MACRO MEALS DESIGN MANUAL 2022

SUPERVISOR
Chris Meudec

STUDENT
Mohsin Tahir (C00250220)

DATE
17-04-2023

Table of Contents

Table of Figures

Introduction

The purpose of this document is to go further into the design of the Macro Meals application. This document will include multiple diagrams to further outline the technical design of the project. These include System Sequence Diagrams, ER Diagram and GUI screenshots.

System Sequence Diagrams - These will outline how each object entity in the application will communicate with each other

ER (Entity Relationship) Diagram - This will describe the database table structure and relationship.

Finally, this document will include the GUI screens for each use case for a further understanding of the application.

System Sequence Diagrams

Receive Personalised Recipe

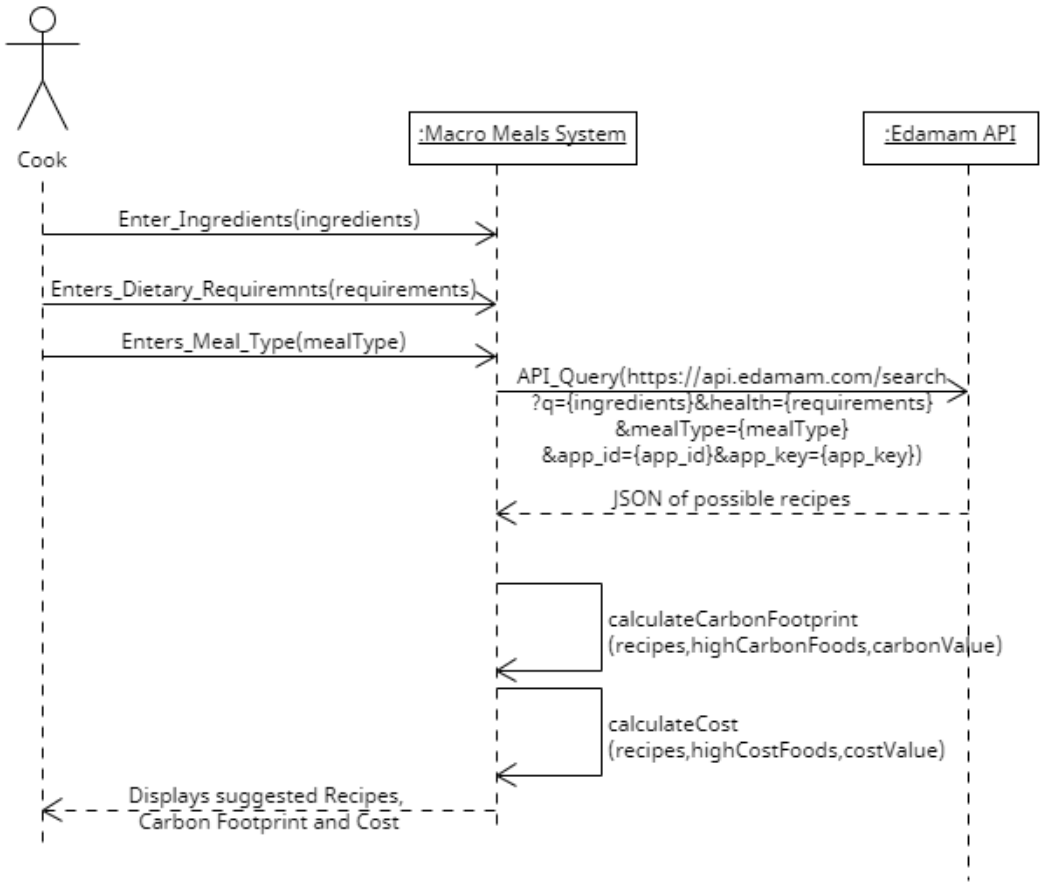

Figure 1. Receive Personalised Recipe System Sequence Diagram.

In the System Sequence Diagram above the Cook enters in available ingredients (ingredients), any dietary requirements (requirements) and the meal type (mealType). This is then passed to the Edamam API in an API query. The query consists of 5 components, 'q' is the list of ingredients the cook has entered. The 'health' is any dietary requirement the cook has. The 'mealType' is the type of meal the cook wishes for. The app_id and app_key are the values used to get authorisation to make the query call to the API. All the parameters in the query are passed in as strings.

The Macro Meals application calculates the carbon footprint of a recipe. This is done by taking in the list of recipes returned by the Edamam API stored in the variable 'recipes', a

list of food items that are high in carbon emissions stored in a variable called ‘highCarbonFoods’ and the carbon emissions of each of the food items stored in a variable called ‘carbonValue’. This algorithm matches the recipe ingredients with the food items in the ‘highCarbonFoods’ list. If there is a match then it extracts the measurement of the ingredient and multiplies it by its carbon value held in the ‘carbonValue’ list. This value is held in a variable called ‘total’. After iterating through each ingredient in the recipe it then divides the total by the recipe serving size if it's greater than 4. If the total is greater than 30 the recipe is deemed to have a high carbon footprint and a red traffic light is shown, if it is less than 30 the recipe is carbon friendly and a green traffic light is shown.

Finally, Macro Meals estimates the cost of the recipe. This is done in a similar fashion to the carbon footprint. The list of recipes stored in a list called ‘recipes’, a list of worldwide high-cost food items stored in a list called ‘highCostFoods’ and their cost per kg stored in a list called ‘costValue’. The algorithm matches the recipe ingredients with the food items in the ‘highCostFoods’ list. If there is a match then it extracts the measurement of the ingredient and multiplies it by its cost value and added to the total. If the serving size is greater than 4, this total is divided by the serving size. If the value is greater than 50 it is deemed as a high-cost recipe, if it's between 25 and 50 is an average-cost recipe and anything lower is a low-cost recipe.

Register

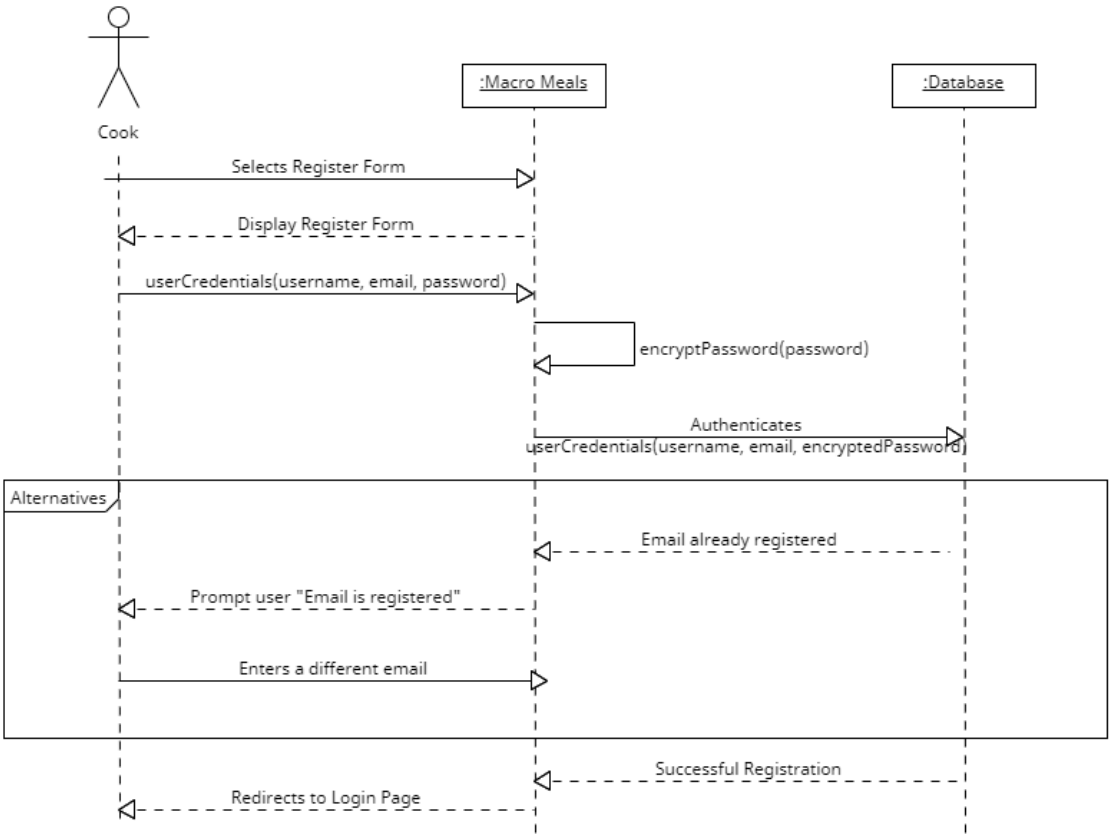

Figure 2. Register System Sequence Diagram

In the diagram above the cook selects the register form and enters a username, email and password. The application encrypts the password before storing it in the database. The passwords are encrypted using the bcrypt Python library. The passwords are encrypted to protect user accounts. If there was a data leak the passwords will not be in plain text and the correct hashing will be needed to decrypt them. If the email provided exists the user is prompted with an error message. If the email is valid then the account is created and a record is added to the database.

Login

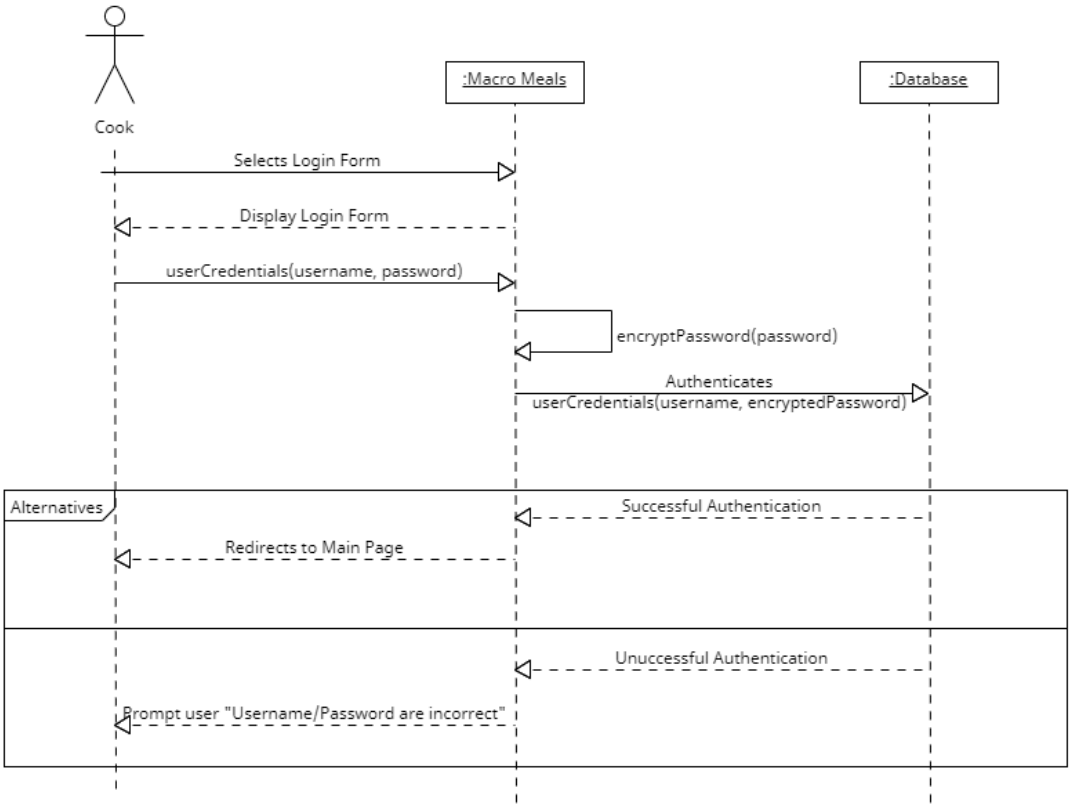

Figure 3. Login System Sequence Diagram

In the diagram above the cook selects the login button and a form is displayed back. The cook enters their credentials which consist of username and password. The password is encrypted and passed into the authenticate function. If these are correct the user is then directed to the main page, if not an error message is displayed.

Reset Password

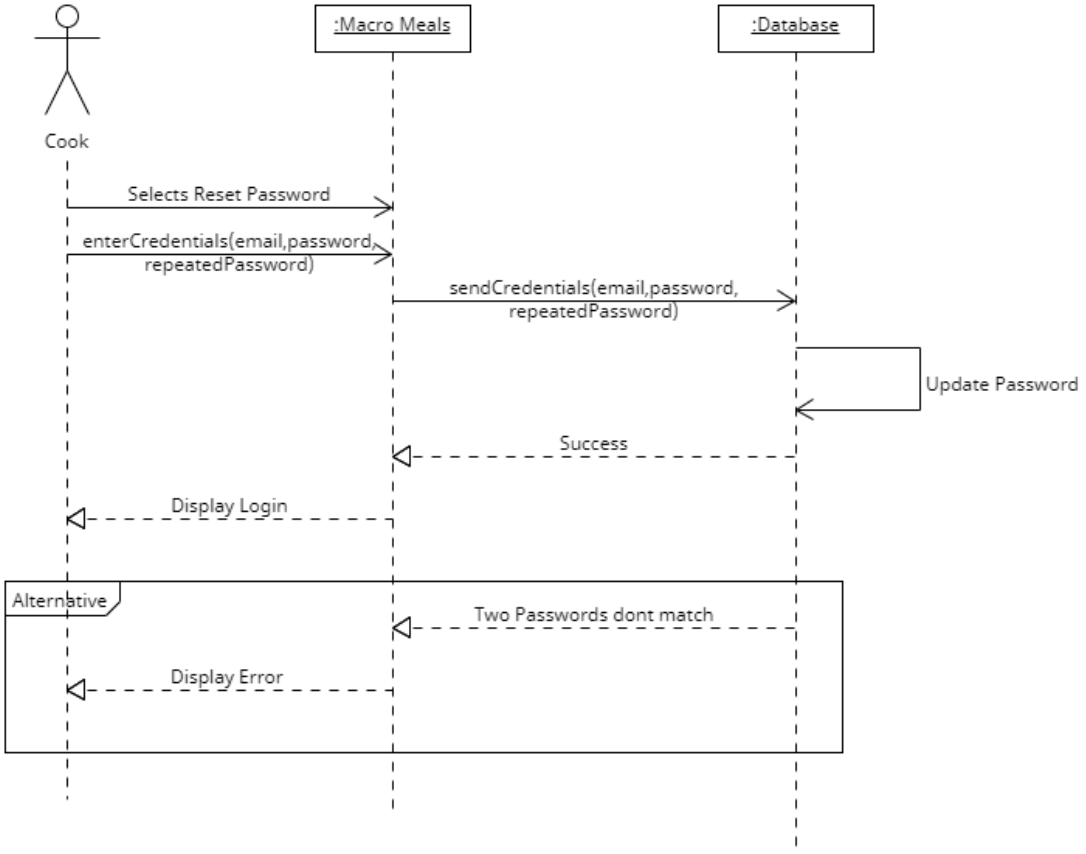

Figure 4. Reset Password System Sequence Diagram

In this diagram the cook selects to reset their password. They enter their email address and the new password they wish. They also repeat the new password. If the two passwords match the database is updated. If they do not match then the cook is alerted with an error message stating so.

Logout

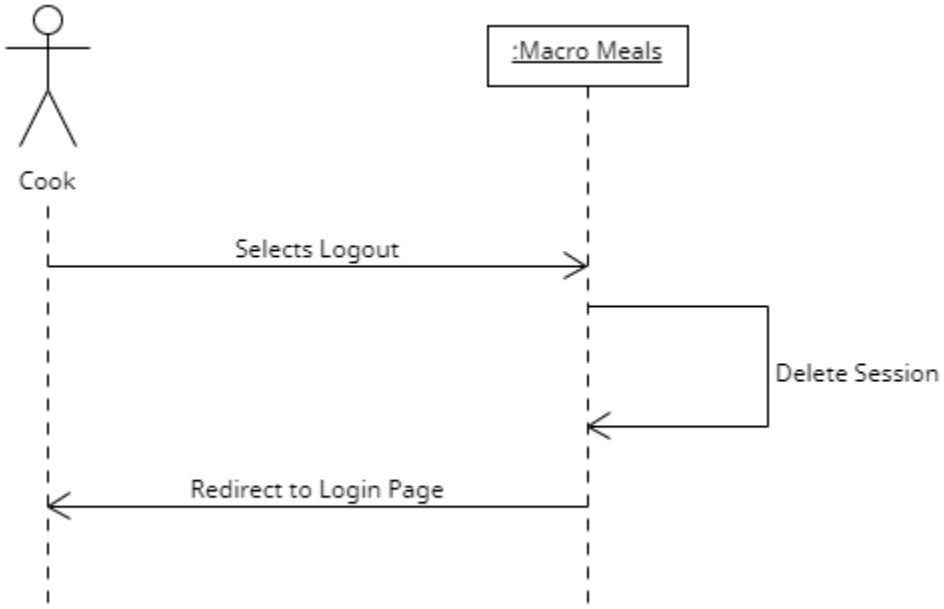

Figure 5. Logout System Sequence Diagram

In this diagram, the cook clicks the logout button and the application deletes the session and logs the user out. They are then redirected back to the login page.

Save Recipe

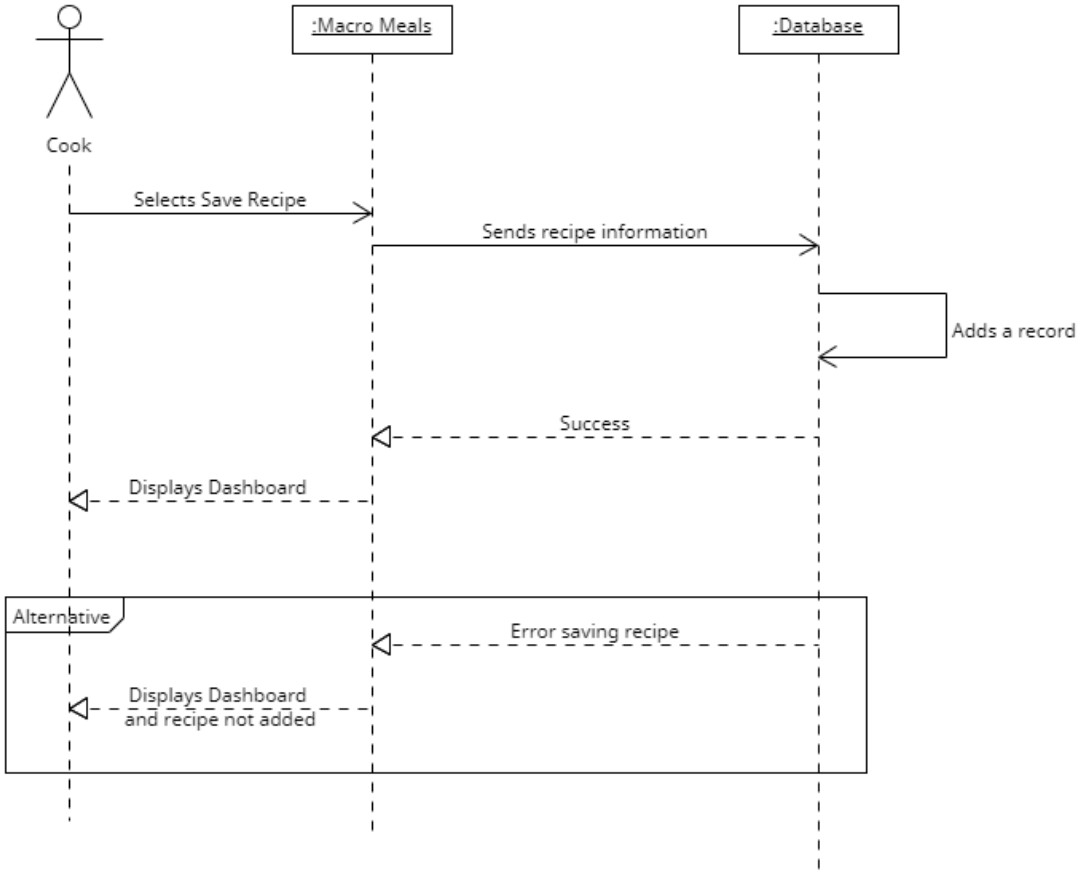

Figure 6. Save Recipe System Sequence Diagram

In this diagram, the cook selects the Save Recipe button. The recipe information is sent to the database and a record is added. If the recipe exists then it is not duplicated and the cook is redirected to the saved recipe's dashboard. No image is saved due to there being a limitation on the API side. There is a 1-hour time limit on the generated image for the recipe card. After 1 hour the image expires and a new one must be generated. Therefore no image gets recorded in the database.

Delete Recipe

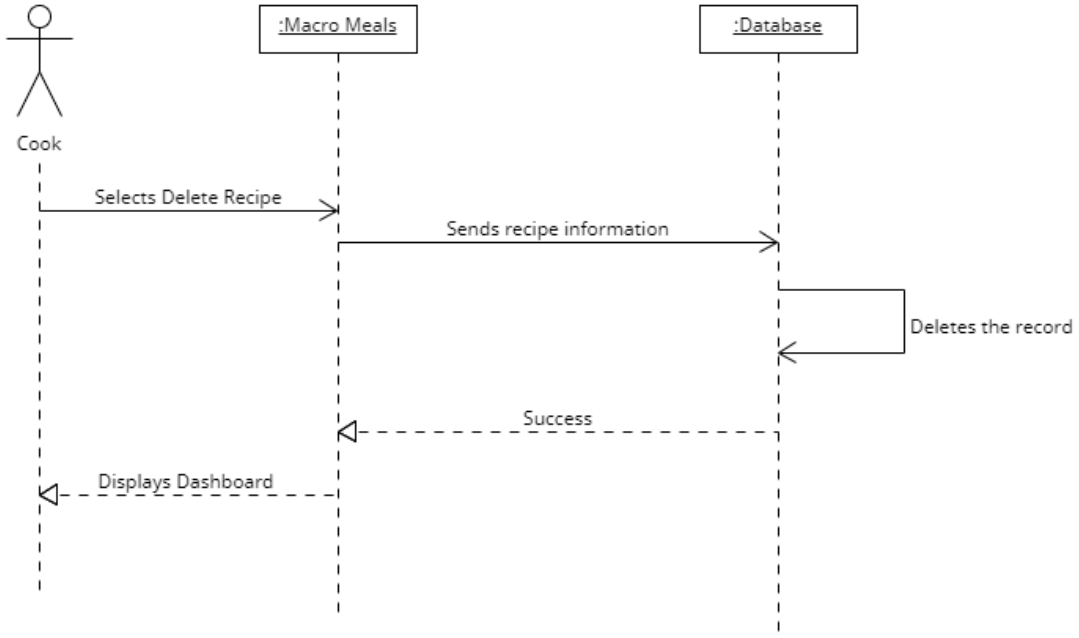

Figure 7. Delete Recipe System Sequence Diagram

In Figure 6 the cook selects to delete a recipe. The recipe information is sent to the database. The database finds this recipe and deletes it from its records. The new and updated saved recipes page is displayed back to the cook.

View Saved Recipes

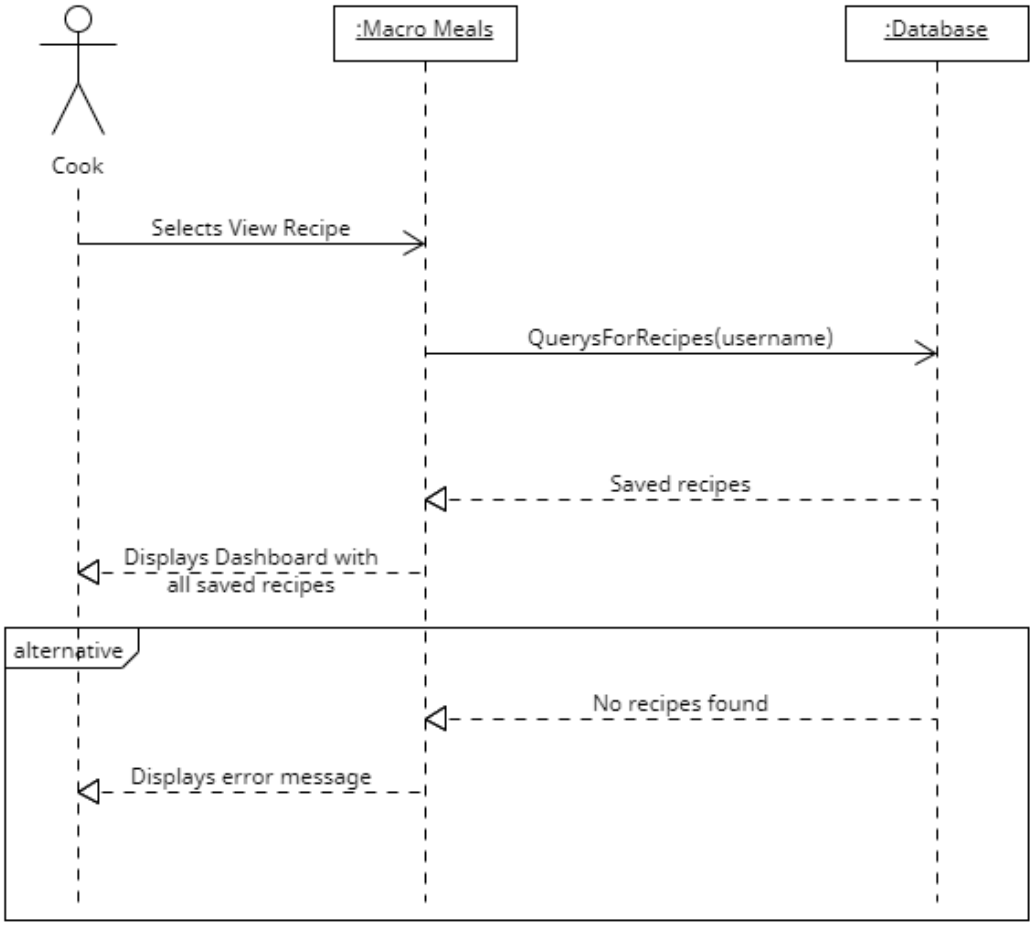

Figure 8. View Saved Recipes System Sequence Diagram

In this diagram the cook selected the view recipes button. The application queries the database for any recipes that match the username. If found they are displayed back to the screen. If none are found an error message alerting the user that there are no saved recipes is returned.

Ask ChatGPT

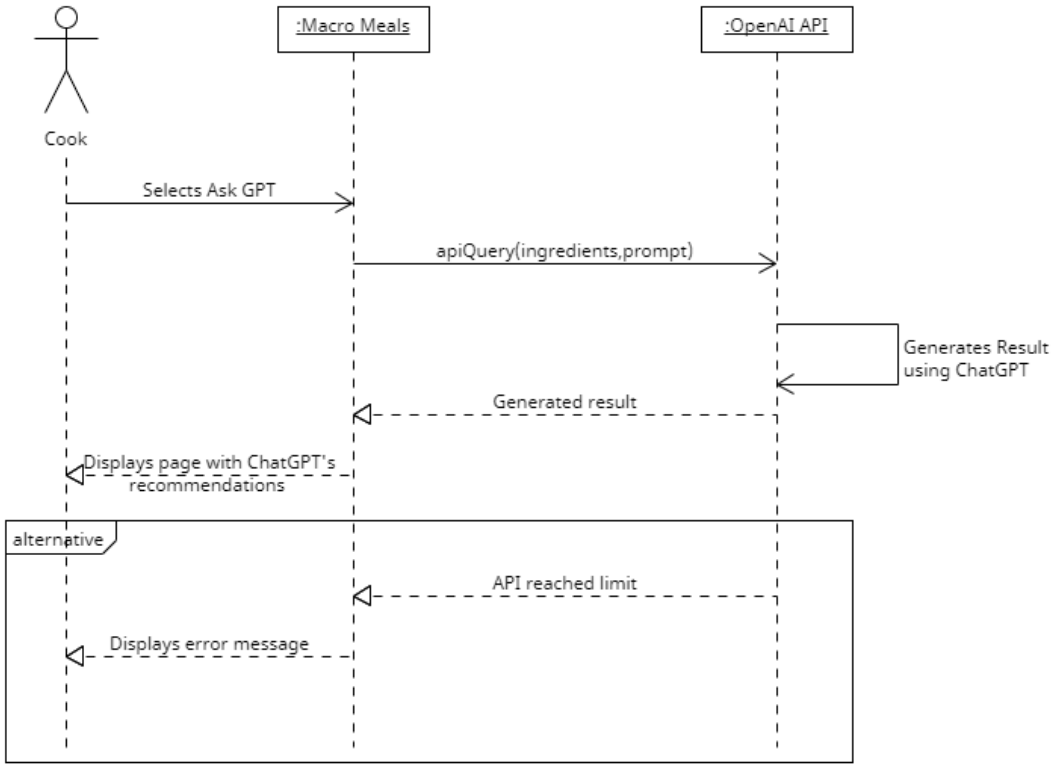

Figure 9. Ask ChatGPT System Sequence Diagram

In this diagram the cook selects the Ask ChatGPT. The application passes the list of ingredients and a prompt to the OpenAI API. The API uses the ChatGPT model to generate an answer and is displayed to the user. The generated solution consists of carbon-friendly food item alternatives for a particular recipe. If the API has reached its monthly limit then an error message is displayed back to the user.

Prompt used: “Print this list of ingredients with carbon footprint-friendly alternatives. Only print a list no explanation please.”

Algorithms

Detecting 404 Pages

If the cook wishes to get the cooking instructions of a recipe they are redirected to an external site. Sometimes the website is not available anymore and returns a 404 page not found error. In my code, I have the retrieve recipes function to send a request to all the

URLs and filter out any 404 status codes returned. This method is effective in removing dead links but sometimes pages return a 200 status code stating the URL is active but instead, it has an image of 404 on it stating that the recipe has been removed. As of right now, there is no workaround for this. One possible method of detecting the image of 404 is using an AI model which is trained to do so but due to time constraints, this is not possible to develop and implement.

GUI Screens

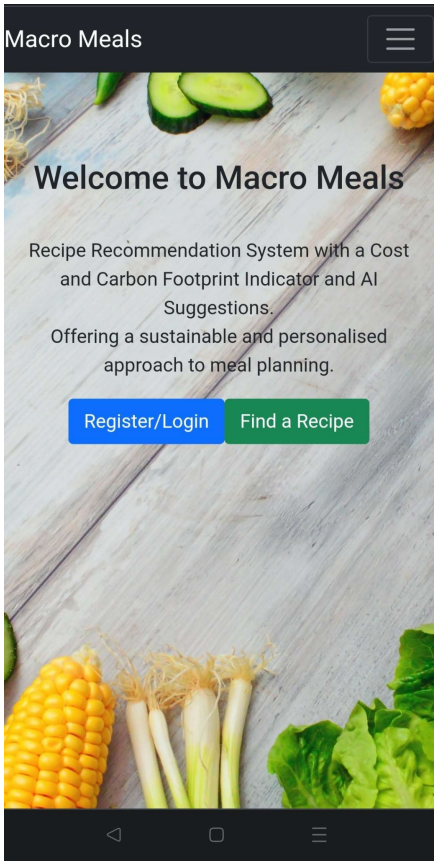

Figure 10. Macro Meals Home Page

This is the home page. It's the first page the cook visits, they can either log in to their account or go straight to searching for recipes.

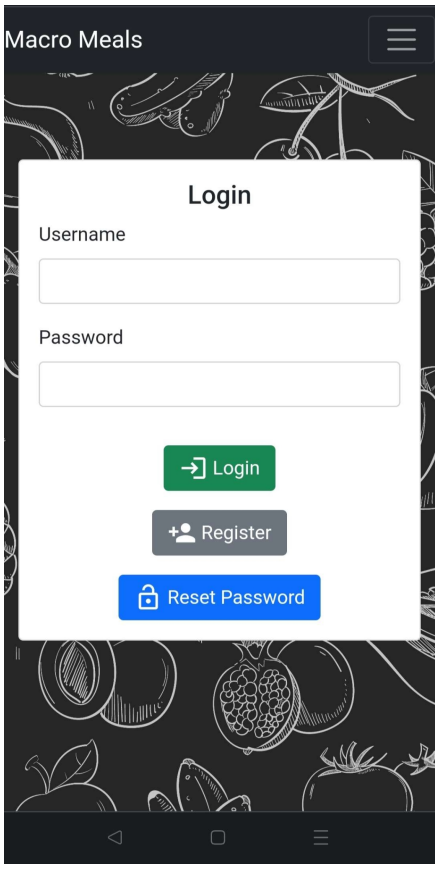

Figure 11. Login Page

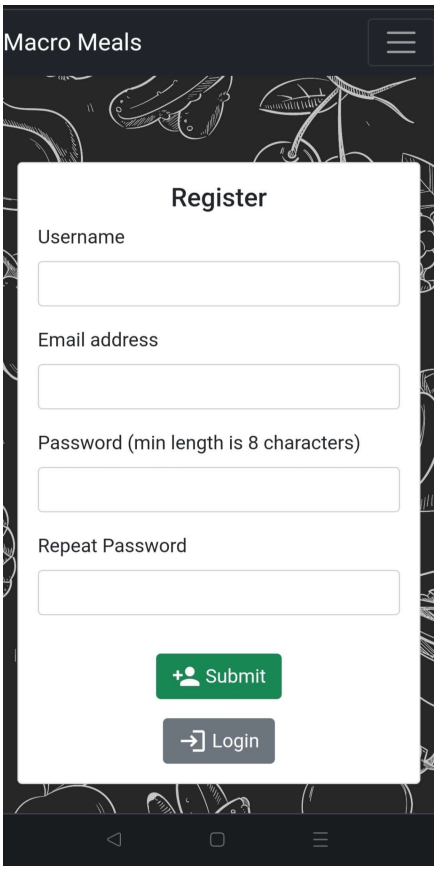

Figure 12. Register Page

On these screens the cook can log in to an existing account by providing their username and password. They can also register for an account by providing a username, email address and password. By registering/logging in the cook can save recipes for future use.

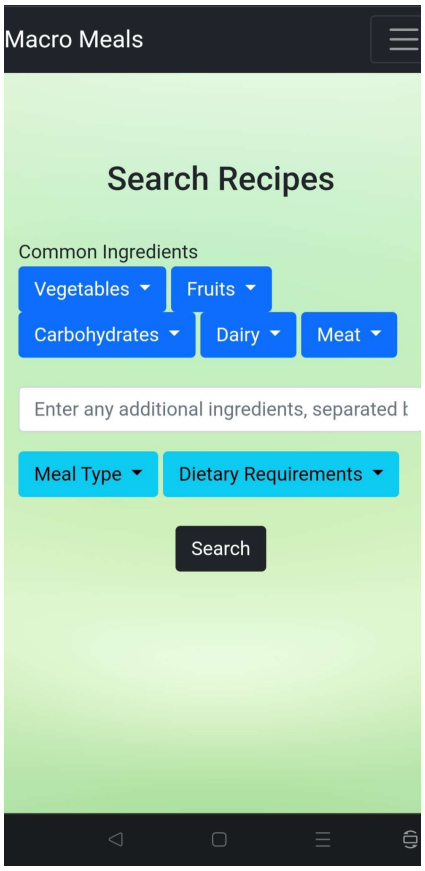

Figure 13. Ingredients Input Page

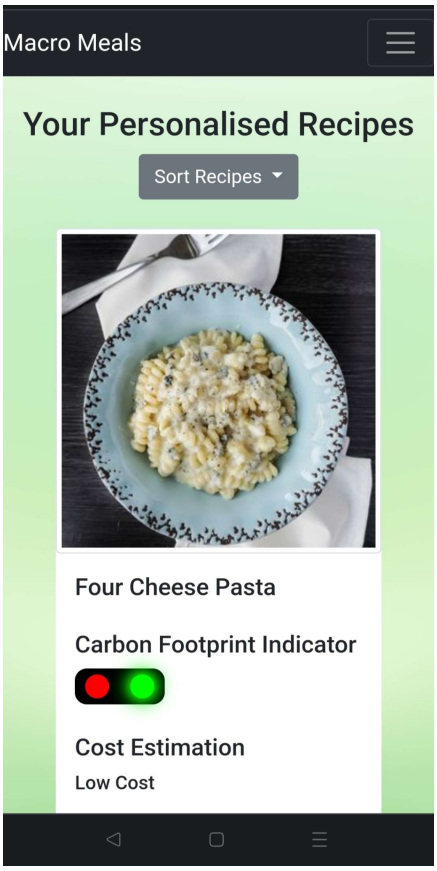

Figure 14. Recipe Results Page

In Figure 13 the screen is shown when a cook wishes to search for a recipe. They can input available ingredients, meal types and any dietary requirements. In Figure 14 all the possible recipes that the user can cook are shown.

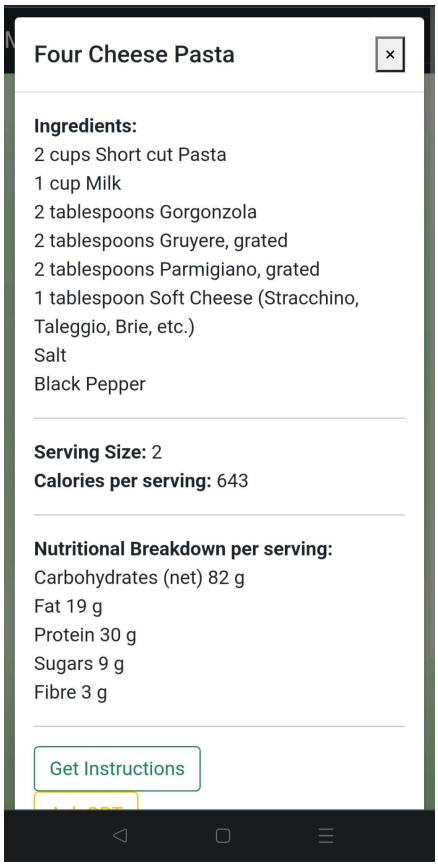

Figure 15. Recipe Breakdown Page

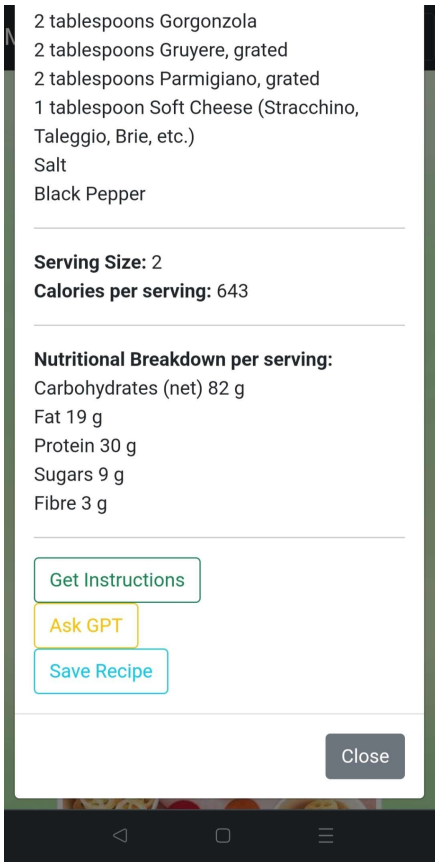

Figure 16. Recipe Breakdown Page Extended

This screen is shown when the cook wishes to get more information about a certain recipe.

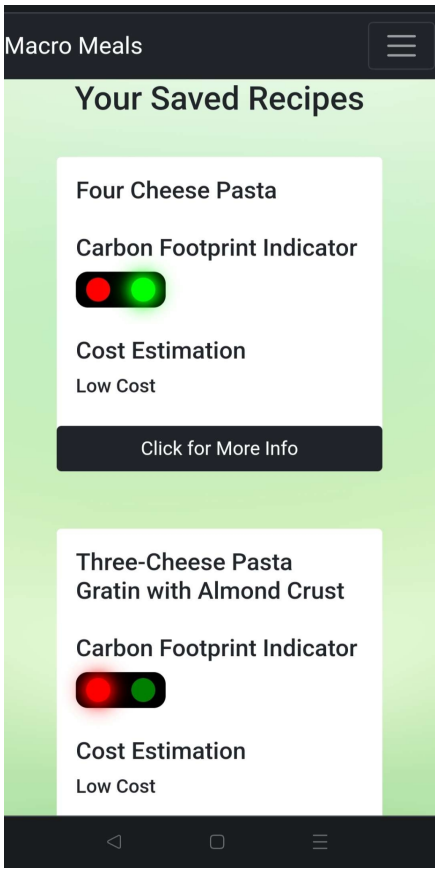

Figure 17. Saved Recipes Page

Figure 18. ChatGPT Recommendation Page

In Figure 17 all the user's saved recipes are shown. They can use this for quick access or future use. When the user selects the "Ask ChatGPT" the recommendations are shown in Figure 18.

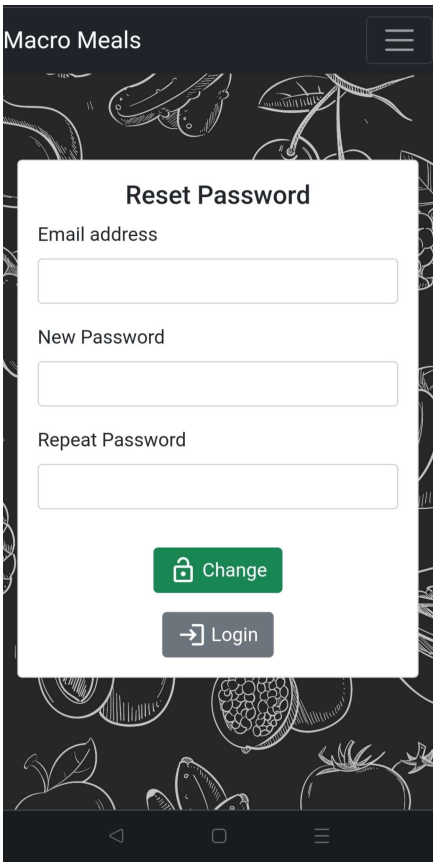

Figure 19. Reset Password Page

This screen is shown when the user wishes to reset their password.

Database Design

A SQL database is used to store usernames, emails, passwords and saved recipes. This is used to authenticate the cook before using the application and it will allow the cook to save any recipe. The passwords are stored using encryption.

ER Diagram

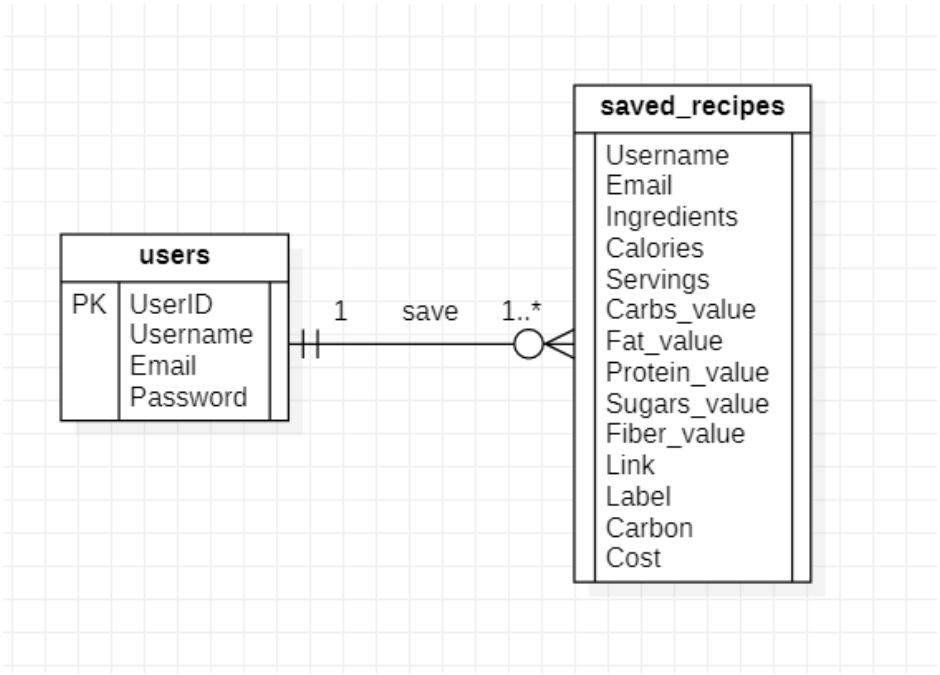

Figure 20. Database ER Diagram

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**UserID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**Username**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	3	**Email**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	4	**Password**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More

Figure 21. Users Table Database Design

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**Username**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	2	**Email**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	3	**Ingredients**	varchar(5500)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	4	**Calories**	int(255)			No	*None*			Change Drop More
☐	5	**Servings**	int(255)			No	*None*			Change Drop More
☐	6	**Carbs_value**	int(255)			No	*None*			Change Drop More
☐	7	**Fat_value**	int(255)			No	*None*			Change Drop More
☐	8	**Protein_value**	int(255)			No	*None*			Change Drop More
☐	9	**Sugars_value**	int(255)			No	*None*			Change Drop More
☐	10	**Fiber_value**	int(255)			No	*None*			Change Drop More
☐	11	**Link**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	12	**Label**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	13	**Carbon**	varchar(55)	*utf8mb4_general_ci*		No	*None*			Change Drop More
☐	14	**Cost**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More

Figure 22. Recipes Table Database Design