**Recipes Book**

**-OOP Project-**

**Author: Deac Melinda-Anca**

**Group: 30421**

**Year: 2nd**

**Description of the project:**

The project has the purpose of being a recipes book. It is simple to add and see recipes in it and also, it has a protection system with access passwords so that children won’t be able to get to recipes that might not be appropriate.

**Use cases:**

As for the use cases, the project has the following:

1. When opening the app, the main page will appear.

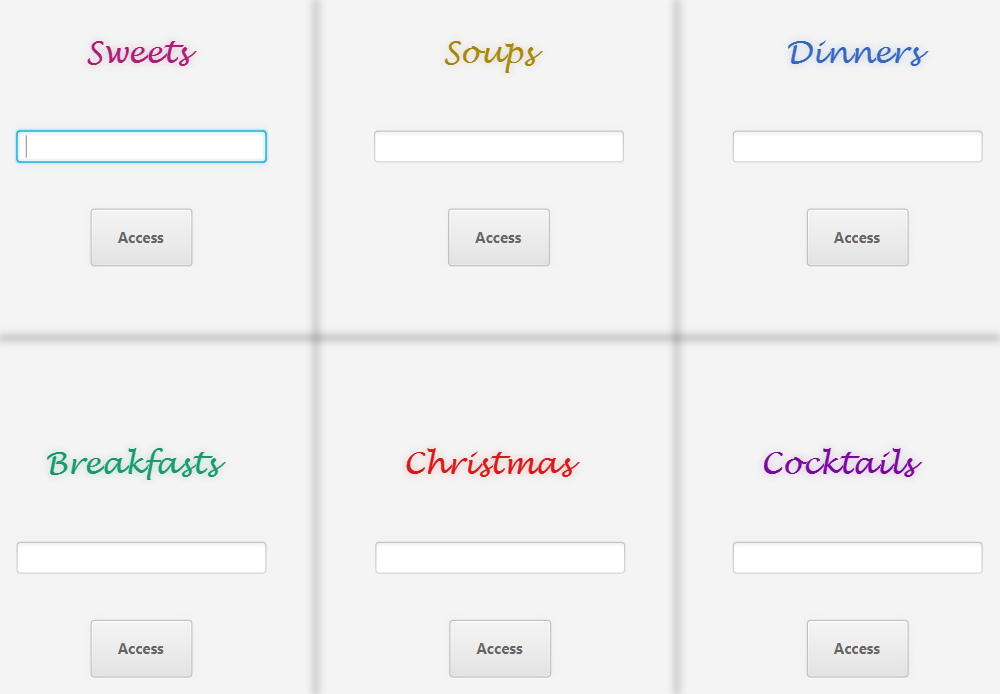
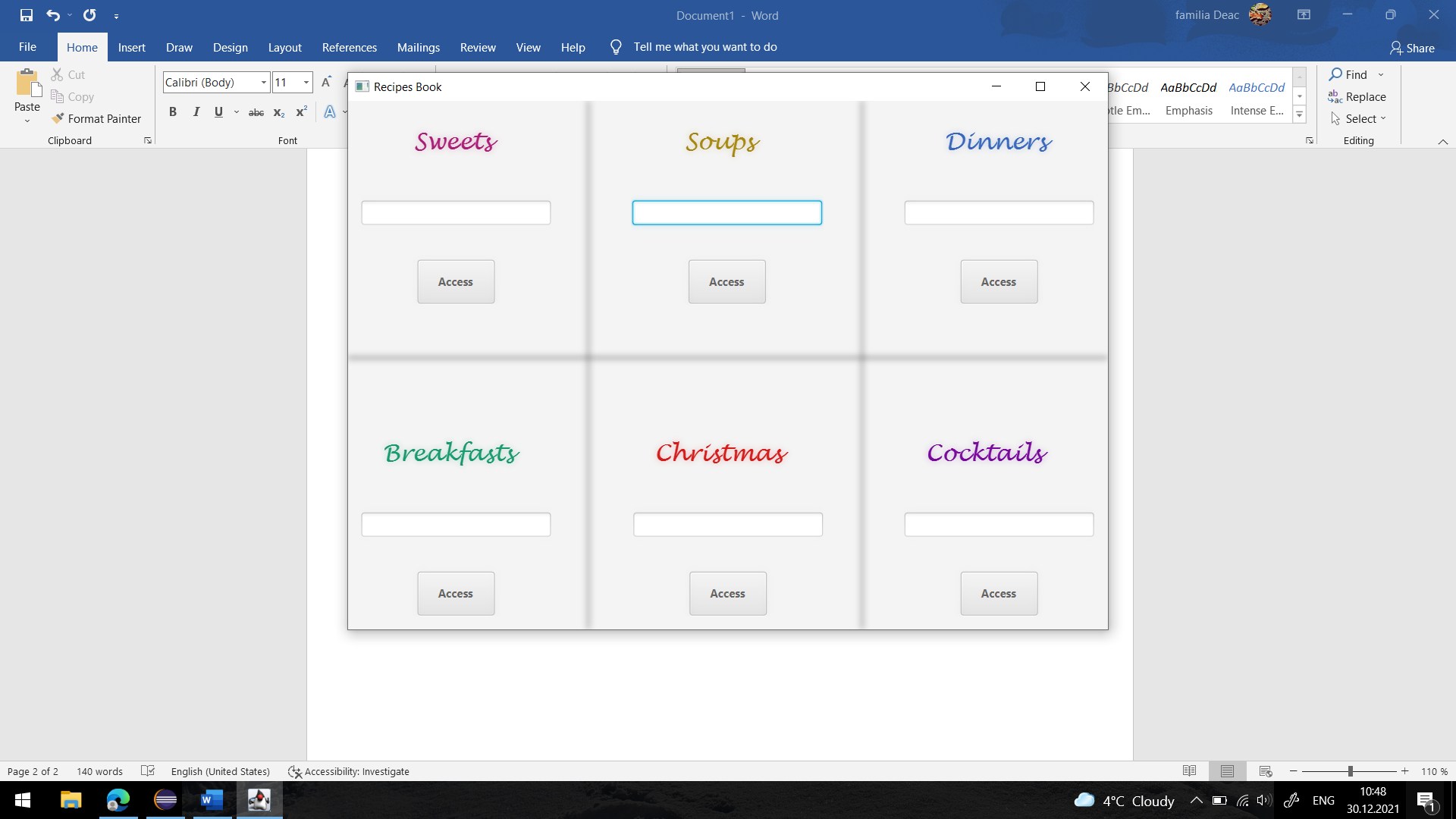


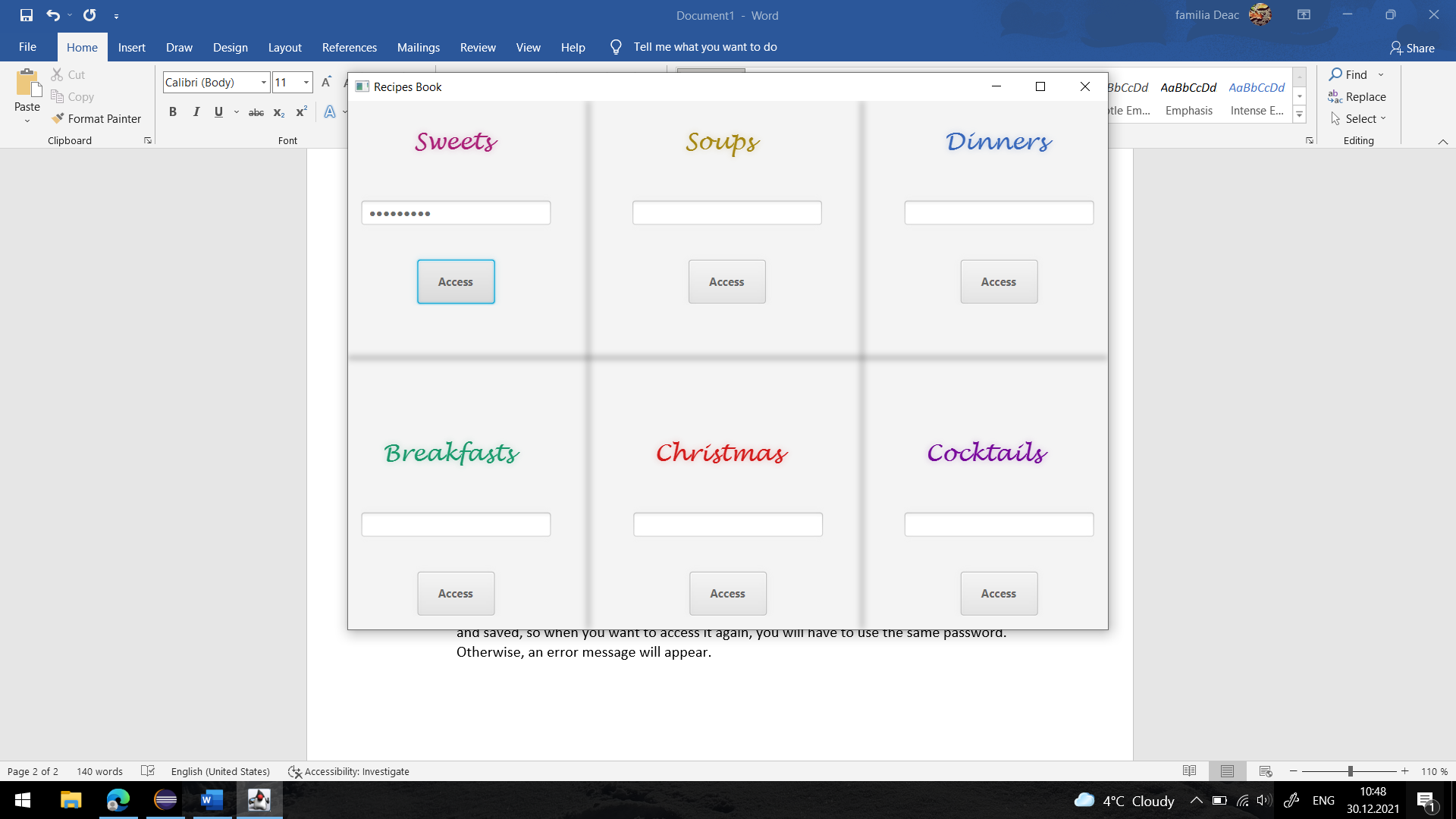
Figure 1. Main Page

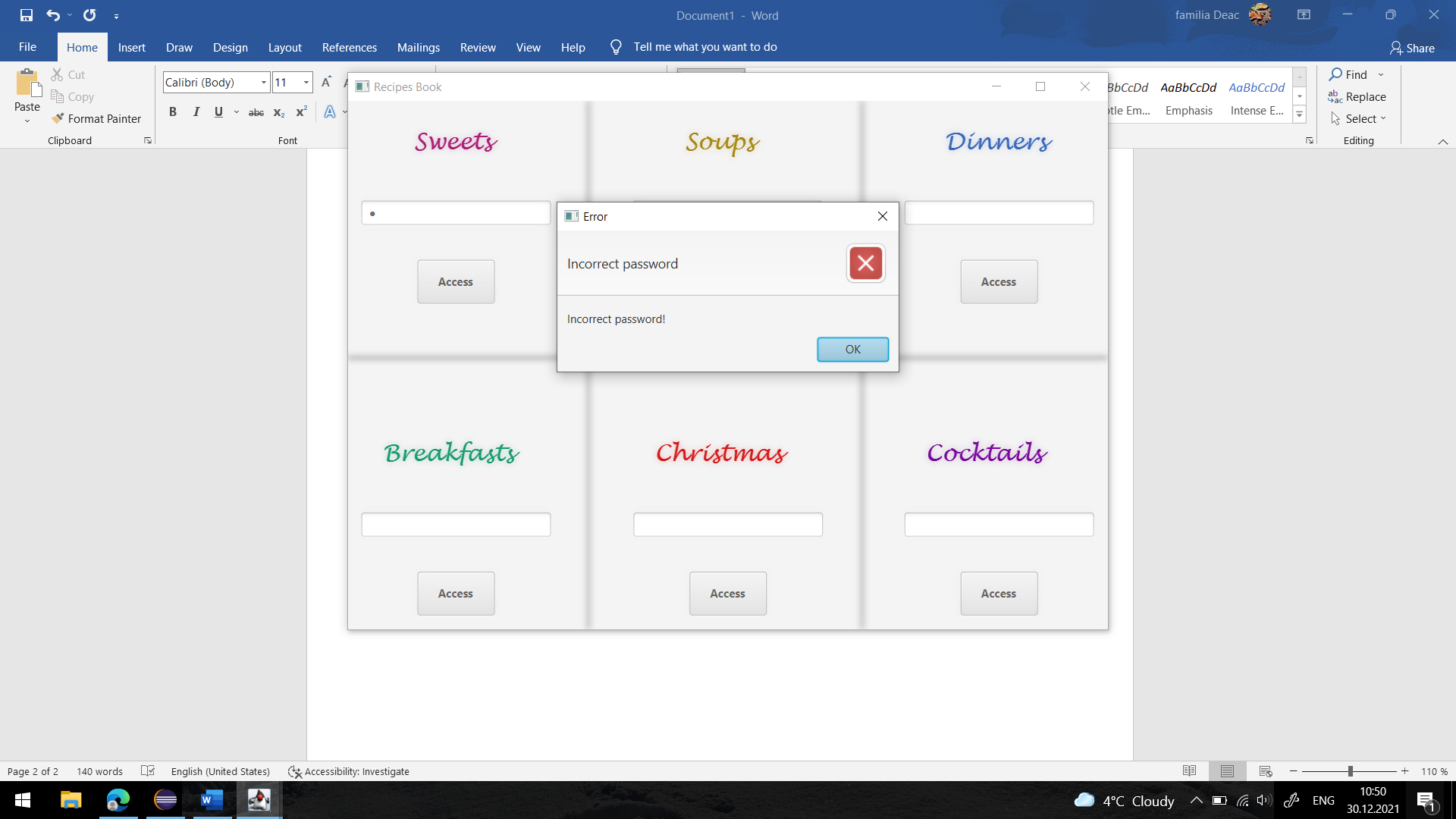
You can access a category of the recipes book by using an access password.



You click the text field, write down the password and click access.

If it is the first time accessing this category, then the password you write will be encrypted and saved, so when you want to access it again, you will have to use the same password. Otherwise, an error message will appear.



In case you entered the wrong password, this appears: .

1. If you entered the correct password, the main window of the category will show. Here, you can introduce a new recipe or view all the recipes. It is also possible to go back to the previous page.

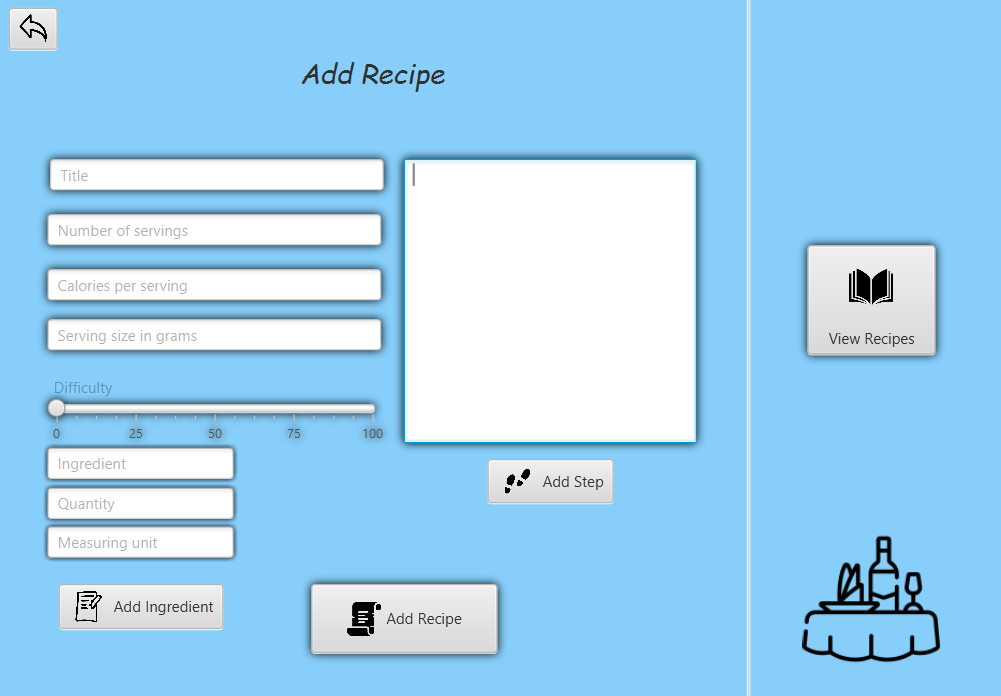
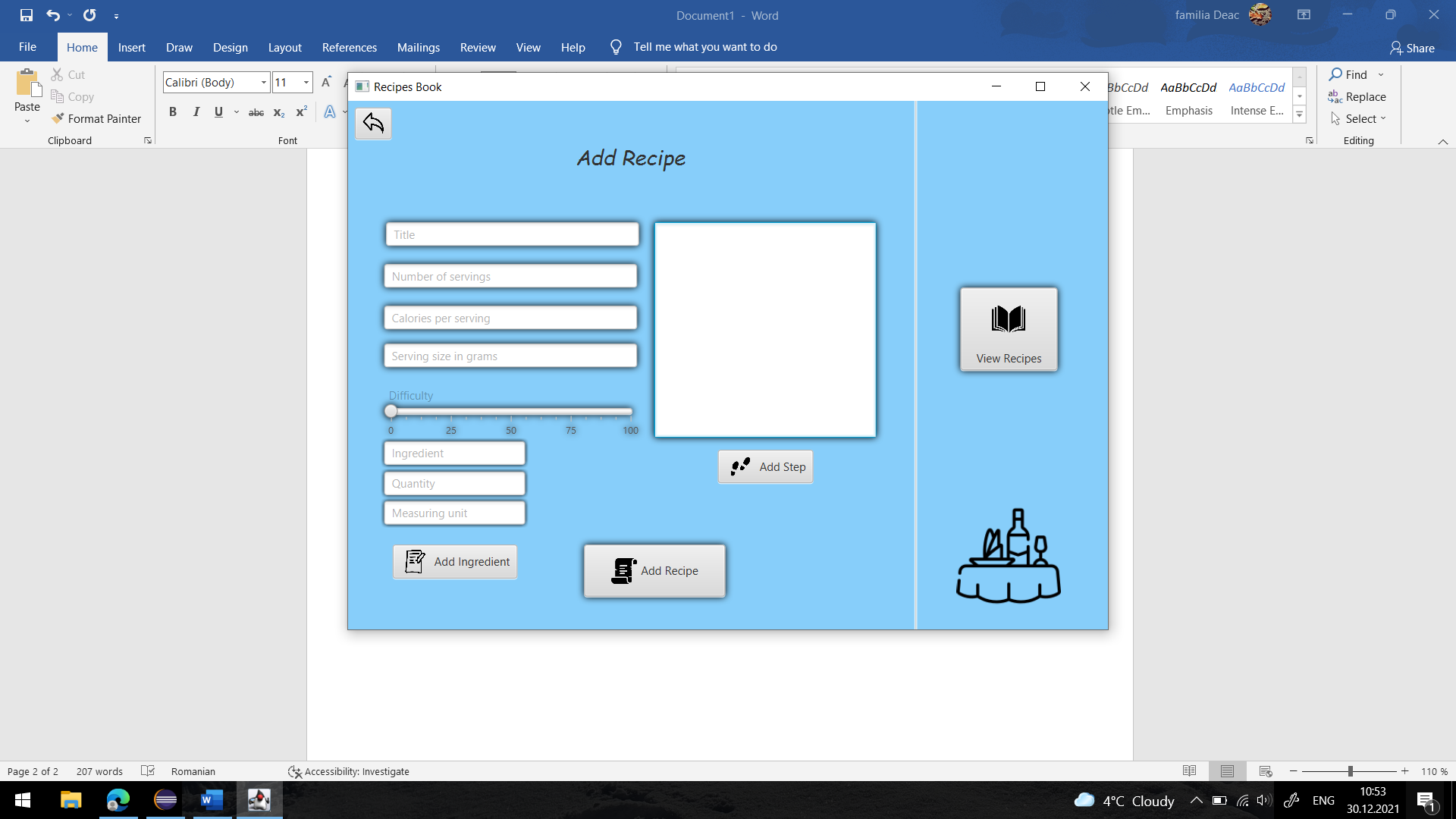
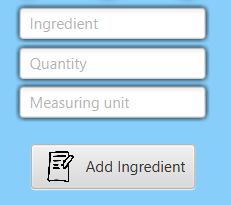


Figure 2. Dinner Recipes Main Page

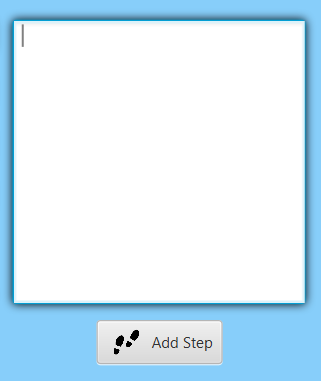
* 1. Introducing a new recipe is done in multiple steps.
     1. First, you have to write down the title, number of servings, calories per serving, serving size in grams and set the difficulty.



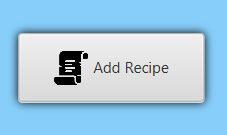
* + 1. Then, you have to introduce each ingredient one by one by setting its name, the quantity and the measurement unit. After writing those details for an ingredient, you must push the „Add Ingredient” button. After each push, the ingredient is added in the Ingredients table from the database.



* + 1. Afterwards, you have to introduce the steps one by one. Write down the steps and after each step, push the „Add Step” button. After each push, the step is added in the Steps table from the database.



* + 1. Once all ingredients have been introduced and also all steps and when the title and all the other details have been set, push the „Add Recipe” button and the recipe will be added in the database.



* 1. In order to view all the recipes from the category, you have to press the „View Recipes” button.



* 1. To go back to the previous page, you have to press the „Go Back” button.



1. Once the „View Recipes” button is pressed, a page containing all recipes from that category will appear. Here, you can scroll through them and for each, you can view the ingredients or the steps.

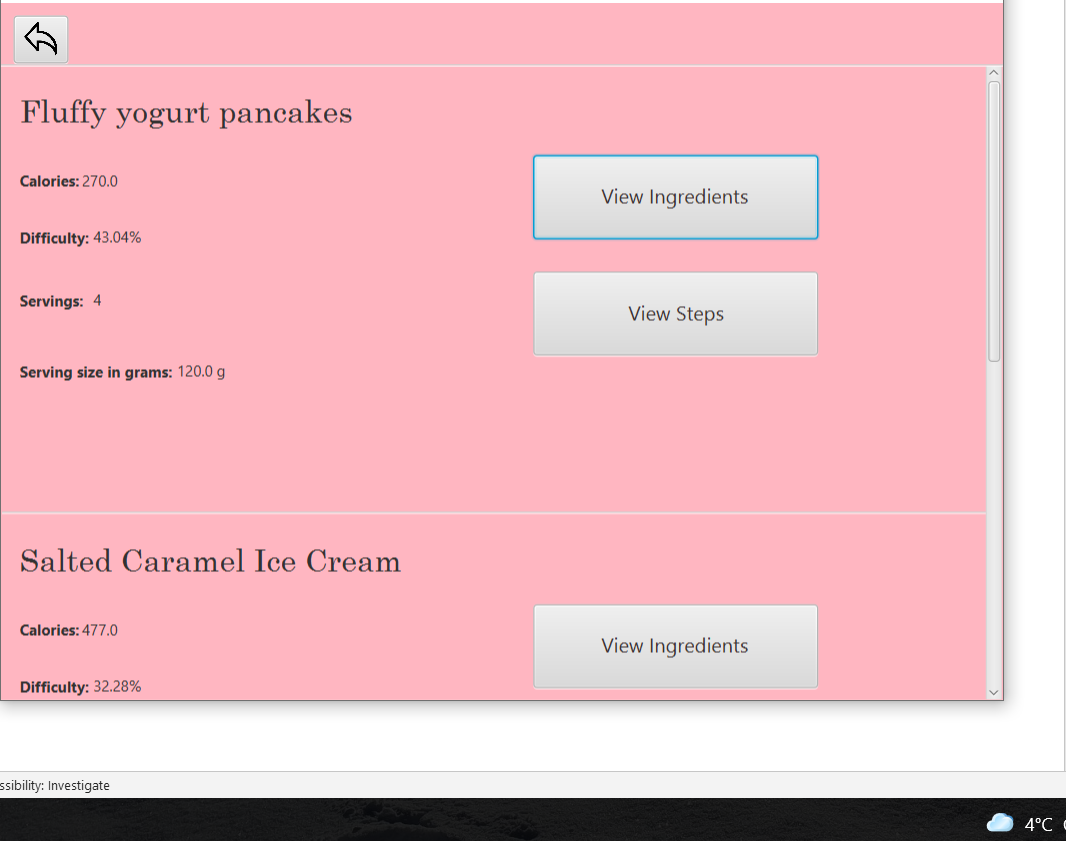
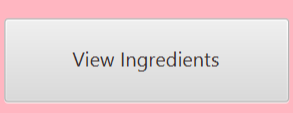
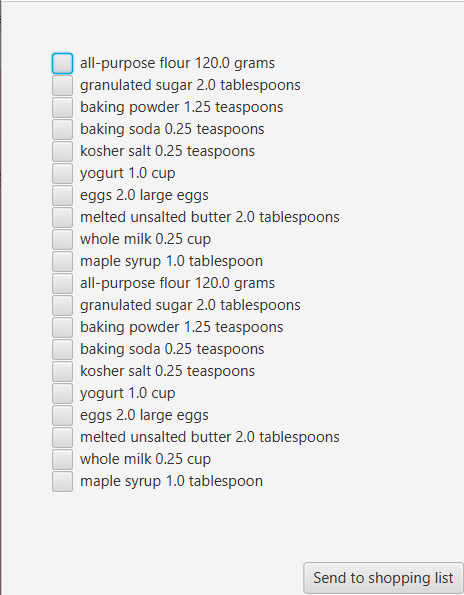


Figure 3. Sweet Recipes

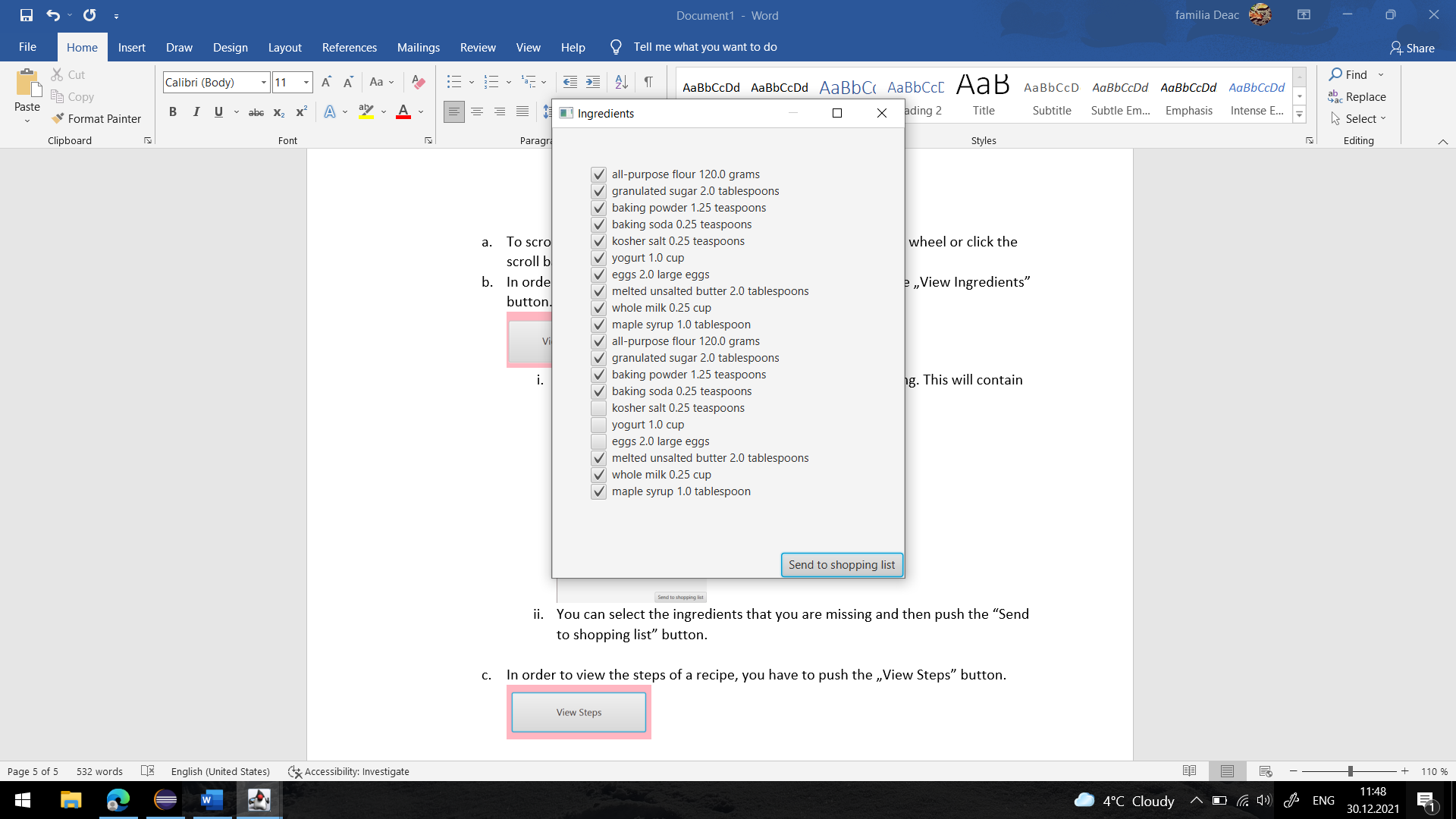
* 1. To scroll, you can use two fingers on the touchpad or the mouse wheel or click the scroll bar present in the page and pull it.
  2. In order to view the ingredients of a recipe, you have to push the „View Ingredients” button.



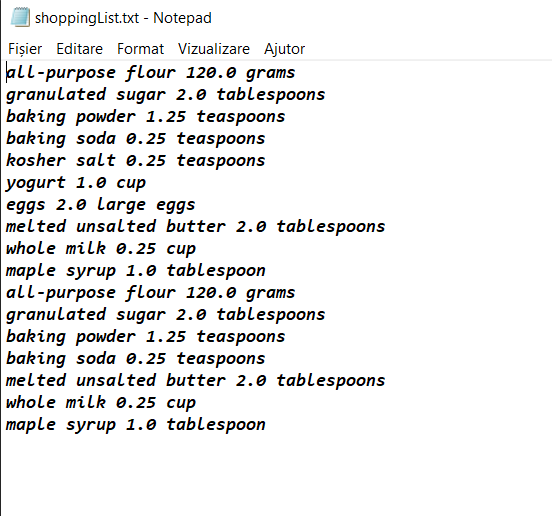
* + 1. Pushing this button will result in a new window appearing. This will contain all the necessary ingredients for the recipe.



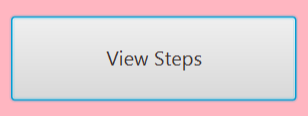
* + 1. You can select the ingredients that you are missing and then push the “Send to shopping list” button.



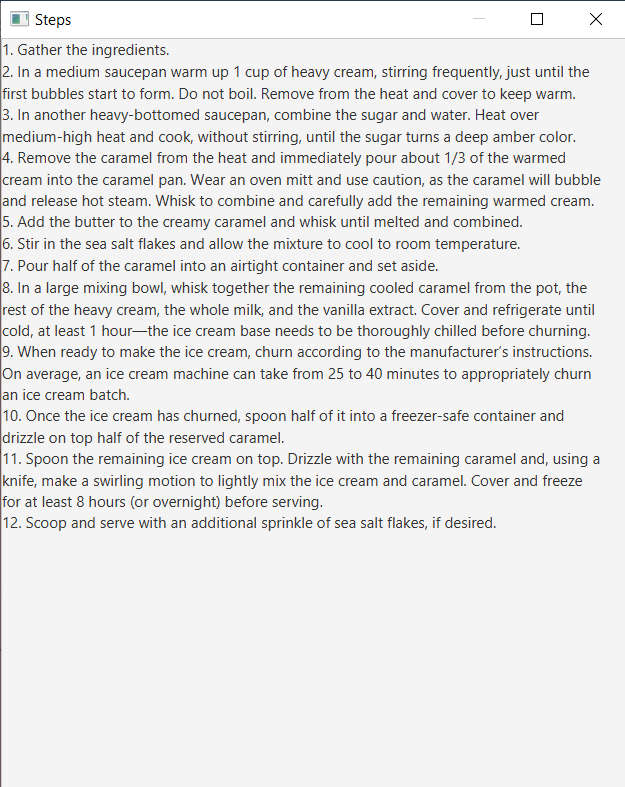
* + 1. After pushing the button, the selected items will be written in a “.txt” file.



* 1. In order to view the steps of a recipe, you have to push the „View Steps” button.



* + 1. After pressing the button, a new window will appear, containing all steps.



* 1. In order to go back to the previous page, you can push the “Go back” button.



**Code and Solution Presentation**

The program is written using JavaFX and SceneBuilder. The architecture I used is **MVC** (Model-View-Controller).

The View section is composed of all “.fxml” files which are used to handle the appearance of the windows.

The Controller section is composed of all files ending with “Controller.java” and also the “DatabaseHelper” and “BCrypt” files.

The Model section is composed of the classes that represent the recipes, the data that I need to hold, which will eventually go in the database.

I have three main classes:

* Recipe. This class holds the structure for each recipe.
* Ingredients. This class holds the structure for each ingredient. In the Recipe class, I have a list of elements of type Ingredients.

**private** List<Ingredients> ingredients;

* Steps. This class holds the structure for each step. In the Recipe class, I have a list of elements of type Steps.

For each window, I created a “.fxml” file in which I created the design using SceneBuilder and also, for each such file, there exists a controller file which handles the action. There are some exceptions, though. The window that appears when you push the “View Recipes” button is completely created using the controller. Here, the view and the controller part have merged. I did this because I wanted the procedure of creating this window to be very clear. So everything is done step by step through code. This is also where I used lambda expressions:

b.setOnAction(event -> {sweetsRecipesGoBackBtnClicked(event);});

For the encryption of the password, I used BCrypt class and saved the encrypted password to a file. As an example, “sweets123” is translated to “$2a$10$PddYwWpawHGWKkTthrzss.yjA2xKg5LE9WuEq72nVeedxgD7lyBLO”. In order to reset the password, you have to delete the contents of the file corresponding to the respective category’s password.

Also, in order to connect to the database I created for this project in PostgreSQL, I created a separated class, in which all the database work is performed. This class comes as a bridge between the program and the database.

**private** **final** String url = "jdbc:postgresql://localhost/recipesbook";

**private** **final** String user = "postgres";

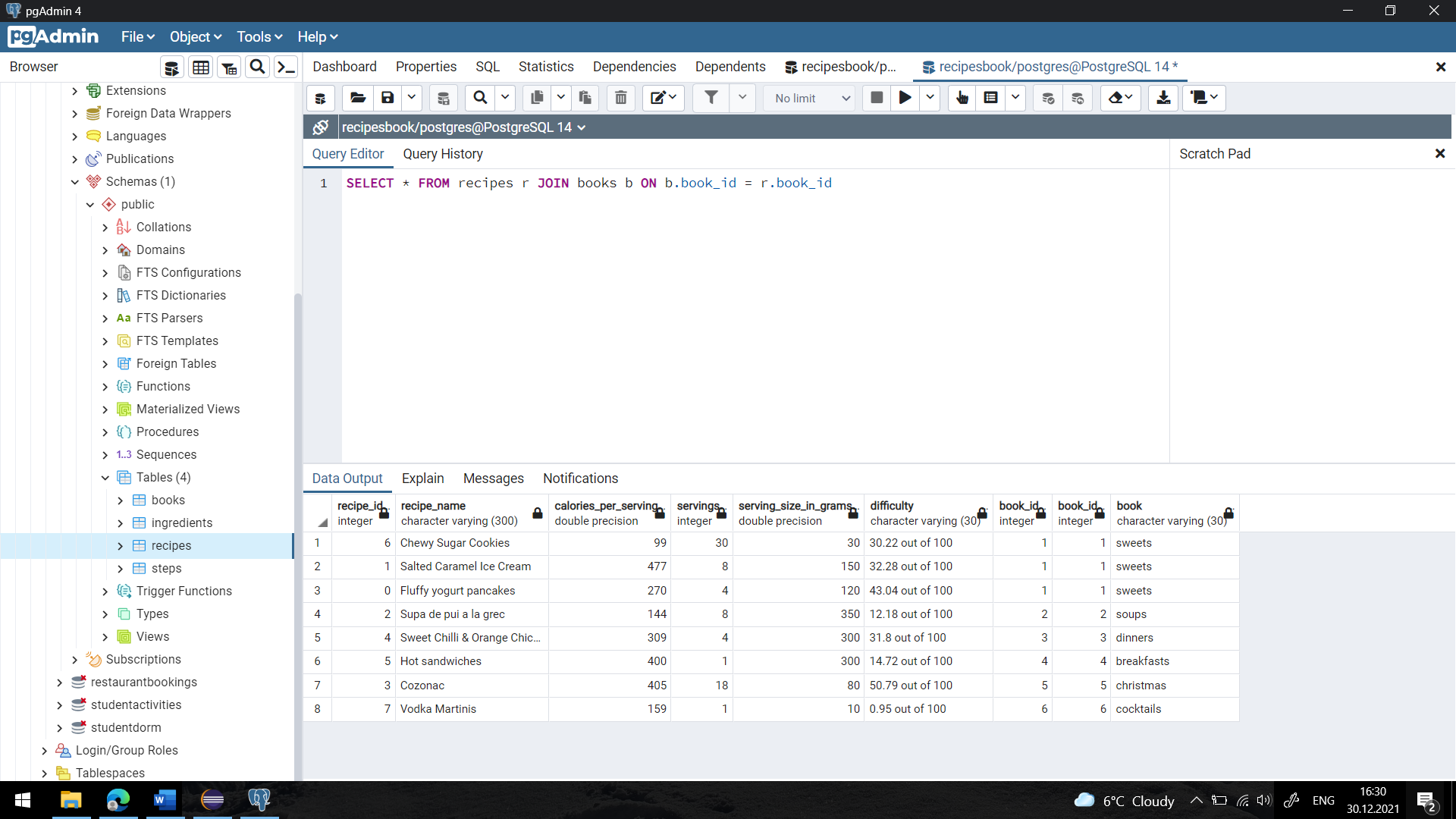
**private** **final** String password = "1234567890";

**public** Connection connect() **throws** SQLException {

**return** DriverManager.*getConnection*(url, user, password);

}

Here I will attach a glimpse of how some of the data in the database looks like.



The code if very beginner like written, but it 100% follows the object oriented programming principles: all fields are private and have getters and setters, I make good use of the constructors, I did not use an interface, because I do not store directly the data. In the case where I would’ve saved the recipes using a list, I would’ve made an interface called Recipe and implement it for each category of recipes. But since I use the database, I did not need such an interface.

**Conclusion**

The idea for this application came to me while madly searching through internet for one tasty recipe I once found. So, for when I do find a recipe that I like, I will insert it in the recipes book application and know where to go looking for it. It’s pretty easy to use, having a friendly user interface.

**Further Improvements**

In further updates of the application, I might add a search option, so that when viewing the recipes, I will also be able to search for keywords. Maybe, an even better improvement would be to be able to search for recipes containing a certain ingredient. Then, it would be useful to be able to sort them relative to the number of calories or to the difficulty. There is plenty of room for improvements, but for a beginning project I believe it is satisfying enough.