Final Project Phase #2

submitted to

Dr. James Papademas

by

Marie Ber Manyanga (A20473921)

As a part of the Object-Oriented Application Development course (ITMD 510)

Illinois Institute of Technology

December 12, 2020

1. **CookHelperApplication.java**

```
package application;

/*
	This class contains the main method used to run the application
	Programmer: Manyanga Marie Ber
*/

import java.io.IOException;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import controllers.MainController;

public class CookHelperApplication extends Application {

	public static void main(String[] args) { launch(args); }
	@Override public void start(Stage stage) throws IOException {

		Scene scene = new Scene(new StackPane());
		MainController mainController = new MainController(scene);
		mainController.initiliaze();

		stage.setTitle("CookHelper");
		stage.setScene(scene);
		stage.show();

	}
```

2. **AddRecipeController.java**

```
package controllers;

/*
	This class is used to control the "add recipe" screen. It handles
	actions performed on that screen.
	Programmer: Manyanga Marie Ber
*/
```

```
import java.io.IOException;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import models.RecipeModel;


public class AddRecipeController {

	@FXML private TextField name;
	@FXML private TextField description;
	@FXML private TextField category;
	@FXML private TextField directions;
	@FXML private Button saveButton;
	@FXML private Button backButton;
	private Scene scene;


	public AddRecipeController(Scene scene)
	{
		this.scene = scene;
	}


	/*
	 *  This method is used to handle actions performed by the user
	 */
	public void handleAddRecipe(MainViewController mainViewController, boolean
flag)
	{

		saveButton.setOnAction((e) -> {

			RecipeModel recipe = new RecipeModel (name.getText(),
description.getText(), category.getText(), directions.getText());
			mainViewController.getRecipeController().addRecipe(recipe);
			mainViewController.showMainView(flag);

		});
```

```
		backButton.setOnAction((e) -> {

			mainViewController.showMainView(flag);

		});


	}


	/*
	 *  This method is used to display the desired page
	 */
	public void showAddRecipe(MainViewController mainViewController, boolean flag)
{
		try {

			FXMLLoader loader = new
FXMLLoader(getClass().getResource("/views/AddRecipeView.fxml"));
			loader.setController(this);
			scene.setRoot((Parent) loader.load());
			handleAddRecipe(mainViewController, flag);
		} catch (IOException ex) {
			ex.printStackTrace();
		}
	}

}
```

3. **EditRecipeController.java**

```
	package controllers;

	/*
		This class is used to control the "edit recipe" screen. It handles
		actions performed on that screen.
		Programmer: Manyanga Marie Ber
	*/


	import java.io.IOException;
	import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import models.RecipeModel;


public class EditRecipeController {

	@FXML private TextField name;
	@FXML private TextField description;
	@FXML private TextField category;
	@FXML private TextField directions;
	@FXML private Button saveButton;
	@FXML private Button backButton;
	private Scene scene;


	public EditRecipeController(Scene scene)
	{
		this.scene = scene;
	}


	/*
	 *  This method is used to handle actions performed by the user
	 */
	public void handleEditRecipe(MainViewController mainViewController,
RecipeModel recipe, boolean flag)
	{

		name.setText(recipe.getName());
		description.setText(recipe.getDescription());
		category.setText(recipe.getCategory());
		directions.setText(recipe.getDirections());

		saveButton.setOnAction((e) -> {

			RecipeModel tempRecipe = new RecipeModel (recipe.getId(),
name.getText(), description.getText(), category.getText(), directions.getText());

mainViewController.getRecipeController().editRecipe(tempRecipe);
			mainViewController.showMainView(flag);
```

```
                });


                backButton.setOnAction((e) -> {

                        mainViewController.showMainView(flag);

                });


        }


        /*
         *  This method is used to display the desired page
         */
        public void showEditRecipe(MainViewController mainViewController,
RecipeModel recipe, boolean flag) {
                try {

                        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/views/EditRecipeView.fxml"));
                        loader.setController(this);
                        scene.setRoot((Parent) loader.load());
                        handleEditRecipe(mainViewController, recipe, flag);
                } catch (IOException ex) {
                        ex.printStackTrace();
                }
        }

}
```

4. LoginController.java

```
package controllers;

/*
        This class is used to control the "login" screen. It handles
        actions performed on that screen.
        Programmer: Manyanga Marie Ber
*/

import java.io.IOException;
```

```
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import models.LoginModel;

public class LoginController {

	@FXML private TextField user;
	@FXML private TextField password;
	@FXML private Button loginButton;

	private Scene scene;
	private LoginModel loginModel;
	private MainViewController mainViewController;

	public LoginController(Scene scene, MainViewController
mainViewController) {

		this.scene = scene;
		loginModel = new LoginModel();
		this.mainViewController = mainViewController;
	}

	/*
	 *  This method is used to handle actions performed by the user
	 */
	public void handleLogin() {

		loginButton.setOnAction((e) -> {

			if(checkCrentials().equals("admin")) {

				System.out.println("Success! You are connected and
your role is : admin");
				mainViewController.showMainView(true);
			}

			else if(checkCrentials().equals("user")) {

				System.out.println("Success! You are connected and
your role is : user");
				mainViewController.showMainView(false);
```

```
                        }

                        else {

                                System.out.println("Wrong username or passowrd!
Please try again...");

                        }

                });

        }

        /*
         *  This method is used to check authorization credentials
         */
        private String checkCrentials() {

                if(loginModel.getCredentials(user.getText(), password.getText())) {

                        return loginModel.getRole();
                }

                return "not found";

        }

        /*
         *  This method is used to display the desired page
         */
        public void showLoginView() {
                try {
                        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/views/LoginView.fxml"));
                        loader.setController(this);
                        scene.setRoot((Parent) loader.load());
                        handleLogin();
                } catch (IOException ex) {
                        ex.printStackTrace();
                }
        }
```

```
    }
```

5. **MainController.java**

```
package controllers;

/*
    This class serves as the main controller for the application. It
    initializes other controllers and displays the first screen.
    Programmer: Manyanga Marie Ber
*/


import javafx.scene.Scene;

public class MainController {

    private LoginController loginController;
    private RecipeController recipeController;
    private MainViewController mainViewController;

    public MainController(Scene scene) {

        recipeController = new RecipeController(scene);
        mainViewController = new MainViewController(scene, recipeController);
        loginController = new LoginController(scene, mainViewController);

    }

    /*
     *  This method is used to initialize the main controller
     */
    public void initiliaze() {

        loginController.showLoginView();

    }


}
```

6. MainViewController.java

```
package controllers;

/*
	This class is used to control the "main view" screen. It handles
	actions performed on that screen.
	Programmer: Manyanga Marie Ber
*/

import java.io.IOException;
import java.util.ArrayList;
import java.util.Comparator;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.transformation.FilteredList;
import javafx.collections.transformation.SortedList;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import models.RecipeModel;

public class MainViewController {

	@FXML private TableView<RecipeModel> tableView;
	@FXML private TableColumn<RecipeModel, String> recipeNameColumn;
	@FXML private TextField search;
	@FXML private Button addButton;
	private Scene scene;
	private RecipeController recipeController;


	public MainViewController(Scene scene, RecipeController recipeController)
	{
```

```
		tableView = new TableView<RecipeModel>();
		this.scene = scene;
		this.recipeController = recipeController;
	}


	public RecipeController getRecipeController() {
		return recipeController;
	}


	/*
	 *  This method is used to handle actions performed by the user
	 */

	public void handleMainView(boolean flag)
	{

		MainViewController mainViewController = this;

		if(!flag) //disable the button if user does not have admin rights
		{

			addButton.setDisable(true);
		}


		addButton.setOnAction((e) -> {



recipeController.getAddRecipeController().showAddRecipe(mainViewController,
flag);

		});

		//display the selected recipe on the "view recipe" screen
		tableView.setOnMouseClicked((EventHandler<? super MouseEvent>)
new EventHandler<MouseEvent>() {
			@Override
			public void handle(MouseEvent event) {

				if (event.getClickCount() == 2) //Check if the user has
clicked twice
```

```
                        {

                                int id =
tableView.getSelectionModel().getSelectedItem().getId();
                                RecipeModel recipe =
recipeController.getRecipeById(id);

recipeController.getViewRecipeController().showViewRecipe(recipe,
mainViewController, flag);
                        }

                    }
                });


                // add recipes' names in the table
                ArrayList<RecipeModel> recipes = recipeController.getRecipes();
                ObservableList<RecipeModel> items =
FXCollections.observableArrayList(recipes);
                recipeNameColumn = (TableColumn<RecipeModel, String>)
tableView.getColumns().get(0);
                recipeNameColumn.setCellValueFactory(data -> new
SimpleStringProperty(data.getValue().getName()));


                //handle search functionality
                FilteredList<RecipeModel> filteredItems = new FilteredList<>(items);
                search.textProperty().addListener((observable, oldValue, newValue) ->
{

                    filteredItems.setPredicate(data -> {
                        if (newValue == null || newValue.isEmpty()) {

                            return true;
                        }

                        if
(data.getName().toLowerCase().contains(newValue.toLowerCase())) {

                            return true;
                        }

                        return false;
                    });
```

```
                });

                //display recipes' names in the table
                SortedList<RecipeModel> sortedItems = new
SortedList<RecipeModel>(filteredItems);
                sortedItems.comparatorProperty().bind((ObservableValue<? extends
Comparator<? super RecipeModel>>) tableView.comparatorProperty());
                tableView.setItems(sortedItems);

        }

        /*
         *  This method is used to display the desired page
         */
        public void showMainView(boolean flag) {
                try {

                        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/views/MainView.fxml"));
                        loader.setController(this);
                        scene.setRoot((Parent) loader.load());
                        handleMainView(flag);
                } catch (IOException ex) {
                        ex.printStackTrace();
                }
        }



}
```

7. RecipeController.java

```
package controllers;

/*
        This class controls the data flow within the application. It
        provides methods needed to interact with the database in
        order to get recipes' information or to perform similar opera-
        tions.
        Programmer: Manyanga Marie Ber
*/

import java.util.ArrayList;
```

```
import javafx.scene.Scene;
import models.RecipeModel;

public class RecipeController {

	private RecipeModel recipeModel;
	private ViewRecipeController viewRecipeController;
	private AddRecipeController addRecipeController;
	private EditRecipeController editRecipeController;

	public RecipeController(Scene scene) {

		recipeModel = new RecipeModel();
		viewRecipeController = new ViewRecipeController(scene);
		addRecipeController = new AddRecipeController(scene);
		editRecipeController = new EditRecipeController(scene);

	}

	public ViewRecipeController getViewRecipeController() {

		return viewRecipeController;
	}

	public AddRecipeController getAddRecipeController() {

		return addRecipeController;
	}

	public EditRecipeController getEditRecipeController() {

		return editRecipeController;
	}

	/*
		This method adds a recipe to the database
	 */
	public void addRecipe(RecipeModel recipe){

		 recipeModel.addRecipe(recipe);

	}

	/*
```

```
        This method changes information related to
        a recipe in the database
     */
    public void editRecipe(RecipeModel recipe){

        recipeModel.editRecipe(recipe);

    }

    /*
        This method gets all recipes from the database
    */
    public ArrayList<RecipeModel> getRecipes(){

        return recipeModel.getRecipes();

    }

    /*
        This method gets a recipe from the database
     */
    public RecipeModel getRecipeById(int id){

        return recipeModel.getRecipeById(id);

    }

    /*
        This method deletes a recipe in the database
     */
    public void deleteRecipeById(int id){

         recipeModel.deleteRecipe(id);

    }

}
```

8. ViewRecipeController.java

```
package controllers;

/*
    This class is used to control the "view recipe" screen. It handles
```

```
	actions performed on that screen.
	Programmer: Manyanga Marie Ber
*/

import java.io.IOException;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import models.RecipeModel;


public class ViewRecipeController {

	@FXML private TextArea name;
	@FXML private TextArea description;
	@FXML private TextArea category;
	@FXML private TextArea directions;
	@FXML private Button deleteButton;
	@FXML private Button editButton;
	@FXML private Button backButton;
	private Scene scene;


	public ViewRecipeController(Scene scene)
	{
		this.scene = scene;
	}

	/*
	 *  This method is used to handle actions performed by the user
	 */
	public void handleViewRecipe(RecipeModel recipe, MainViewController
mainViewController, boolean flag)
	{

		name.setText(recipe.getName());
		description.setText(recipe.getDescription());
		category.setText(recipe.getCategory());
		directions.setText(recipe.getDirections());
```

```
                if(!flag) //disable buttons if user does not have admin rights
                {

                        deleteButton.setDisable(true);
                        editButton.setDisable(true);
                }


                backButton.setOnAction((e) -> {

                        mainViewController.showMainView(flag);


                });


                deleteButton.setOnAction((e) -> {


mainViewController.getRecipeController().deleteRecipeById(recipe.getId());
                        mainViewController.showMainView(flag);

                });



                editButton.setOnAction((e) -> {



mainViewController.getRecipeController().getEditRecipeController().showEditRecip
e(mainViewController, recipe, flag);


                });


        }

        /*
         *  This method is used to display the desired page
         */
        public void showViewRecipe(RecipeModel recipe, MainViewController
mainViewController, boolean flag) {
```

```
                try {

                        FXMLLoader loader = new
        FXMLLoader(getClass().getResource("/views/ViewRecipe.fxml"));
                        loader.setController(this);
                        scene.setRoot((Parent) loader.load());
                        handleViewRecipe(recipe, mainViewController, flag);
                } catch (IOException ex) {
                        ex.printStackTrace();
                }
        }

    }
```

9. DaoModel.java

```
package dao;

/*
    This class contains methods used to perform database related
    operations such as creating a table, inserting records, etc.

    Programmer: Manyanga Marie Ber
*/

import java.sql.SQLException;
import java.sql.Statement;


public class DaoModel {

    protected DBConnect connection = null;
    protected Statement statement = null;


    public DaoModel() {

        connection = new DBConnect();
    }

    /*
     *  This method is used to create a table to store users information
     */
```

```
	public void createUsersTable() {

		try {

			statement = connection.connect().createStatement();
			String sql = "CREATE TABLE cookhelper_users " +
					"(id INTEGER NOT NULL AUTO_INCREMENT, " +
					" username VARCHAR(50) NOT NULL, " +
					" password VARCHAR(50) NOT NULL, " +
					" admin BOOLEAN NOT NULL, " +
					" PRIMARY KEY ( id ))";

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}

	/*
	 *  This method is used to add a user with admin rights in the database
	 */
	public void addAdmin() {

		try {

			statement = connection.connect().createStatement();
			String sql = "INSERT INTO cookhelper_users (username, password,
admin) " +
					"VALUES ('admin', 'admin', true)";

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}

	/*
	 *  This method is used to add a user in the database (no admin rights)
```

```
	 */
	public void addUser() {

		try {

			statement = connection.connect().createStatement();
			String sql = "INSERT INTO cookhelper_users (username, password,
admin) " +
					"VALUES ('user', 'user', false)";

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}

	/*
	 *  This method is used to create a table to store recipes
	 */
	public void createRecipeTable() {

		try {

			statement = connection.connect().createStatement();
			String sql = "CREATE TABLE cookhelper_recipe " +
					"(id INTEGER NOT NULL AUTO_INCREMENT, " +
					" name VARCHAR(150) NOT NULL, " +
					" description VARCHAR(255) NOT NULL, " +
					" category VARCHAR(50) NOT NULL, " +
					" directions TEXT NOT NULL, " +
					" PRIMARY KEY ( id ))";

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}
```

```
}
```

10. DBConnect.java

```
package dao;

/*
	This class is used to connect to the database
	Programmer: Manyanga Marie Ber
 */

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

public class DBConnect {

	// Code database URL
	static final String DB_URL =
"jdbc:mysql://www.papademas.net:3307/510fp?autoReconnect=true&useSSL=false";
	// Database credentials
	static final String USER = "fp510", PASS = "510";

	public Connection connect() throws SQLException {

		return DriverManager.getConnection(DB_URL, USER, PASS);

	}

}
```

11. LoginModel.java

```
package models;

/*
	This class serves as "a model" for the Login Controller. It
	interacts with the database and provides information needed
	by the controller.
	Programmer: Manyanga Marie Ber
*/

import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import dao.DaoModel;

public class LoginModel extends DaoModel {

	private String role;

	public String getRole() {

		return role;

	}

	/*
	 *  This method is used to get credentials fom the database
	 */
	public boolean getCredentials(String username, String password){

		try {

			statement = connection.connect().createStatement();
			String sql = "SELECT * FROM cookhelper_users";
			ResultSet rs = statement.executeQuery(sql);

			while(rs.next()){

				String tempUsername = rs.getString("username");
				String tempPassword = rs.getString("password");
				boolean isAdmin = rs.getBoolean("admin");

				if ( username.equals(tempUsername) &&
password.equals(tempPassword) ) {


					if(isAdmin) {

						role = "admin";
					}

					else {


						role = "user";
					}
```

```
                                        return true;

                                }

                        }

                        rs.close();
                        connection.connect().close();

                }

                catch (SQLException se) {
                        se.printStackTrace();
                }

                return false;

        }

}
```

12. RecipeModel.java

```
package models;

/*
        This class serves as "a model" for the Recipe Controller. It
        interacts with the database and provides information needed
        by the controller.
        Programmer: Manyanga Marie Ber
*/

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import dao.DaoModel;

public class RecipeModel extends DaoModel{

        private int id;
        private String name;
        private String description;
        private String category;
        private String directions;
```

```
	public RecipeModel() {

	}

	public RecipeModel(int id, String name, String description, String category, String
directions) {

		this.id = id;
		this.name = name;
		this.description = description;
		this.category = category;
		this.directions = directions;
	}

	public RecipeModel(String name, String description, String category, String
directions) {

		this.name = name;
		this.description = description;
		this.category = category;
		this.directions = directions;
	}


	public int getId() {
		return id;
	}

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

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
```

```
	}
	public String getCategory() {
		return category;
	}
	public void setCategory(String category) {
		this.category = category;
	}
	public String getDirections() {
		return directions;
	}
	public void setDirections(String directions) {
		this.directions = directions;
	}


	/*
		This method adds a recipe to the database
	*/
	public void addRecipe(RecipeModel recipeModel) {

		String name = recipeModel.getName();
		String description = recipeModel.getDescription();
		String category = recipeModel.getCategory();
		String directions = recipeModel.getDirections();

		try {

			statement = connection.connect().createStatement();
			String sql = "INSERT INTO cookhelper_recipe (name, description,
category, directions) " +
					"VALUES (" + "'" + name + "', " + "'" + description +
"', " + "'" + category + "', " + "'" + directions + "')";

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}

	/*
		This method changes information related to
```

```
		a recipe in the database
	 */
	public void editRecipe(RecipeModel recipeModel) {

		int id = recipeModel.getId();
		String name = recipeModel.getName();
		String description = recipeModel.getDescription();
		String category = recipeModel.getCategory();
		String directions = recipeModel.getDirections();

		try {

			statement = connection.connect().createStatement();
			String sql = "UPDATE cookhelper_recipe SET name =" + "'" + name
+ "', description = " + "'" + description + "', category = " + "'" + category + "', directions = " +
"'" + directions
				+ "'" + " where id = " + id;

			statement.executeUpdate(sql);
			connection.connect().close();

		}catch (SQLException se) {
			se.printStackTrace();
		}

	}


	/*
		This method gets all recipes from the database
	*/
	public ArrayList<RecipeModel> getRecipes(){

		ArrayList<RecipeModel> recipes = new ArrayList<RecipeModel>();

		try {

			statement = connection.connect().createStatement();
			String sql = "SELECT * FROM cookhelper_recipe";
			ResultSet rs = statement.executeQuery(sql);

			while(rs.next()){

				int id = rs.getInt("id");
```

```
                String name = rs.getString("name");
                String description = rs.getString("description");
                String category = rs.getString("category");
                String directions = rs.getString("directions");

                RecipeModel recipe = new RecipeModel(id, name, description,
category, directions);
                recipes.add(recipe);

            }

            rs.close();
            connection.connect().close();

        }

        catch (SQLException se) {
            se.printStackTrace();
        }

        return recipes;

    }

    /*
        This method gets a recipe from the database
    */
    public RecipeModel getRecipeById(int recipeId){

        RecipeModel recipe = null;

        try {

            statement = connection.connect().createStatement();
            String sql = "SELECT * FROM cookhelper_recipe where id = " +
recipeId;
            ResultSet rs = statement.executeQuery(sql);

            while(rs.next()){

                int id = rs.getInt("id");
                String name = rs.getString("name");
                String description = rs.getString("description");
                String category = rs.getString("category");
```

```
                String directions = rs.getString("directions");

                recipe = new RecipeModel(id, name, description, category,
directions);

            }

            rs.close();
            connection.connect().close();

        }

        catch (SQLException se) {
            se.printStackTrace();
        }

        return recipe;

    }

    /*
        This method deletes a recipe in the database
     */
    public void deleteRecipe(int recipeId) {

        try {

            statement = connection.connect().createStatement();
            String sql = "DELETE FROM cookhelper_recipe where id = " +
recipeId;

            statement.executeUpdate(sql);
            connection.connect().close();

        }catch (Exception se) {
            se.printStackTrace();
        }

    }


}
```

13. AddRecipeView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<GridPane alignment="center" hgap="10" vgap="10"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/11.0.1">

   <Label text="Name: " GridPane.columnIndex="0" GridPane.rowIndex="1">
</Label>
   <TextField fx:id="name" prefHeight="40" GridPane.columnIndex="1"
GridPane.rowIndex="1" />

   <Label text="Description: " GridPane.columnIndex="0" GridPane.rowIndex="2">
</Label>
   <TextField fx:id="description" prefHeight="100.0" prefWidth="200.0"
GridPane.columnIndex="1" GridPane.rowIndex="2" />

   <Label text="Category: " GridPane.columnIndex="0" GridPane.rowIndex="3">
</Label>
   <TextField fx:id="category" prefHeight="40" GridPane.columnIndex="1"
GridPane.rowIndex="3" />

   <Label text="Directions: " GridPane.columnIndex="0" GridPane.rowIndex="4">
</Label>
   <TextField fx:id="directions" prefHeight="200.0" prefWidth="400.0"
GridPane.columnIndex="1" GridPane.rowIndex="4" />

   <Button fx:id="saveButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Save" GridPane.columnIndex="1"
GridPane.rowIndex="5" />
   <Button fx:id="backButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Back" GridPane.columnIndex="2"
GridPane.rowIndex="5" />

  <columnConstraints>
    <ColumnConstraints />
```

```
    <ColumnConstraints maxWidth="471.0" minWidth="336.0" prefWidth="342.0"
/>
    <ColumnConstraints maxWidth="142.0" minWidth="7.0" prefWidth="136.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
  </rowConstraints>


</GridPane>
```

14. EditRecipeView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<GridPane alignment="center" hgap="10" vgap="10"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/11.0.1">

   <Label text="Name: " GridPane.columnIndex="0" GridPane.rowIndex="1">
</Label>
   <TextField fx:id="name" prefHeight="40" GridPane.columnIndex="1"
GridPane.rowIndex="1" />

   <Label text="Description: " GridPane.columnIndex="0" GridPane.rowIndex="2">
</Label>
   <TextField fx:id="description" prefHeight="100.0" prefWidth="200.0"
GridPane.columnIndex="1" GridPane.rowIndex="2" />

   <Label text="Category: " GridPane.columnIndex="0" GridPane.rowIndex="3">
</Label>
   <TextField fx:id="category" prefHeight="40" GridPane.columnIndex="1"
GridPane.rowIndex="3" />
```

```
   <Label text="Directions: " GridPane.columnIndex="0" GridPane.rowIndex="4">
</Label>
   <TextField fx:id="directions" prefHeight="200.0" prefWidth="400.0"
GridPane.columnIndex="1" GridPane.rowIndex="4" />

   <Button fx:id="saveButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Save" GridPane.columnIndex="1"
GridPane.rowIndex="5" />
   <Button fx:id="backButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Back" GridPane.columnIndex="2"
GridPane.rowIndex="5" />

  <columnConstraints>
    <ColumnConstraints />
    <ColumnConstraints maxWidth="471.0" minWidth="336.0" prefWidth="342.0"
/>
    <ColumnConstraints maxWidth="142.0" minWidth="7.0" prefWidth="136.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
  </rowConstraints>


</GridPane>
```

15. LoginView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.paint.*?>

<StackPane prefWidth="231.0" xmlns:fx="http://javafx.com/fxml">
```

```
 <children>
  <StackPane>
   <children>
    <VBox spacing="10.0">
     <children>
      <GridPane>
       <children>
        <Label text="Username:" GridPane.columnIndex="0"
GridPane.rowIndex="0" />
        <Label text="Password:" GridPane.columnIndex="0"
GridPane.rowIndex="1" />
        <TextField fx:id="user" promptText="Use &amp;quot;user&amp;quot; to login"
text="user" GridPane.columnIndex="1" GridPane.rowIndex="0" />
        <TextField fx:id="password" promptText="Use &amp;quot;user&amp;quot; to login"
text="user" GridPane.columnIndex="1" GridPane.rowIndex="1" />
       </children>
       <columnConstraints>
        <ColumnConstraints hgrow="SOMETIMES" maxWidth="148.0"
minWidth="10.0" prefWidth="109.0" />
        <ColumnConstraints hgrow="SOMETIMES" maxWidth="228.0"
minWidth="10.0" prefWidth="189.0" />
       </columnConstraints>
       <rowConstraints>
        <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
       </rowConstraints>
      </GridPane>
      <StackPane prefHeight="-1.0" prefWidth="-1.0">
       <children>
        <Button fx:id="loginButton" alignment="CENTER" defaultButton="true"
mnemonicParsing="false" text="Login" StackPane.alignment="CENTER_RIGHT" />
       </children>
      </StackPane>
     </children>
    </VBox>
   </children>
  </StackPane>
 </children>
 <padding>
  <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
 </padding>
</StackPane>
```

16. MainView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>

<AnchorPane minWidth="315.0" prefHeight="300.0" prefWidth="315.0"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/11.0.1">
  <children>
    <TableView fx:id="tableView" layoutX="42.0" layoutY="40.0"
prefWidth="264.0" tableMenuButtonVisible="false"
AnchorPane.bottomAnchor="10.0" AnchorPane.leftAnchor="11.0"
AnchorPane.rightAnchor="9.0" AnchorPane.topAnchor="40.0">
      <columns>
        <TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="120.0"
text="Recipes" />
      </columns>
<columnResizePolicy>
<TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
</columnResizePolicy>
    </TableView>
    <HBox id="HBox" alignment="CENTER" spacing="5.0"
AnchorPane.leftAnchor="10.0" AnchorPane.rightAnchor="10.0"
AnchorPane.topAnchor="10.0">
      <children>
        <Label text="Search:" />
        <TextField fx:id="search" prefWidth="-1.0" HBox.hgrow="ALWAYS" />
      </children>
      <children>
      <Button fx:id="addButton" defaultButton="true" mnemonicParsing="false"
text="Add"/>
      </children>
    </HBox>
  </children>
</AnchorPane>
```

17. ViewRecipe.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<GridPane alignment="center" hgap="10" vgap="10"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/11.0.1">

   <Label text="Name: " GridPane.columnIndex="0" GridPane.rowIndex="1">
</Label>
   <TextArea fx:id="name" editable="false" prefHeight="40"
GridPane.columnIndex="1" GridPane.rowIndex="1" />

   <Label text="Description: " GridPane.columnIndex="0" GridPane.rowIndex="2">
</Label>
   <TextArea fx:id="description" editable="false" prefHeight="100.0"
prefWidth="200.0" wrapText="true" GridPane.columnIndex="1"
GridPane.rowIndex="2" />

   <Label text="Category: " GridPane.columnIndex="0" GridPane.rowIndex="3">
</Label>
   <TextArea fx:id="category" editable="false" prefHeight="40"
GridPane.columnIndex="1" GridPane.rowIndex="3" />

   <Label text="Directions: " GridPane.columnIndex="0" GridPane.rowIndex="4">
</Label>
   <TextArea fx:id="directions" editable="false" prefHeight="200.0"
prefWidth="400.0" wrapText="true" GridPane.columnIndex="1"
GridPane.rowIndex="4" />

   <Button fx:id="deleteButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Delete" GridPane.columnIndex="1"
GridPane.columnSpan="2" GridPane.halignment="CENTER"
GridPane.rowIndex="5" GridPane.rowSpan="1" />
```

```
   <Button fx:id="editButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Edit" GridPane.columnIndex="1"
GridPane.rowIndex="5" />
   <Button fx:id="backButton" defaultButton="true" mnemonicParsing="false"
prefHeight="61.0" prefWidth="121.0" text="Back" GridPane.columnIndex="2"
GridPane.rowIndex="5" />

  <columnConstraints>
    <ColumnConstraints />
    <ColumnConstraints maxWidth="471.0" minWidth="336.0" prefWidth="342.0"
/>
    <ColumnConstraints maxWidth="142.0" minWidth="7.0" prefWidth="136.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
    <RowConstraints />
  </rowConstraints>


</GridPane>
```