# Final Project

# CSYE 6200

**NOSTALGIC FOOD AND RECIPES**

**By Team**

# SOFTWARE SIZZLERS

**TEAM MEMBERS:**

**Abhishek Chintapalli - 002837203,**

**Venkata Naga Sri Sai Sujhata Meghana Tadikonda - 002642314,**

**Abhinav Eeranti - 002883528**

**COURSE:**

Software Engineering Systems, Northeastern University.

**EMAIL ID'S:**

chintapalli.a@northeastern.edu

tadikonda.v@northeasten.edu

eeranti.a@northeastern.edu

## I. ABSTRACT:

This project addresses the common cooking challenges faced by students through the development of a Java-based mobile application. The scope encompasses the creation of a comprehensive solution that allows students to easily access and prepare a wide range of recipes, including those contributed by their parents or favorite chefs. The proposed application offers a recipe repository with features such as user profiles, recipe uploads, and personalized collections. Users can browse, search, and select recipes based on their preferences, while parents and chefs can effortlessly contribute to the growing database.

The project methodology involves leveraging Java as the primary programming language and utilizing the Eclipse integrated development environment. GitHub serves as the version control system, ensuring efficient collaboration among team members. Task management is streamlined through Trello, facilitating organized development and tracking of project milestones. The design focuses on creating a user-friendly interface with modules for recipe management, user profiles, and step-by-step guides.

Key outcomes of the project include a functional mobile application that serves as a centralized hub for diverse recipes. Users can create profiles, personalize their recipe collections, and benefit from a step-by-step cooking guide with images or videos. The app facilitates recipe uploads from parents and chefs, ensuring a continuously expanding and diverse recipe database. Accurate measurements and ingredient list aid users in efficient ingredient shopping. The platform promotes user interaction, allowing individuals to ask questions about recipes and share their experiences with the community. User feedback and ratings contribute to the app's continuous improvement, enhancing the overall user experience.

In conclusion, this project addresses a pressing issue faced by students, providing a solution that not only improves their cooking skills but also promotes healthier eating habits, self-sufficiency, and better concentration on studies. The methodology, design, and outcomes collectively contribute to the development of a user-centric and feature-rich mobile application for an enhanced culinary experience among students.

## II. PROBLEM DESCRIPTION:

**Introduction:** Cooking is an essential life skill, yet many students find themselves grappling with its challenges, especially when transitioning to new environments where access to familiar recipes may be limited. The fundamental background of this project lies in recognizing the critical role nutrition plays in the overall well-being and academic performance of students. Malnutrition, weakness, and an inability to concentrate on studies are identified consequences of insufficient cooking skills and lack of access to diverse recipes. This project aims to bridge this gap by developing a Java-based mobile application tailored to the unique needs of students, offering a comprehensive solution to enhance their culinary experience and, consequently, their overall health and academic success.

**Scope:** The scope of this project is to create a user-centric mobile application that serves as a centralized platform for a diverse range of recipes. The application goes beyond conventional recipe repositories by incorporating features that allow users to access recipes from their parents or favorite chefs, fostering a sense of familiarity and personalization. The scope extends to facilitating recipe uploads, not only empowering students but also creating a collaborative environment where parents and chefs can effortlessly contribute to the platform, ensuring a continually expanding and diverse recipe database. Additionally, the application's scope includes personalized user profiles, step-by-step cooking guides, accurate measurements, and a feedback mechanism, creating an all-encompassing solution for users with varying cooking skills and preferences.

**Purpose:** This innovative project seeks to empower students, particularly new immigrants, by addressing the dual challenges of adapting to a new environment and honing essential cooking skills. At its core, the mobile application aims to alleviate the cooking obstacles faced by students, promoting healthier eating habits, self-sufficiency, and improved focus on academic pursuits. Rooted in the understanding that being away from home can evoke a longing for familiar tastes, the app is uniquely tailored to store and share cherished home-cooked recipes, providing a comforting connection to one's cultural roots.
Designed with the student experience in mind, the application not only serves as a comprehensive recipe repository but also fosters a sense of community. Users can interact, ask questions, and share their cooking experiences, creating a space for collective learning. Recognizing cooking as a vital life skill, the platform goes beyond preserving recipes; it actively encourages users to enhance their culinary expertise. This dual functionality not only addresses the practical challenges of cooking in a new

environment but also contributes to the overall well-being and nutrition of students. By leveraging the wisdom of revered chefs, particularly mothers, the app ensures a continuous exchange of cultural and culinary knowledge, fostering a holistic development that goes beyond the kitchen.

## III. ANALYSIS:

**Summary of the Project:** The project addresses the cooking challenges faced by students through the development of a Java-based mobile application, aiming to provide a comprehensive solution that includes a diverse recipe repository, user profiles, recipe uploads, and personalized features. The analysis focuses on understanding the fundamental issues students encounter in cooking and proposes a user-centric platform to enhance their culinary skills.

**Previous Works:** While various recipe apps exist, the majority lack a personalized touch and the ability to incorporate recipes from parents or specific chefs. Existing solutions often do not cater specifically to the needs of students, and the lack of a collaborative recipe upload feature limits the diversity of available recipes.

**Findings:** The analysis identifies the primary challenges faced by students in cooking, emphasizing malnutrition, weakness, and a lack of concentration on studies. The findings suggest a need for a solution that not only provides recipes but also addresses the personalization aspect by including contributions from parents and chefs.

**Shortcomings:** Common shortcomings observed in existing solutions include a lack of personalization, limited diversity in recipes, and a disconnect between users and their preferred sources of recipes. Additionally, many apps do not offer a collaborative platform for recipe uploads, limiting the variety of recipes available to users.

**Existing Solutions or Technologies:** Current recipe apps often focus on providing a vast collection of recipes without considering the personalization factor. Few platforms incorporate collaborative features for users to contribute recipes. Some existing technologies include machine learning algorithms for personalized recipe recommendations and interactive cooking guides.

## IV. SYSTEM DESIGN:

**Addressing the Identified Problem:** The primary problem identified is the lack of cooking skills among students, leading to issues such as malnutrition and an inability to concentrate on studies. The proposed system design aims to address this problem by providing a user-friendly mobile application that serves as a centralized hub for diverse recipes, including contributions from parents and chefs. The system emphasizes personalized user profiles, collaborative recipe uploads, and step-by-step cooking guides to enhance the cooking experience for students.

**System Architecture:** The system architecture comprises three main components: the mobile application interface, the backend server, and the database.

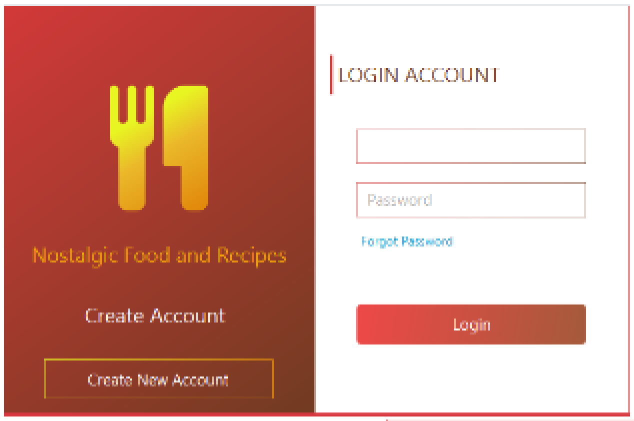
- **Mobile Application Interface:** This is the front-end component accessible to users. It includes features such as recipe browsing, user profiles, and interactive cooking guides.
- **Backend Server:** Responsible for processing user requests, managing user profiles, handling recipe uploads, and facilitating communication with the database.
- **Database:** Stores user data, recipes, and related information. It provides the necessary data to the backend server for seamless user interactions.

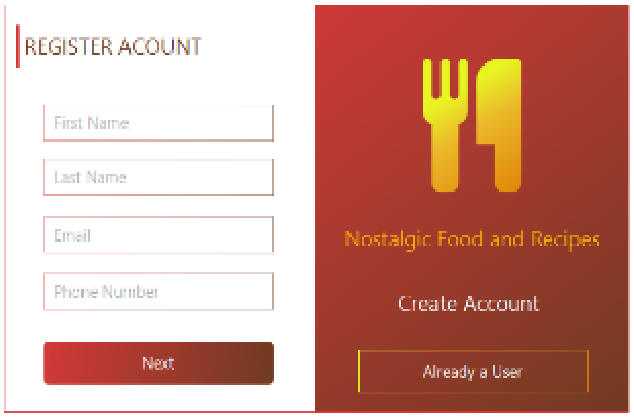
The communication between these components follows a client-server model, ensuring a responsive and scalable system.

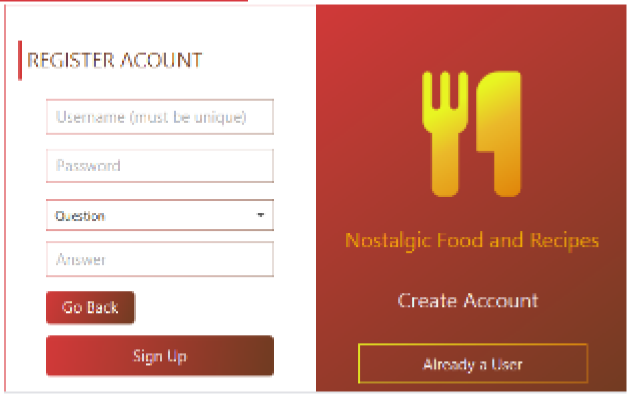
**UI Design:** The user interface (UI) is designed to be intuitive and user-friendly, catering to users with varying cooking skills. The main screens include:

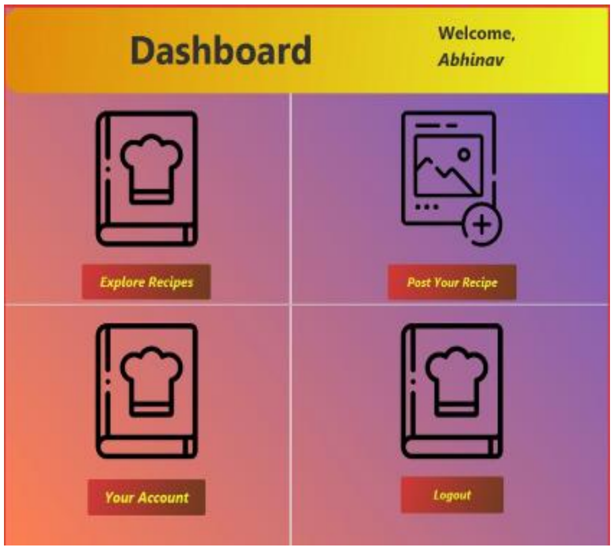
1. **Login Screen:** The project begins with a user-friendly login popup, enabling users to either log in with existing credentials or seamlessly create a new account by providing essential information. The interactive design, featuring hover effects and transition animations, enhances the user experience, ensuring a visually engaging process with the option to backtrack if needed.
2. **Home Screen:** Displays featured recipes, personalized recommendations, and quick links to user profiles and recent activities.
3. **Recipe Browser:** Allows users to search and browse recipes based on categories, ingredients, or personal preferences.
4. **User Profile:** Enables users to create and manage their profiles, including preferences, favorite recipes, and personal contributions.

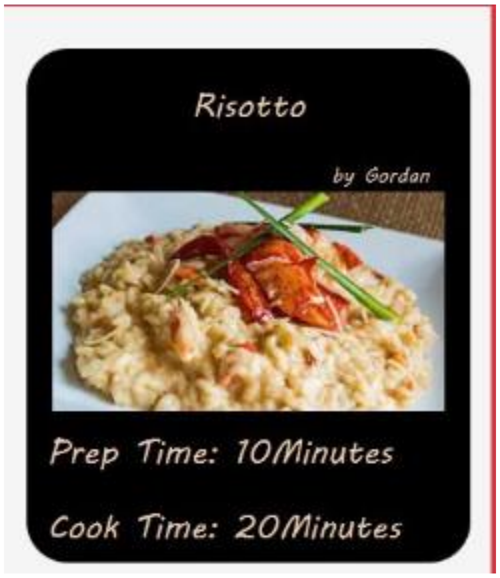
5. **Recipe Details:** Provides detailed information about a selected recipe, including ingredients, step-by-step guides, and user ratings.
6. **Upload Recipe:** Allows parents and chefs to contribute recipes, including text instructions, images, and additional details.
7. **Interactive Cooking Guide:** Displays step-by-step instructions with images or videos, guiding users through the cooking process.

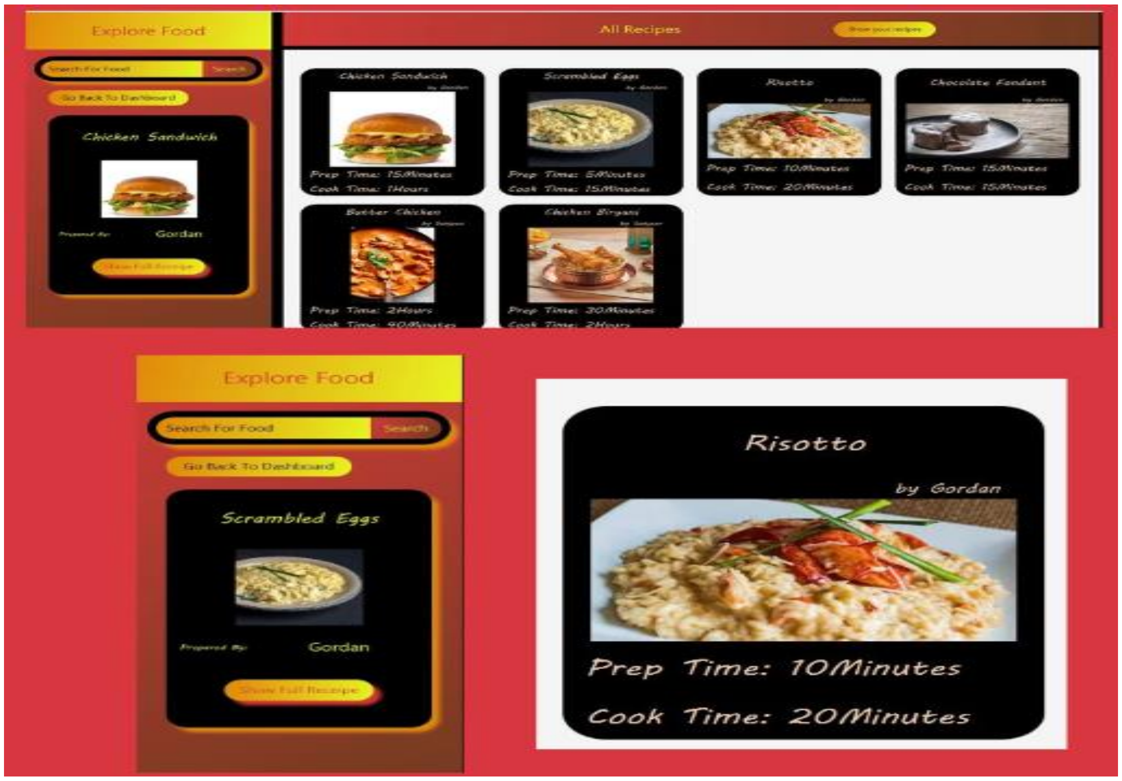


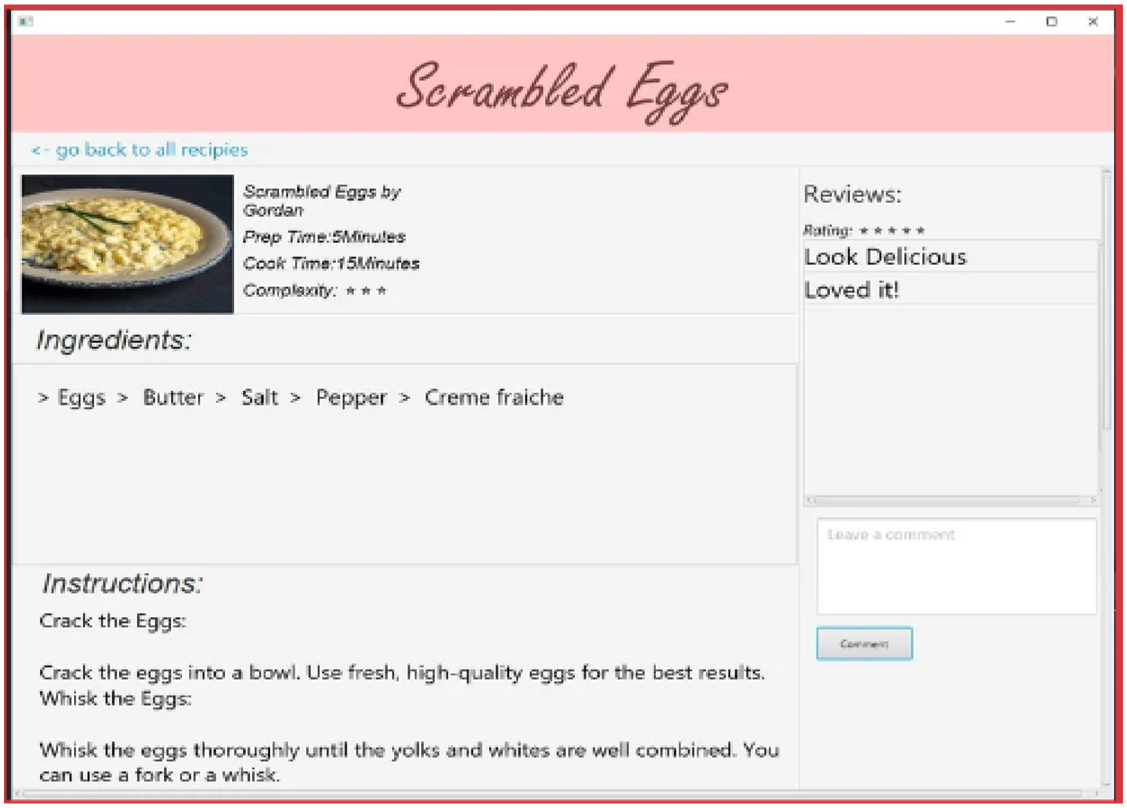












*Scrambled Eggs by Gordan*
*Prep Time:5Minutes*
*Cook Time:15Minutes*
*Complexity: ★ ★ ★*

*Ingredients:*

> Eggs > Butter > Salt > Pepper > Creme fraiche

*Instructions:*

Crack the Eggs:

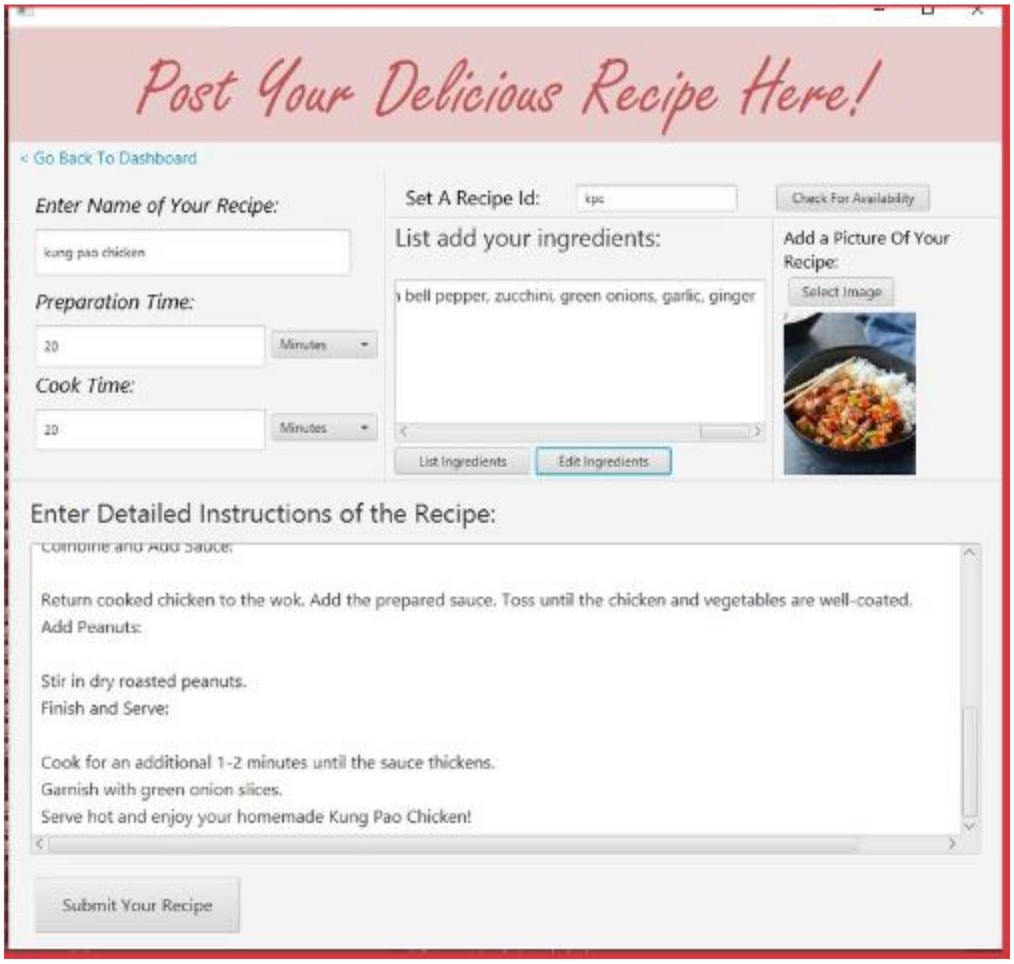
Crack the eggs into a bowl. Use fresh, high-quality eggs for the best results.
Whisk the Eggs:

Whisk the eggs thoroughly until the yolks and whites are well combined. You can use a fork or a whisk.

*Rating: ★ ★ ★ ★ ★*

Look Delicious

Loved it!



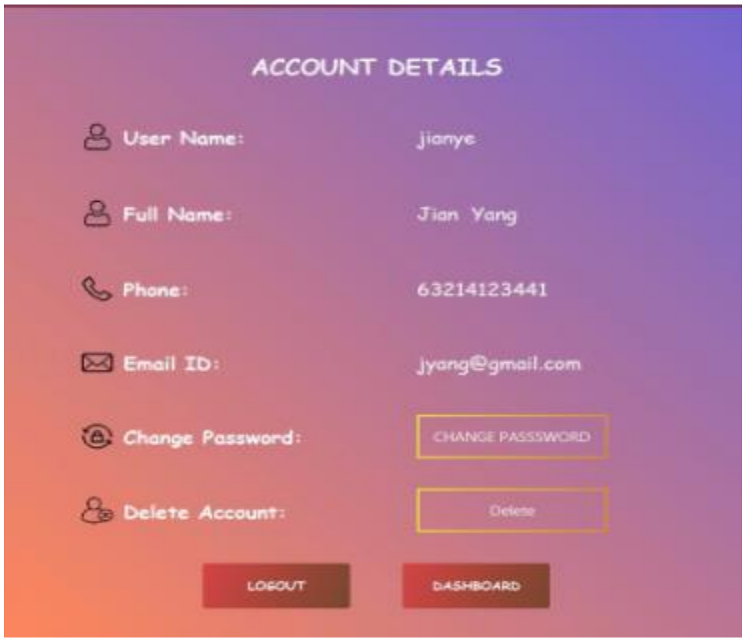
i bell pepper, zucchini, green onions, garlic, ginger

Combine and Add Sauce:

Return cooked chicken to the wok. Add the prepared sauce. Toss until the chicken and vegetables are well-coated.
Add Peanuts:

Stir in dry roasted peanuts.
Finish and Serve:

Cook for an additional 1-2 minutes until the sauce thickens.
Garnish with green onion slices.
Serve hot and enjoy your homemade Kung Pao Chicken!



| | |
|---|---|
| User Name: | jianye |
| Full Name: | Jian Yang |
| Phone: | 63214123441 |
| Email ID: | jyang@gmail.com |
| Change Password: | CHANGE PASSSWORD |
| Delete Account: | Delete |

## V. IMPLEMENTATION:

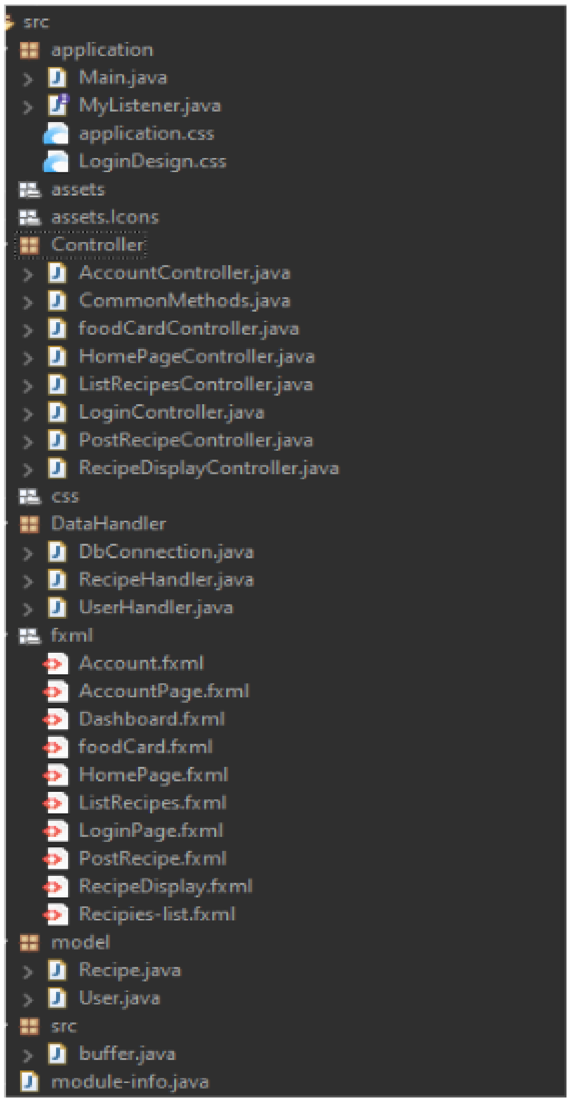
### Structure:

1. Application Package:
Main: The main class responsible for initializing the code and launching the application.
LoginDesign.css: A CSS file for styling aspects across the project.
2. Controller Package:
AccountController: Manages the account-related functionalities.
CommonMethods: Contains reusable methods shared across controllers.
FoodCardController: Controls the behavior of food card components.
HomePageController: Handles events and actions on the home page.
ListRecipesController: Manages the listing and display of recipes.
LoginController: Controls the login functionality.
PostRecipeController: Manages the posting of new recipes.
RecipeDisplayController: Handles the display of individual recipes.
3. DataHandlers Package:
DbConnection: Establishes and manages the database connection.
RecipeHandler: Manages interactions with the recipe-related data in the database.
UserHandler: Handles user-related data operations.
4. FXML Package:
Account.fxml: The FXML file for the account page.
AccountPage.fxml: FXML file for the account details page.
Dashboard.fxml: FXML file for the dashboard.
FoodCard.fxml: FXML file defining the structure of the food card.
HomePage.fxml: FXML file for the main home page.
ListRecipes.fxml: FXML file for listing recipes.
LoginPage.fxml: FXML file for the login page.
PostRecipe.fxml: FXML file for posting new recipes.
RecipeDisplay.fxml: FXML file for displaying individual recipes.
RecipeList.fxml: FXML file for listing multiple recipes.
5. Model Package:
Recipe.java: Represents the structure of a recipe.
User.java: Represents the structure of a user.

### Project Execution:

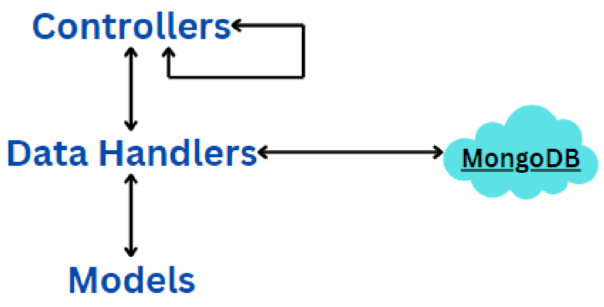
Upon successful implementation and execution, the project seamlessly integrates the specified packages and functionalities, providing a well-organized, modular, and maintainable architecture.

**Code Order:**


```
src
  application
    Main.java
    MyListener.java
    application.css
    LoginDesign.css
  assets
  assets.Icons
  Controller
    AccountController.java
    CommonMethods.java
    foodCardController.java
    HomePageController.java
    ListRecipesController.java
    LoginController.java
    PostRecipeController.java
    RecipeDisplayController.java
  css
  DataHandler
    DbConnection.java
    RecipeHandler.java
    UserHandler.java
  fxml
    Account.fxml
    AccountPage.fxml
    Dashboard.fxml
    foodCard.fxml
    HomePage.fxml
    ListRecipes.fxml
    LoginPage.fxml
    PostRecipe.fxml
    RecipeDisplay.fxml
    Recipies-list.fxml
  model
    Recipe.java
    User.java
  src
    buffer.java
  module-info.java
```

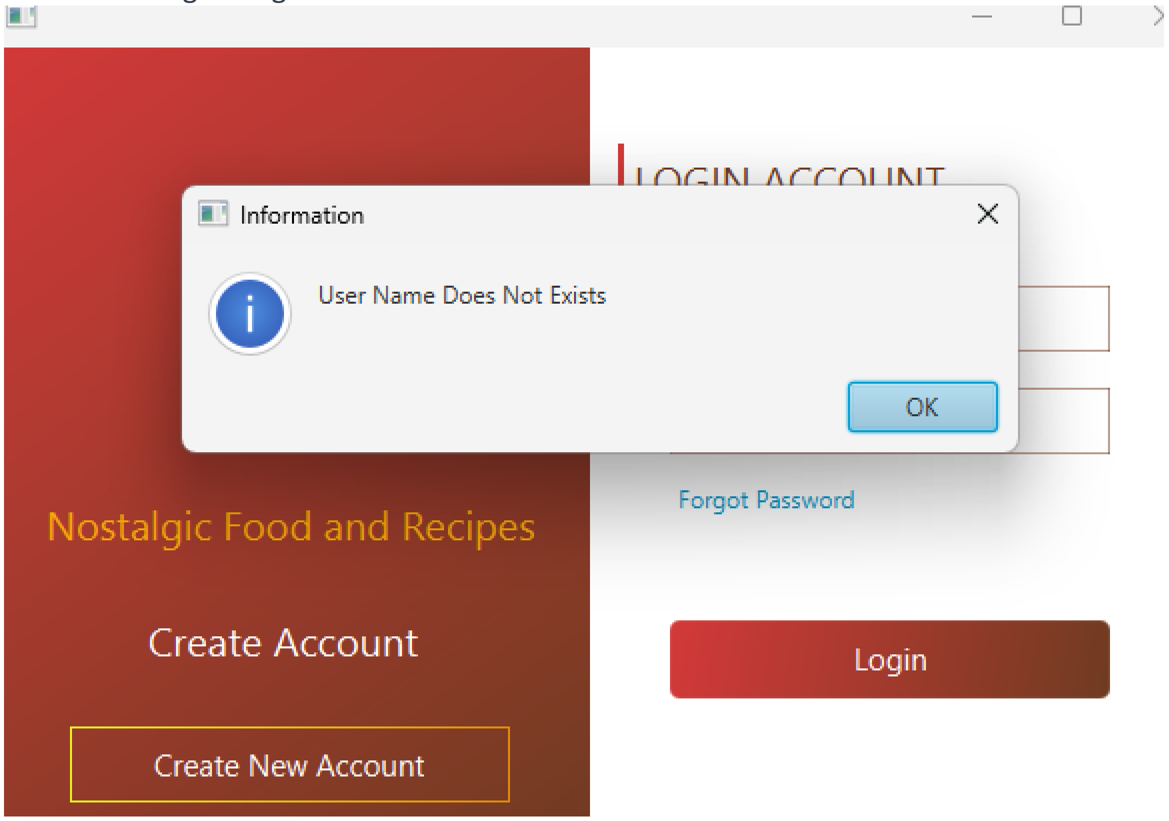
## VI. EVALUATION:

The Working flow of the project is:



Testing:

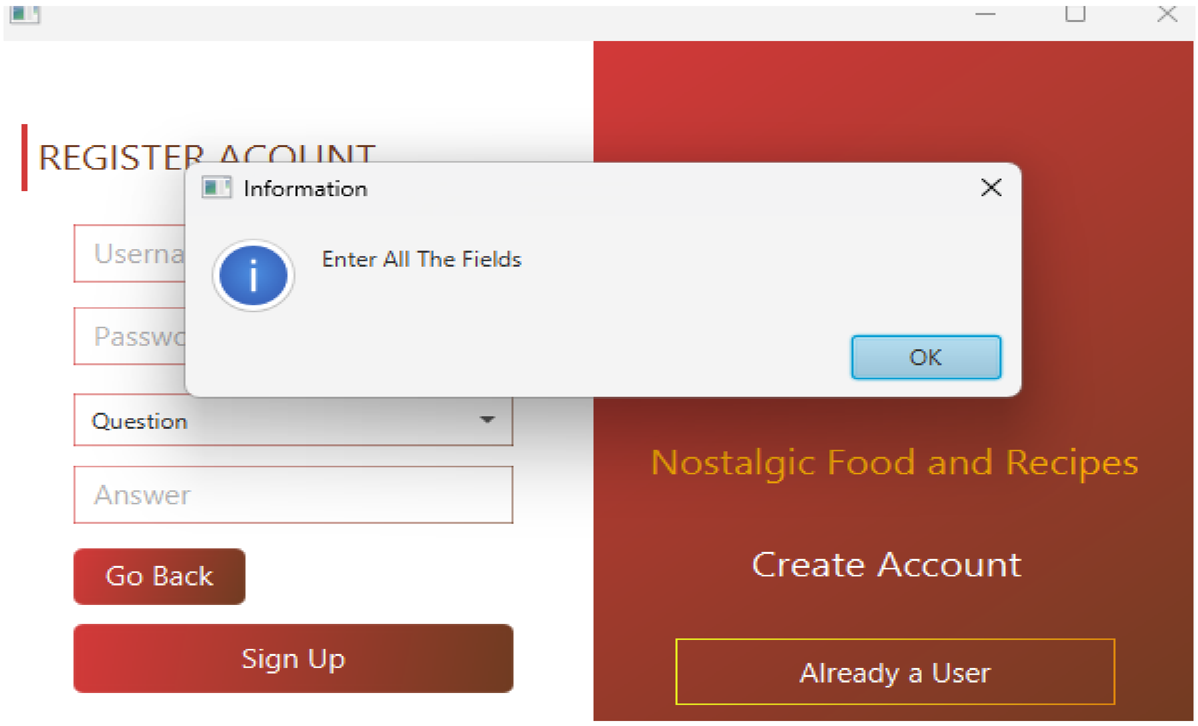
When Entering wrong Credentials:



Information

User Name Does Not Exists

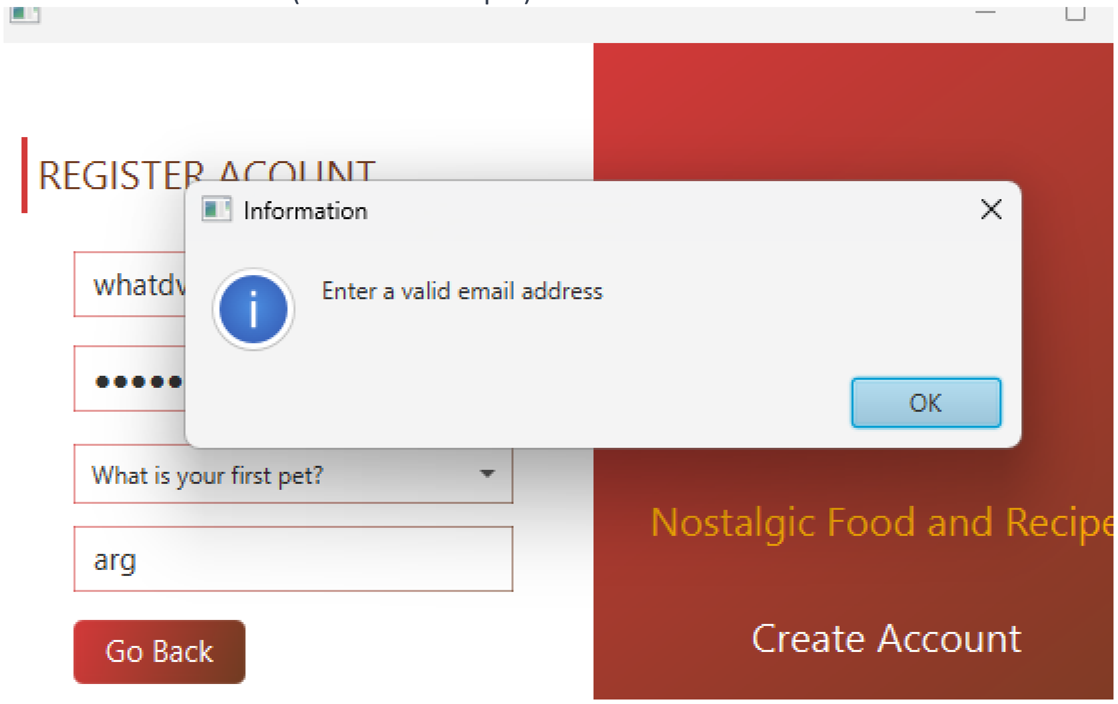
When all the fields are not entered:



Information

Enter All The Fields

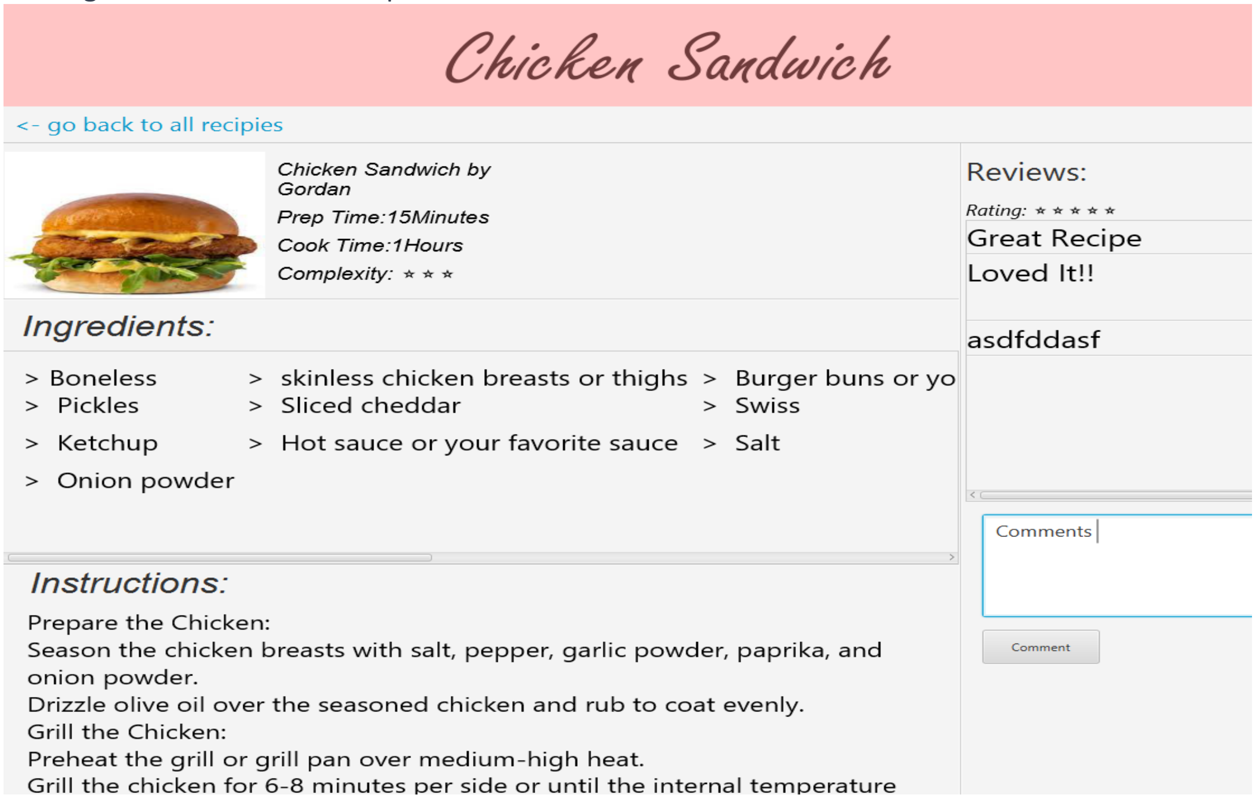
Validation at all levels (Email as Example):


REGISTER ACOUNT

Information

Enter a valid email address

Posting Comments to the recipe:


Chicken Sandwich

<- go back to all recipies

Chicken Sandwich by Gordan
Prep Time:15Minutes
Cook Time:1Hours
Complexity: ★ ★ ★

Ingredients:

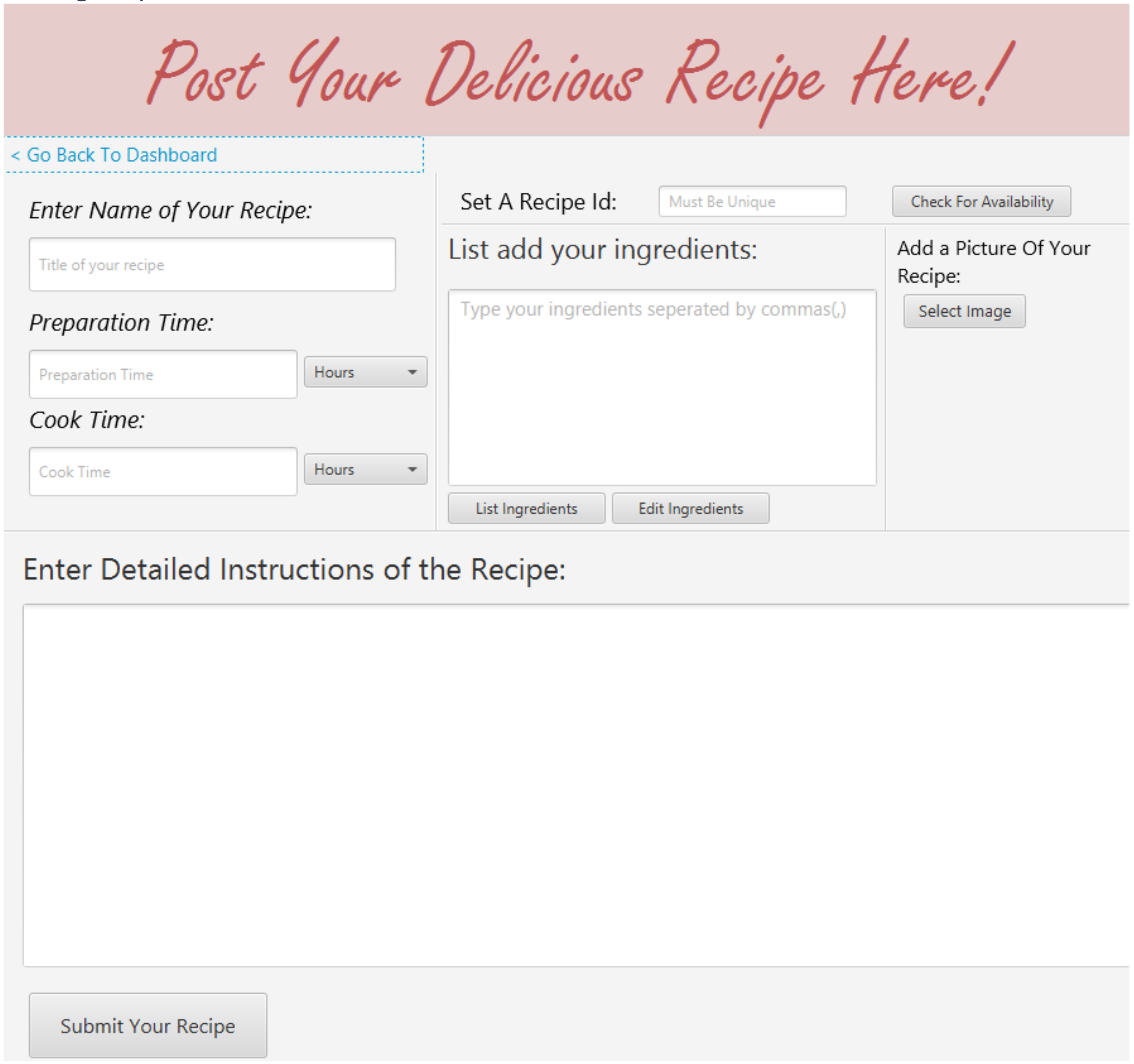
| | | |
|---|---|---|
| > Boneless | > skinless chicken breasts or thighs | > Burger buns or yo |
| > Pickles | > Sliced cheddar | > Swiss |
| > Ketchup | > Hot sauce or your favorite sauce | > Salt |
| > Onion powder | | |

Instructions:

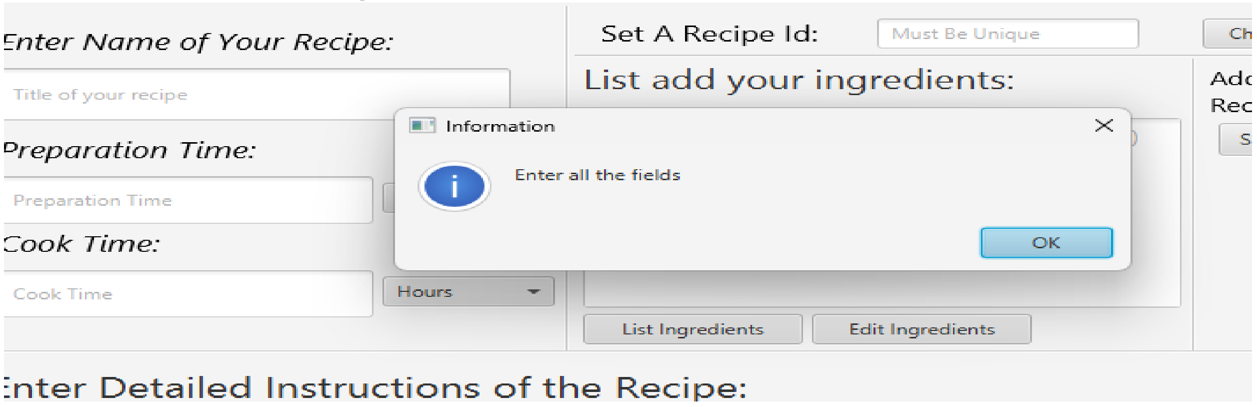
Prepare the Chicken:
Season the chicken breasts with salt, pepper, garlic powder, paprika, and onion powder.
Drizzle olive oil over the seasoned chicken and rub to coat evenly.
Grill the Chicken:
Preheat the grill or grill pan over medium-high heat.
Grill the chicken for 6-8 minutes per side or until the internal temperature

Reviews:
Rating: ★ ★ ★ ★ ★
Great Recipe
Loved It!!
asdfddasf

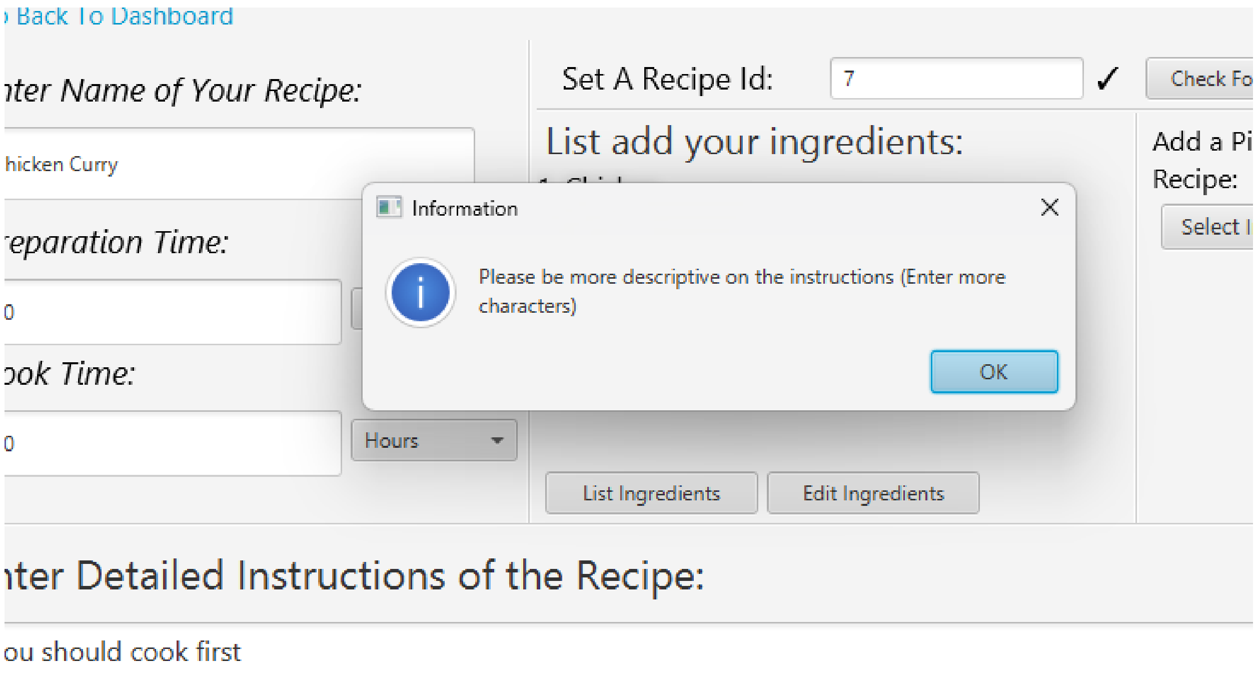
Posting recipe:



If instructions are not enough then:



Enter all the fields



Please be more descriptive on the instructions (Enter more characters)

## Primary Codes:

**Main.java**

```
package application;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;


public class Main extends Application {
	@Override
	public void start(Stage primaryStage) {
		try {
			Parent root = 
FXMLLoader.load(getClass().getResource("/fxml/LoginPage.fxml"));
			Scene scene = new Scene(root,610,400);
			
	scene.getStylesheets().add(getClass().getResource("applicat
ion.css").toExternalForm());
			primaryStage.setScene(scene);
			primaryStage.show();
		} catch(Exception e) {
			e.printStackTrace();
		}
	}

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

**LoginDesign.css**

```
.grdient{
	-fx-background-color: linear-gradient(to bottom right, 
#d33939, #6f3b21);
}

.light-gradient{
	-fx-background-color: linear-gradient(to top right, 
#ff7f50, #6a5acd);
	-fx-border-color: black;
	-fx-border-width: 10;
}

.white-form{
	-fx-background-color: #fff;

}
.head-label{
	-fx-border-color: #d33939;
	-fx-border-width: 0 0 0 3px;
	-fx-padding: 0.0 0.0 0.0 5px;
}

.login-button{
	-fx-background-color: linear-gradient(to bottom right, 
#d33939, #6f3b21);
	-fx-background-radius: 5px;
	-fx-cursor:hand;
	-fx-text-fill: #fff;
	-fx-font-size: 15px;
}

.login-button:hover{
	-fx-background-color: linear-gradient(to bottom right, 
#ee4848, #a45a3b);
}

.textfield{
	-fx-background-color: transparent;
```

```
     -fx-border-color: linear-gradient(to bottom right, #d33939,
#6f3b21);
     -fx-border-width: .5px;
}

.create-button{
     -fx-background-color: transparent;
     -fx-cursor: hand;
     -fx-text-fill: #fff;
     -fx-border-color: linear-gradient(to bottom right, #e8f522,
#e1880b);
     -fx-border-width: 1px;
     -fx-font-size: 15px;
}

.create-button:hover{
     -fx-background-color: linear-gradient(to bottom right,
#e8f522, #e1880b);
     -fx-text-fill: #dd2c08;
}

.nav-dash{
     -fx-background-color: linear-gradient(to bottom right,
#e1880b, #e8f522);
}

.searchBar{
     -fx-background-color: #000000;
     -fx-background-radius: 100;
}

.searchButton{
     -fx-background-color: linear-gradient(to bottom right,
#d33939, #6f3b21);
     -fx-background-radius: 0 30 30 0;
}

.searchButton:hover{
     -fx-background-color: linear-gradient(to bottom right,
#ff6262, #ab6544);
     -fx-cursor: hand;
}

.searchTextArea{
     -fx-prompt-text-fill: #000000;
     -fx-background-color: linear-gradient(to bottom right,
#e1880b, #e8f522);
```

```
     /*-fx-border-color: linear-gradient(to bottom right,
#d33939, #6f3b21);*/
     -fx-background-radius: 30 0 0 30;
}

.foodcard{
     -fx-background-color: #000000;
     -fx-background-radius: 30;
}
.foodcard:hover{
     -fx-cursor: hand;
     -fx-background-color: rgba(0,0,0,0.92);
}

.shadow{
     -fx-effect:dropshadow(three-pass-box, #e1880b, 10, 0, 10,
10);
}

.shadow2{
     -fx-effect:dropshadow(three-pass-box, #d33939, 10, 0, 10,
10);
}

.cardButton{
     -fx-background-radius: 100;
     -fx-background-color: linear-gradient(to bottom right,
#e1880b, #e8f522);
}

.cardButton:hover{
     -fx-background-color: linear-gradient(to bottom right,
#f3951c, #e8e618);
     -fx-cursor: hand;
}

.black{
     -fx-background-color: #000000;
}

.scroll-Pane{
     -fx-background-color: rgba(0,0,0,0);
}

.dash-boardButtons{
     -fx-background-color: linear-gradient(to bottom right,
#d33939, #6f3b21);
```

```
}

.dash-boardButtons:hover{
      -fx-background-color: linear-gradient(to bottom right,
#ee4848, #a45a3b);
      -fx-cursor: hand;
}
```

**Logincontroller.java**

```java
package Controller;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import DataHandler.UserHandler;
import javafx.animation.TranslateTransition;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Hyperlink;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.TextFormatter;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import javafx.util.Duration;
import model.User;

@SuppressWarnings("unused")
public class LoginController implements Initializable {

    @FXML
    private Hyperlink si_forgotPassword;

    @FXML
    private Button si_loginButton;

    @FXML
    private AnchorPane si_loginForm;
    @FXML
    private PasswordField si_password;

    @FXML
    private TextField si_username;

    @FXML
    private Button side_createButton;

    @FXML
    private AnchorPane side_form;

    @FXML
    private TextField su_answer;

    @FXML
    private PasswordField su_password;

    @FXML
    private TextField su_firstName;

    @FXML
    private TextField su_lastName;

    @FXML
    private TextField su_email;

    @FXML
    private TextField su_phno;
    @FXML
    private ComboBox<String> su_question;

    @FXML
    private Button su_signupButton;

    @FXML
    private AnchorPane su_signupForm;

    @FXML
    private AnchorPane su_signupForm1;

    @FXML
    private TextField su_username;

    @FXML
    private Button side_allHave;
    private User loggedInUser;

    //method for slide animation
    @SuppressWarnings("exports")
     public void slideForm(ActionEvent event) {

     TranslateTransition sliderform = new TranslateTransition();

     if(event.getSource() == side_createButton){
          sliderform.setNode(side_form);
          sliderform.setToX(300);
          sliderform.setDuration(Duration.seconds(.5));

          sliderform.setOnFinished((ActionEvent e) -> {
               side_allHave.setVisible(true);
               side_createButton.setVisible(false);
          });
          sliderform.play();
     }else if(event.getSource() == side_allHave){
          sliderform.setNode(side_form);
          sliderform.setToX(0);
          sliderform.setDuration(Duration.seconds(.5));
          sliderform.setOnFinished((ActionEvent e) -> {
               side_allHave.setVisible(false);
               side_createButton.setVisible(true);
          });
          sliderform.play();
     }
    }


//     SignUp Helper Functions

//     Method to check if all fields are entered
    private boolean checkFields() {
     if(su_firstName.getText().isEmpty() ||
                su_lastName.getText().isEmpty() ||
                su_email.getText().isEmpty() ||
                su_phno.getText().isEmpty() ||
                su_username.getText().isEmpty() ||
                su_password.getText().isEmpty() ||
                su_answer.getText().isEmpty()) {
          CommonMethods.showAlert("Enter All The Fields");
          return false;
     } else if(su_question.getValue()==null) {
          CommonMethods.showAlert("Please Select A Question
Before Setting An Answer");
          return false;
     }
     return true;
    }

//    Method to check if email is valid
    private boolean isValidEmail(String email) {
        // Define the pattern for a valid email address
        String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\.[a-zA-Z0-
9_+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";
        // Create a Pattern object
        Pattern pattern = Pattern.compile(emailRegex);
        // Create a Matcher object
        Matcher matcher = pattern.matcher(email);
        // Return true if the email matches the pattern
        return matcher.matches();
    }

//    Method to perform form validation
    private boolean performFormValidation() {
     String email = su_email.getText();
     String userName = su_username.getText();
     String password = su_password.getText();
     if(!isValidEmail(email)) {
          CommonMethods.showAlert("Enter a valid email
address");
          return false;
     }else if(userName.length()<5) {
          CommonMethods.showAlert("Username should by greater
than 5 characters");
          return false;
     }else if(password.length()<6) {
          CommonMethods.showAlert("Password length should be
greater than 6");
          return false;
     }
     return true;

    }

//    Method to check if userName already exists
    private boolean checkUserNameExists() {
     if(UserHandler.checkIfUserExists(su_username.getText())) {
          CommonMethods.showAlert("User Name is Already Taken");
          return true;
     }
     return false;
    }

    private User getUserDetails() {
     User user = new User();
     user.setFirstName(su_firstName.getText());
     user.setLastName(su_lastName.getText());
     user.setEmail(su_email.getText());
     user.setPhoneNo(su_phno.getText());
     user.setUserName(su_username.getText());
     user.setPassword(su_password.getText());
     user.setSeqQuestion(su_question.getValue());
     user.setSeqAnswer(su_answer.getText());


     return user;
    }


//     Login Helper Functions
//     Methods which checks entered login details and returns a
boolean if details are valid
    public boolean performLoginChecks(String userName, String
password) {
     if(!UserHandler.checkIfUserExists(userName)) {
          CommonMethods.showAlert("User Name Does Not Exists");
          return false;
     }else if(!UserHandler.checkPasswordMatch(userName,
password)){
          CommonMethods.showAlert("Invalid Password");
          return false;
     }
     return true;
    }

//     Method to redirect to HomePage after successful login
    private void redirectToHomePage() {
     si_loginButton.getScene().getWindow().hide();
     Stage stage = new Stage();
     try {
               FXMLLoader fxmlloader = new FXMLLoader();

     fxmlloader.setLocation(getClass().getResource("/fxml/Dashbo
ard.fxml"));

               Parent root = fxmlloader.load();
               HomePageController hpController =
fxmlloader.getController();
               hpController.setUserHomePage(loggedInUser);

          Scene scene = new Scene(root);

          stage.setScene(scene);
          stage.show();
     }catch(Exception e) {
          e.printStackTrace();
     }
    }


    @FXML
    void loginSubmit(ActionEvent event) {
     String enteredUserName = si_username.getText();
     String enteredPassword = si_password.getText();
     if(performLoginChecks(enteredUserName,enteredPassword)) {
          CommonMethods.showAlert("Login Successful");

          loggedInUser =  UserHandler.getUser(enteredUserName);
          redirectToHomePage();
     }
    }
    @FXML
    void signUpSubmit(ActionEvent event) {
     if(checkFields() && performFormValidation() &&
!checkUserNameExists()) {
          String message =
UserHandler.addUser(getUserDetails());
          CommonMethods.showAlert(message);
     }

    }

    //set visibility of the anchorpane 1
    @FXML
    void nextHandler(ActionEvent event) {
     su_signupForm1.setVisible(true);
    }

    //turn off the visibility of the second anchor pane to
accommodate the next slide for logging in
    @FXML
    void goBack(ActionEvent event) {
     su_signupForm1.setVisible(false);
    }

//     Text Formatter for phonenumber;
    TextFormatter<String> phoneNo = new TextFormatter<>(change -
> {
        if (!change.getText().matches("[\\d+]*")) {
            change.setText(""); // Remove non-numeric characters
        }
        return change;
    });

     @Override
     public void initialize(URL arg0, ResourceBundle arg1) {

          su_phno.setTextFormatter(phoneNo);

          String[] comboquestions = {
                    "What is your first pet?",
                    "Who is your your first ex?",
                    "What is the name of your Elementary
School?",
                    "What is your Mother's maiden name?",
                    "What is your favourite Color?"
          };
          su_question.getItems().addAll(comboquestions);


     }



}
```

**DataHandlers:**

**DbConnection:**

```
package DataHandler;
import org.bson.Document;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

@SuppressWarnings("unused")
public class DbConnection {
//   URI of the mangodb cluster of our database
```

```
     private static String uri =
"mongodb+srv://eerantia:eerantia@testcluster.pu9tsrc.mongodb.net
/?retryWrites=true&w=majority";
     private static MongoClient mongoClient;

//   Function to start connection to our database and returns
database object, on which we can make our calls
    public static MongoClient startConnection() {
     if(mongoClient==null) {
          try {
               System.out.println("Client instance is getting
created");
               mongoClient = MongoClients.create(uri);
                return mongoClient;
          }catch(Exception e) {return null;}
     }else {
          System.out.println("Client instance is already
created");
          return mongoClient;
     }
    }

//    call this function when user is logged out
    public static void endConnection() {
     if (mongoClient != null) {
            mongoClient.close();
        }
    }
}
```

**UserHandler:**

```
package DataHandler;
import static com.mongodb.client.model.Filters.eq;

import java.util.ArrayList;
import java.util.List;

import org.bson.Document;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

import model.Recipe;
import model.User;
```

```
@SuppressWarnings("unused")
public class UserHandler {
//	A mongoClient variable which refers to same object whole
session
	private static MongoClient mongoClient =
DbConnection.startConnection();

//	Method to get users collection from database
	public static MongoCollection<Document> getCollection(){
	MongoDatabase database = mongoClient.getDatabase("recipe-
javafx");
	    System.out.println("Connected to database:\n" +
database);
	    MongoCollection<Document> collection =
database.getCollection("users");
	    return collection;
	}

//  Method to get a user from database, it take userId as
parameter, returns a Recipe object
	public static User getUser(String userName) {
		MongoCollection<Document> collection =
getCollection();
		Document userDoc = collection.find(eq("userName",
userName)).first();

		if(userDoc== null) {
			return null;
		}
		User resultUser = docToUser(userDoc);
		return resultUser;
	   }

//	Method to return all the users object as User List
	public static List<User> getAllUsers(){
		MongoCollection<Document> collection =
getCollection();
		MongoCursor<Document> cursor =
collection.find().iterator();

		List<User> userList = new ArrayList<>();
		try {
		  while (cursor.hasNext()) {
			Document userDocument = cursor.next();
			// Convert the Document to a User object
			User user = docToUser(userDocument);
			// Add the User object to the list
```

```
                userList.add(user);
            }
        } finally {
            cursor.close();
        }
        return userList;
    }

//   Method to add user to DB, takes user object as input and
returns a message
     public static String addUser(User user) {
          String message;
          MongoCollection<Document> collection =
getCollection();
          try {

     if(collection.find(eq("userName",user.getUserName())).first
()!=null){
                    return "A User With Same UserName is already
present, please change the userId and try again";
               }
               else {
                    Document userDocument = userToDoc(user);
                    collection.insertOne(userDocument);
                    return "Account Created SuccessFully";
               }
          }catch(Exception e) {
          return "An error has occured while processing your
request";
     }
     }

//  Method to update User, Takes user name and updated user
object and returns meessage
     public static String updateUser(String userName, User user)
{
          MongoCollection<Document> collection =
getCollection();
          Document userDoc = userToDoc(user);
     Document filter = new Document("userName", userName);
     try{
          collection.findOneAndUpdate(filter, new
Document("$set",userDoc));
          return "User Update Successfull";
     }catch(Exception e) {
          return e.getMessage();
     }
     }

//   Method to delete a user from DB, takes user name as input
and retu
     public static String deleteUser(String userName) {
          MongoCollection<Document> collection =
getCollection();
     String message;
     try {
          Document result =
collection.findOneAndDelete(eq("userName",userName));
          if(result == null) {
               message = "No such recipe found to delete";
          }
          else {
               message = "Delete Successfull";
          }
     }catch(Exception e){
          message = e.getMessage();
     }

     return message;
     }

//   Method to check if user exists, takes user name as input
and return boolean
     public static boolean checkIfUserExists(String userName) {
          MongoCollection<Document> collection =
getCollection();
     if( collection.find(eq("userName",userName)).first() !=
null ) {
          return true;
     }
     return false;
     }

//   Method to check if password matches, takes user name as
passsword as input

     public static boolean checkPasswordMatch(String userName,
String password) {
          MongoCollection<Document> collection =
getCollection();
          Document userDoc =
collection.find(eq("userName",userName)).first();
          User user  = docToUser(userDoc);
          if(user.getPassword().matches(password)) {
               return true;
          }
          return false;
     }




//Helper funcitons
     private static Document userToDoc(User user) {
         Document userDocument = new Document()
                 .append("firstName", user.getFirstName())
                 .append("lastName", user.getLastName())
                 .append("email", user.getEmail())
                 .append("phoneNo", user.getPhoneNo())
                 .append("userName", user.getUserName())
                 .append("password", user.getPassword())
                 .append("seqQuestion", user.getSeqQuestion())
                 .append("seqAnswer", user.getSeqAnswer())
                 .append("recipeIds", user.getRecipeIds());

         return userDocument;
}

@SuppressWarnings("unchecked")
     private static User docToUser(Document userDocument) {
         User user = new User(); // Assuming User is your model
class for User

         if (userDocument != null) {

user.setFirstName(userDocument.getString("firstName"));

user.setLastName(userDocument.getString("lastName"));
             user.setEmail(userDocument.getString("email"));
             user.setPhoneNo(userDocument.getString("phoneNo"));

user.setUserName(userDocument.getString("userName"));

user.setPassword(userDocument.getString("password"));

user.setSeqQuestion(userDocument.getString("seqQuestion"));

user.setSeqAnswer(userDocument.getString("seqAnswer"));
             user.setRecipeIds((List<String>)
userDocument.get("recipeIds"));
         }
         return user;
     }

//   External call actor to test out the method, to be deleted
     public static void externalCallActor() {
          User exampleUser = new User();
        exampleUser.setFirstName("Abhinav");
        exampleUser.setLastName("Chary");
        exampleUser.setEmail("abhinav.doe@example.com");
        exampleUser.setPhoneNo("+616161");
        exampleUser.setUserName("abbu1502");
        exampleUser.setPassword("secretpassword");
        exampleUser.setSeqQuestion("What is your favorite
color?");
        exampleUser.setSeqAnswer("Blue");

        // Adding an example recipe id to the user
        exampleUser.addRecipeId("recipe122");
        exampleUser.addRecipeId("recipe63");

        System.out.println(addUser(exampleUser));


     }


     public static void main(String[] args) {
//              externalCallActor();


     }

}
```

## VII. DISCUSSION:

**Problems Faced:** Traditional relational databases may not always be the best fit for modern, flexible data structures, leading to complications in data modeling and retrieval.

As your application scales, coordinating and managing concurrent code execution becomes challenging, especially when dealing with complex operations.

Redundant code can lead to maintenance issues, code duplication, and increased chances of errors.

JavaFX UI might lack modern and visually appealing icons, making it challenging to create an aesthetically pleasing user interface.

**Overcame Methods:** To bridge the gap between our JavaFX application and MongoDB, we employed the MongoDB Java driver. This driver served as the key facilitator in seamlessly integrating MongoDB into our JavaFX architecture. It allowed us to interact with the database using Java, making the data access layer of our application more intuitive and efficient.

## VIII. CONCLUSION AND FUTURE WORK:

**Conclusion:** In conclusion, the envisioned Java-based mobile application presents a comprehensive solution to the culinary challenges encountered by students, particularly new immigrants. This innovative tool not only offers a user-friendly interface and an extensive recipe repository but also proposes a significant enhancement to the user experience through the incorporation of video tutorials. By featuring visual guidance, especially from cherished chefs like mothers, the app strives to make the cooking process more intuitive and enjoyable. This addition aligns with the project's core objective of fostering a sense of home in a foreign setting while promoting healthier eating habits and self-sufficiency.

Moreover, the application's emphasis on user personalization, diverse recipes, and interactive features sets it apart as a holistic tool for novice cooks. With step-by-step guidance and a community-driven platform, the app goes beyond being a mere recipe repository, empowering students to develop essential cooking skills and build connections with their cultural roots. In essence, the proposed Java-based mobile application not only addresses the practical aspects of cooking challenges but also enriches the overall well-being and nutritional development of students, contributing to their holistic growth.

**Advantages and Benefits of the Solution:**

1. **Nutritional Improvement:** The application addresses the risk of malnutrition by facilitating access to a wide range of recipes, promoting healthier eating habits among students.
2. **Enhanced Learning Experience:** The step-by-step cooking guides, coupled with interactive features, create an engaging and educational experience for users, fostering culinary skills and self-sufficiency.

3. **Community Building:** The platform encourages a sense of community by allowing users to share experiences, ask questions, and provide feedback, fostering a collaborative learning environment.
4. **Convenience for Recipe Contributors:** Parents and chefs benefit from a streamlined recipe upload process, contributing to the diversity and richness of the application's recipe database.

**Problems Found During Development but Not Explored:**

1. **User Accessibility:** While the proposal outlines feature for personalization, additional attention to accessibility features would ensure inclusivity for users with varying levels of cooking expertise and diverse needs.
2. **Security Measures:** As the application involves user profiles and community interactions, further exploration into data security and privacy measures is essential to protect user information.

**If the Team Has More Time, What to Improve:**

1. **Enhanced User Experience:** Invest additional time in refining the user interface and overall user experience to make the app more intuitive and visually appealing.
2. **Integration of Artificial Intelligence:** Explore the integration of AI algorithms to enhance recipe suggestions based on user preferences and provide more personalized recommendations.
3. **Offline Functionality:** Implement offline functionality to allow users to access recipes and cooking guides even without an internet connection.
4. **Gamification Elements:** Introduce gamification elements to make the learning process more engaging, such as achievement badges, challenges, or rewards for completing cooking tasks.
5. **Cross-Platform Compatibility:** Extend the application's compatibility to multiple platforms (iOS, Android) to cater to a broader user base.

In summary, the proposed application not only addresses immediate cooking challenges but also lays the foundation for continuous improvement, ensuring a dynamic and evolving solution that meets the evolving needs of its users.

## IX. JOB ASSIGNMENT:

**Abhishek (002837203):**
I played a pivotal role in the project, focusing on the design and functionality of login-related components. My contributions encompassed working on crucial elements such as LoginDesign, Login Page, Controllers, Dashboard, and Explore Food Section Recipes. I took an active role in creating and managing the following FXML files:

Account
AccountPage
Dashboard
FoodCard
HomePage
ListRecipes
LoginPage
RecipeDisplay
In terms of controllers, I specifically handled HomePageController, ListRecipesController, LoginController, HomePageController, PostRecipeController, and RecipeDisplayController.

**Abhinav (002883528):**
I served as a linchpin in the backend development of the application, delving into Controllers, DataBase, and DataHandlers. My contributions spanned various critical aspects of the project, and I actively participated in the creation and management of the following FXML files:

Account
AccountPage
Dashboard
FoodCard
HomePage
ListRecipes
LoginPage
PostRecipe
RecipeDisplay
Recipe-List
In the realm of controllers, I took charge of AccountController, CommonMethods, FoodCardController, HomePageController, ListRecipesController, LoginController, PostRecipeController, and RecipeDisplayController. Additionally, I managed DataHandlers, overseeing DbConnection, RecipeHandler, and UserHandler.

**Meghana (002642314):**
I played a pivotal role in crafting and implementing the Account Page, channeling my efforts into design and controllers. My specific involvement extended to the creation and management of the following FXML files:

Account
Dashboard
FoodCard
AccountPage
RecipeDisplay
Recipe-List
In terms of controllers, I handled AccountController, HomePageController, CommonMethods, HomePageController, and ListRecipesController.

Collectively, our distinct and valuable contributions ensured a well-rounded and cohesive application, bringing together expertise from different facets of the project.

## X. REFERENCES:

Every material provided in the canvas under csye6200 and additionally:
Transition Slides:
https://docs.oracle.com/javafx/2/api/javafx/animation/Transition.html

List of animation packages:
https://docs.oracle.com/javase/8/javafx/api/toc.html

Font awesome icon references:
https://www.w3schools.com/icons/icons_reference.asp

JavaFX CSS References:
https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

JavaFX Controllers:
https://docs.oracle.com/javafx/2/fxml_get_started/custom_control.htm