Food Saver Report

Eliminate Food Waste

Pavel Krivopustov | Conard James Faraon | Alex Mackinen | Lan Yang

Introduction

According to study performed by USDA, “In the United States, 31 percent—or 133 billion pounds—of the 430 billion pounds of the available food supply at the retail and consumer levels in 2010 went uneaten. The estimated value of this food loss was $161.6 billion using retail prices” (Buzby). Our goal is to develop a database that will be able to provide consumers with recipes based on their leftovers. Users will be able to specify leftover ingredients they have, ingredients they want or do not want in their recipes, and the time to prepare the food. Our database is able to retrieve the matching recipes to reflect user’s preferences.

Entity-Relationship Diagram

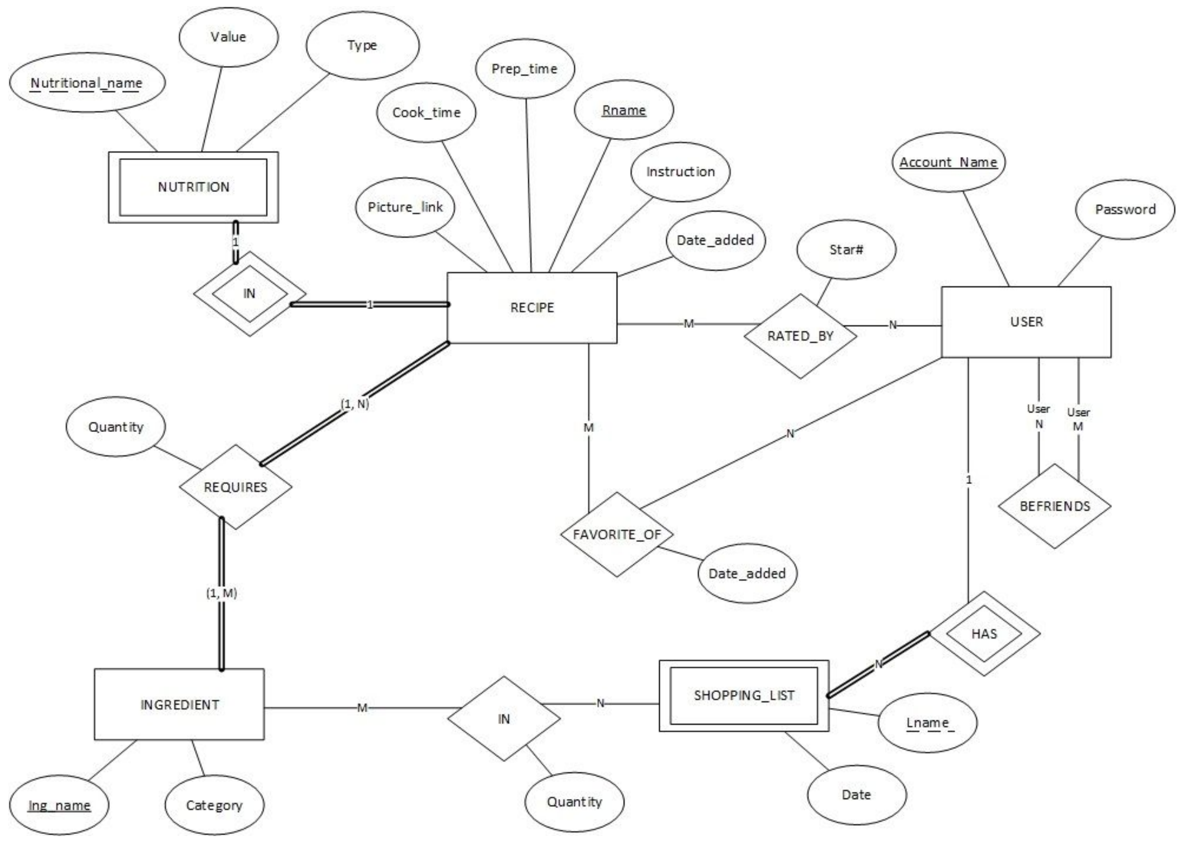


Figure 1: Entity-Relationship for Food Saver database

Relational Data Model

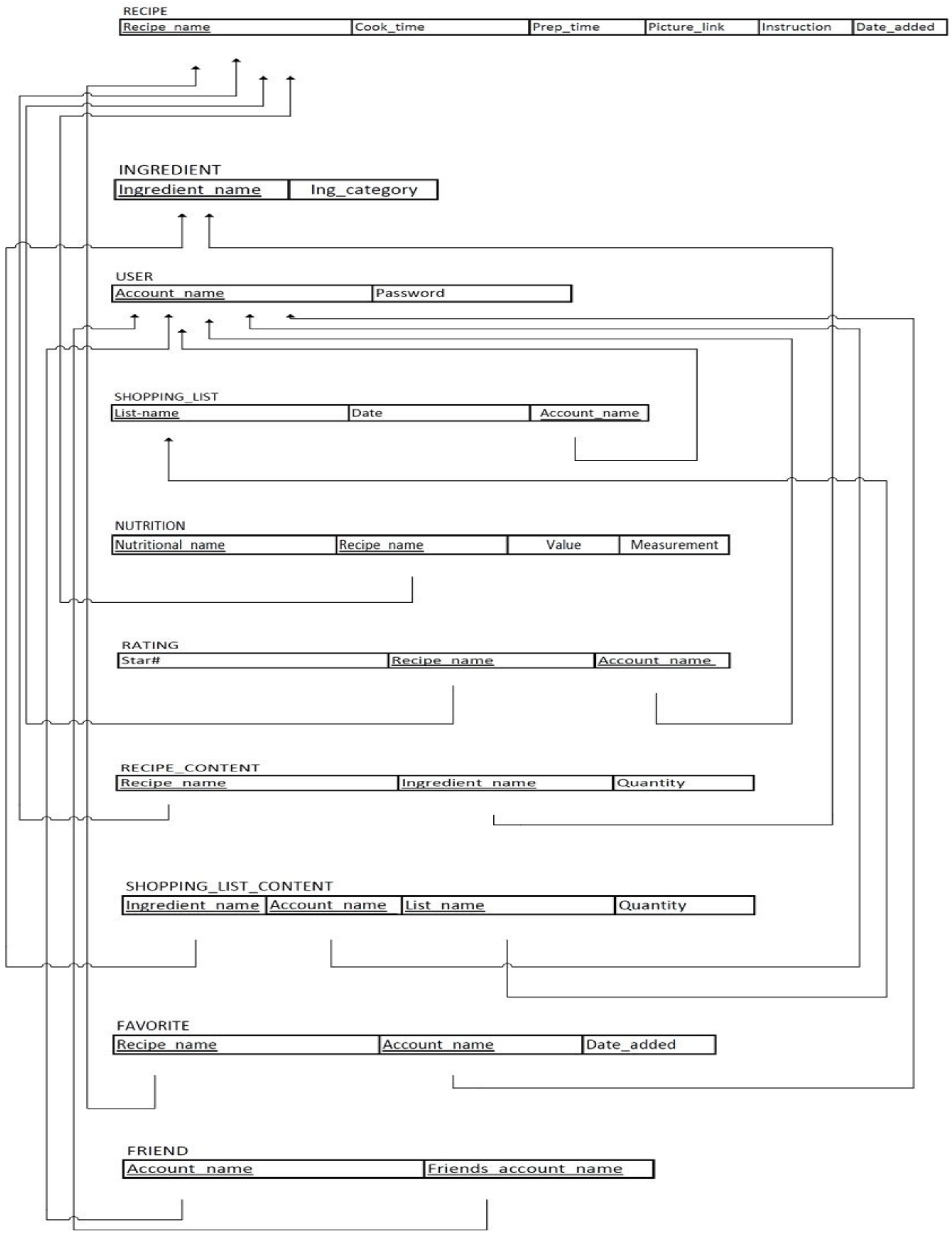


Figure 2: Representation of relational data for Food Saver database

Tables Used in the Database

Table Name	Information Stored in the Table	Normal Form
FAVORITE	Stores user’s favorite recipes, and the date the recipe was added.	BCNF
FRIEND	Stores the username of a friend.	BCNF
INGREDIENT	Stores ingredient name and ites respective category.	BCNF
NUTRITION	Stores nutritional values of the recipe and the respective measurement units.	BCNF
RATING	Stores the number of stars given to the recipe along with the user who gave the rating.	BCNF
RECIPE	Stores the name of the recipe, cook and preparation times, link to a picture, instructions and the date the recipe was added.	BCNF
RECIPE_CONTENT	Stores the recipe name, ingredients within the recipe and the quantity.	BCNF
SHOPPING_LIST	Stores the name of the list, date it was added, and the account name to whom it belongs.	BCNF
SHOPPING_LIST_CONTENT	Stores the name of the list, account name to whom it belongs, ingredient name and the quantity needed of the corresponding ingredient.	BCNF
USER	Stores the user’s name and their password.	BCNF

Table 1: Names of the tables used in the database and what they store.

Constraints for Each of the Tables

Table Name	Constraints
FAVORITE	PRIMARY KEY (Recipe_name, Account_name), CONSTRAINT FAFK1 FOREIGN KEY (Recipe_name) REFERENCES RECIPE(Recipe_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT FAFK2 FOREIGN KEY (Account_name) REFERENCES USER(Account_name) ON UPDATE CASCADE ON DELETE CASCADE
FRIEND	PRIMARY KEY (Account_name, Friends_account_name) CONSTRAINT FFK1 FOREIGN KEY (Account_name) REFERENCES USER(Account_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT FFK2 FOREIGN KEY (Friends_account_name) REFERENCES USER(Account_name) ON UPDATE CASCADE
INGREDIENT	PRIMARY KEY (Ingredient_name), CONSTRAINT ICK1 CHECK(Ingredient_category in('Dairy', 'Meats', 'Oils', 'Soup', 'Beverages', 'Vegetables', 'Fruits', 'Spices', 'Fish', 'Baking & Grains', 'Seafood', 'Added Sweetener', 'Seasonings', 'Nuts', 'Condiments', 'Desserts & Snacks', 'Dairy Alternatives', 'Legumes', 'Sauces', 'Alcohol', 'Liquids', 'Eggs'))
NUTRITION	PRIMARY KEY (Nutritional_name, Recipe_name),

	CONSTRAINT NFK1 FOREIGN KEY(Recipe_name) REFERENCES RECIPE(Recipe_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT NCK1 CHECK(Nutritional_value >= 0) CONSTRAINT NCK2 CHECK (Nutritional_name in ('Calories', 'Carbs', 'Protein', 'Sodium', 'Fat', 'Cholesterol')) CONSTRAINT NCK3 CHECK (Nutrition_type in ('kcal', 'g', 'mg'))
RATING	PRIMARY KEY (Recipe_name, Account_name), CONSTRAINT RFK1 FOREIGN KEY(Recipe_name) REFERENCES RECIPE(Recipe_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT RFK2 FOREIGN KEY(Account_name) REFERENCES USER(Account_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT RCK1 CHECK(Star_num >= 1 AND Star_num <= 5)
RECIPE	PRIMARY KEY (Recipe_name), CONSTRAINT RCK1 CHECK(Cook_time > 0), CONSTRAINT RCK2 CHECK(Prep_time > 0)
RECIPE_CONTENT	PRIMARY KEY (Recipe_name, Ingredient_name), CONSTRAINT RCFK1 FOREIGN KEY(Recipe_name)

	REFERENCES RECIPE(Recipe_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT RCFK2 FOREIGN KEY (Ingredient_name) REFERENCES INGREDIENT(Ingredient_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT RCCK1 CHECK(Quantity >= 0)
SHOPPING_LIST	PRIMARY KEY (Shopping_listname, Account_name), CONSTRAINT SLFK1 FOREIGN KEY(Account_name) REFERENCES USER(Account_name) ON UPDATE CASCADE ON DELETE CASCADE
SHOPPING_LIST_CONTENT	PRIMARY KEY (Account_name, Ingredient_name, Shopping_listname), CONSTRAINT SLCFK1 FOREIGN KEY(Account_name, Shopping_listname) REFERENCES SHOPPING_LIST(Account_name, Shopping_listname) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT SLCFK2 FOREIGN KEY(Ingredient_name) REFERENCES INGREDIENT(Ingredient_name) ON UPDATE CASCADE ON DELETE CASCADE, CONSTRAINT SLCCK1 CHECK(Quantity >= 0)
USER	PRIMARY KEY (Account_name)

Table 2: Describes the constraints for each entity.

Sample Database Queries

SQL statement	**Purpose**
SELECT Recipe_name FROM RECIPE WHERE prep_time + cook_time < 60;	Search recipe with total cook time less than 60 minutes
SELECT Ingredient_name FROM SHOPPING_LIST_CONTENT WHERE Shopping_listname = 'For My Birthday';	Search all the ingredients are in the shopping list 'For My Birthday'
SELECT Friends_account_name FROM FRIEND WHERE Account_name = 'ProCook';	Find ProCook's friends
SELECT Recipe_name FROM FAVORITE WHERE Account_name = 'Kristin';	Find all the recipes in the Favorite of account 'Kristin'
SELECT F.Recipe_name, SLC.Ingredient_name, SLC.Quantity FROM FAVORITE as F, SHOPPING_LIST_CONTENT as SLC, USER as U, RECIPE_CONTENT as RC WHERE F.Account_name = 'ProCook' and F.Recipe_name = 'Elegant Brunch Chicken Salad' and SLC.Account_name = 'ProCook' GROUP BY SLC.Ingredient_name;	List required ingredients to cook a user's favorite meal.

SELECT
Recipe_name
FROM
RATING as Ra, FRIEND as Fr
WHERE
Fr.Account_name = 'ProCook' and Ra.Account_name = Fr.Friends_account_name and Star_num > 3
GROUP BY
Recipe_name;
	List my friend's recipes which they have rated above 3 stars.

SELECT
Recipe_name, Nutritional_name, Nutritional_value, Nutrition_type
FROM
NUTRITION
WHERE
Nutritional_name = 'Calories' and Nutritional_value < 400;
	Find recipe that contains less than 400 calories

SELECT
Recipe_name
FROM
FAVORITE
WHERE
Account_name = 'ProCook' and Date_added = '2017-27-03';
	Find the recipe that was added by ProCook on March 27, 2017

SELECT
Re.Recipe_name, avg(Star_num) as Rating
FROM
RECIPE as Re, RATING as Ra
WHERE
Re.Recipe_name = Ra.Recipe_name
GROUP BY
Re.Recipe_name
HAVING
avg(Star_num) > 4;
	Find recipes that have rating of above 4 stars

SELECT
Rc.Recipe_name, Cook_time, Prep_time, Picture_link, Instruction
FROM
RECIPE as Re, RECIPE_CONTENT as Rc
WHERE
Re.Recipe_name = Rc.Recipe_name AND
	List the details of recipes which content onions as ingredient

Ingredient_name = 'Onions'
GROUP BY
    Re.Recipe_name;	

SELECT
    Re.Recipe_name, Nutritional_name, Nutritional_value, Nutrition_type
FROM
    RECIPE as Re, NUTRITION as Nu
WHERE
    Re.Recipe_name = Nu.Recipe_name AND Re.Recipe_name = 'Seared Scallops with Jalapeno Vinaigrette';
	List the nutritions of the recipe 'Seared Scallops with Jalapeno Vinaigrette'

SELECT
    Rc.Recipe_name, Rc.Ingredient_name, Rc.Quantity
FROM
    RECIPE_CONTENT as Rc
WHERE
    Rc.Recipe_name = 'Good Old Fashioned Pancakes';
	List the ingredients of recipe 'Good Old Fashioned Pancakes'

INSERT into FAVORITE values
('Chili Burgers', 'ProCook', '2017-03-07');
	Insert a new favorite recipe to ProCook’s favorites

SELECT
    Account_name
FROM
    USER;
	Lists out all the active users for Admin use.

UPDATE
    RECIPE
SET
    Prep_time=7
WHERE
    Recipe_name = 'Spaghetti with Meatballs';
	Allows an Admin to change the prep time for a Recipe.

SELECT
    Recipe_name
FROM
    RECIPE
WHERE
    Date_added=date('now');
	Lists the Recipes added on the current day for an admin.

UPDATE
    RECIPE
SET
	Allows an admin change the cook time of an Recipe.

Cook_time=10
WHERE
Recipe_name = 'Spaghetti with Meatballs';	

UPDATE
RECIPE
SET
Instruction='just boil noodles'
WHERE
Recipe_name = 'Spaghetti with Meatballs';
	Allows an admin to update the Recipe instructions. To allow changes/fixes

Table 3: A set of possible queries that can be made on the database along with a description of what it does.

Computer Graphical User Interface

An.actual JavaFX prototype has been coded to practice integrating local database to an application. **All source files for the JavaFX prototype is included with the submission**. Below are some screenshots of the Graphical User Interface that was developed to reflect the results of the database.

Search Screen

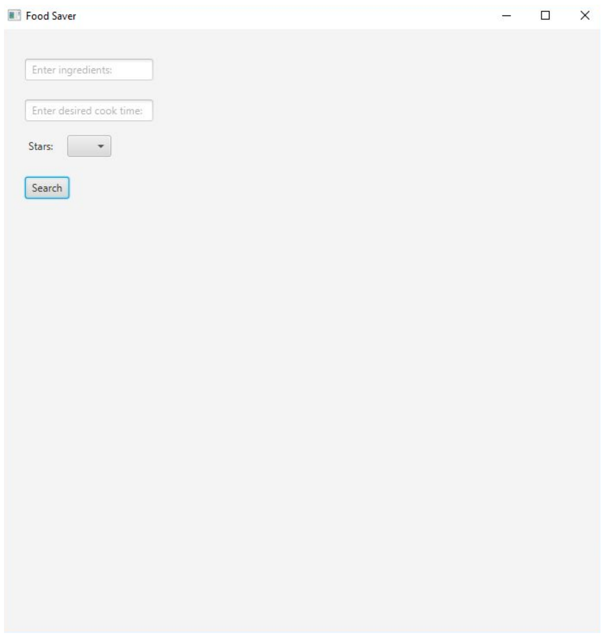
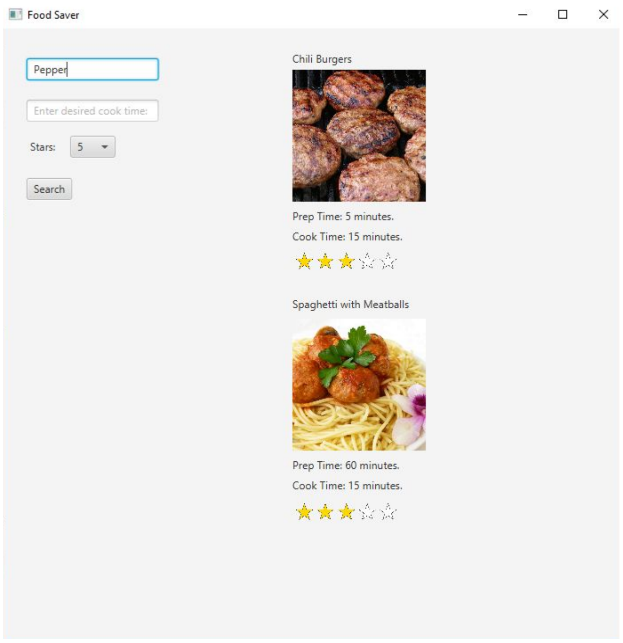


Figure 3: Search screen where the user can specify a leftover ingredient, time it takes to cook (combination of prep and cook time), and the number of stars, representing the rating for the recipes to show.

Search Screen With Results After Specifying Pepper as Leftover Ingredient



Pepper

Enter desired cook time:

Stars: 5

Chili Burgers

Prep Time: 5 minutes.

Cook Time: 15 minutes.

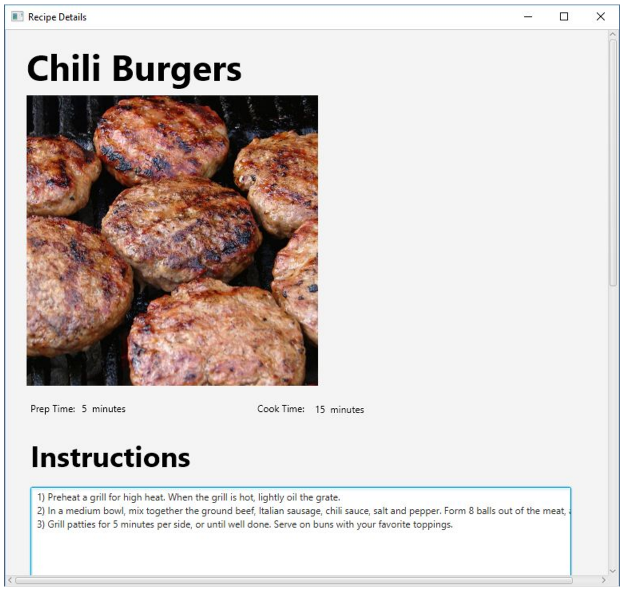
Spaghetti with Meatballs

Prep Time: 60 minutes.

Cook Time: 15 minutes.

Figure 4: Once the search button is clicked, results will be shown on the right within the same screen. Results show the recipe name, picture, rating, prep and cook times. Clicking on the image will open a detailed view of the recipe. (See Figure 5)

Recipe Details Screen



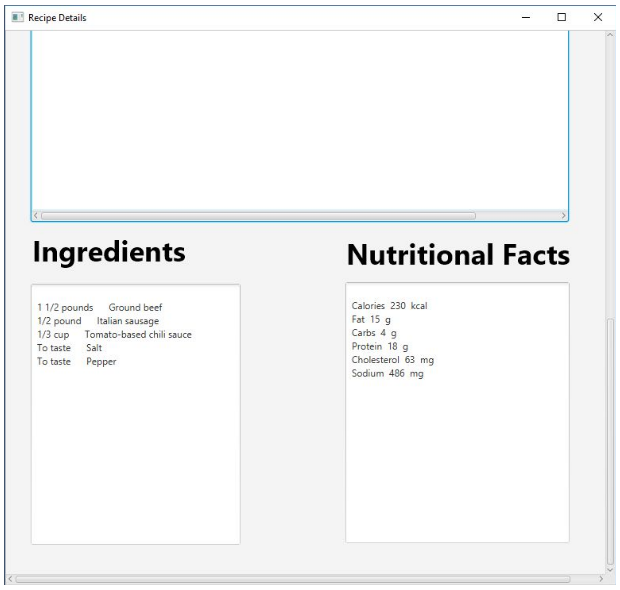
Chili Burgers

Prep Time: 5 minutes

Cook Time: 15 minutes

Instructions

1) Preheat a grill for high heat. When the grill is hot, lightly oil the grate.
2) In a medium bowl, mix together the ground beef, Italian sausage, chili sauce, salt and pepper. Form 8 balls out of the meat,
3) Grill patties for 5 minutes per side, or until well done. Serve on buns with your favorite toppings.



Ingredients

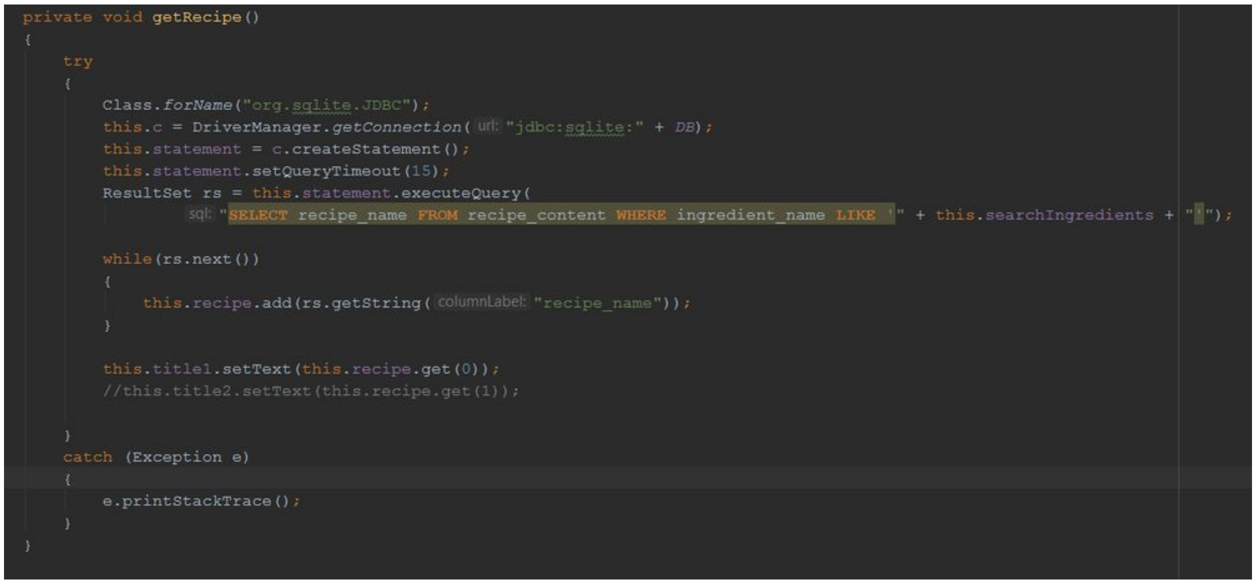
1 1/2 pounds	Ground beef
1/2 pound	Italian sausage
1/3 cup	Tomato-based chili sauce
To taste	Salt
To taste	Pepper

Nutritional Facts

Calories	230	kcal
Fat	15	g
Carbs	4	g
Protein	18	g
Cholesterol	63	mg
Sodium	486	mg

Figure 5: Details screen shows all of the information about the selected recipe. This includes recipe name, picture, rating, prep and cook times, instructions on how to cook, ingredients required, and the nutritional facts for that meal.

Sample Screenshot of the GUI's Code



```
private void getRecipe()
{
    try
    {
        Class.forName("org.sqlite.JDBC");
        this.c = DriverManager.getConnection( url: "jdbc:sqlite:" + DB);
        this.statement = c.createStatement();
        this.statement.setQueryTimeout(15);
        ResultSet rs = this.statement.executeQuery(
                sql: "SELECT recipe_name FROM recipe_content WHERE ingredient_name LIKE '" + this.searchIngredients + "'");

        while(rs.next())
        {
            this.recipe.add(rs.getString( columnLabel: "recipe_name"));
        }

        this.title1.setText(this.recipe.get(0));
        //this.title2.setText(this.recipe.get(1));

    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Figure 6: Sample snipped of code from the GUI development.