UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING, UTM

SEMESTER I, SESSION 2023/2024

PROJECT PHASE 3: DATABASE

LOGICAL DESIGN

SECD2523: DATABASE

SECTION 02

NAME :

1. AINA NURAIN BINTI MOHD AROFF B23CS0022
2. DANESH MUTHU KRISNAN B23CS0034
3. MUHAMMAD FIRDAUS BIN MOHIDEEN ABDUL CADER B23CS0056

COURSE : BACHELOR OF COMPUTER SCIENCE (COMPUTER NETWORKS AND SECURITY)

SUBMISSION DATE : 16 - JANUARY - 2024

LECTURER'S NAME : DR. IZYAN IZZATI BINTI KAMSANI

TABLE OF CONTENTS

1.0 Introduction

Following phase 2, we will continue to proceed with the next phase of database design which is logical database design for phase 3 of this project. The main purpose of this phase is to determine the logical data structures that are required to support the storage of our "Food Recipe Finder" system's data systems. This will allow us to create a final database that will contain well structured tables that properly reflect our system's business environment.

Logical database design bridges the 2 other phases of database design, which are the conceptual and physical design levels. In this phase, we will be developing a more detailed description of our previous conceptual model and focusing on refining our database's data structures, relationships and constraints. We will be defining attributes for each entity, specifying their data types, primary keys, foreign keys, as well as normalizing our data to ensure data integrity and to eliminate any potential insertion, deletion, and update anomalies in our database. This phase is crucial to the implementation of a database as the consequences of an incomplete or flawed logical design to the means of data collection, storage, and data integrity will be very costly. A well-planned logical design could curb any of those potential risks, as well as allow us to successfully implement and test our database in the next phase.

In a nutshell, this report will be focusing on translating our previous conceptual model into a more detailed representation by specifying the attributes, keys, and relationships for the data that will be stored in our "Food Recipe Finder" system. The completion of this phase can help to ensure data independence, physical database flexibility, data integrity, and also user satisfaction. By completing all three levels of database design, we can make sure that our system's data can stay organized, easily accessible, and valuable to people who may access them.

2.0 Overview of Project

Our “Food Recipe Finder” system offers a convenient and easy way to share and save recipes, search for recipes and make connections with other users. This report aims to deliver a comprehensive layout of our database’s logical design which includes a few points such as our updated business rules which were derived from our proposed business rules in phase 2. Next are our system’s conceptual and logical ERDs, which intend to visualize data structures, relationships and constraints involved in our database. The way to achieve a concise logical ERD is by conducting 1NF, 2NF, 3NF and BCNF (Boyce Codd’s Normal Form) normalization on our conceptual ERD in order to prevent any potential update, delete or insertion anomalies present in our database. Once completed with those steps, we then moved on to updating our data dictionary. We also included our relational DB schemas which were derived from the completed logical ERD as well as SQL statements consisting of DML (Data Manipulation Language) and DDL (Data Definition Language) statements. All of these deliverables aim to showcase the steps we have taken to complete our “Food Recipe Finder” project’s phase 3 in a comprehensive manner. In order to summarize this report, we also proceeded to lay out the key advantages and outcomes that will be gained with the implementation of this system, showcasing the potential for users to go through an enjoyable and efficient user experience while using our “Food Recipe Finder” system.

3.0 Database Conceptual Design

3.1 Updated Business Rules

1. Only one recipe has one to many recipe courses.
2. Only one recipe has one to many recipe food categories.
3. Only one recipe has one to many cuisines.
4. Only one recipe has one to many levels.
5. Only one course has one to many recipe courses.
6. Only one food category has one to many recipe food categories.
7. Only one user finds zero to many recipes.
8. Only one recipe has one to many recipe nutritional information.
9. Only one recipe has one to many recipe ingredients.
10. Only one recipe has one to many menu recipes.
11. Only one nutritional information has one to many recipe nutritional information.
12. Only one measurement is included in one to many recipe ingredients.
13. Only one ingredient is included in one to many recipe ingredients.
14. Only one menu has one to many menu recipes.

3.2 Conceptual ERD

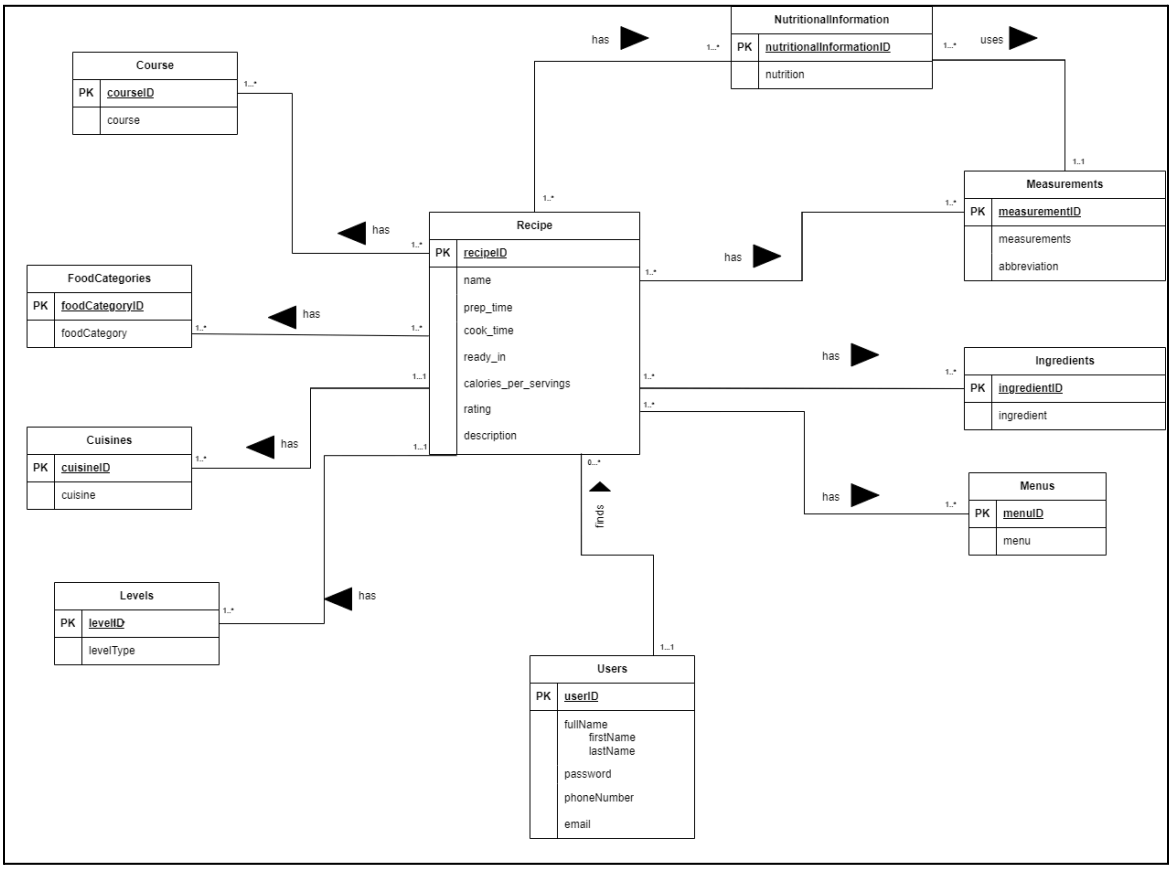


Figure 1.1: Conceptual ERD

4.0 Database Logical Design

4.1 Logical ERD

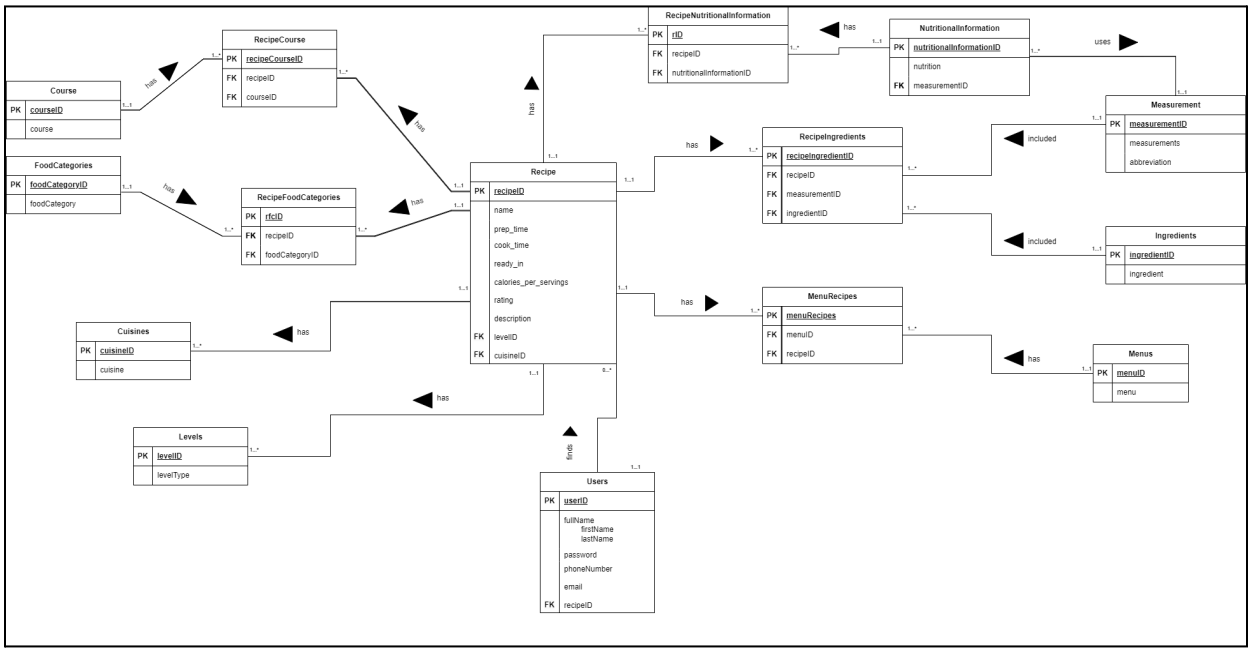


Figure 1.2: Logical ERD

4.2 Updated Data Dictionary

4.2.1 Description of Entity

Entity	Description	Occurrence
Users	Holds user information	- User signs up and logs in. - User finds recipe. - User adds recipe to bookmark.
Recipe	Holds recipe information	- Recipe information is shown after the user finds the recipe. - User manages (add,edit,delete) the recipe.
MenuRecipes	Retrieve recipe and menu information	- When recipe information is displayed.
Menus	Holds menu information	- When recipe information is displayed. - User manages (add,edit,delete) the recipe.
Recipe Ingredients	Holds recipe and ingredient information	- When recipe information is displayed.
Measurement	Holds measurement information	- Shows measurement information related to recipe ingredients and nutritional information when displaying the recipe information. - User manages (add,edit,delete) the recipe.
Ingredients	Holds ingredient information	- Shows ingredients information when displaying the recipe information. - User manages (add,edit,delete) the recipe.
Nutritional Information	Holds nutritional information	- Shows nutritional information when displaying the recipe information - User manages (add,edit,delete) the recipe.

RecipeNutritionalInformation	Holds recipe and nutritional information	- When recipe information is displayed.
Course	Holds course information	- Shows course information when displaying the recipe information. - User manages (add, edit,delete) the recipe
RecipeCourse	Holds recipe and course Information.	- When recipe information is displayed.
FoodCategories	Holds food categories information	- Shows food categories information when displaying the recipe information. - User manages (add,delete,edit) the recipe.
RecipeFood Categories	Holds recipe and food categories information	- When recipe information is displayed.
Cuisines	Holds cuisine information	- Shows cuisine information when displaying the recipe information. - User manages (add,delete,edit) the recipe.
Levels	Holds level information	- Shows levels information when displaying the recipe information. - User manages (add,delete,edit) the recipe.

Table 1.1: Entities table

4.2.2 Description of Relationship

Entity	Multiplicity	Relationship	Multiplicity	Entity
Recipe	1..1	has	1..*	RecipeCourse
	1..1	has	1..*	RecipeFoodCategories
	1..1	has	1..*	Cuisines
	1..1	has	1..*	Levels
	1..1	has	1..*	RecipeNutritionalInformation
	1..1	has	1..*	RecipeIngredients
	1..1	has	1..*	MenuRecipes
Course	1..1	has	1..*	RecipeCourse
FoodCategories	1..1	has	1..*	RecipeFoodCategories
Users	1..1	finds	0..*	Recipe
NutritionalInformation	1..1	has	1..*	RecipeNutritionalInformation
	1..*	uses	1..1	Measurement
Measurement	1..1	included	1..*	RecipeIngredients
Ingredients	1..1	included	1..*	RecipeIngredients
Menus	1..1	has	1..*	MenuRecipes

Table 1.2: Relationship table

4.2.3 Description of Attributes

Entity	Attributes	Data Type	Field Size	Key	Description
Recipe	recipeID	NUMBER		PK	Recipe unique ID
	name	VARCHAR	255		Recipe name
	prep_time	NUMBER			Preparation time in minutes
	cook_time	NUMBER			Total time (preparation + cook) in minutes
	ready_in	NUMBER			Cooking time in minutes
	calories_per_servings	NUMBER			Total calories per serving in grams
	rating	DECIMAL			Rating of recipe
	description	VARCHAR	1000		Description of recipe
	levelID	NUMBER		FK	Difficulty level unique ID
	cuisineID	VARCHAR		FK	Cuisine type unique ID
Users	userID	VARCHAR	50	PK	User unique ID
	fullName	VARCHAR	255		Full name of user
	firstName	VARCHAR	255		First name of user
	lastName	VARCHAR	255		Last name of user
	password	VARCHAR	255		Password of user account

	phoneNumber	VARCHAR	255		User phone number
	email	VARCHAR	255		User email
	recipeID	NUMBER		FK	Recipe unique ID
Cuisines	cuisineID	NUMBER		PK	Cuisine type unique ID
	cuisine	VARCHAR	255		Cuisine type
Levels	levelID	NUMBER		PK	Level type unique ID
	levelType	VARCHAR	255		Difficulty level of the recipe
FoodCategories	foodCategoryID	NUMBER		PK	Food category unique ID
	foodCategory	VARCHAR	255		Food category list (beverages, grilling, desserts)
RecipeFoodCategories	rfcID	NUMBER		PK	Recipe food categories unique ID
	recipeID	NUMBER		FK	Recipe unique ID
	foodCategoryID	NUMBER		FK	Food category unique ID
Course	courseID	NUMBER		PK	Course unique ID
	course	VARCHAR	255		Course list (lunch, dinner)
RecipeCourse	recipeCourseID	NUMBER		PK	Recipe course unique ID
	recipeID	NUMBER		FK	Recipe unique ID
	courseID	NUMBER		FK	Course unique ID
Menus	menuID	NUMBER	50	PK	Menu unique ID

	menu	VARCHAR	255		Menu holds recipes
MenuRecipes	menuRecipe	NUMBER		PK	The list of recipes in menu
	menuID	NUMBER		FK	Menu unique ID
	recipeID	NUMBER		FK	Recipe unique ID
RecipeIngredients	recipeIngredientID	NUMBER		PK	Recipe ingredient unique ID
	recipeID	NUMBER		FK	Recipe unique ID
	measurementID	NUMBER		FK	Measurement unique ID
	ingredientID	NUMBER		FK	Ingredient unique ID
Measurement	measurementsID	NUMBER		PK	Measurement unique ID
	measurements	VARCHAR	255		Measurements of ingredient and nutrition
	abbreviation	VARCHAR	50		Abbreviation (grams, cups) for ingredients and measurements
Ingredients	ingredientID	NUMBER		PK	Ingredient unique ID
	ingredient	VARCHAR	255		Ingredients list
NutritionalInformation	nutritionalInforationID	NUMBER		PK	Nutritional information unique ID
	nutrition	VARCHAR	255		Lust of nutrition
	measurementID	NUMBER		FK	Measurement unique ID
RecipeNutritionalInformation	rID	NUMBER		PK	Recipe nutritional

					information unique ID
	recipeID	NUMBER		FK	Recipe unique ID
	nutritionalInformationID	NUMBER		FK	Nutritional information unique ID

Table 1.3: Attributes table

4.3 Normalization

UNF (Unnormalized Form)

Recipe (recipeID, name, prep_time, cook_time, ready_in, calories_per_servings, rating, description, courseID, course, foodCategoryID, foodCategory, cuisineID, cuisine, levelID, levelType, nutritionalInformationID, nutrition, measurementID, measurements, abbreviation, ingredientID, ingredient, menuID, menu, userID, fullName, firstName, lastName, password, phoneNumber, email)

1NF (Primary Key)

Recipe (recipeID, name, prep_time, cook_time, ready_in, calories_per_servings, rating, description, courseID, course, foodCategoryID, foodCategory, cuisineID, cuisine, levelID, levelType, nutritionalInformationID, nutrition, measurementID, measurements, abbreviation, ingredientID, ingredient, menuID, menu, userID, fullName, firstName, lastName, password, phoneNumber, email)

2NF (Partial Dependencies)

Recipe (recipeID, name, prep_time, cook_time, ready_in, calories_per_servings, rating, description, levelID, cuisineID)

- Primary Key: recipeID
- Foreign Key: levelID references Levels (levelID)
- Foreign Key: cuisineID references Cuisines (cuisineID)

Course (courseID, course)

- Primary Key: courseID

RecipeCourse (recipeCourseID, recipeID, courseID)

- Primary Key: recipeCourseID
- Foreign Key: recipeID references Recipe (recipeID)
- Foreign Key: courseID references Course (courseID)

FoodCategories (foodCategoryID, foodCategory)

- Primary Key: foodCategoryID

RecipeFoodCategories (rfcID, recipeID, foodCategoryID)

- Primary Key: rfcID
- Foreign Key: recipeID references Recipe (recipeID)
- Foreign Key: foodCategoryID references FoodCategories (foodCategoryID)

Cuisines (cuisineID, cuisine)

- Primary Key: cuisineID

Levels (levelID, levelType)

- Primary Key: levelID

NutritionalInformation (nutritionalInformationID, nutrition, measurementID)

- Primary Key: nutritionalInformationID
- Foreign Key: measurementID references Measurements (measurementID)

RecipeNutritionalInformation (rID, recipeID, nutritionalInformationID)

- Primary Key: nutritionalInformationID
- Foreign Key: recipeID references Recipe (recipeID)
- Foreign Key: nutritionalInformationID references NutritionalInformation (nutritionalInformationID)

Measurement (measurementID, measurements, abbreviation)

- Primary Key: measurementID

Ingredients (ingredientID, ingredient)

- Primary Key: ingredientID

RecipeIngredients(recipeIngredientID, recipeID, measurementID, ingredientID)

- Primary Key: recipeIngredientID
- Foreign Key: recipeID references Recipe (recipeID)
- Foreign Key: measurementID references Measurements (measurementID)
- Foreign Key: ingredientID references Ingredients (ingredientID)

Menus (menuID, menu)

- Primary Key: menuID

MenuRecipes (menuRecipes, menuID, recipeID)

- Primary Key: menuRecipes
- Foreign Key: menuID references Menu (menuID)
- Foreign Key: recipeID references Recipe (recipeID)

Users (userID, fullName, firstName, lastName, password, phoneNumber, email, recipeID)

- Primary Key: userID
- Foreign Key: recipeID references Recipe (recipeID)

5.0 Relational DB Schemas (After Normalization)

Listed below are the set of relational database schemas for our "Food Recipe Finder" system after going through normalization.

Recipe	(recipeID, name, prep_time, cook_time, ready_in, calories_per_servings, rating, description, levelID, cuisineID)
Course	(courseID, course)
RecipeCourse	(recipeCourseID, recipeID, courseID)
FoodCategories	(foodCategoryID, foodCategory)
RecipeFoodCategories	(rfcID, recipeID, foodCategoryID)
Cuisines	(cuisineID, cuisine)
Levels	(levelID, levelType)
NutritionalInformation	(nutritionalInformationID, nutrition, measurementID)
RecipeNutritionalInformation	(rID, recipeID, nutritionalInformationID)
Measurement	(measurementID, measurements, abbreviation)
Ingredients	(ingredientID, ingredient)
RecipeIngredients	(recipeIngredientID, recipeID, measurementID, ingredientID)

Menus (menuID, menu)

MenuRecipes (menuRecipes, menuID, recipeID)

Users (userID, fullName, firstName, lastName, password, phoneNumber, email, recipeID)

Recipe

recipeID (PK)	name	prep_time	cook_time	ready_in	calories_per_servings	rating	description	levelID (FK)	cuisineID (FK)

Course

courseID (PK)	course

RecipeCourse

recipeCourse (PK)	recipeID (FK)	courseID (FK)

FoodCategories

foodCategoryID (PK)	foodCategory

RecipeFoodCategories

rfcID (PK)	recipeID (FK)	foodCategoryID (FK)

Cuisines

cuisineID (PK)	cuisine

Levels

levelID (PK)	levelType

RecipeNutritionalInformation

rID (PK)	recipeID (FK)	nutritionalInformationID (FK)

NutritionalInformation

nutritionalInformationID (PK)	nutrition	measurementID (FK)

Measurement

measurementID (PK)	measurements	abbreviation

Ingredients

ingredientID (PK)	ingredient

RecipeIngredients

recipeIngredientID (PK)	recipeID (FK)	measurementID (FK)	ingredientID (FK)

Menus

menuID (PK)	menu

MenuRecipes

menuRecipes (PK)	menuID (FK)	recipeID (FK)

Users

userID (PK)	fullName	firstName	lastName	password	phoneNumber	email	recipeID (FK)

6.0 SQL Statements

6.1 Data Definition Language

```
CREATE TABLE users(
    userID VARCHAR2(50) PRIMARY KEY,
    fullName VARCHAR2(255),
    firstName VARCHAR2(255),
    lastName VARCHAR2(255),
    password VARCHAR2(255),
    phoneNumber VARCHAR2(255),
    email VARCHAR2(255),
    recipeID NUMBER,
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID)
);

CREATE TABLE RecipeNutritionalInformation (
    rID NUMBER PRIMARY KEY,
    recipeID NUMBER,
    nutritionalInformationID NUMBER,
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID),
    FOREIGN KEY (nutritionalInformationID) REFERENCES
NutritionalInformation (nutritionalInformationID)
);

CREATE TABLE recipeIngredients (
    recipeIngredientID NUMBER PRIMARY KEY,
    recipeID NUMBER,
    measurementID NUMBER,
    ingredientID NUMBER,
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID),
    FOREIGN KEY (measurementID) REFERENCES Measurement
(measurementID),
    FOREIGN KEY (ingredientID) REFERENCES Ingredients
(ingredientID)
);

CREATE TABLE RecipeFoodCategories (
    rfcID NUMBER PRIMARY KEY,
    recipeID NUMBER,
    foodCategoryID NUMBER,
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID),
    FOREIGN KEY (foodCategoryID) REFERENCES FoodCategories
```

```
(foodCategoryID)
);

CREATE TABLE RecipeCourse (
    recipeCourseID NUMBER PRIMARY KEY,
    recipeID NUMBER,
    courseID NUMBER,
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID),
    FOREIGN KEY (courseID) REFERENCES Course (courseID)
);

CREATE TABLE Recipe (
    recipeID NUMBER PRIMARY KEY,
    name VARCHAR2(255),
    prep_time NUMBER,
    cook_time NUMBER,
    ready_in NUMBER,
    calories_per_servings NUMBER,
    levelID NUMBER,
    rating DECIMAL,
    cuisineID NUMBER,
    description VARCHAR(1000),
    FOREIGN KEY (levelID) REFERENCES Levels (levelID),
    FOREIGN KEY (cuisineID) REFERENCES Cuisines (cuisineID)
);

CREATE TABLE NutritionalInformation (
    nutritionalInformationID NUMBER PRIMARY KEY,
    nutrition VARCHAR2(255),
    measurementID NUMBER,
    FOREIGN KEY (measurementID) REFERENCES Measurement
(measurementID)
);

CREATE TABLE MenuRecipes (
    menuRecipes NUMBER PRIMARY KEY,
    menuID NUMBER,
    recipeID NUMBER,
    FOREIGN KEY (menuID) REFERENCES Menus (menuID),
    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID)
);

CREATE TABLE Measurement (
    measurementID NUMBER PRIMARY KEY,
    measurements VARCHAR2(255),
    abbreviation VARCHAR2(50)
```

```
);

CREATE TABLE Menus (
    menuID NUMBER PRIMARY KEY,
    menu VARCHAR2(255)
);

CREATE TABLE Levels (
    levelID NUMBER PRIMARY KEY,
    levelType VARCHAR2(255)
);

CREATE TABLE Ingredients (
    ingredientID NUMBER PRIMARY KEY,
    ingredient VARCHAR2(255)
);

CREATE TABLE FoodCategories (
    foodCategoryID NUMBER PRIMARY KEY,
    foodCategory VARCHAR2(255)
);

CREATE TABLE Cuisines (
    cuisineID NUMBER PRIMARY KEY,
    cuisine VARCHAR2(255)
);

CREATE TABLE Course (
    courseID NUMBER PRIMARY KEY,
    course VARCHAR2(255)
);
```

6.2 Data Manipulation Language

6.2.1 DML 1 (Insert Data)

COURSE TABLE

```
 INSERT INTO Course (courseID, course) VALUES (1, 'Appetizer');
 INSERT INTO Course (courseID, course) VALUES (2, 'Main
Course');
 INSERT INTO Course (courseID, course) VALUES (3, 'Dessert');
 INSERT INTO Course (courseID, course) VALUES (4, 'Side Dish');
 INSERT INTO Course (courseID, course) VALUES (5, 'Beverage');
```

CUISINES TABLE

```
INSERT INTO Cuisines (cuisineID, cuisine) VALUES (1,
'Italian');
INSERT INTO Cuisines (cuisineID, cuisine) VALUES (2,
'Mexican');
INSERT INTO Cuisines (cuisineID, cuisine) VALUES (3,
'Chinese');
INSERT INTO Cuisines (cuisineID, cuisine) VALUES (4, 'Indian');
INSERT INTO Cuisines (cuisineID, cuisine) VALUES (5,
'Japanese');
```

FOODCATEGORIES TABLE

```
INSERT INTO FoodCategories (foodCategoryID, foodCategory)
VALUES (1, 'Appetizers");
INSERT INTO FoodCategories (foodCategoryID, foodCategory)
VALUES (2, 'Main Dishes');
INSERT INTO FoodCategories (foodCategoryID, foodCategory)
VALUES (3,'Desserts');
INSERT INTO FoodCategories (foodCategoryID, foodCategory)
VALUES (4,'Salads')
INSERT INTO FoodCategories (foodCategoryID, foodCategory)
VALUES (5,'Beverages');
```

INGREDIENTS TABLE

```
INSERT INTO Ingredients (ingredientID, ingredient) VALUES (1,
'Salt');
INSERT INTO Ingredients (ingredientID, ingredient) VALUES (2,
'Pepper');
INSERT INTO Ingredients (ingredientID, ingredient) VALUES (3,
'Sugar');
INSERT INTO Ingredients (ingredientID, ingredient) VALUES (4,
'Flour');
INSERT INTO Ingredients (ingredientID, ingredient) VALUES (5,
'Butter');
```

LEVELS TABLE

```
INSERT INTO Levels (levelID, levelType) VALUES (1, 'Beginner');
INSERT INTO Levels (levelID, levelType) VALUES (2,
'Intermediate');
INSERT INTO Levels (levelID, levelType) VALUES (3, 'Advanced');
INSERT INTO Levels (levelID, levelType) VALUES (4, 'Expert');
INSERT INTO Levels (levelID, levelType) VALUES (5, 'Master');
```

MENU TABLE

```
INSERT INTO Menus (menuID, menu) VALUES  (1, 'Breakfast');
INSERT INTO Menus (menuID, menu) VALUES  (2, 'Lunch');
INSERT INTO Menus (menuID, menu) VALUES  (3, 'Dinner');
INSERT INTO Menus (menuID, menu) VALUES  (4, 'Snacks');
INSERT INTO Menus (menuID, menu) VALUES  (5, 'Dessert');
```

MEASUREMENT TABLE

```
INSERT INTO Measurement (measurementID, measurements,
abbreviation) VALUES (1, 'Cup', 'C');
INSERT INTO Measurement (measurementID, measurements,
abbreviation) VALUES (2, 'Teaspoon', 'tsp');
INSERT INTO Measurement (measurementID, measurements,
abbreviation) VALUES (3, 'Tablespoon', 'tbsp');
INSERT INTO Measurement (measurementID, measurements,
abbreviation) VALUES (4, 'Ounce', 'oz');
INSERT INTO Measurement (measurementID, measurements,
abbreviation) VALUES (5, 'Gram', 'g');
```

RECIPE TABLE

```
INSERT INTO Recipe (recipeID, name, prep_time, cook_time,
ready_in, calories_per_servings,levelID,rating,cuisineID,
description)
VALUES
  (1,'Spaghetti Bolognese',20,30,50,500,2,4.5,1,'combine ground
beef, onion, garlic, carrot, and celery with olive oil, crushed
tomatoes, tomato paste, red wine (optional), oregano, and
basil, seasoning it to taste. Serve the flavorful sauce over
cooked spaghetti, garnish with grated Parmesan, and optionally
add fresh basil.');

INSERT INTO Recipe (recipeID, name, prep_time, cook_time,
ready_in, calories_per_servings,levelID,rating,cuisineID
description)
VALUES
  (2, 'Chicken Parmesan', 15, 45, 60, 600, 3, 4.2, 2,'First,
dipping chicken cutlets in beaten eggs and coating them with
seasoned breadcrumbs. Fry the cutlets until golden brown on
both sides. Transfer the fried cutlets to a baking dish, top
them with marinara sauce and mozzarella cheese, and bake until
the cheese is melted and bubbly. Meanwhile, cook spaghetti
according to your taste. Serve the chicken over spaghetti and
garnish with fresh basil.');

INSERT INTO Recipe (recipeID, name, prep_time, cook_time,
ready_in, calories_per_servings,levelID,rating,cuisineID,
description)
VALUES
  (3, 'Beef Stir Fry', 10, 20, 30, 400, 1, 4.8, 3,'Start by
marinating thinly sliced beef in soy sauce and sesame oil.
Stir-fry the marinated beef in a wok until browned, then set it
aside. In the same wok, stir-fry garlic, ginger, and assorted
vegetables until crisp-tender. Add the cooked beef back to the
wok, toss everything together.');

INSERT INTO Recipe (recipeID, name, prep_time, cook_time,
ready_in, calories_per_servings,levelID,rating,cuisineID,
description)
VALUES
  (4, 'Vegetable Curry', 25, 40, 65, 300, 2, 4.0, 4,'Create
Vegetable Curry by sautéing chopped onion and garlic until
softened. Add a blend of curry spices and stir to release their
flavors. Introduce chopped vegetables, tomatoes, and coconut
milk to the pot, simmering until the vegetables are tender and
```

```
the flavors meld together beautifully.');

INSERT INTO Recipe (recipeID, name, prep_time, cook_time,
ready_in, calories_per_servings,levelID,rating,cuisineID,
description)
VALUES
  (5,'Salmon with Lemon Butter
Sauce',12,25,37,450,3,4.6,5,'Season the salmon fillets with
salt and pepper. Sear the salmon until golden on both sides. In
the same pan, melt butter, sauté minced garlic, and squeeze
fresh lemon juice into the mix. Optionally, add white wine for
extra depth. Drizzle this luscious lemon-infused butter sauce
over the salmon fillets and garnish with chopped fresh parsley
before serving.');
```

MENURECIPE TABLE

```
INSERT INTO MenuRecipes (menuRecipes, menuID, recipeID) VALUES
(1, 1, 1);
INSERT INTO MenuRecipes (menuRecipes, menuID, recipeID) VALUES
(2, 1, 2);
INSERT INTO MenuRecipes (menuRecipes, menuID, recipeID) VALUES
(3, 2, 3);
INSERT INTO MenuRecipes (menuRecipes, menuID, recipeID) VALUES
(4, 2, 4);
INSERT INTO MenuRecipes (menuRecipes, menuID, recipeID) VALUES
(5, 3, 5);
```

NUTRITIONALINFORMATION TABLE

```
INSERT INTO NutritionalInformation
(nutritionalInformationID,nutrition,measurementID)
VALUES (1, 'Protein', 1);

INSERT INTO NutritionalInformation
(nutritionalInformationID,nutrition,measurementID)
VALUES (2, 'Carbohydrates', 1);

INSERT INTO NutritionalInformation
(nutritionalInformationID,nutrition,measurementID)
VALUES (3, 'Fat', 1);

INSERT INTO NutritionalInformation
(nutritionalInformationID,nutrition,measurementID)
VALUES (4, 'Fiber', 1);
```

```
INSERT INTO NutritionalInformation
(nutritionalInformationID,nutrition,measurementID)
VALUES (5, 'Sugar', 1);
```

RECIPE COURSE TABLE

```
INSERT INTO RecipeCourse (recipeCourseID, recipeID, courseID)
VALUES (1, 1, 1);
INSERT INTO RecipeCourse (recipeCourseID, recipeID, courseID)
VALUES (2, 2, 3);
INSERT INTO RecipeCourse (recipeCourseID, recipeID, courseID)
VALUES (3, 3, 2);
INSERT INTO RecipeCourse (recipeCourseID, recipeID, courseID)
VALUES (4, 4, 1);
INSERT INTO RecipeCourse (recipeCourseID, recipeID, courseID)
VALUES (5, 5, 3);
```

RECIPE FOODCATEGORY TABLE

```
INSERT INTO RecipeFoodCategories (rfcID, recipeID,
foodCategoryID) VALUES (1, 1, 1);
INSERT INTO RecipeFoodCategories (rfcID, recipeID,
foodCategoryID) VALUES (2, 2, 3);
INSERT INTO RecipeFoodCategories (rfcID, recipeID,
foodCategoryID) VALUES (3, 3, 2);
INSERT INTO RecipeFoodCategories (rfcID, recipeID,
foodCategoryID) VALUES (4, 4, 1);
INSERT INTO RecipeFoodCategories (rfcID, recipeID,
foodCategoryID) VALUES (5, 5, 4);
```

RECIPEINGREDIENTS TABLE

```
INSERT INTO recipeIngredients
(recipeIngredientID,recipeID,measurementID,ingredientID) VALUES
(1, 1, 1, 1);
INSERT INTO recipeIngredients
(recipeIngredientID,recipeID,measurementID,ingredientID) VALUES
(2, 1, 2, 2);
INSERT INTO recipeIngredients
(recipeIngredientID,recipeID,measurementID,ingredientID) VALUES
(3, 2, 3, 3);
INSERT INTO recipeIngredients
(recipeIngredientID,recipeID,measurementID,ingredientID) VALUES
(4, 2, 4, 4);
INSERT INTO recipeIngredients
(recipeIngredientID,recipeID,measurementID,ingredientID) VALUES
```

```
(5, 3, 5, 5);
```

RECIPENUTRITIONALINFORMATION TABLE

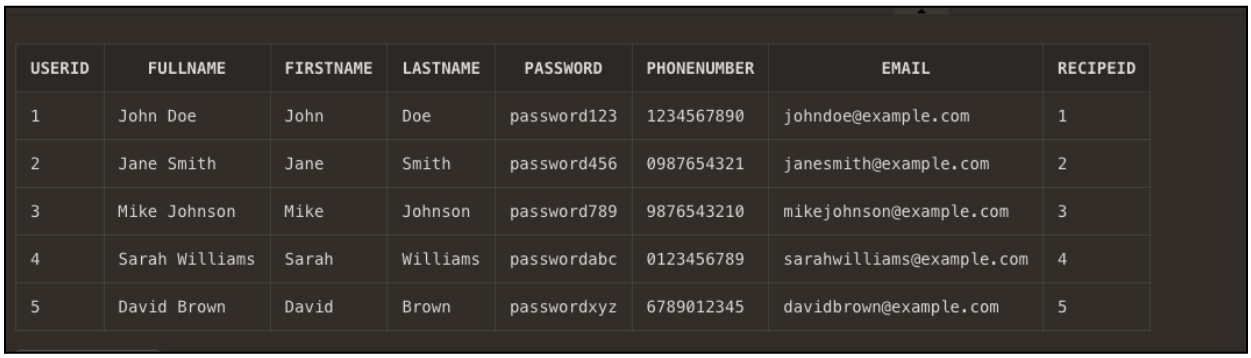
```
INSERT INTO RecipeNutritionalInformation (rID, recipeID,
nutritionalInformationID) VALUES (1, 1, 1);
INSERT INTO RecipeNutritionalInformation (rID, recipeID,
nutritionalInformationID) VALUES (2, 2, 2);
INSERT INTO RecipeNutritionalInformation (rID, recipeID,
nutritionalInformationID) VALUES (3, 3, 3);
INSERT INTO RecipeNutritionalInformation (rID, recipeID,
nutritionalInformationID) VALUES (4, 4, 4);
INSERT INTO RecipeNutritionalInformation (rID, recipeID,
nutritionalInformationID) VALUES (5, 5, 5);
```

USERS TABLE

```
INSERT INTO users (userID,
fullName,firstName,lastName,password,phoneNumber,email,recipeID
) VALUES ('1','John
Doe','John','Doe','password123','1234567890','johndoe@example.c
om',1);

INSERT INTO users (userID,
fullName,firstName,lastName,password,phoneNumber,email,recipeID
) VALUES ( '2','Jane
Smith','Jane','Smith','password456','0987654321','janesmith@exa
mple.com',2);

INSERT INTO users (userID,
fullName,firstName,lastName,password,phoneNumber,email,recipeID
) VALUES ('3','Mike
Johnson','Mike','Johnson','password789','9876543210','mikejohns
on@example.com',3);

INSERT INTO users (userID,
fullName,firstName,lastName,password,phoneNumber,email,recipeID
) VALUES ('4','Sarah
Williams','Sarah','Williams','passwordabc','0123456789','sarahw
illiams@example.com',4);

INSERT INTO users (userID,
fullName,firstName,lastName,password,phoneNumber,email,recipeID
) VALUES ('5','David
Brown','David','Brown','passwordxyz','6789012345','davidbrown@e
xample.com',5);
```

6.2.2 DML 2 (Display Table)

```
SELECT * FROM Users;
```



USERID	FULLNAME	FIRSTNAME	LASTNAME	PASSWORD	PHONENUMBER	EMAIL	RECIPEID
1	John Doe	John	Doe	password123	1234567890	johndoe@example.com	1
2	Jane Smith	Jane	Smith	password456	0987654321	janesmith@example.com	2
3	Mike Johnson	Mike	Johnson	password789	9876543210	mikejohnson@example.com	3
4	Sarah Williams	Sarah	Williams	passwordabc	0123456789	sarahwilliams@example.com	4
5	David Brown	David	Brown	passwordxyz	6789012345	davidbrown@example.com	5

```
SELECT * FROM Course;
```

COURSEID	COURSE
1	Appetizer
2	Main Course
3	Dessert
4	Side Dish
5	Beverage

```
SELECT * FROM Cuisine;
```

CUISINEID	CUISINE
1	Italian
2	Mexican
3	Chinese
4	Indian
5	Japanese

```
SELECT * FROM FoodCategories;
```

FOODCATEGORYID	FOODCATEGORY
1	Appetizers
2	Main Dishes
3	Desserts
4	Salads
5	Beverages

```
SELECT * FROM Ingredients;
```

INGREDIENTID	INGREDIENT
1	Salt
2	Pepper
3	Sugar
4	Flour
5	Butter

```
SELECT * FROM Levels;
```

LEVELID	LEVELTYPE
1	Beginner
2	Intermediate
3	Advanced
4	Expert
5	Master

```
SELECT * FROM Measurement;
```

MEASUREMENTID	MEASUREMENTS	ABBREVIATION
1	Cup	C
2	Teaspoon	tsp
3	Tablespoon	tbsp
4	Ounce	oz
5	Gram	g

```
SELECT * FROM Menus;
```

MENUID	MENU
1	Breakfast
2	Lunch
3	Dinner
4	Snacks
5	Dessert

```
SELECT * FROM MenuRecipes;
```

MENURECIPES	MENUID	RECIPEID
1	1	1
2	1	2
3	2	3
4	2	4
5	3	5

```
SELECT * FROM NutritionalInformation;
```

NUTRITIONALINFORMATIONID	NUTRITION	MEASUREMENTID
1	Protein	1
2	Carbohydrates	1
3	Fat	1
4	Fiber	1
5	Sugar	1

```
SELECT * FROM Recipe;
```

RECIPEID	NAME	PREP_TIME	COOK_TIME	READY_IN	CALORIES_PER_SERVINGS	LEVELID	RATING	CUISINEID	DESCRIPTION
1	Spaghetti Bolognese	20	30	50	500	2	5	1	combine ground beef, onion, garlic, carrot, and celery with olive oil, crushed tomatoes, tomato paste, red wine (optional), oregano, and basil, seasoning it to taste. Serve the flavorful sauce over cooked spaghetti, garnish with grated Parmesan, and optionally add fresh basil.
2	Chicken Parmesan	15	45	60	600	3	4	2	First, dipping chicken cutlets in beaten eggs and coating them with seasoned breadcrumbs. Fry the cutlets until golden brown on both sides. Transfer the fried cutlets to a baking dish, top them with marinara sauce and mozzarella cheese, and bake until the cheese is melted and bubbly. Meanwhile, cook spaghetti according to your taste. Serve the chicken over spaghetti and garnish with fresh basil.
3	Beef Stir Fry	10	20	30	400	1	5	3	Start by marinating thinly sliced beef in soy sauce and sesame oil. Stir-fry the marinated beef in a wok until browned, then set it aside. In the same wok, stir-fry garlic, ginger, and assorted vegetables until crisp-tender. Add the cooked beef back to the wok, toss everything together.
4	Vegetable Curry	25	40	65	300	2	4	4	Create Vegetable Curry by sautéing chopped onion and garlic until softened. Add a blend of curry spices and stir to release their flavors. Introduce chopped vegetables, tomatoes, and coconut milk to the pot, simmering until the vegetables are tender and the flavors meld together beautifully.

5	Salmon with Lemon Butter Sauce	12	25	37	450	3	5	5	Season the salmon fillets with salt and pepper. Sear the salmon until golden on both sides. In the same pan, melt butter, sauté minced garlic, and squeeze fresh lemon juice into the mix. Optionally, add white wine for extra depth. Drizzle this luscious lemon-infused butter sauce over the salmon fillets and garnish with chopped fresh parsley before serving.

```
SELECT * FROM RecipeFoodCategories;
```

RFCID	RECIPEID	FOODCATEGORYID
1	1	1
2	2	3
3	3	2
4	4	1
5	5	4

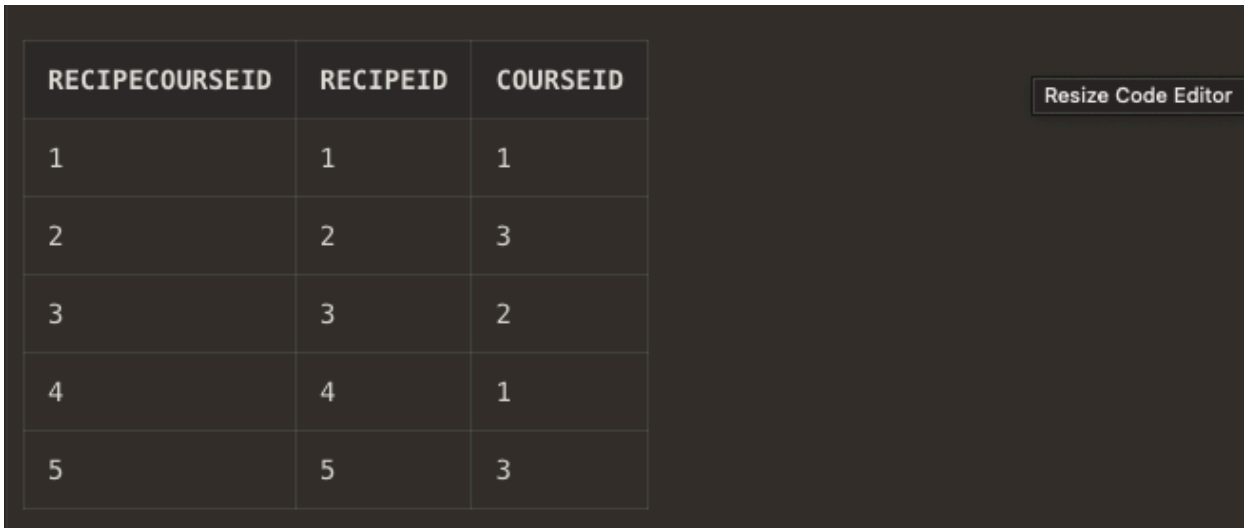
```
SELECT * FROM RecipeIngredients;
```

RECIPEINGREDIENTID	RECIPEID	MEASUREMENTID	INGREDIENTID
1	1	1	1
2	1	2	2
3	2	3	3
4	2	4	4
5	3	5	5

```
SELECT * FROM RecipeNutritionalInformation;
```

RID	RECIPEID	NUTRITIONALINFORMATIONID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

```
SELECT * FROM RecipeCourse;
```



RECIPECOURSEID	RECIPEID	COURSEID
1	1	1
2	2	3
3	3	2
4	4	1
5	5	3

6.2.3 DML 3 (Join & Delete)

6.2.3.1 Display User and Recipe Name Information

```
SELECT
  u.*,
  r.name
FROM
  Users u
  LEFT JOIN Recipe r ON u.recipeID = r.recipeID
```

USERID	FULLNAME	FIRSTNAME	LASTNAME	PASSWORD	PHONENUMBER	EMAIL	RECIPEID	NAME
1	John Doe	John	Doe	password123	1234567890	johndoe@example.com	1	Spaghetti Bolognese
2	Jane Smith	Jane	Smith	password456	0987654321	janesmith@example.com	2	Chicken Parmesan
3	Mike Johnson	Mike	Johnson	password789	9876543210	mikejohnson@example.com	3	Beef Stir Fry
4	Sarah Williams	Sarah	Williams	passwordabc	0123456789	sarahwilliams@example.com	4	Vegetable Curry
5	David Brown	David	Brown	passwordxyz	6789012345	davidbrown@example.com	5	Salmon with Lemon Butter Sauce

6.2.3.2 Display User Profile Information

```
SELECT userID, firstName, lastName, phoneNumber, email
FROM Users
WHERE userID = 1;
```

FIRSTNAME	LASTNAME	PHONENUMBER	EMAIL
John	Doe	1234567890	johndoe@example.com

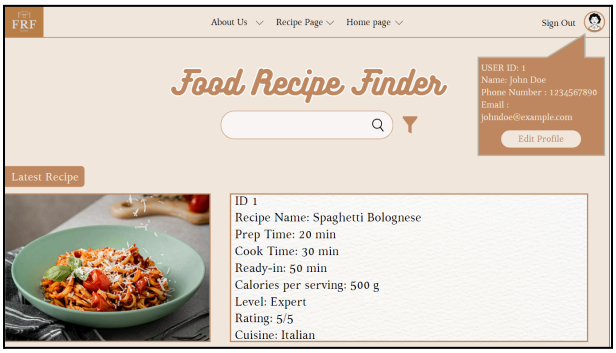


Figure 2.1: Interface to view user profile

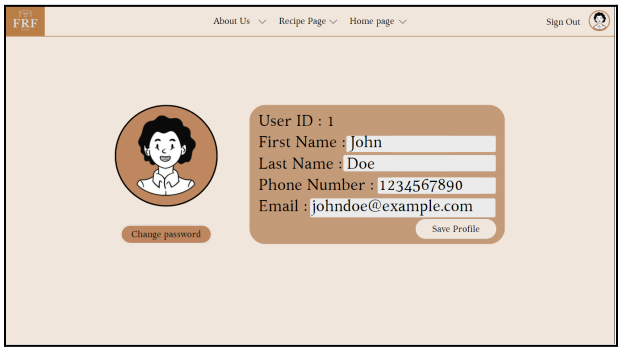


Figure 2.2: Interface to edit user profile

6.2.3.3 Display List of Recipes

```
SELECT
  r.recipeID,
  r.name,
  r.prep_time,
  r.cook_time,
  r.ready_in,
  r.calories_per_servings,
  l.levelType,
  r.rating,
  c.cuisine
FROM
  Recipe r
  JOIN Cuisines c ON r.cuisineID = c.cuisineID
  JOIN Levels l ON r.levelID = l.levelID
ORDER BY r.recipeID ASC;
```

RECIPEID	NAME	PREP_TIME	COOK_TIME	READY_IN	CALORIES_PER_SERVINGS	LEVELTYPE	RATING	CUISINE
1	Spaghetti Bolognese	20	30	50	500	Intermediate	5	Italian
2	Chicken Parmesan	15	45	60	600	Advanced	4	Mexican
3	Beef Stir Fry	10	20	30	400	Beginner	5	Chinese
4	Vegetable Curry	25	40	65	300	Intermediate	4	Indian
5	Salmon with Lemon Butter Sauce	12	25	37	450	Advanced	5	Japanese
6	French Toast	10	10	20	500	Beginner	5	French
7	Mushroom Soup	15	20	35	300	Intermediate	5	Western
8	Risotto	20	50	70	400	Expert	5	Italian
9	Korean Beef Stew	30	50	80	500	Master	5	Korean
10	French Fries	10	10	20	400	Beginner	4	French

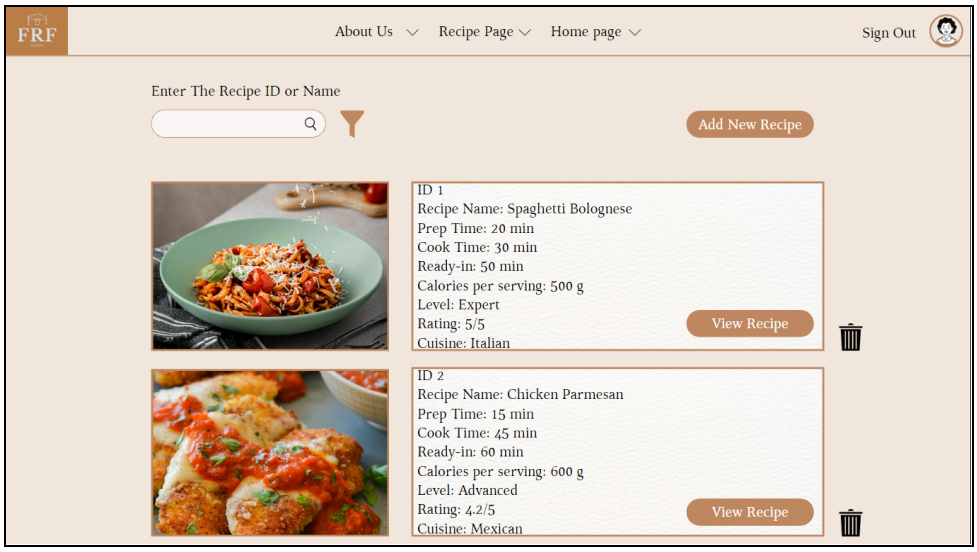


Figure 2.3: Interface to view list of recipes

6.2.3.4 Delete Recipe

```
DELETE FROM Recipe
WHERE recipeID = 8;
```

The previous query is used again to display the updated recipe table after deletion.

RECIPEID	NAME	PREP_TIME	COOK_TIME	READY_IN	CALORIES_PER_SERVINGS	LEVELTYPE	RATING	CUISINE
1	Spaghetti Bolognese	20	30	50	500	Intermediate	5	Italian
2	Chicken Parmesan	15	45	60	600	Advanced	4	Mexican
3	Beef Stir Fry	10	20	30	400	Beginner	5	Chinese
4	Vegetable Curry	25	40	65	300	Intermediate	4	Indian
5	Salmon with Lemon Butter Sauce	12	25	37	450	Advanced	5	Japanese
6	French Toast	10	10	20	500	Beginner	5	French
7	Mushroom Soup	15	20	35	300	Intermediate	5	Western
9	Korean Beef Stew	30	50	80	500	Master	5	Korean
10	French Fries	10	10	20	400	Beginner	4	French

6.2.3.5 Display Recipe Details

```
SELECT
  r.recipeID,
  r.name,
  r.prep_time,
  r.cook_time,
  r.ready_in,
  r.calories_per_servings,
  l.levelType,
  r.rating,
  c.cuisine,
  co.course,
  fc.foodcategory,
  i.ingredient,
  r.description
FROM
  Recipe r
  JOIN Cuisines c ON r.cuisineID = c.cuisineID
  JOIN Levels l ON r.levelID = l.levelID
  JOIN RecipeCourse rc ON rc.recipeID = r.recipeID
  JOIN Course co ON co.courseID = rc.courseID
  JOIN RecipeFoodCategories rfc ON rfc.recipeID = r.recipeID
  JOIN FoodCategories fc ON fc.foodCategoryID =
       rfc.foodCategoryID
  JOIN RecipeIngredients ri ON ri.recipeID = r.recipeID
  JOIN Ingredients i ON i.ingredientID = ri.ingredientID
WHERE r.recipeID = 1
ORDER BY r.recipeID ASC;
```

RECIPEID	NAME	PREP_TIME	COOK_TIME	READY_IN	CALORIES_PER_SERVINGS	LEVELTYPE	RATING	CUISINE	COURSE	FOODCATEGORY	INGREDIENT	DESCRIPTION
1	Spaghetti Bolognese	20	30	50	500	Intermediate	5	Italian	Appetizer	Pasta	Salt	combine ground beef, onion, garlic, carrot, and celery with olive oil, crushed tomatoes, tomato paste, red wine (optional), oregano, and basil, seasoning it to taste. Serve the flavorful sauce over cooked spaghetti, garnish with grated Parmesan, and optionally add fresh basil.
1	Spaghetti Bolognese	20	30	50	500	Intermediate	5	Italian	Appetizer	Pasta	Pepper	combine ground beef, onion, garlic, carrot, and celery with olive oil, crushed tomatoes, tomato paste, red wine (optional), oregano, and basil, seasoning it to taste. Serve the flavorful sauce over cooked spaghetti, garnish with grated Parmesan, and optionally add fresh basil.

Figure 2.4: Interface to view recipe details

Figure 2.5: Interface to edit recipe details

7.0 Summary

To summarize, the “Food Recipe Finder” system is designed to be an efficient platform that allows various users to find, manage, save and share recipes with other users on the platform. This report serves the purpose of outlining the processes that we have gone through in order to create a final database for our system that properly reflects our system’s business environment. For this phase, this pertains to the process of specifying our attribute’s data types, primary keys, foreign keys, and also normalizing the data to remove any anomalies in our database. This can be seen arranged throughout our report via our updated business rules, conceptual and logical ERDs, normalization process, updated data dictionaries and relational DB schemas. Once the adjustments are complete, everything is then implemented with SQL statements. This tool aids in creating a real database and provides the capability of effortlessly adding, modifying or locating our data. This procedural approach culminates into an efficiently functioning and solidly structured database that adheres strictly to its rules manifested clearly through diagrams and its accompanying explanations. All of this will hopefully develop into a robust database that will allow users to have a smooth and easy user experience.