SUBJECT: OBJECT ORIENTED PROGRAMMING

ASSIGNMENT #: LAB ASSIGNMENT:4

GROUP MEMBERS

FATIMA KASHIF (SP24-BSE-144)

ZARA BAIG (SP24-BSE-128)

AYESHA KAMRAN(SP24-BSE-021)

SECTION: A

DUE DATE: 13-12-2024

RECIPE BOOK APPLICATION

HELLOAPPLICATION.JAVA

```
package com.example.recipebookapplication;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.geometry.Orientation;
import javafx.scene.control.Separator;

import java.io.*;

public class HelloApplication extends Application {

    @Override
```

```
        public void start(Stage primaryStage) {
            File file = new File("user.abc");

            GridPane loginPane = new GridPane();
            loginPane.setPadding(new Insets(50));
            Scene loginScene = new Scene(loginPane, 400, 450);


            Image img = new
Image(getClass().getResource("/blueOpacity.png").toExternalForm(
));
            BackgroundImage backgroundImage = new
BackgroundImage( img, BackgroundRepeat.NO_REPEAT,
BackgroundRepeat.NO_REPEAT,
                    BackgroundPosition.CENTER,
BackgroundSize.DEFAULT);

            Background background = new
Background(backgroundImage);
            loginPane.setBackground(background);


            Label label = new Label("Recipe Book");
            loginPane.add(label, 0,0);
            label.setFont(Font.font("Comic Sans MS",32));
            label.setAlignment(Pos.CENTER);
            label.setPadding(new Insets(0,0,20,45));

            Label user = new Label("Enter Username");
            loginPane.add(user, 0,1);
            user.setFont(Font.font("verdana",12));
```

```
            user.setPadding(new Insets(0,0,7,0));


            TextField nameField = new TextField();
            nameField.setPromptText("User Name");
            nameField.setFont(Font.font("verdana", 12));
            loginPane.add(nameField, 0, 2);
            nameField.setMinWidth(250);
            nameField.setMinHeight(32);
            nameField.setPadding( new Insets(5,0,5,15));


            GridPane.setMargin(nameField, new
Insets(0,0,15,0));


                PasswordField passwordField = new
PasswordField();
            passwordField.setPromptText("Password");
            loginPane.add(passwordField, 0, 3);
            passwordField.setPadding( new Insets(5,0,5,15));
            passwordField.setMinHeight(32);


            CheckBox showPasswordCheckBox = new CheckBox("Show
Password");
            loginPane.add(showPasswordCheckBox, 0, 4);
            showPasswordCheckBox.setCursor(Cursor.HAND);
            showPasswordCheckBox.setPadding(new
Insets(8,0,20,0));
            showPasswordCheckBox.setFont(Font.font("Verdana",
12));
            showPasswordCheckBox.setStyle("-fx-body-color:
black; -fx-mark-color: white");
```

```
            Button loginButton = new Button("Login");
            loginPane.add(loginButton, 0, 6);
            loginButton.setFont(Font.font("Courier New", 20));
            loginButton.setTextFill(Color.WHITE);
            loginButton.setStyle("-fx-background-color:
#000000; -fx-background-radius: 50; -fx-border-radius: 50;");
            loginButton.setMinWidth(250);
            loginButton.setMinHeight(40);
            loginButton.setCursor(Cursor.HAND);
            GridPane.setMargin(loginButton, new
Insets(20,0,10,30));

            Button signUpButton = new Button("Sign Up");
            loginPane.add(signUpButton, 0, 7);
            signUpButton.setFont(Font.font("Courier New", 20));
            signUpButton.setTextFill(Color.WHITE);
            signUpButton.setStyle("-fx-background-color:
#000000; -fx-background-radius: 50; -fx-border-radius: 50;");
            signUpButton.setCursor(Cursor.HAND);
            signUpButton.setMinWidth(250);
            signUpButton.setMinHeight(40);
            GridPane.setMargin(signUpButton, new
Insets(10,0,20,10));

            Label loginMessage = new Label();
            loginMessage.setFont(Font.font("verdana", 12));
            loginMessage.setTextFill(Color.RED);
            loginPane.add(loginMessage, 0, 5);
            GridPane.setMargin(loginButton, new Insets(10));
```

```
            loginPane.setAlignment(Pos.TOP_CENTER);

            Stage mainStage = new Stage();
            mainStage.setTitle("Recipe Book");
            mainStage.setResizable(false);
            BorderPane borderPane = new BorderPane();
            Scene appScene = new Scene(borderPane, 900, 700);


            loginButton.setOnAction(e -> {

                String name = nameField.getText().trim();
                String password =
passwordField.getText().trim();
                boolean found = false;
                passwordField.clear();
                nameField.clear();

                try (BufferedReader reader = new
BufferedReader(new FileReader(file))) {
                    String line;

                    while ((line = reader.readLine()) != null)
{

                        String[] parts = line.split(" \\| ");

                        if (parts[0].equals(name) &&
parts[2].equals(password)) {
```

```
                            found = true;
                            mainStage.setScene(appScene);
                            mainStage.setFullScreen(true);
                            mainStage.show();
                            primaryStage.close();
                            break;
                        }
                    }

                    if (!found)
                        loginMessage.setText("*User  Not
Found");

                } catch (IOException ex) {
                    throw new RuntimeException(ex);
                }
            });

            TextField pass = new TextField();
            pass.setPromptText("Password");
            pass.setVisible(false);
            pass.setMinHeight(32);
            pass.setPadding(new Insets(5,0,5,15));
            loginPane.add(pass, 0, 3);
            GridPane.setMargin(loginPane, new
Insets(5,0,5,15));

            showPasswordCheckBox.setOnAction(e -> {
                if (showPasswordCheckBox.isSelected()) {
```

```
                pass.setText(passwordField.getText());
                pass.setVisible(true);
                passwordField.setVisible(false);
            } else {

                passwordField.setText(pass.getText());
                passwordField.setVisible(true);
                pass.setVisible(false);
            }
        });

        signUpButton.setOnAction(e -> {

            Stage signUpStage = new Stage();

            primaryStage.close();

            signUpStage.setTitle("Sign Up");
            signUpStage.setResizable(false);

            GridPane signUpPane = new GridPane();

            signUpPane.setPadding(new Insets(50));

            Scene signUpScene = new Scene(signUpPane, 400,
450);
            signUpStage.setScene(signUpScene);
            signUpStage.show();
```

```
                Image image1 = new
Image(getClass().getResource("/login_background.jpg").toExternal
Form());

                BackgroundImage backgroundImage1 = new
BackgroundImage(image1, BackgroundRepeat.NO_REPEAT,
BackgroundRepeat.NO_REPEAT,

                        BackgroundPosition.CENTER,
BackgroundSize.DEFAULT);

                Background background1 = new
Background(backgroundImage1);

                signUpPane.setBackground(background1);


                Label nameLabel1 = new Label("Enter Name: ");

                nameLabel1.setPadding(new Insets(0,0,5,0));

                nameLabel1.setFont(Font.font("Verdana", 12));

                signUpPane.add(nameLabel1, 0, 0);


                TextField nameField1 = new TextField();

                nameField1.setPromptText("User  Name");

                nameField1.setPadding(new Insets(5,0,5,15));

                nameField1.setMinHeight(32);

                GridPane.setMargin(nameField1, new
Insets(0,0,10,0));

                signUpPane.add(nameField1, 0, 1);


                Label emailLabel = new Label("Enter Email: ");

                emailLabel.setFont(Font.font("Verdana", 12));

                emailLabel.setPadding(new Insets(0,0,5,0));

                signUpPane.add(emailLabel, 0, 2);
```

```
                TextField emailField = new TextField();
                emailField.setPromptText("Email");
                emailField.setPadding(new Insets(5,0,5,15));
                emailField.setMinHeight(32);

                GridPane.setMargin(emailField, new
Insets(0,0,10,0));
                signUpPane.add(emailField, 0, 3);

                Label passwordLabel1 = new Label("Enter
Password: ");
                passwordLabel1.setFont(Font.font("Verdana",
12));
                passwordLabel1.setPadding(new Insets(0,0,5,0));
                signUpPane.add(passwordLabel1, 0, 4);

                TextField passwordField1 = new TextField();
                passwordField1.setPromptText("Password");
                passwordField1.setMinHeight(32);
                passwordField1.setPadding(new
Insets(5,0,5,15));
                GridPane.setMargin(passwordField1, new
Insets(0,0,10,0));
                signUpPane.add(passwordField1, 0, 5);

                Label confirmPasswordLabel = new Label("Confirm
Password: ");
                confirmPasswordLabel.setPadding(new
Insets(0,0,5,0));
                signUpPane.add(confirmPasswordLabel, 0, 6);
```

```
confirmPasswordLabel.setFont(Font.font("Verdana", 12));


                TextField confirmPasswordField = new
TextField();
                confirmPasswordField.setPromptText("Confirm
Password");
                confirmPasswordField.setPadding(new
Insets(5,0,5,15));
                confirmPasswordField.setMinHeight(32);
                GridPane.setMargin(confirmPasswordField, new
Insets(0,0,20,0));
                signUpPane.add(confirmPasswordField, 0, 7);


                Button signInButton = new Button("Sign Up");
                signUpPane.add(signInButton, 0, 9);
                signInButton.setPadding(new Insets(5));
                signInButton.setTextFill(Color.WHITE);
                signInButton.setStyle("-fx-background-color:
LIGHTBLUE; -fx-background-radius: 50; -fx-border-radius: 50; -
fx-font-weight:700;");
                signInButton.setFont(Font.font("Courier New",
15));
                signInButton.setCursor(Cursor.HAND);
                signInButton.setMinWidth(250);
                signInButton.setMinHeight(36);


                GridPane.setMargin(signInButton, new
Insets(7));


                Button backButton = new Button("Back");
                signUpPane.add(backButton, 0, 10);
```

```
                backButton.setPadding(new Insets(5));

                backButton.setTextFill(Color.WHITE);

                backButton.setStyle("-fx-background-color:
LIGHTBLUE; -fx-background-radius: 50; -fx-border-radius: 50; -
fx-font-weight:700;");

                backButton.setFont(Font.font("Courier New",
15));

                backButton.setMinWidth(250);

                backButton.setMinHeight(36);

                backButton.setCursor(Cursor.HAND);

                GridPane.setMargin(backButton, new Insets(7));


                Label messageLabel = new Label();

                messageLabel.setTextFill(Color.RED);

                signUpPane.add(messageLabel, 0, 8);


                signUpPane.setAlignment(Pos.TOP_CENTER);


                backButton.setOnAction(event ->
signUpStage.setScene(loginScene));


                signInButton.setOnAction(event -> {


                    primaryStage.close();


                    String name = nameField1.getText().trim();

                    String email = emailField.getText().trim();

                    String password =
passwordField1.getText().trim();

                    String confirmPassword =
confirmPasswordField.getText().trim();
```

```
                    if (name.isEmpty() || email.isEmpty() ||
password.isEmpty() || confirmPassword.isEmpty()) {
                        messageLabel.setText("*Fields cannot be
empty!");
                        return;
                    }

                    if (!password.equals(confirmPassword)) {
                        messageLabel.setText("*Passwords do not
match!");
                        return;
                    }

                    try (BufferedReader reader = new
BufferedReader(new FileReader(file))) {
                        String line;
                        boolean found = false;

                        while ((line = reader.readLine()) !=
null) {
                            String[] parts = line.split(" \\|
");

                            if (parts[0].equals(name)) {
                                messageLabel.setText("*Username
Already Exists");
                                found = true;
                                break;
                            }
                            if (parts[1].equals(email)) {
```

```
                                messageLabel.setText("*Email
Already Registered");
                                found = true;
                                break;
                            }

                            if (parts[2].equals(password)) {
                                messageLabel.setText("*Password
Already Exists");
                                found = true;
                                break;
                            }
                        }

                        if (!found) {
                            try (BufferedWriter writer = new
BufferedWriter(new FileWriter(file, true))) {
                                writer.write(name + " | " +
email + " | " + password);
                                writer.newLine();

                                try {
                                    Thread.sleep(800);
                                } catch (InterruptedException
ex) {
                                    throw new
RuntimeException(ex);
                                }

messageLabel.setText("Registration Successful!");
```

```
                                mainStage.setScene(appScene);
                                mainStage.setFullScreen(true);
                                mainStage.show();
                                signUpStage.close();

                            } catch (IOException ex) {
                                throw new RuntimeException(ex);
                            }
                        }
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }
                });
            });

            Image image = new
Image(getClass().getResource("borderpanetop.jpg").toExternalForm
());
            ImageView imageView = new ImageView(image);
            imageView.setFitWidth(955);
            imageView.setFitHeight(625);
            BorderPane.setAlignment(imageView, Pos.TOP_CENTER);


            Pane leftRegion = new Pane();
            leftRegion.setBackground(new Background(new
BackgroundFill(Color.LIGHTBLUE, CornerRadii.EMPTY,
Insets.EMPTY)));
            leftRegion.setPrefWidth(290);

leftRegion.prefHeightProperty().bind(borderPane.heightProperty()
);
```

```
            Button addRecipe = new Button("ADD A RECIPE");
            Button browseRecipe = new Button("BROWSE ALL
RECIPES");

            browseRecipe.setLayoutX(40);
            browseRecipe.setLayoutY(80);
            browseRecipe.setMinHeight(40);
            browseRecipe.setMinWidth(165);
            browseRecipe.setFont(Font.font("Comic Sans MS",
16));
            browseRecipe.setTextFill(Color.WHITE);
            browseRecipe.setStyle("-fx-background-color:
#000000;");
            browseRecipe.setCursor(Cursor.HAND);

            addRecipe.setLayoutX(55);
            addRecipe.setLayoutY(135);
            addRecipe.setMinHeight(32);
            addRecipe.setMinWidth(165);
            addRecipe.setFont(Font.font("Comic Sans MS", 16));
            addRecipe.setTextFill(Color.WHITE);
            addRecipe.setStyle("-fx-background-color:
#000000;");
            addRecipe.setCursor(Cursor.HAND);

            leftRegion.getChildren().addAll(browseRecipe,
addRecipe);

            borderPane.setLeft(leftRegion);
```

```
            Pane rightPane = new Pane();
            rightPane.setBackground(new Background(new
BackgroundFill(Color.LIGHTBLUE, CornerRadii.EMPTY,
Insets.EMPTY)));
            rightPane.setPrefWidth(290);

rightPane.prefHeightProperty().bind(borderPane.heightProperty())
;
            borderPane.setRight(rightPane);


            Button logoutButton = new Button("LOGOUT");
            rightPane.getChildren().add(logoutButton);


            logoutButton.setLayoutX(100);
            logoutButton.setLayoutY(80);
            logoutButton.setMinWidth(110);
            logoutButton.setMinHeight(35);
            logoutButton.setFont(Font.font("Comic Sans MS",
16));
            logoutButton.setTextFill(Color.WHITE);
            logoutButton.setStyle("-fx-background-color:
#000000;");
            logoutButton.setCursor(Cursor.HAND);


            logoutButton.setOnAction(e ->
mainStage.setScene(loginScene));


            ComboBox<String> findRecipe = new ComboBox<>();
            findRecipe.setPromptText("What would you like
today?");
```

```
            findRecipe.setEditable(true);
            findRecipe.setMinWidth(330);
            findRecipe.setMinHeight(38);

            Button findButton = new Button("FIND");
            findButton.setMinHeight(32);
            findButton.setMinWidth(110);
            findButton.setFont(Font.font("Comic Sans MS", 16));
            findButton.setTextFill(Color.WHITE);
            findButton.setStyle("-fx-background-color:
#000000;");
            findButton.setCursor(Cursor.HAND);

            HBox hbox = new HBox();
            hbox.setSpacing(50);
            hbox.getChildren().addAll(findRecipe,findButton);
            hbox.setAlignment(Pos.CENTER);

            hbox.setPadding(new Insets(100));

            VBox vBox = new VBox();
            vBox.getChildren().addAll(imageView,hbox);

            borderPane.setCenter(vBox);

            Stage addingStage = new Stage();
            BorderPane borderPane1 = new BorderPane();
            Scene addingScene = new Scene(borderPane1,
850,550);
```

```
            GridPane addPane = new GridPane();

            Separator separator = new
Separator(Orientation.HORIZONTAL);

            Separator separator1 = new
Separator(Orientation.HORIZONTAL);

            Separator separator2 = new
Separator(Orientation.HORIZONTAL);

            Separator separator3 = new
Separator(Orientation.HORIZONTAL);

            Separator separator4 = new
Separator(Orientation.HORIZONTAL);

            Separator separator5 = new
Separator(Orientation.HORIZONTAL);

            Separator separator6 = new
Separator(Orientation.HORIZONTAL);

            Separator separator7 = new
Separator(Orientation.HORIZONTAL);


            Label addTitle = new Label("ADD A RECIPE");

            addTitle.setFont(Font.font("Comic Sans MS", 35));


            Label recipeName = new Label("Name of Recipe: ");

            TextField recipeNameField = new TextField();

            recipeName.setFont(Font.font("Courier New", 18));


            Label cusinineLabel = new Label("Type of Cuisine");

            cusinineLabel.setFont(Font.font("Courier New",
18));


            ComboBox<String> cuisine = new ComboBox<>();
```

```
cuisine.getItems().addAll("Continental","Pakistani", "Arab",
"Chinese", "Pan Asian", "English", "European");
            cuisine.setValue("Select an option");


            Label classLabel = new Label("Classify Dish As");
            classLabel.setFont(Font.font("Courier New", 18));


            ComboBox<String> classification = new ComboBox<>();
            classification.getItems().addAll("Breakfast",
"Brunch", "Lunch", "Snack", "Dinner");
            classification.setValue("Select an option");


            Label requires = new Label("Requires");
            requires.setFont(Font.font("Courier New", 18));


            ComboBox<String> requiresMethod = new ComboBox<>();

requiresMethod.getItems().addAll("Baking","Grilling", "Cooking",
"No Bake", "Freezing", "Roasting");
            requiresMethod.setValue("Select an option");


            TableView table = new TableView();
            ObservableList<Ingredients> ingredientList =
FXCollections.observableArrayList();


            TableColumn category = new
TableColumn<>("Category");
            TableColumn ingredients = new
TableColumn<>("Ingredients");
            TableColumn measurement = new
TableColumn<>("Measurements");
```

```
            TableColumn quantity = new
TableColumn<>("Quantity");


            category.setCellValueFactory(new
PropertyValueFactory<>("category"));
            ingredients.setCellValueFactory(new
PropertyValueFactory<>("name"));
            measurement.setCellValueFactory(new
PropertyValueFactory<>("measurement"));
            quantity.setCellValueFactory(new
PropertyValueFactory<>("quantity"));


            table.getColumns().addAll(category, ingredients,
measurement, quantity);
            table.setItems(ingredientList);


            ComboBox<String> categoryComboBox = new
ComboBox<>();
            categoryComboBox.getItems().addAll("Spices",
"Herbs", "Vegetables", "Fruits", "Dairy", "Meat", "Poultry",
"Seafood", "Grains and Legumes", "Oils and Fats", "Nuts and
Seeds", "Condiments", "Sweeteners", "Liquids");
            categoryComboBox.setPromptText("Select Category");


            ComboBox<String> ingredientComboBox = new
ComboBox<>();
            ingredientComboBox.setPromptText("Select
Ingredient");


            ComboBox<String> measurementComboBox = new
ComboBox<>();
            measurementComboBox.getItems().addAll("grams",
"kilograms", "teaspoons", "tablespoons", "cups", "ounces",
"millilitres", "liters");
```

```
            measurementComboBox.setPromptText("Select
Measurement");

            TextField quantityField = new TextField();
            quantityField.setPromptText("Quantity");

            TextField quantityOfIngredients = new TextField();
            quantityOfIngredients.setPromptText("Enter
Quantity");

            category.setMinWidth(200);
            ingredients.setMinWidth(200);
            measurement.setMinWidth(150);
            quantity.setMinWidth(100);

            Button addButton = new Button("Add Ingredient");

            Label instruction = new Label("Instructions");
            instruction.setFont(Font.font("Courier New", 18));

            TextArea instructionsTextarea = new TextArea();
            instructionsTextarea.setPromptText("Add all
instructions here.....");
            instructionsTextarea.setFont(Font.font("Verdana",
13));
            instructionsTextarea.setPadding(new Insets(5));
            instructionsTextarea.setLayoutX(50);

            Image image1 = new
Image(getClass().getResource("/right-
add.jpg").toExternalForm());
```

```
            ImageView imageView1 = new ImageView(image1);

imageView1.fitHeightProperty().bind(borderPane1.heightProperty()
);
            imageView1.setFitWidth(490);

            Button backButton = new Button("BACK TO MAIN");
            backButton.setPadding(new Insets(5));
            backButton.setLayoutY(60);
            backButton.setFont(Font.font("Comic Sans MS", 16));
            backButton.setTextFill(Color.WHITE);
            backButton.setStyle("-fx-background-color:
#000000;");
            backButton.setCursor(Cursor.HAND);

            backButton.setOnAction(e -> {
                mainStage.setScene(appScene);
                addingStage.close();
                mainStage.show();
            });

            ScrollPane scrollPane = new ScrollPane();
            scrollPane.setContent(addPane);
            scrollPane.setStyle("-fx-background: LIGHTBLUE;");

            borderPane1.setRight(backButton);
            borderPane1.setLeft(imageView1);
```

```
            borderPane1.setBackground(new Background(new
BackgroundFill(Color.LIGHTBLUE, CornerRadii.EMPTY,
Insets.EMPTY)));


            Label imageSelectTitle = new Label("Image");
            imageSelectTitle.setFont(Font.font("Courier New",
18));


            Label imageSelect = new Label("Select an image");
            imageSelect.setFont(Font.font("Verdana", 16));
            imageSelect.setAlignment(Pos.BOTTOM_LEFT);
            imageSelect.setPadding(new Insets(0,0,10,0));


            VBox imageContainer = new VBox();


            Button imageButton = new Button("Choose Image");
            imageButton.setFont(Font.font("Courier New", 15));
            imageButton.setTextFill(Color.WHITE);
            imageButton.setStyle("-fx-background-color:
#000000;");
            imageButton.setCursor(Cursor.HAND);


            imageButton.setOnAction(e -> {
                FileChooser fileChooser = new FileChooser();
                fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Image Files", ".png", ".jpg",
".jpeg", ".gif"));


                File selectedFile =
fileChooser.showOpenDialog(addingStage);
```

```
                if (selectedFile != null) {
                    Image selectedImage = new
Image(selectedFile.toURI().toString());
                    ImageView imageView3 = new
ImageView(selectedImage);
                    imageView3.setFitWidth(200);
                    imageView3.setFitHeight(150);

                    HBox imageBox = new HBox(imageView3);
                    imageBox.setAlignment(Pos.TOP_CENTER);
                    imageBox.setPadding(new Insets(0,0,10,0));
                    imageContainer.getChildren().add(imageBox);
                }
            });

            imageContainer.getChildren().addAll(imageSelect,
imageButton);
            imageContainer.setPrefSize(200, 200);
            imageContainer.setAlignment(Pos.CENTER);
            imageContainer.setBorder(new Border(new
BorderStroke(
                    Color.WHITE,
                    BorderStrokeStyle.SOLID,
                    new CornerRadii(0),
                    BorderStroke.DEFAULT_WIDTHS
            )));

            addRecipe.setOnAction(e -> {

                addingStage.setFullScreen(true);
```

```
addingStage.setTitle("Add New Recipe");
addingStage.setScene(addingScene);
mainStage.close();
addingStage.show();

addPane.add(addTitle,0,0);
addPane.add(separator,0,1);
addPane.add(separator1,1,1);
addPane.add(recipeName, 0,2);
addPane.add(recipeNameField,1,2);
addPane.add(cusinineLabel,0,3);
addPane.add(cuisine,1,3);
addPane.add(classLabel, 0,4);
addPane.add(classification,1,4);
addPane.add(requires,0,5);
addPane.add(requiresMethod,1,5);
addPane.add(separator2,0,6);
addPane.add(separator3,1,6);
addPane.add(table,0,7);
addPane.add(separator4,0,8);
addPane.add(separator5, 1,8);
addPane.add(instruction, 0,9);
addPane.add(instructionsTextarea,0,10);
addPane.add(separator6,0,11);
addPane.add(separator7, 1,11);
addPane.add(imageSelectTitle,0,12);
addPane.add(imageContainer,0,13);
```

```
            addPane.setVgap(20);
            addPane.setHgap(10);
            borderPane1.setCenter(scrollPane);
            addPane.setAlignment(Pos.BASELINE_RIGHT);
            addPane.setPadding(new Insets(50));

        });

        Stage browsingStage = new Stage();
        GridPane browsingPane = new GridPane();
        Scene browsingScene = new Scene(browsingPane,
850,500);


        browseRecipe.setOnAction(e -> {

            browsingStage.setResizable(false);
            browsingStage.setTitle("All Recipes");
            browsingStage.setFullScreen(true);
            browsingStage.setScene(browsingScene);
            browsingStage.show();
            mainStage.close();
        });

        primaryStage.setTitle("Recipe Book Login");
        primaryStage.setResizable(false);
        primaryStage.setScene(loginScene);
        primaryStage.show();
    }
```

```
        public static void main(String[] args) {
            launch();
        }
    }
```

==========

INGREDIENTS.JAVA

```
    package com.example.recipebookapplication;

    import javafx.collections.FXCollections;
    import javafx.collections.ObservableList;

    public class Ingredients {

        public static final ObservableList<String> SPICES =
FXCollections.observableArrayList(
                "Salt", "Pepper", "Cinnamon", "Paprika", "Red
Chili",
                "Turmeric", "Cloves", "Cardamom", "Nutmeg",
"Cumin",
                "Coriander Seeds", "Mustard Seeds", "Bay Leaf",
"Fenugreek",
                "Star Anise", "Saffron", "Curry Powder", "Garam
Masala");

        public static final ObservableList<String> HERBS =
FXCollections.observableArrayList(
                "Basil", "Parsley", "Thyme", "Rosemary",
"Oregano",
```

```
                "Coriander", "Mint", "Cilantro", "Lemongrass",
"Marjoram");


        public static final ObservableList<String> VEGETABLES =
FXCollections.observableArrayList(
                "Carrot", "Potato", "Onion", "Tomato",
"Ginger",
                "Garlic", "Broccoli", "Cauliflower",
"Cucumber",
                "Bell Peppers", "Spinach", "Capsicum",
                "Lettuce", "Cabbage", "Eggplant", "Green
Beans",
                "Peas", "Mushrooms", "Sweet Corn");


        public static final ObservableList<String> FRUITS =
FXCollections.observableArrayList(
                "Apple", "Banana", "Orange", "Grapes", "Mango",
                "Pineapple", "Strawberries", "Blueberries",
                "Raspberries", "Watermelon", "Cantaloupe",
                "Peach", "Pear", "Plum", "Cherry",
"Pomegranate",
                "Kiwi", "Avocado", "Lemon", "Lime");


        public static final ObservableList<String> DAIRY =
FXCollections.observableArrayList(
                "Milk", "Cheese", "Butter", "Yogurt", "Cream",
                "Ghee", "Paneer", "Sour Cream", "Condensed
Milk",
                "Evaporated Milk", "Whipped Cream",
"Buttermilk");
```

```
        public static final ObservableList<String> MEAT =
FXCollections.observableArrayList(
                "Chicken", "Beef", "Lamb", "Goat", "Veal",
"Bacon", "Salami", "Sausage");


        public static final ObservableList<String> POULTRY =
FXCollections.observableArrayList(
                "Chicken Eggs", "Turkey Eggs", "Duck Eggs",
"Goose Eggs", "Quail Eggs");


        public static final ObservableList<String> SEAFOOD =
FXCollections.observableArrayList(
                "Shrimp", "Prawns", "Crab", "Lobster", "Clams",
                "Mussels", "Scallops", "Squid", "Octopus",
"Salmon",
                "Tuna", "Cod", "Sardines");


        public static final ObservableList<String>
GRAINS_AND_LEGUMES = FXCollections.observableArrayList(
                "Rice", "Wheat", "Oats", "Barley", "Quinoa",
                "Corn", "Chickpeas", "Lentils", "Black Beans",
                "Kidney Beans", "Pinto Beans", "Peas",
                "Soybeans", "Millet");


        public static final ObservableList<String>
OILS_AND_FATS = FXCollections.observableArrayList(
                "Olive Oil", "Vegetable Oil", "Canola Oil",
                "Coconut Oil", "Sesame Oil", "Sunflower Oil",
                "Butter", "Ghee");


        public static final ObservableList<String>
NUTS_AND_SEEDS = FXCollections.observableArrayList(
```

```
                "Almonds", "Walnuts", "Cashews", "Pistachios",
                "Hazelnuts", "Peanuts",
                "Chia Seeds", "Flaxseeds", "Sesame Seeds",
"Sunflower Seeds",
                "Pumpkin Seeds", "Pine Nuts");

        public static final ObservableList<String> CONDIMENTS =
FXCollections.observableArrayList(
                "Ketchup", "Mustard", "Mayonnaise", "Soy
Sauce",
                "Fish Sauce", "Hot Sauce",
                "Vinegar", "Barbecue Sauce", "Honey", "Jam",
                "Pickles", "Chutney", "Tahini", "Hummus");

        public static final ObservableList<String> SWEETENERS =
FXCollections.observableArrayList(
                "Sugar", "Brown Sugar", "Honey", "Maple Syrup",
                "Corn Syrup", "Artificial Sweeteners");

        public static final ObservableList<String> LIQUIDS =
FXCollections.observableArrayList(
                "Water", "Milk", "Lemon Juice", "Vinegar", "Soy
Sauce",
                "Fish Sauce", "Broth", "Cream", "Olive Oil",
"Coconut Milk");

        public static ObservableList<String>
getIngredientsByCategory(String category) {
            switch (category) {
                case "Spices":
                    return SPICES;
```

```
case "Herbs":
    return HERBS;
case "Vegetables":
    return VEGETABLES;
case "Fruits":
    return FRUITS;
case "Dairy":
    return DAIRY;
case "Meat":
    return MEAT;
case "Poultry":
    return POULTRY;
case "Seafood":
    return SEAFOOD;
case "Grains and Legumes":
    return GRAINS_AND_LEGUMES;
case "Oils and Fats":
    return OILS_AND_FATS;
case "Nuts and Seeds":
    return NUTS_AND_SEEDS;
case "Condiments":
    return CONDIMENTS;
case "Sweeteners":
    return SWEETENERS;
case "Liquids":
    return LIQUIDS;
default:
    return FXCollections.observableArrayList();
```

```
                }

            }

        }
```