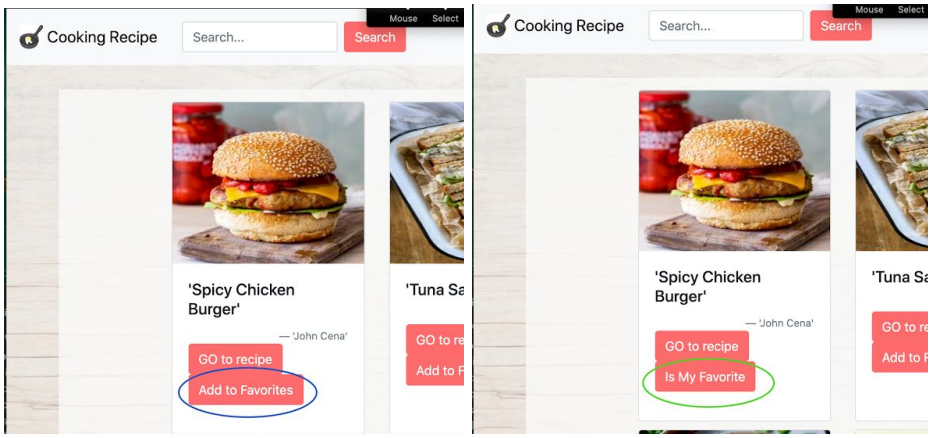
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <meta charset="UTF-8">
    <title>Add New Recipe</title>
    <link rel="icon shortcut" href="./Images/icon.png">
    <link rel="stylesheet" href="Style.css">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h8
</head>
<style>
    body {
        background-image: url("https://i.picsum.photos/id/307/5760/3840.jpg?hmac=XtsDQJ477ylS15H1VvpO9qCUlvrvBcDVQBu9ZU1B-SI");
        background-size: 100% 100%;
    }
</style>

<body>
<div class="container-fluid">
    <div class="row flex-xl-nowrap">
        <div class="col-md-3 col-xl-2">

        </div>

        <div class="col-md-9 col-xl-8 py-md-3 pl-md-5 bd-content main-background">

            <form action = "/addrecipe" method = "post" enctype = "multipart/form-data">

                <div>
                    <label><b><em>Recipe Name</em></b></label>
                    <input type="text" placeholder="Enter Recipe Name" name="addRecipeName" required>
                </div>
```

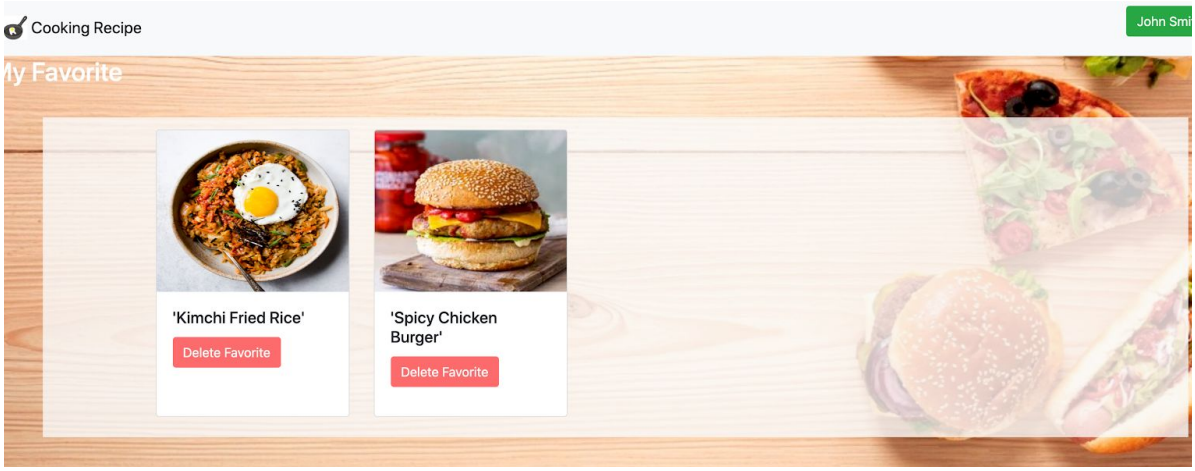
addRecipe.jsp

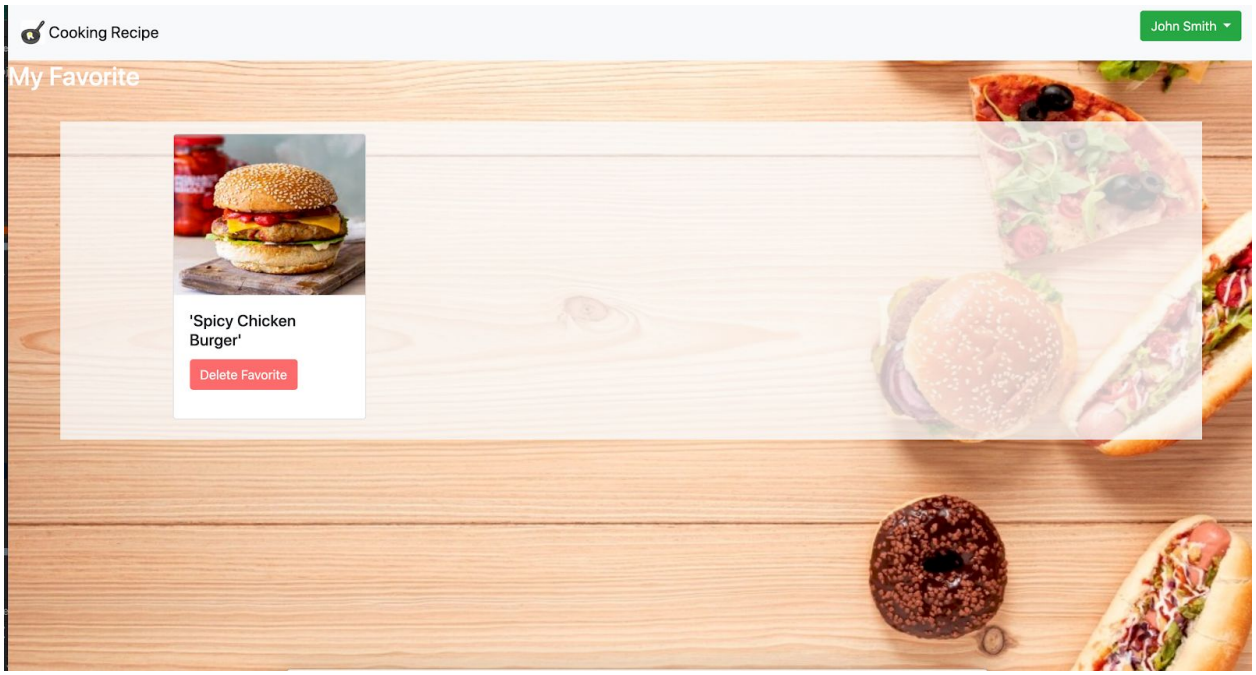
```
                    <input type="text" placeholder="Enter Recipe Name" name="addRecipeName" required>
                </div>
                <div>
                    <label><b><em>Recipe Type</em></b></label>
                    <select name="addFoodType">
                        <option>unknown</option>
                        <option>main-dish</option>
                        <option>side-dish</option>
                        <option>dessert</option>
                    </select>
                </div>
                <div>
                    <h3> Choose File to Upload in Server </h3>
                    <input type="file" name="dataFile" id="fileChooser"/><br/><br/>
                </div>

                <div>
                    <label><b><em>Recipe Step</em></b></label>
                    <img src="./Images/tomato.jpg" width="36px;">
                </div>
                <div>
                    <textarea name="addRecipeStep" rows="15" cols="100"></textarea>
                </div>
                <input type="submit" value="Upload Recipe" class="btn btn-primary"/>
            </form>
        </div>
        <div class="d-none d-xl-block col-xl-2 bd-toc">
        </div>
    </div>
</div>
</body>
</html>
```

- **Add to My Favorite** (Frontend Implementation)

We have a button Add to favourite in each recipe icon on the main page. We user click Add to favorites button the button name is changed if Is my favourite and that particular recipe is stored in user my favourite page. If the user does not like to see Add to My Favourite anymore they can have the ability to delete the Favourite recepie.


Cooking Recipe
Search...
Search
'Spicy Chicken Burger'
— 'John Cena'
GO to recipe
Add to Favorites
'Tuna Sa
Cooking Recipe
Search...
Search
'Spicy Chicken Burger'
— 'John Cena'
GO to recipe
Is My Favorite
'Tuna Sa


Cooking Recipe
John Smi
My Favorite
'Kimchi Fried Rice'
Delete Favorite
'Spicy Chicken Burger'
Delete Favorite


Cooking Recipe
John Smith
My Favorite
'Spicy Chicken Burger'
Delete Favorite

myfavorite.jsp

```
10      To change this template use File | Settings | File Templates.
11  --%>
12  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
13  <html>
14  <head...>
28
29  <style...>
37  <nav class="navbar navbar-expand-lg navbar-light bg-light"...>
71  <h2 style="...">My Favorite</h2>
72
73  <div class="container-fluid">
74      <div class="row flex-xl-nowrap">
75          <div class="mx-auto col-xl-11 main-background">
76              <!--display recipes-->
77              <div class="container" style="...">
78                  <div class="row">
79                      <%
80                          WebHandler webHandler = new WebHandler();
81                          DBHandler dbHandler = new DBHandler();
82
83                          /**************************************************************************
84                           * Displaying My Favorite
85                           *************************************************************************/
86                          HttpSession sess = request.getSession(false); //use false to use the existing session
87                          String currentUser = (String) sess.getAttribute("currentUserName");//this will return username anytime in the session
88                          int currentUserID = (int) sess.getAttribute("currentUserID");//this will return id anytime in the session
89                          try {
90                              Connection connection = dbHandler.startConnection();
91                              Statement stmt = connection.createStatement();
92                              ResultSet rs = stmt.executeQuery("SELECT * FROM recipe NATURAL JOIN myfavorite WHERE account_id='"+currentUserID+"'");
93                              int recipeID;
```

```
2                       ResultSet rs = stmt.executeQuery("SELECT * FROM recipe NATURAL JOIN myfavorite WHERE account_id='"+currentUserID+"'");
3                       int recipeID;
4                       String recipeName;
5                       String imagePath;
6                       int index = 0;
7
8                       String delete = "";
9                       while (rs.next()) {
0                           recipeID = rs.getInt("recipe_id");
1                           recipeName = rs.getString("recipe_name");
2                           imagePath = webHandler.getImage(recipeID);
3
4                           if ((index % 4) == 0 || index == 0) {
5                               out.println("<div class=\"row\">\n");
6                           }
7                           delete = "delete"+recipeID;
8                           out.println("<div class=\"col d-flex\" name ='" + recipeID + "'" + ">\n" +
9                                   "        <div class=\"card\" style=\"width: 15rem;\">\n" +
0                                   "            <img src='" + imagePath + "'" + " class=\"card-img-top\" alt=\"...\" width=\"150\" height=\"200\" >\n" +
1                                   "              <div class=\"card-body\">\n" +
2                                   "                  <h5 class=\"card-title\">'" + recipeName + "'</h5>\n" + "" +
3                                   "                  <footer class=\"blockquote-footer text-right\" style=\"color: green\">Public</footer>" +
4                                   "                   <form action=\"/myfavorite\" method=\"post\">\n" +
5                                   "                        <button class=\"btn btn-primary\" type=\"submit\" name='"+delete+"'\" + \">Delete Favorite\n
6                                   "                    </form>\n"  +
7                                   "                </div>\n" +
8                                   "        </div>\n" +
9                                   "    </div>\n");
0                           if ((index % 4) == 3) {
1                               out.println("</div>\n");
```

(Back end)

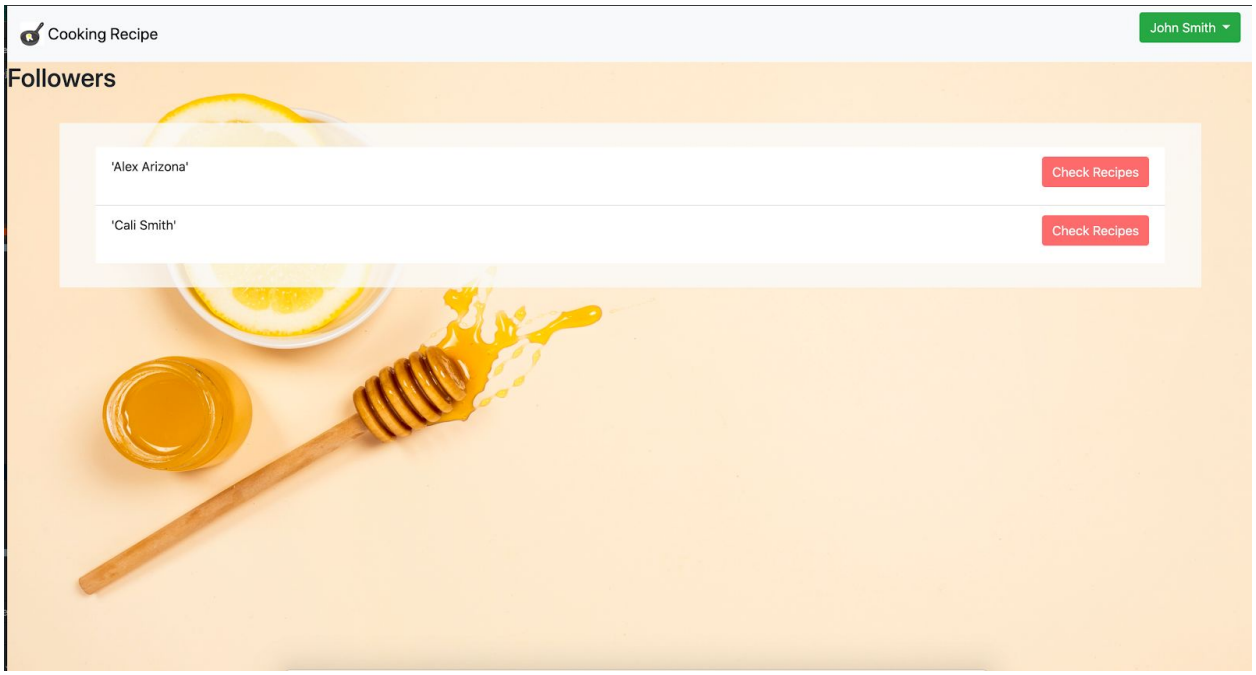
MyFavorite.java

```java
import ...

@WebServlet(name ="myfavorite")
public class MyFavorite extends HttpServlet {
    WebHandler webHandler = new WebHandler();
    DBHandler dbHandler = new DBHandler();
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int recipeCount = webHandler.getRecipeCount() + 1;
        int recipeID = 0;
        for (int index = 0; index < recipeCount; index++){

            if (request.getParameter( s: "delete"+String.valueOf(index)) != null){
                recipeID = index;
                break;
            }
        }
        if (request.getParameter( s: "delete" + recipeID) != null){

            try{
                Connection connection = dbHandler.startConnection();
                Statement statement = connection.createStatement();
                statement.executeUpdate( s: "DELETE FROM myfavorite WHERE recipe_id='"+recipeID+"'");
                dbHandler.closeConnection();
            }
            catch (Exception e){
                e.printStackTrace();
            }

            RequestDispatcher dispatcher = getServletContext().getRequestDispatcher( s: "/myfavorite.jsp");
            dispatcher.forward(request,response);
        }
```

```java
            if ((index % 4) == 3) {
                out.println("</div>\n");
            }
            index++;
        }
        rs.close();
        stmt.close();
        connection.close();
    }
    catch(Exception e) {
        System.out.println("SQLException caught: " + e.getMessage());
    }

    /**************************************************************************
     * Go back to mainpage
     *************************************************************************/
    if (request.getParameter("home") != null){...}
    /************************************************************************** ...*/
    if (request.getParameter("logoutInDisplay") != null){...}
    /************************************************************************** ...*/
    if (request.getParameter("myRecipe") != null){...}

    /************************************************************************** ...*/
    if (request.getParameter("myFavorites") != null){...}
    /************************************************************************** ...*/
    if (request.getParameter("addRecipe") != null){...}
    /************************************************************************** ...*/
    if (request.getParameter("following") != null){...}
%>
```

- **Followers** (Frontend Implementation)

The followers button directs users to the follower web page where users can keep track of users who are following them and also check for their follower recipes.

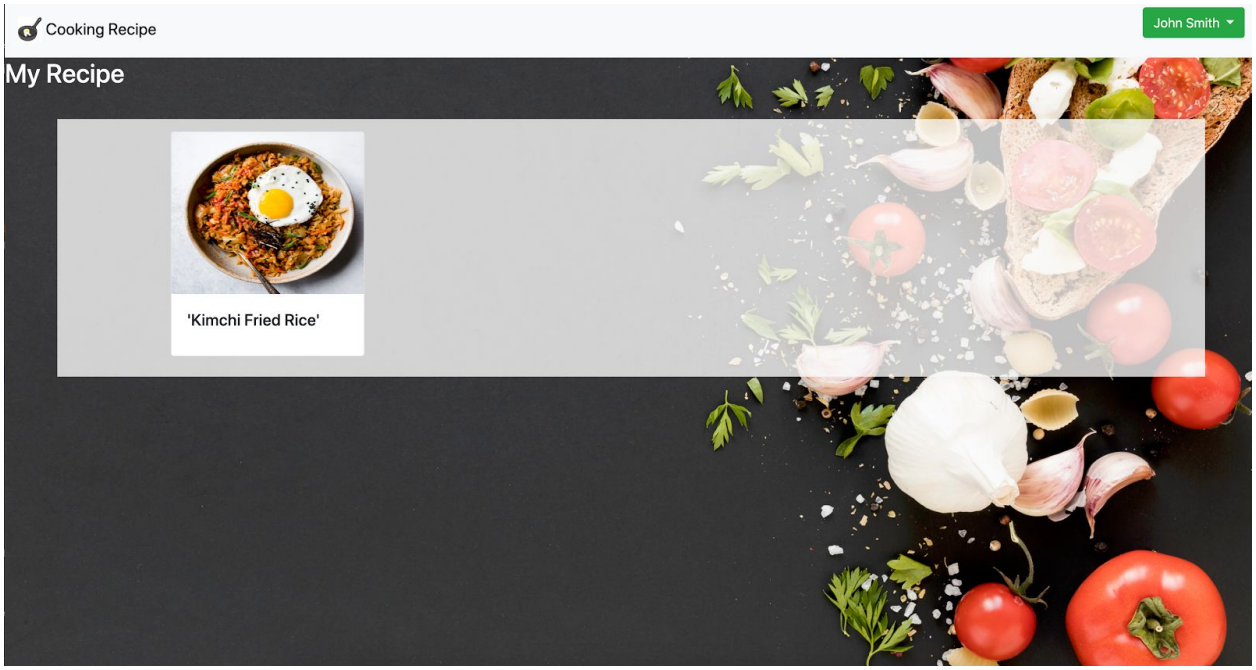



```
13
14 •  SELECT
15        a.account_id,
16        a.username
17    FROM
18        account a
19    JOIN
20        following f ON
21        f.follower_id = a.account_id
22    WHERE
23        f.account_id=2
24
```

100% 5:26

Result Grid Filter Rows: Search Export:

account_id	username
3	Alex Arizona
4	Cali Smith



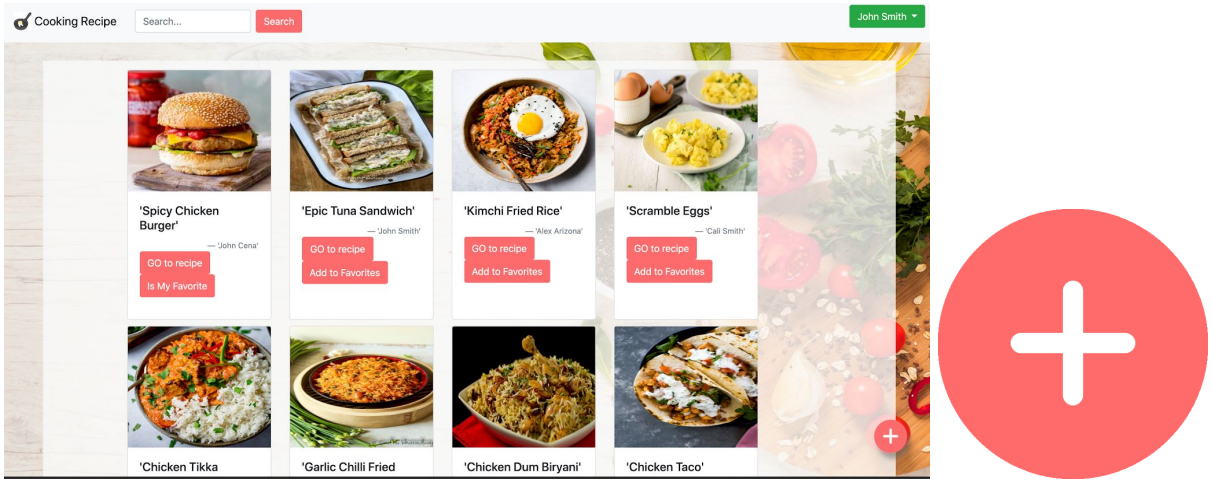
follower.jsp

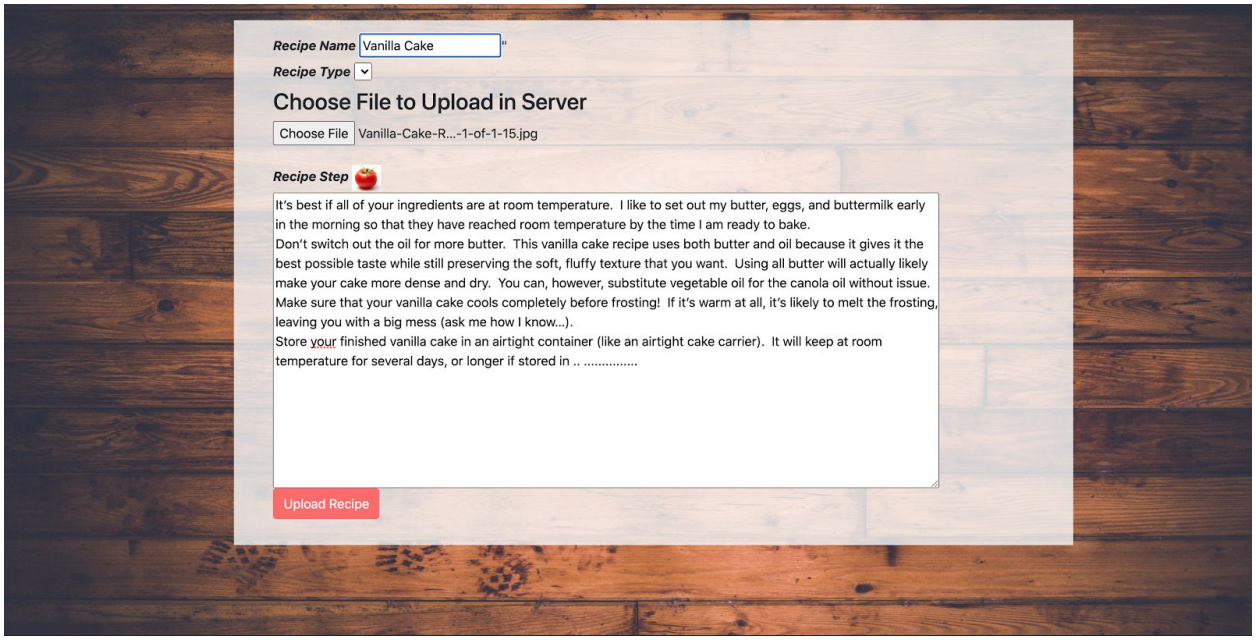
```
<%
    DBHandler dbHandler = new DBHandler();
    WebHandler webHandler = new WebHandler();
    HttpSession sess = request.getSession(false); //use false to use the existing session
    String currentUser = (String) sess.getAttribute("currentUserName");//this will return username anytime in the session
    int currentUserID = (int) sess.getAttribute("currentUserID");//this will return id anytime in the session
    ArrayList<Integer> followerIDList = new  ArrayList<~>();
    ArrayList<String> followerNameList = new  ArrayList<~>();
    try{
        Connection connection = dbHandler.startConnection();
        Statement statement = connection.createStatement();
        ResultSet rs = statement.executeQuery("SELECT a.account_id, a.username FROM account a JOIN following f ON f.follower_id =

        while (rs.next()){
            followerIDList.add(rs.getInt("account_id"));
            followerNameList.add(rs.getString("username"));
        }
        rs.close();
        statement.close();
        connection.close();
        for (int index = 0; index <followerIDList.size(); index++ ){
            System.out.print(followerNameList.get(index) + " "+ followerIDList.get(index));

            out.println("<li class=\"list-group-item\">\n" +
                    "      <form action=\"/follower\" method=\"post\">\n" +
                    "          <label>'"+followerNameList.get(index)+"'</label>\n" +
                    "                <button style=\"float: right;\" type=\"submit\"  name='"+followerIDList.get(index)+"'\" cla
                    "          </form>\n" +
                    "      </li>");

        }
```

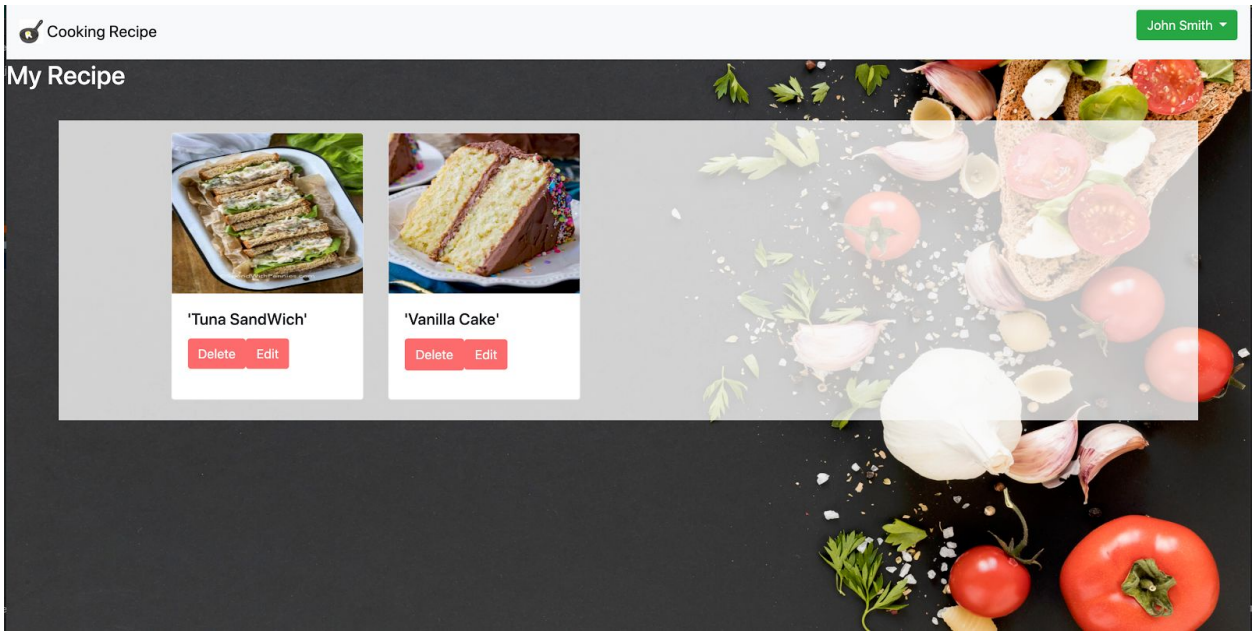
```
vorite.jsp × follower.jsp ×
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head...>

<style>
    body {...}
</style>
<nav class="navbar navbar-expand-lg navbar-light bg-light"...>
<h2>Followers</h2>
<div class="container-fluid">
    <div class="row flex-xl-nowrap">
        <div class="mx-auto col-xl-11 main-background">
            <!--display followers-->
            <div style="...">
                <ul class="list-group list-group-flush">
                    <%...%>

                </ul>
            </div>
        </div>
    </div>
</div>

</body>
</html>
```

- **Add New Recipe** (Frontend Implementation)

This + is a button located in the right bottom corner and it will allow users to add a new recipe and when users hover the mouse on the icon will have a small note “Add new recipe”.



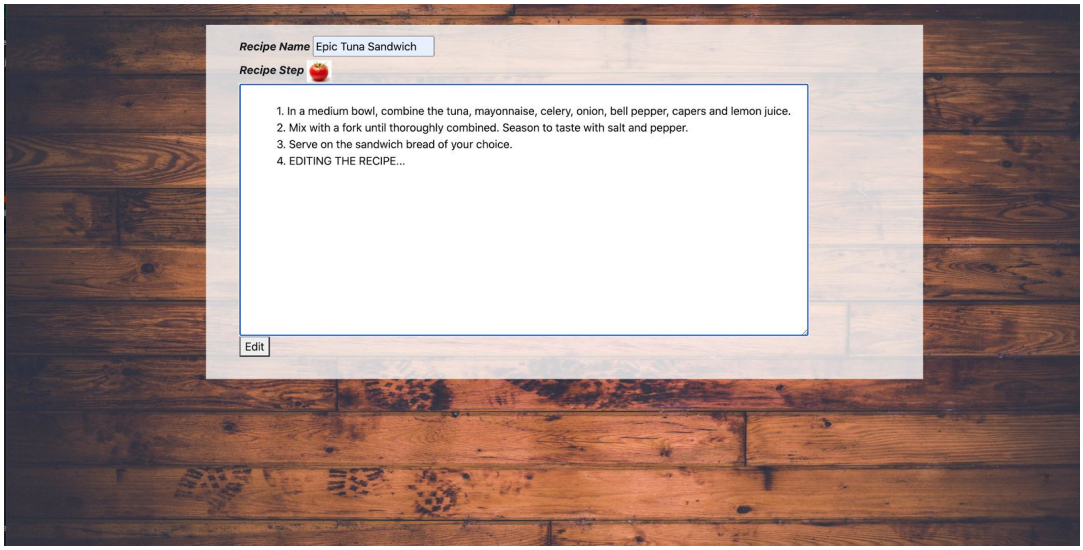





```
            </div>
        </div>
    </div>
    <a href="/addRecipe.jsp"><img class="add-recipe-btn" title="Add new recipe" src="./Images/plus.png"></a>

</div>
```

- **Edit Recipe**

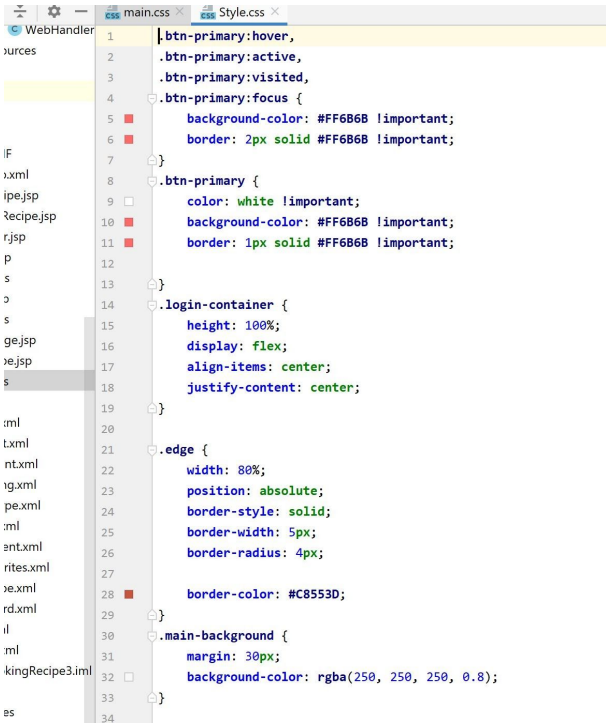
The edit recipe is the Update functionality that users can make in my recipe page. In the edit recipe user can edit recipe name and description to the existing recipe which is then updated in the database.

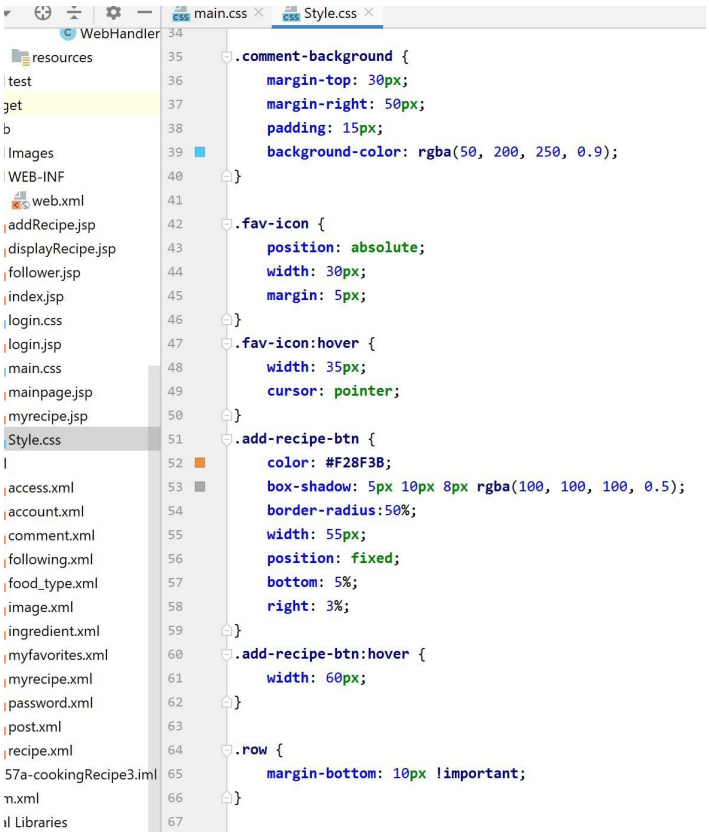


- **Delete Recipe**

The Delete recipe is the Delete functionality that users can make in my recipe page. Users can click the delete button from my recipe page which then deletes the recipe and from my recipe page and deleted from the database.

(Front end CSS Code)





b. Non-Functional Requirements

I. Scalable (**Desired**)

A. the program would be able to work well from 1 to 'n' number of cooking recipes.

II. Smooth and Intuitive UI (**Desired**)

A. Simple looking and easy to follow UI that knockout user experience.

III. Responsive Interface (**Desired**)

A. UI adapts to different screen sizes.

IV. High Reliability (**Desired**)

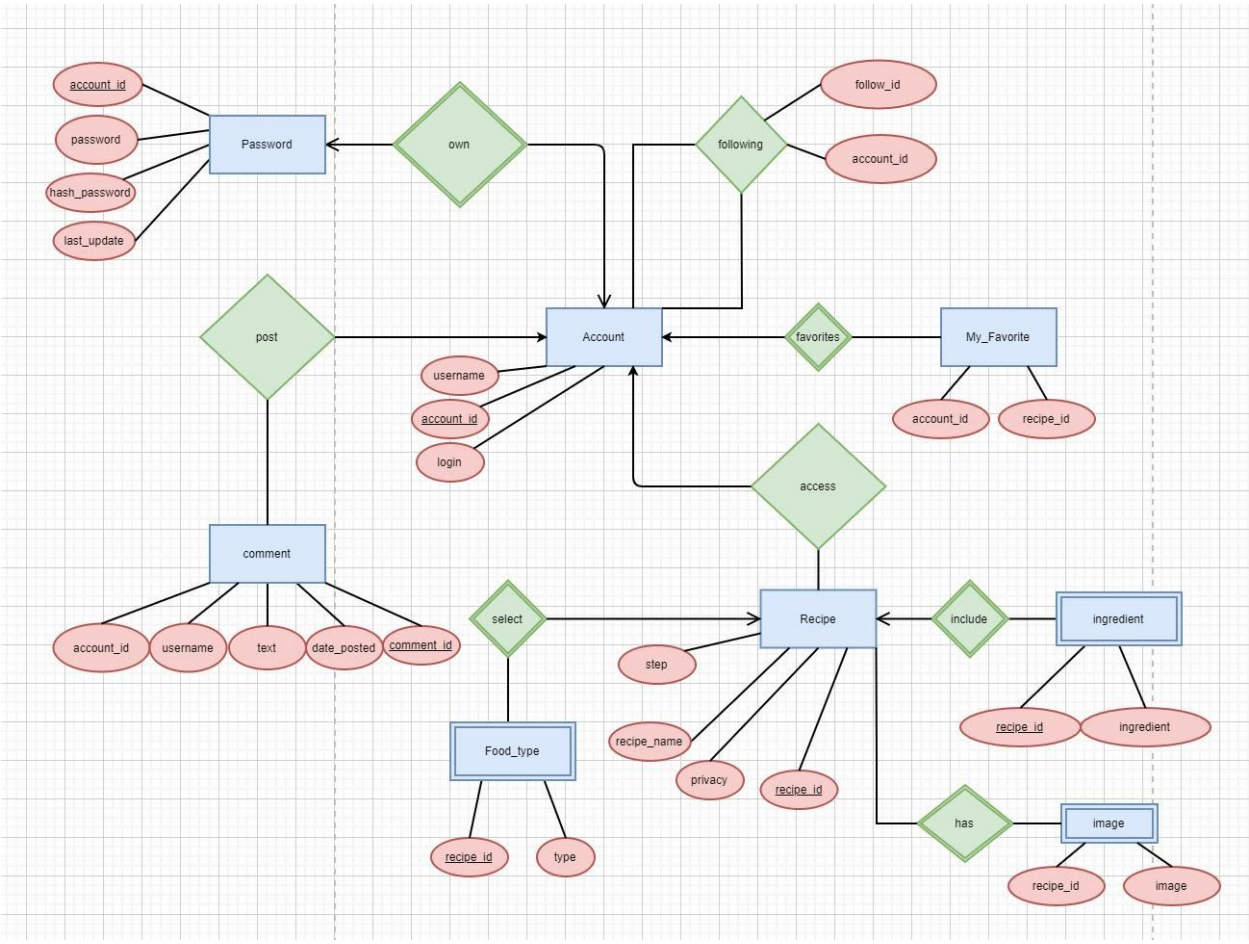
A. It's always consistent.

V. Fun (**Essential**)

c. Cooking with "Cooking Recipes" is interactive to the user

IV. ERD Diagram

Below is an illustration of our project 'Cooking Recipe' on where it provides a server-client system.



V. Database Entities, Relations, and Properties

Database Name: cooking_recipe

Entities:

1. Accounts(<u>account_id</u>, username, login)
2. Recipe(<u>recipe_id</u>, recipe_name, step, privacy)
3. Password (<u>account_id</u>, password, hash_password, last_update)

4. Comment(account_id, recipe_id, username, text, date_posted)
5. Food_type(recipe_id, type)
6. Ingredient(recipe_id, ingredient)
7. image(recipe_id, image)
8. myfavorite(account_id, recipe_id)

Relation:

1. following(account_id, follower_id)
2. post(account_id, date_posted)
3. access(account_id, recipe_id)

Description:

1. **Accounts**: Accounts is an entity that represents the attributes of a user in the database. Accounts consist of attributes like account_id which is the primary key for the Accounts table, username and login status.
2. **Recipe**: Recipe is an entity in which represents the scope of different recipes in the database table. This table has recipe_id as an attribute which is the primary key followed by other attributes like step, recipe_name and privacy.
3. **Password**: Password is the weak entity in the database whose existence is dependent on the Account table. This table has a weak relationship with the Account table with attributes like account_id as primary key, password, hash_password and last_update as date. Each account can have at least one password.
4. **Comment**: Comment is an entity whose scope in the database is to store user comments. This table has attributes like account_id as primary key, username, text, date_posted and post_id.
5. **Food_type:** Food_type is a weak entity table in the database whose existence is totally dependent on the recipe entity. Food_type must have attributes like recipe_id and type. Each repice has exactly one food_type.
6. **Ingredient:** Ingredient is a weak entity table in the database whose existence is dependent on the recipe table. This table must have attributes like recipe_id as primary key, ingredient. Each recipe has exactly at least one ingredient.

7. **Image:** Image is an entity which shares a weak relationship with the recipe table. Each recipe has an image, the image table can not exist without a recipe table. Image must have recipe_id and image as attributes.
8. **Following:** Following is the relationship table between account table. This is the table which stores account_id and account_following as an must have attribute.
9. **Post:** The post table establishes the relationship between the account and the comment table using the account_id. The main purpose of the post table is to locate the posted comment from the particular account in the comment table.
10. **Access:** Access is a relationship between account and recipe table. Account can access recipes.
11. **Myfavorite:** A list of recipes in which the user added to their favorites.

VI. Tables Output (screenshots)

1. **account** (account_id, username, login) table screenshot.

account_id	username	login
1	John Cena	0
2	John Smith	0
3	Alex Arizona	1
4	Cali Smith	0
5	Karen Cena	0
6	Demi Gonzales	0
7	Belarus King	1
8	Sarah Jen	0
9	Damian Richards	0
10	Bob Junior	1
NULL	NULL	NULL

2. **password** (account_id, password, hash_password, last_update)

account_id	password	hash_password	last_update
1	John Cena	mandjemandjeldign3m5nldign3m5n	2006-02-14
2	John Smith	adfm19fnc5kgfbadfm19fnc5kgfb5n	2011-02-14
3	Alex Arizona	adjfdkdajadfm19fnc5kgfbldf25as	2012-02-14
4	Cali Smith	a3kasdf45wadfm19fnc5kgfbdfajdf	2012-02-14
5	Karen Cena	a3kasdf45wadfm19fnc5kgfbdfajdf	2012-02-14
6	Demi Gonzales	jdie93jdie93utndndutndndasdwfs	2012-02-14
7	Belarus King	d1nfhddjdie93utndndj45mdneasde	2012-02-14
8	Sarah Jen	aeif3i8jdie93utndnddhndsdeavks	2012-02-14
9	Damian Richardsa	aeif3i8jdie93ufgdsgdhndsfeavks	2012-02-14
10	Bob Junior	aeif3i8jdie9sufgdsgdhndsfeaasc	2012-02-14
NULL	NULL	NULL	NULL

3. **recipe**(recipe_id, recipe_name, step, privacy)

recipe_id	recipe_name	step	privacy
1	Spicy Chicken Burger	Step 1: Make the Frank's® Red Hot lim…	0
2	Tuna SandWich	1. In a medium bowl, combine the tuna,…	0
3	Kimchi Fried Rice	1. In a large saucepan of 3 cups water,…	1
4	Scramble Eggs	1. BEAT eggs, milk, salt and pepper in a…	1
5	Chicken Tikka Masala	1. In a large saucepan of 2 cups water,…	1
6	Garlic Chilli Fried Rice	Step by step recipe for Chilli Garlic sauc…	1
7	Chicken Dum Biryani	1. Wash rice few times till the water run…	1
8	Chicken Taco	1. In a small bowl, combine chili powder…	1
9	Breakfast Burrito	1. Cook the potatoes in bacon fat.…	1
10	Chicken Wings	1. Preheat oven to 425°F. Dab wings wit…	1
NULL	NULL	NULL	NULL

4. **ingredient**(<u>recipe_id</u>, ingredient)

recipe_id	ingredient
1	1 cup sour cream 2 tablespoons F…
2	2 (6-ounce) cans solid, water packed tuna…
3	1 1/2 cups jasmine rice 1 cup kim…
4	4 large potatoes, peeled and cut into strips…
5	4 large eggs ¼ cups of milk…
6	1 cup basmati rice 1 1/2 tablespoo…
7	2 tbsp. extra-virgin olive oil 3 chick…
8	500 grams chicken 4 cups basmat…
9	3 cups shredded Rotisserie chicken…
10	chicken wings, flour, olive oil, salt and pepper.

5. **image**(recipe_id, image)

recipe_id	image
1	Images/1.jpeg
2	Images/2.jpg
3	Images/3.jpg
4	Images/4.jpg
5	Images/5.jpg
6	Images/6.jpg
7	Images/7.jpg
8	Images/8.jpg
9	Images/9.jpg
10	Images/10.jpeg

6. **comment**(comment_id, account_id, username, text, date_posted)

account_id	recipe_id	username	text	date_posted
1	1	John Cena	I love chicken wings, how do you make it?	2020-06-23
2	1	John Smith	I love Chicken Tikka Masala, Can you share the…	2020-06-23
3	2	Alex Arizona	I love Chicken Tikka Masala, Can you share the…	2020-06-23
3	7	Cali Smith	YUM YUM...Chicken Dum Biryani, how do you…	2020-06-24
4	8	Karen Cena	Delicious Breakfast Burrito, Share your recipes	2020-06-24
5	3	Demi Gonzales	I love chicken mushroom pizza, how do you ma…	2020-06-25
6	9	Belarus King	WOW my favourite chicken curry, how do you m…	2020-06-25
7	5	Sarah Jen	All time favourite Appetizer Samosa chat, how d…	2020-06-25
8	6	Damian Richards	Amazing Food!!!	2020-06-26
9	8	Bob Junior	I love chicken momo, how do you make it?	2020-06-27

7. **food_type**(recipe_id, type)

recipe_id	type
1	unknown
2	unknown
3	main-dish
4	side-dish
5	side-dish
6	main-dish
7	unknown
8	main-dish
9	main-dish
10	main-dish

8. **following**(account_id, follow_id)

account_id	follower_id
1	2
1	3
1	4
2	3
2	4

9. **post** (account_id, date_posted)

account_id	date_posted
1	2020-06-23
2	2020-06-23
3	2020-06-23
4	2020-06-24
5	2020-06-24
6	2020-06-25
7	2020-06-25
8	2020-06-25
9	2020-06-26
10	2020-06-27

10. **access**(account_id, recipe_id)

recipe_id	account_id
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

11.myfavorite(account_id, recipe_id)

account_id	recipe_id
1	0
1	2
1	0
1	5
2	0
2	3
4	0
4	5
1	0
1	3

IX. Lesson Learned

Min-yuan Lee (Mina)

Learning how to build a full stack database application in the summer section is quite intense because the time is short and there are so many things that need to be done. I have learned a lot during our project building. Like before I only used Github for my own project, but this time we shared one project and we all did the push and pull to contribute this project. I have also learned how to use 3-Tier structure to build a local web application and use Java, JSP, HTML, CSS, and MySQL which I have never had experience how to combine all the different languages before. Throughout the project we really put a lot of effort to work on the design for user experience, and putting together the front end and back end since we all do not have experience with that. I also understand the importance of designing a database because it will allow you to get the data that you need from the database. Sometimes, we just realized that we need to add another attribute to the table for building some functionality.

Overall our team made it, and I think this project really helped me learn a lot and not just using MySQL statements, but also how to collaborate as a team and implement what we learn to the real world situation.

Guiller Dalit (Gel)

What I have learned in the course of our project development has given me insight into how three-tier-architecture operates. I have learned how to manipulate data inputs and data definitions using java.sql libraries. And that with the combination of the javax.servlet libraries I was able to handle inputs from the user into the database. Before I was only using .txt to save the information and data from the user, but in the process of our project development, and so I realized the importance of a database to keep the current state and information of the program. It is far more efficient to mutate and access the data than using .txt files. And another thing that I learn is that the design of the ERD diagram determines the structure of the program. I realized this after coding the back-end of our project on where some of the tables we have are missing key attributes and there are missing relations between our tables. For example, the favorites recipe of the user, this functional requirement was not included in our ERD diagram. Another thing that I learned the most is how the front-end handles the input from the users on the web. I have learned how to acquire, store, and manipulate the data from the front-end. I learned how to connect a jsp file to a java servlet class by mapping the two in the web.xml. I also learned how to configure mysql-connector to MySQL workbench and apache-tomcat to run a Web application by

configuring the dependency and the .jar files. Another thing I have learned is that certain .jar files support certain functionalities, for example, in our project I had to download the .jar files of common-io and commons-fileupload and configure it in our project in order for our web application to handle images uploads from the user and save the image uploaded to our local directories and store the location of the images in our database. I have learned a lot in our project development in a short amount of time that I have never known before.

Sushil M S Karki(sushil)

This project gave me a clear understanding of the architecture of a full stack web application. I was quite nervous at the beginning because I did not have much experience but during the project journey I learned so many things and I was very fortunate to have supportive and hard working teammates who put countless effort to make this project work and helped me learn through this process.

I also learned to use different user tools and configurations that are required to run the project. This project gave me better understanding on working on the version control system git github. I learned how to use Java as a backend with Tomcat as a local server and making changes on the database system.I learned how to implement 3-tier applications with JSP, JavaServlets, Tomcat Server and MySql databases. I learned how

to implement ERD diagrams into tables in MySql and make changes in the database system when the user makes changes on the frontend. For this project I worked majorly on the front-end part designing SingOn/SignUp which have enhanced my skills in html css and javascript. Most importantly I learned how to work in a team environment and contribute what you have learned and make an effort to help your team best. This project gives me motivation to develop other full-stack projects where I can implement the experience from this project to learn more about frontend, backend and RDBMS systems. I want to thank Prof. Mike Wu for giving me the opportunity to work on this class project.

X. Github Link: https://github.com/gelier/cs157a-cr-eclipse-git