

Subject: OOps

Assignment #: lab assignment:4

Fatima kashif (sp24-bse-144)

Zara baig(sp24-bse-128)

Ayesha Kamran (sp24-bse-021)

Section:A

Due date:20-12-2024

HelloApplication:

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.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

import java.io.\*;

public class HelloApplication extends Application {

private ObservableList<Ingredient> ingredients = FXCollections.observableArrayList();

private static final List<String> CATEGORIES = Arrays.asList(

"Spices", "Herbs", "Vegetables", "Fruits", "Dairy", "Meat", "Poultry",

"Liquids", "Pasta and Noodles", "Dry Goods", "Bakery", "Seafood",

"Grains and Legumes", "Oils and Fats", "Nuts and Seeds", "Condiments", "Sweeteners");

private static final List<String> MEASUREMENTS = Arrays.asList(

"grams", "kilograms","quantity", "teaspoons", "tablespoons", "cups", "ounces", "millilitres", "liters"

);

File recipeFile = new File("recipe.abc");

Stage mainStage = new Stage();

@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);

mainStage.setTitle("Recipe Book");

mainStage.setResizable(true);

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").

toExternalForm());

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);

Button removeRecipeButton = new Button("REMOVE RECIPE");

removeRecipeButton.setLayoutX(55);

removeRecipeButton.setLayoutY(190);

removeRecipeButton.setMinHeight(32);

removeRecipeButton.setMinWidth(165);

removeRecipeButton.setFont(Font.font("Comic Sans MS", 16));

removeRecipeButton.setTextFill(Color.WHITE);

removeRecipeButton.setStyle("-fx-background-color: #000000;");

removeRecipeButton.setCursor(Cursor.HAND);

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

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("Select a recipe");

findRecipe.setEditable(false);

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","French", "English","American",

"European", "Italian");

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", "Lunch", "Snack","Dessert", "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","Frying", "Cooking", "No Bake", "Freezing", "Roasting");

requiresMethod.setValue("Select an option");

TableView<Ingredient> table = new TableView<>();

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

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

TextField quantityField = new TextField();

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

quantityField.setPrefWidth(75);

Button addButton = new Button("Add");

addButton.setPrefWidth(60);

addButton.setPadding(new Insets(3));

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

addButton.setTextFill(Color.WHITE);

addButton.setStyle("-fx-background-color: #000000;");

addButton.setCursor(Cursor.HAND);

TableColumn<Ingredient, String> categoryCol = new TableColumn<>("Category");

categoryCol.setCellValueFactory(new PropertyValueFactory<>("category"));

TableColumn<Ingredient, String> ingredientCol = new TableColumn<>("Ingredient");

ingredientCol.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<Ingredient, String> measurementCol = new TableColumn<>("Measurement");

measurementCol.setCellValueFactory(new PropertyValueFactory<>("measurement"));

TableColumn<Ingredient, String> quantityCol = new TableColumn<>("Quantity");

quantityCol.setCellValueFactory(new PropertyValueFactory<>("quantity"));

table.getColumns().addAll(categoryCol, ingredientCol,quantityCol, measurementCol);

table.setItems(ingredients);

categoryCol.setMinWidth(190);

ingredientCol.setMinWidth(190);

measurementCol.setMinWidth(150);

quantityCol.setMinWidth(100);

categoryComboBox.setItems(FXCollections.observableArrayList(CATEGORIES));

categoryComboBox.setPromptText("Select Category");

categoryComboBox.setOnAction(e -> {

String selectedCategory = categoryComboBox.getValue();

if (selectedCategory != null) {

List<String> categoryIngredients = getIngredientsForCategory(selectedCategory);

ingredientComboBox.setItems(FXCollections.observableArrayList(categoryIngredients));

}

});

ingredientComboBox.setPromptText("Select Ingredient");

measurementComboBox.setItems(FXCollections.observableArrayList(MEASUREMENTS));

measurementComboBox.setPromptText("Select Measurement");

quantityField.setPromptText("Quantity");

addButton.setOnAction(e ->{

String category = categoryComboBox.getValue();

String ingredient = ingredientComboBox.getValue();

String measurement = measurementComboBox.getValue();

String quantity = quantityField.getText();

if (category != null && ingredient != null && measurement != null && !quantity.isEmpty()) {

Ingredient newIngredient = new Ingredient(category, ingredient, measurement, quantity);

ingredients.add(newIngredient);

try(BufferedWriter recipeWriter = new BufferedWriter(new FileWriter(recipeFile, true))) {

recipeWriter.write(category + " ");

recipeWriter.write(ingredient + " ");

recipeWriter.write(quantity + " ");

recipeWriter.write(measurement);

recipeWriter.newLine();

} catch (IOException ex) {

throw new RuntimeException(ex);

}

table.setItems(ingredients);

table.refresh();

categoryComboBox.setValue(null);

ingredientComboBox.setValue(null);

measurementComboBox.setValue(null);

quantityField.clear();

} else {

Alert alertBox = new Alert(Alert.AlertType.WARNING);

alertBox.setContentText("Fields Can Not Be Left Empty!");

alertBox.showAndWait();

}

});

Button removeButton = new Button("Remove");

removeButton.setMinHeight(25);

removeButton.setMinWidth(65);

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

removeButton.setTextFill(Color.WHITE);

removeButton.setStyle("-fx-background-color: #000000;");

removeButton.setCursor(Cursor.HAND);

removeButton.setLayoutX(250);

removeButton.setPadding(new Insets(8));

removeButton.setOnAction(e -> {

Ingredient selectedIngredient = table.getSelectionModel().getSelectedItem();

if (selectedIngredient != null) {

ingredients.remove(selectedIngredient);

table.refresh();

} else {

Alert alertBox = new Alert(Alert.AlertType.WARNING);

alertBox.setContentText("Please select an ingredient to remove!");

alertBox.showAndWait();

}

});

VBox tableBox = new VBox(10);

tableBox.setPadding(new Insets(20));

HBox inputBox = new HBox(10);

inputBox.getChildren().addAll(categoryComboBox, ingredientComboBox, measurementComboBox, quantityField, addButton);

tableBox.getChildren().addAll(inputBox, table);

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);

final String[] selectedImagePath = {null};

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) {

selectedImagePath[0] = selectedFile.getAbsolutePath();

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

)));

Button saveButton = new Button("Save");

saveButton.setPadding(new Insets(10));

saveButton.setFont(Font.font("Courier New", 20));

saveButton.setTextFill(Color.WHITE);

saveButton.setStyle("-fx-background-color: #000000;");

saveButton.setCursor(Cursor.HAND);

saveButton.setMinWidth(100);

saveButton.setMinHeight(12);

saveButton.setOnAction(e -> {

String steps = instructionsTextarea.getText();

String name = recipeNameField.getText();

String cuisineType = cuisine.getValue();

String classType = classification.getValue();

String require = requiresMethod.getValue();

if (steps.isEmpty() || name.isEmpty() || cuisineType.isEmpty() || classType.isEmpty() || require.isEmpty()){

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setContentText("Cannot Save Unless All Fields Are Filled!");

alert.showAndWait();

}

else {

try (BufferedWriter recipeWriter = new BufferedWriter(new FileWriter(recipeFile, true))) {

recipeWriter.write("START\_RECIPE");

recipeWriter.newLine();

recipeWriter.write("|NAME|" + name);

recipeWriter.newLine();

recipeWriter.write("|CUISINE|" + cuisineType);

recipeWriter.newLine();

recipeWriter.write("|MEAL\_TYPE|" + classType);

recipeWriter.newLine();

recipeWriter.write("|COOKING\_METHOD|" + require);

recipeWriter.newLine();

recipeWriter.write("|IMAGE\_PATH|" + (selectedImagePath[0] != null ? selectedImagePath[0] : "No Image"));

recipeWriter.newLine();

recipeWriter.write("|INGREDIENTS|");

recipeWriter.newLine();

for (Ingredient ingredient : ingredients) {

recipeWriter.write(ingredient.getName() + ";" + ingredient.getQuantity() + " " + ingredient.getMeasurement());

recipeWriter.newLine();

}

recipeWriter.write("|INSTRUCTIONS|");

recipeWriter.newLine();

recipeWriter.write(steps);

recipeWriter.newLine();

recipeWriter.write("END\_RECIPE");

recipeWriter.newLine();

recipeWriter.newLine();

} catch (IOException ex) {

throw new RuntimeException(ex);

}

instructionsTextarea.clear();

recipeNameField.clear();

cuisine.setValue(null);

classification.setValue(null);

requiresMethod.setValue(null);

table.getItems().clear();

addingStage.close();

mainStage.setScene(appScene);

mainStage.show();

Alert confirm = new Alert(Alert.AlertType.INFORMATION);

confirm.setContentText("Recipe Successfully Added!");

confirm.showAndWait();

}

});

browseRecipe.setOnAction(e -> {

Stage browsingStage = new Stage();

BorderPane borderPane2 = new BorderPane();

Scene browsingScene = new Scene(borderPane2, 850, 500);

Image image2 = new Image(getClass().getResourceAsStream("/browse\_left.jpg"));

ImageView imageView2 = new ImageView(image2);

imageView2.fitHeightProperty().bind(borderPane2.heightProperty());

imageView2.setFitWidth(400);

Image image3 = new Image(getClass().getResourceAsStream("/browse\_right.jpg"));

ImageView imageView3 = new ImageView(image3);

imageView3.fitHeightProperty().bind(borderPane2.heightProperty());

imageView3.setFitWidth(380);

ScrollPane scrollPane1 = new ScrollPane();

GridPane gridPane = new GridPane();

gridPane.setHgap(20);

gridPane.setVgap(20);

gridPane.setPadding(new Insets(20));

scrollPane1.setContent(gridPane);

scrollPane1.setFitToWidth(true);

borderPane2.setCenter(scrollPane1);

borderPane2.setLeft(imageView2);

borderPane2.setRight(imageView3);

browsingStage.setScene(browsingScene);

browsingStage.setTitle("Browse All Recipes");

Button browseBackButton = new Button("Back to Main");

browseBackButton.setFont(Font.font("Comic Sans MS", 16));

browseBackButton.setStyle("-fx-background-color: #000000; -fx-text-fill: white;");

browseBackButton.setCursor(Cursor.HAND);

browseBackButton.setOnAction(event -> {

browsingStage.close();

mainStage.setScene(appScene);

mainStage.setFullScreen(true);

});

gridPane.add(browseBackButton, 0, 0);

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

loadRecipes(gridPane);

browsingStage.setFullScreen(true);

browsingStage.show();

});

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(tableBox,0,7);

addPane.add(removeButton, 0, 8);

addPane.add(separator4,0,9);

addPane.add(separator5, 1,9);

addPane.add(instruction, 0,10);

addPane.add(instructionsTextarea,0,11);

addPane.add(separator6,0,12);

addPane.add(separator7, 1,12);

addPane.add(imageSelectTitle,0,13);

addPane.add(imageContainer,0,14);

addPane.add(saveButton,1,15);

addPane.setVgap(20);

addPane.setHgap(10);

borderPane1.setCenter(scrollPane);

addPane.setAlignment(Pos.BASELINE\_RIGHT);

addPane.setPadding(new Insets(50));

});

findButton.setOnAction(e -> {

String selectedRecipe = findRecipe.getValue();

if (selectedRecipe != null) {

FindRecipe.showRecipeDetails(selectedRecipe, mainStage);

} else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setContentText("Please select a recipe first!");

alert.showAndWait();

}

});

removeRecipeButton.setOnAction(e -> {

RemoveRecipe.show();

});

primaryStage.setTitle("Recipe Book Login");

primaryStage.setResizable(false);

primaryStage.setScene(loginScene);

primaryStage.show();

FindRecipe.loadRecipesToComboBox(findRecipe);

}

private List<String> getIngredientsForCategory(String category) {

switch (category) {

case "Spices":

return Arrays.asList("Salt", "Pepper", "Cinnamon", "Paprika", "Red Chili", "Turmeric", "Cloves",

"Cardamom", "Nutmeg", "Cumin", "Sumac", "Za'atar", "Saffron", "Fenugreek",

"Coriander Powder", "Anise", "Star Anise", "Garam Masala", "Black Peppercorns",

"Fennel Seeds", "White Pepper", "Bay Leaves");

case "Herbs":

return Arrays.asList("Basil", "Parsley", "Thyme", "Rosemary", "Oregano", "Coriander", "Mint",

"Cilantro", "Lemongrass", "Marjoram", "Sage", "Dill", "Bay Leaves",

"Chives", "Tarragon");

case "Vegetables":

return Arrays.asList("Carrot", "Potato", "Onion", "Tomato", "Ginger", "Garlic", "Broccoli",

"Cauliflower", "Cucumber", "Bell Peppers", "Spinach", "Cabbage",

"Lettuce", "Capsicum","Pickles", "Eggplant", "Peas", "Bitter Gourd", "Mushrooms",

"Pak Choi", "Spring Onions", "Radish");

case "Fruits":

return Arrays.asList("Apple", "Banana", "Orange", "Grapes", "Mango", "Pineapple", "Strawberries",

"Blueberries", "Raspberries", "Watermelon", "Dates", "Figs", "Pomegranate",

"Guava", "Papaya", "Melon", "Kiwi", "Cantaloupe", "Lemon", "Tangerine");

case "Dairy":

return Arrays.asList("Milk", "Cheese Slices", "Butter", "Yogurt", "Cream", "Ghee", "Paneer",

"Sour Cream", "Condensed Milk", "Evaporated Milk", "Lassi",

"Ricotta", "Mozzarella","Cheddar", "Parmesan", "Feta Cheese", "Kefir", "Labneh");

case "Meat":

return Arrays.asList( "Chicken", "Beef", "Lamb", "Goat","Fish", "Veal", "Salami", "Sausage",

"Mutton", "Duck", "Quail", "Rabbit");

case "Poultry":

return Arrays.asList("Chicken Eggs", "Turkey Eggs", "Duck Eggs", "Goose Eggs", "Quail Eggs");

case "Seafood":

return Arrays.asList("Shrimp", "Prawns", "Crab", "Lobster", "Clams", "Mussels",

"Scallops", "Squid", "Octopus", "Salmon");

case "Grains and Legumes":

return Arrays.asList( "Rice", "Wheat", "Oats", "Barley", "Quinoa", "Corn", "Chickpeas",

"Lentils", "Black Beans", "Kidney Beans", "Couscous", "Freekeh",

"Bulgur", "Farro", "Millet", "Split Peas", "Moong Dal", "Chana Dal");

case "Oils and Fats":

return Arrays.asList( "Oil", "Olive Oil", "Vegetable Oil", "Canola Oil", "Coconut Oil", "Sesame Oil",

"Sunflower Oil", "Butter", "Ghee", "Mustard Oil", "Palm Oil",

"Avocado Oil", "Clarified Butter");

case "Nuts and Seeds":

return Arrays.asList( "Almonds", "Walnuts", "Cashews", "Pistachios", "Hazelnuts", "Peanuts",

"Chia Seeds", "Flaxseeds", "Sesame Seeds", "Sunflower Seeds",

"Pumpkin Seeds", "Pine Nuts", "Poppy Seeds", "Chestnuts");

case "Condiments":

return Arrays.asList( "Ketchup", "Mustard", "Mayonnaise", "Soy Sauce", "Fish Sauce", "Hot Sauce",

"Vinegar", "Barbecue Sauce", "Honey", "Jam", "Pickle", "Chutney",

"Harissa", "Tartar Sauce", "Hoisin Sauce", "Tahini", "Pesto");

case "Sweeteners":

return Arrays.asList( "Sugar","Granulated Sugar", "Brown Sugar", "Honey", "Maple Syrup", "Corn Syrup",

"Vanilla Extract", "Artificial Sweeteners", "Date Syrup", "Molasses", "Jaggery");

case "Liquids":

return Arrays.asList( "Water", "Milk", "Lemon Juice", "Vinegar", "Soy Sauce", "Fish Sauce",

"Broth", "Cream", "Olive Oil", "Coconut Milk", "Rose Water", "Orange Juice",

"Apple Cider", "Buttermilk", "Saffron Water");

case "Pasta and Noodles":

return Arrays.asList("Spaghetti", "Penne", "Fettuccine", "Macaroni", "Lasagna Sheets",

"Tagliatelle", "Ravioli", "Udon Noodles", "Rice Noodles", "Soba Noodles",

"Egg Noodles", "Vermicelli");

case "Dry Goods":

return Arrays.asList("All-Purpose Flour", "Whole Wheat Flour", "Bread Flour", "Cake Flour",

"Cocoa Powder", "Cornmeal", "Semolina", "Cornstarch", "Tapioca Starch", "Rice Flour",

"Yeast", "Baking Powder", "Baking Soda");

case "Bakery":

return Arrays.asList("White Bread", "Whole Wheat Bread", "Sourdough Bread", "Baguette", "Focaccia",

"Ciabatta", "Rye Bread", "Multigrain Bread", "Pita Bread", "Naan");

default:

return Arrays.asList();

}

}

public static void main(String[] args) {

launch();

}

public void loadRecipes(GridPane gridPane) {

List<Recipe> recipes = readRecipesFromFile();

int row = 1;

int col = 0;

for (Recipe recipe : recipes) {

VBox recipeBox = createRecipeBox(recipe);

gridPane.add(recipeBox, col, row);

col++;

if (col == 3) {

col = 0;

row++;

}

}

}

public VBox createRecipeBox(Recipe recipe) {

VBox box = new VBox(10);

box.setPadding(new Insets(10));

box.setStyle("-fx-background-color: WHITE; -fx-border-color: #cccccc; -fx-border-radius: 0;");

ImageView imageView = new ImageView();

imageView.setFitWidth(200);

imageView.setFitHeight(150);

imageView.setPreserveRatio(true);

if (recipe.getImagePath() != null && !recipe.getImagePath().equals("No Image")) {

try {

File imageFile = new File(recipe.getImagePath());

if (imageFile.exists()) {

Image image = new Image(imageFile.toURI().toString());

imageView.setImage(image);

} else {

imageView.setImage(new Image("file:placeholder.png"));

}

} catch (Exception e) {

imageView.setImage(new Image("file:placeholder.png"));

}

} else {

imageView.setImage(new Image("file:placeholder.png"));

}

Label nameLabel = new Label(recipe.getName());

nameLabel.setStyle("-fx-font-weight: bold; -fx-font-size: 14px;");

Button viewButton = new Button("View Recipe");

viewButton.setCursor(Cursor.HAND);

viewButton.setOnAction(e -> FindRecipe.showRecipeDetails(recipe.getName(), mainStage));

viewButton.setPadding(new Insets(8));

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

viewButton.setTextFill(Color.WHITE);

viewButton.setStyle("-fx-background-color: #000000;");

viewButton.setCursor(Cursor.HAND);

box.getChildren().addAll(imageView, nameLabel, viewButton);

return box;

}

public List<Recipe> readRecipesFromFile() {

boolean endRecipeReached = false;

List<Recipe> recipes = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(recipeFile))) {

String line;

Recipe currentRecipe = null;

List<String> currentIngredients = new ArrayList<>();

StringBuilder instructions = new StringBuilder();

boolean readingIngredients = false;

boolean readingInstructions = false;

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

line = line.trim();

if (line.equals("START\_RECIPE")) {

currentRecipe = null;

currentIngredients = new ArrayList<>();

instructions = new StringBuilder();

readingIngredients = false;

readingInstructions = false;

endRecipeReached = false;

} else if (line.equals("END\_RECIPE")) {

if (currentRecipe != null && !endRecipeReached) {

currentRecipe.setIngredients(currentIngredients);

currentRecipe.setInstructions(instructions.toString().trim());

recipes.add(currentRecipe);

endRecipeReached = true;

}

} else if (line.startsWith("|")) {

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

if (parts.length > 2) {

String key = parts[1];

String value = parts[2];

if (currentRecipe == null && key.equals("NAME")) {

currentRecipe = new Recipe(value.trim(), "");

} else if (currentRecipe != null) {

switch (key) {

case "CUISINE":

currentRecipe.setCuisine(value.trim());

break;

case "MEAL\_TYPE":

currentRecipe.setMealType(value.trim());

break;

case "COOKING\_METHOD":

currentRecipe.setCookingMethod(value.trim());

break;

case "IMAGE\_PATH":

currentRecipe.setImagePath(value.trim());

break;

case "INGREDIENTS":

readingIngredients = true;

readingInstructions = false;

break;

case "INSTRUCTIONS":

readingIngredients = false;

readingInstructions = true;

break;

}

}

}

} else if (!line.isEmpty()) {

if (readingIngredients) {

currentIngredients.add(line);

} else if (readingInstructions) {

instructions.append(line).append("\n");

}

}

}

} catch (IOException e) {

System.out.println("Error loading recipes: " + e.getMessage());

}

return recipes;

}

public class Recipe {

private String name;

private String imagePath;

private List<String> ingredients;

private String instructions;

private String cuisine;

private String mealType;

private String cookingMethod;

public Recipe(String name, String imagePath) {

this.name = name;

this.imagePath = imagePath;

this.ingredients = new ArrayList<>();

this.instructions = "";

}

public String getCuisine() {

return cuisine;

}

public void setCuisine(String cuisine) {

this.cuisine = cuisine;

}

public String getMealType() {

return mealType;

}

public void setMealType(String mealType) {

this.mealType = mealType;

}

public String getCookingMethod() {

return cookingMethod;

}

public void setCookingMethod(String cookingMethod) {

this.cookingMethod = cookingMethod;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getImagePath() {

return imagePath;

}

public void setImagePath(String imagePath) {

this.imagePath = imagePath;

}

public void setIngredients(List<String> ingredients) {

this.ingredients = ingredients;

}

public List<String> getIngredients() {

return new ArrayList<>(ingredients);

}

public void setInstructions(String instructions) {

this.instructions = instructions;

}

public String getInstructions() {

return instructions;

}

}

}

Developer login:

…package com.example.recipebookapplication;

import java.io.\*;

public class DeveloperLogin {

static File developerFile = new File("developers.abc");

public static boolean authenticate(String username, String password) {

try (BufferedReader reader = new BufferedReader(new FileReader(developerFile))) {

String line;

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

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

if (parts.length == 2 && parts[0].equals(username) && parts[1].equals(password)) {

return true;

}

}

} catch (IOException e) {

System.out.println("Error reading developer file: " + e.getMessage());

}

return false;

}

}

RemoveRecipe:

package com.example.recipebookapplication;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Cursor;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.Background;

import javafx.scene.layout.BackgroundFill;

import javafx.scene.layout.CornerRadii;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

import static com.example.recipebookapplication.DeveloperLogin.developerFile;

public class RemoveRecipe {

private static final String RECIPE\_FILE = "recipe.abc";

public static void show() {

Stage stage = new Stage();

VBox layout = new VBox(10);

layout.setPadding(new Insets(20));

layout.setAlignment(Pos.CENTER);

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

TextField usernameField = new TextField();

usernameField.setPromptText("Developer Username");

PasswordField passwordField = new PasswordField();

passwordField.setPromptText("Developer Password");

Button loginButton = new Button("Login");

loginButton.setFont(Font.font("Courier New", 16));

loginButton.setPadding(new Insets(5));

loginButton.setCursor(Cursor.HAND);

loginButton.setStyle("-fx-background-color: #000000; -fx-text-fill: white;");

layout.getChildren().addAll(

new Label("Developer Login"),

usernameField,

passwordField,

loginButton

);

loginButton.setOnAction(e -> {

if (DeveloperLogin.authenticate(usernameField.getText(), passwordField.getText())) {

showRemoveRecipeInterface(stage);

} else {

showAlert("Login Failed", "Invalid developer credentials.");

}

});

try( BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(developerFile))) {

bufferedWriter.write("admin:123");

} catch (IOException e) {

throw new RuntimeException(e);

}

Scene scene = new Scene(layout, 300, 200);

stage.setScene(scene);

stage.setTitle("Remove Recipe - Developer Login");

stage.show();

}

private static void showRemoveRecipeInterface(Stage stage) {

VBox layout = new VBox(10);

layout.setPadding(new Insets(20));

layout.setAlignment(Pos.CENTER);

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

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

Button removeButton = new Button("Remove Recipe");

removeButton.setFont(Font.font("Courier New", 16));

removeButton.setPadding(new Insets(5));

removeButton.setCursor(Cursor.HAND);

removeButton.setStyle("-fx-background-color: #000000; -fx-text-fill: white;");

loadRecipes(recipeComboBox);

removeButton.setOnAction(e -> {

String selectedRecipe = recipeComboBox.getValue();

if (selectedRecipe != null) {

removeRecipe(selectedRecipe);

recipeComboBox.getItems().remove(selectedRecipe);

showAlert("Recipe Removed", "The recipe has been successfully removed.");

} else {

showAlert("No Recipe Selected", "Please select a recipe to remove.");

}

});

Label remove = new Label("Select Recipe to Remove:");

remove.setFont(Font.font("Verdana", 18));

layout.getChildren().addAll(

remove,

recipeComboBox,

removeButton

);

Scene scene = new Scene(layout, 400, 300);

stage.setScene(scene);

stage.setTitle("Remove Recipe");

}

private static void loadRecipes(ComboBox<String> recipeComboBox) {

recipeComboBox.getItems().clear();

try (BufferedReader reader = new BufferedReader(new FileReader(RECIPE\_FILE))) {

String line;

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

if (line.startsWith("|NAME|")) {

String recipeName = line.substring(6).trim();

recipeComboBox.getItems().add(recipeName);

}

}

} catch (IOException e) {

System.err.println("Error loading recipes: " + e.getMessage());

}

}

private static void removeRecipe(String recipeName) {

List<String> updatedRecipes = new ArrayList<>();

boolean skip = false;

try (BufferedReader reader = new BufferedReader(new FileReader(RECIPE\_FILE))) {

String line;

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

if (line.equals("START\_RECIPE")) {

String nextLine = reader.readLine();

if (nextLine != null && nextLine.startsWith("|NAME|") && nextLine.substring(6).trim().equals(recipeName)) {

skip = true;

continue;

} else {

updatedRecipes.add(line);

updatedRecipes.add(nextLine);

}

} else if (line.equals("END\_RECIPE")) {

if (skip) {

skip = false;

} else {

updatedRecipes.add(line);

}

} else if (!skip) {

updatedRecipes.add(line);

}

}

} catch (IOException e) {

System.err.println("Error reading recipes: " + e.getMessage());

return;

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter(RECIPE\_FILE))) {

for (String recipe : updatedRecipes) {

writer.write(recipe);

writer.newLine();

}

} catch (IOException e) {

System.err.println("Error writing updated recipes: " + e.getMessage());

}

}

private static void showAlert(String title, String content) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

}

Findrecipe:

package com.example.recipebookapplication;

import javafx.geometry.Insets;

import javafx.scene.Cursor;

import javafx.scene.Scene;

import javafx.scene.control.\*;

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.Stage;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class FindRecipe {

private static final String RECIPE\_FILE = "recipe.abc";

public static void loadRecipesToComboBox(ComboBox<String> findRecipe) {

findRecipe.getItems().clear();

File recipeFile = new File(RECIPE\_FILE);

if (!recipeFile.exists()) {

return;

}

try (BufferedReader reader = new BufferedReader(new FileReader(recipeFile))) {

String line;

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

if (line.startsWith("|NAME|")) {

String recipeName = line.substring(6).trim();

findRecipe.getItems().add(recipeName);

}

}

} catch (IOException e) {

System.err.println("Error loading recipes: " + e.getMessage());

}

}

public static void showRecipeDetails(String recipeName, Stage mainStage) {

Stage stage = new Stage();

VBox layout = new VBox(20);

layout.setPadding(new Insets(40));

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

ScrollPane scrollPane = new ScrollPane(layout);

scrollPane.setFitToWidth(true);

Scene scene = new Scene(scrollPane, 800, 600);

stage.setScene(scene);

stage.setFullScreen(true);

stage.setTitle("Recipe Details: " + recipeName);

Map<String, String> recipeDetails = new HashMap<>();

List<String> ingredients = new ArrayList<>();

StringBuilder instructions = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(RECIPE\_FILE))) {

String line;

boolean foundRecipe = false;

boolean readingIngredients = false;

boolean readingInstructions = false;

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

if (line.equals("START\_RECIPE")) {

recipeDetails.clear();

ingredients.clear();

instructions.setLength(0);

foundRecipe = false;

readingIngredients = false;

readingInstructions = false;

} else if (line.equals("END\_RECIPE")) {

if (foundRecipe) break;

} else if (line.startsWith("|")) {

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

if (parts.length > 2) {

String key = parts[1];

String value = parts[2];

recipeDetails.put(key, value);

if (key.equals("NAME") && value.equalsIgnoreCase(recipeName)) {

foundRecipe = true;

}

}

if (parts[1].equals("INGREDIENTS")) {

readingIngredients = true;

readingInstructions = false;

} else if (parts[1].equals("INSTRUCTIONS")) {

readingIngredients = false;

readingInstructions = true;

}

} else if (readingIngredients && !line.trim().isEmpty()) {

ingredients.add(line.trim());

} else if (readingInstructions) {

instructions.append(line).append("\n");

}

}

if (foundRecipe) {

displayRecipeDetails(layout, recipeDetails, ingredients, instructions.toString(), mainStage);

} else {

layout.getChildren().add(new Label("Recipe not found: " + recipeName));

}

} catch (IOException e) {

layout.getChildren().add(new Label("Error loading recipe details: " + e.getMessage()));

}

stage.show();

}

private static void displayRecipeDetails(VBox layout, Map<String, String> recipeDetails, List<String> ingredients,

String instructions, Stage mainStage) {

Label titleLabel = new Label(recipeDetails.get("NAME"));

titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

layout.getChildren().add(titleLabel);

layout.getChildren().add(new Label("Cuisine: " + recipeDetails.get("CUISINE")));

layout.getChildren().add(new Label("Meal Type: " + recipeDetails.get("MEAL\_TYPE")));

layout.getChildren().add(new Label("Cooking Method: " + recipeDetails.get("COOKING\_METHOD")));

HBox contentBox = new HBox(20);

VBox imageBox = new VBox(10);

VBox ingredientsBox = new VBox(10);

String imagePath = recipeDetails.get("IMAGE\_PATH");

if (imagePath != null && !imagePath.isEmpty() && !imagePath.equals("No Image")) {

try {

File imageFile = new File(imagePath);

if (imageFile.exists()) {

Image image = new Image(imageFile.toURI().toString());

ImageView imageView = new ImageView(image);

imageView.setFitWidth(300);

imageView.setPreserveRatio(true);

imageBox.getChildren().add(imageView);

} else {

imageBox.getChildren().add(new Label("Image not found: " + imagePath));

}

} catch (Exception e) {

System.err.println("Error loading image: " + e.getMessage());

imageBox.getChildren().add(new Label("Error loading image"));

}

} else {

imageBox.getChildren().add(new Label("No image available"));

}

Label ingredientsLabel = new Label("Ingredients:");

ingredientsLabel.setStyle("-fx-font-weight: bold;");

ingredientsBox.getChildren().add(ingredientsLabel);

for (String ingredient : ingredients) {

String[] parts = ingredient.split(";");

if (parts.length > 1) {

ingredientsBox.getChildren().add(new Label("• " + parts[0] + " - " + parts[1]));

} else {

ingredientsBox.getChildren().add(new Label("• " + ingredient));

}

}

contentBox.getChildren().addAll(imageBox, ingredientsBox);

layout.getChildren().add(contentBox);

Label instructionsLabel = new Label("Instructions:");

instructionsLabel.setStyle("-fx-font-weight: bold;");

layout.getChildren().add(instructionsLabel);

TextArea instructionsArea = new TextArea(instructions);

instructionsArea.setWrapText(true);

instructionsArea.setEditable(false);

instructionsArea.setPrefHeight(200);

layout.getChildren().add(instructionsArea);

Button backButton = new Button("Back to Main");

backButton.setFont(Font.font("Comic Sans MS", 16));

backButton.setPadding(new Insets(5));

backButton.setCursor(Cursor.HAND);

backButton.setStyle("-fx-background-color: #000000; -fx-text-fill: white;");

backButton.setOnAction(e -> {

((Stage) layout.getScene().getWindow()).close();

mainStage.setFullScreen(true);

});

layout.getChildren().add(backButton);

}

}

Ingredients:

package com.example.recipebookapplication;

public class Ingredient {

private String category;

private String name;

private String measurement;

private String quantity;

public Ingredient(String category, String name, String measurement, String quantity) {

this.category = category;

this.name = name;

this.measurement = measurement;

this.quantity = quantity;

}

public String getCategory() {

return category;

}

public String getName() {

return name;

}

public String getMeasurement() {

return measurement;

}

public String getQuantity() {

return quantity;

}

}