Requirements Analysis & Specification

1- Scenarios

Scenario 1: Maria plans to prepare dinner once she returns home after a day of classes. Upon reaching home, she decides to download and install the Differeat application. After creating an account, Maria logs into the app. To find a suitable recipe, she enters the ingredients available in her kitchen and may optionally specify a preferred type of cuisine. The application considers the entered ingredients and cuisine preference and presents her with five recipe suggestions. After carefully considering the options, she selects one and follows the provided cooking instructions to prepare her dinner. Finally, she savours the meal and thoroughly enjoys it.

Scenario 2: *Jason wants to plan his meals for the week and ensure he has all the necessary ingredients. He opens the Differeat application and logs into his account. He accesses the "Shopping List" feature and reviews the ingredients needed for the recipes he has selected. He adds any missing ingredients to his shopping list and specifies the quantities needed. The application generates the complete shopping list based on the selected recipes and quantities. Jason goes to the grocery store, checks off the items from his shopping list as he purchases them, and ensures he has everything he needs for the upcoming meals.*

Scenario 3: *Paolo, a cooking enthusiast, uses the Differeat application to enhance his culinary experience. He saves favourite recipes and manages them in the "Favourites" section. Paolo receives notifications for new recipes and personalized recommendations, which he can manage in his notification preferences. He can also access detailed nutritional information for each recipe, including calories, macronutrients, vitamins, and minerals. These features enhance Paolo's cooking experience and encourage him to explore more recipes.*

Scenario 4: *Greta, a student with specific dietary restrictions, wants to find suitable recipes. She logs into the Differeat application and accesses her account settings. In the "Dietary Preferences" section, she indicates her dietary restrictions, such as being gluten-free and lactose intolerant. The application saves her preferences. Greta then proceeds to search for recipes by entering the available ingredients she has at home. The application matches the ingredients with recipes that comply with her dietary restrictions and presents the top 5 suitable recipes. Greta selects a recipe and follows the provided cooking instructions, ensuring it aligns with her dietary needs. She successfully prepares a delicious meal that meets her dietary requirements and enjoys the satisfaction of finding a recipe tailored to her needs.*

2- Use cases

From the scenario 1, we retrieve these use cases:

- Sign Up User
- Login
- Input Ingredients
- Select a cuisine
- Choose a Recipe

UC1	**Sign Up User**

Actors	User
Entry Condition	The user has installed the application on his/her device
Flow of Events	1. Click on “Sign up” button 2. Fill all the necessary information 3. Click on “Confirm” button 4. The system saves the data
Exit Condition	The user has successfully registered and can use the application
Exceptions	1. The user is already signed up 2. The user didn’t fill all the mandatory fields with valid data 3. The username is already taken 4. The e-mail is already registered 5. All the exceptions are handled by notifying the user and taking him back to the sign-up activity.

UC2	**Login**
Actors	User
Entry Condition	The user is previously successfully signed up
Flow of Events	1. The user opens the application on his device 2. He enters his credentials in the “Username” and “Password” fields of the home page 3. The user clicks on the “Log in” button 4. The user is successfully logged in his session
Exit Condition	The user is successfully redirected to his/her session
Exceptions	1. The user enters invalid Username 2. The user enters invalid Password 3. All the exceptions are handled by notifying the user and taking him back to the login activity

UC3	**Input Ingredients**
Actors	User
Entry Condition	User has successfully logged in
Flow of Events	1. User opens the application. 2. User selects the "Input Ingredients" option. 3. Application displays a list of ingredients from the database. 4. User selects the ingredients they have available.

	5. User submits the selected ingredients to the application.
Exit Condition	The application receives the user's inputted ingredients and proceeds to find matching recipes
Exceptions	If the user submits none, invalid or non-existent ingredients, the application displays an error message and prompts the user to input valid ingredients

UC4	**Select a cuisine**
Actors	User
Entry Condition	The user has already inputted ingredients
Flow of Events	1. User select none, one or more type of cuisine 2. User submits his selection to the application
Exit Condition	The application receives the user's inputted cuisine and filters matching recipes
Exceptions	If the user selects an invalid or non-existent cuisine, the application displays an error message and prompts the user to select a valid cuisine

UC5	**Choose a Recipe**
Actors	User
Entry Condition	The user has already inputted ingredients and specified or not a type of cuisine
Flow of Events	1. The application displays a list of maximum five recipes 2. User selects one recipe 3. User submit his selection to the application
Exit Condition	The application displays the recipe name, ingredients, cooking instructions, and cooking time
Exceptions	If the user selects none, invalid or non-existent recipe, the application displays an error message and prompts the user to select a valid recipe

From the scenario 2:

UC6	**Create Meal Plan**
Actors	User
Entry Condition	The user has logged into their Differeat account.

Flow of Events	1. The user accesses the "Meal Plan" feature in the Differeat application. 2. The user selects the desired recipes for their meal plan, specifying the meals they want to prepare for the week. 3. The user reviews the selected recipes and confirms their meal plan. 4. The application generates a weekly meal plan based on the selected recipes.
Exit Condition	The application displays the complete meal plan for the week, including the chosen recipes for each meal.
Exceptions	If the user does not select any recipes or provides incomplete information, the application displays an error message and prompts the user to select valid recipes and specify all required details.

UC7	**Generate Shopping List**
Actors	User
Entry Condition	The user has a confirmed meal plan for the week.
Flow of Events	1. The user navigates to the "Shopping List" feature in the Differeat application. 2. The application retrieves the ingredients needed for the selected recipes in the user's meal plan. 3. The user reviews the generated shopping list and has the option to modify quantities or remove specific items if necessary. 4. The user finalizes the shopping list.
Exit Condition	The application presents the user with a complete shopping list that includes all the necessary ingredients for their chosen recipes.
Exceptions	I If there are no ingredients required for the selected recipes, the application displays a message indicating that no shopping list is available.

UC8	**Shopping List Management**
Actors	User
Entry Condition	The user has a generated shopping list.
Flow of Events	1. The user opens the shopping list in the Differeat application.

	2. The user goes to the grocery store and checks off the items from their shopping list as they purchase them. 3. If the user cannot find a specific item or decides not to purchase it, they have the option to remove it from the shopping list. 4. After completing their shopping, the user confirms that they have obtained all the necessary items.
Exit Condition	The user's shopping list is updated to reflect the items they have purchased or removed.
Exceptions	If the user encounters any issues or discrepancies with the shopping list, they can report them through the application's support or feedback channels.

From scenario 3:

UC9	**Save Recipe as Favourite**
Actors	User
Entry Condition	The user is logged into his account and viewing a recipe.
Flow of Events	1. The user selects the "Save as Favourite" option for a recipe. 2. The application associates the selected recipe with the user's account and adds it to his favourites. 3. The application confirms the successful addition of the recipe to the user's favourites.
Exit Condition	The selected recipe is successfully saved as a favourite for the user.
Exceptions	If the recipe is already saved as a favourite, the application displays a message indicating that the recipe is already in the user's favourites.

UC10	**Manage Favourite Recipes**
Actors	User
Entry Condition	The user is logged into their account and accessing the "Favourites" section.
Flow of Events	1. The user navigates to the "Favourites" section in the application. 2. The application displays a list of the user's favourite recipes. 3. The user can perform actions such as removing a recipe from their favourites,

	editing recipe details, or organizing the favourite recipes. 4. The application updates the user's favourites list according to the performed actions.
Exit Condition	The user successfully manages their favourite recipes, including removing recipes, editing details, or organizing the list.
Exceptions	If the user has no favourite recipes, the application displays a message indicating that there are no recipes saved as favourites.

UC11	**View Nutritional Information**
Actors	User
Entry Condition	The user is viewing a recipe.
Flow of Events	1. The user selects the "View Nutritional Information" option for a recipe. 2. The application retrieves the nutritional information for the selected recipe based on the ingredients and quantities used. 3. The application displays the nutritional information, including details such as calories, macronutrients (carbohydrates, proteins, fats), vitamins, and minerals.
Exit Condition	The user successfully views the nutritional information for the selected recipe.
Exceptions	If the nutritional information is not available for a recipe, the application displays a message indicating that the information is not provided.

From scenario 4:

UC12	**Set Dietary Preferences**
Actors	User
Entry Condition	The user has logged into their Differeat account.
Flow of Events	1. The user navigates to the "Account Settings" section in the Differeat application. 2. The user selects the "Dietary Preferences" option. 3. The user indicates their specific dietary restrictions, such as being gluten-free and lactose intolerant. 4. The application saves the user's dietary preferences.

Exit Condition	The application saves the user's dietary preferences for future reference.
Exceptions	If the user encounters any issues while setting their dietary preferences, such as invalid selections or technical errors, the application displays an error message and prompts the user to provide valid information.

3- Functional requirements:

From the prototype scope define on the feasibility study, we refine functionality we need to achieve that:

S1 - User Input:

- R1 - The application should allow users to select ingredients they have from a predefined list of ingredients in a database.

S2 - Recipe Matching:

- R2 - The application should have a matching algorithm that takes the user-inputted ingredients and matches them with recipes in a database.
- R3 - The matching algorithm should prioritize recipes that have a higher number of matching ingredients.
- R4 - The matching algorithm should consider the cuisine preference selected by the user, if any.

S3 - Recipe Display:

- R5 - The application should display the top 5 matching recipes to the user.
- R6 - The application should display the recipe name, ingredients, cooking instructions, and cooking time for each recipe.

S4 - Cuisine Selection:

- R7 - The application should allow users to choose a specific cuisine (e.g., Turkish, Chinese, Italian) for the recipe recommendations.
- R8 - The application should have a database of recipes for each cuisine available.

S5 - Basic User Management:

- R9 - The application should allow users to create a new account and login.
- R10 - The application should allow users to update their account information (e.g., name, email address, password).

For further development these functionalities will be covered by these other requirements:

S6 - Favourite Recipes:

- *R11 - The application should allow users to save recipes as their favourites, associating them with their user account.*

- *R12 - The application should provide a "Favourites" section where users can view and manage their saved favourite recipes.*
- *R13 - The application should allow users to remove recipes from their favourites list if desired.*

S7 - Search History:

- *R14 - The application should provide a search history feature that records and displays the user's previous search queries.*
- *R15 - The search history feature should include information such as the search keywords, date and time of the search, and the matching recipes.*
- *R16 - The application should allow users to clear their search history if desired.*

S8 - Personalization

- *R17 - The application should provide an interface for users to set their dietary preferences and restrictions, such as vegetarian, vegan, gluten-free, and others.*
- *R18 - The system should use the user's dietary preferences and restrictions to generate personalized recipe recommendations that align with their chosen preferences.*

S8 - Notifications

- *R19 - The application should send notifications to users for updates on new recipes, personalized recommendations, and alerts related to their favourite recipes.*
- *R20 - Users should be able to manage their notification preferences, such as enabling or disabling specific types of notifications.*

S9 - Nutritional Information:

- *R21 - The application should display nutritional information for each recipe, including details such as calories, macronutrients (carbohydrates, proteins, fats), vitamins, and minerals.*
- *R22 - The nutritional information should be based on the ingredients and quantities used in the recipe.*

S10 – Shopping list

- *R23 - The application should provide a shopping list feature that generates a list of ingredients required for the selected recipes.*
- *R24 - Users should be able to view and manage their shopping list, including adding, removing, and updating quantities of ingredients.*

S11 – Quantity management

- *R25 - The application should provide a quantity management feature that allows users to specify the quantity of each ingredient they have and adjust it as needed.*
- *R26 - Users should receive reminders or notifications when the quantity of certain ingredients in their inventory falls below a specified threshold.*
- *R27 - The admin management section should allow administrative users to manage user accounts, including creating, editing, and deleting accounts.*

S12 – Admin Management Section

- *R28 - The admin management section should provide functionalities to manage the ingredient and recipe database, including adding new ingredients, updating existing ingredients, and removing ingredients that are no longer needed.*
- *R29 - Administrative users should be able to add, edit, and delete recipes from the database, ensuring the availability of up-to-date recipes for users.*
- *R30 - The admin management section should have proper authentication and access controls to restrict access to authorized administrative users only.*

* Sn for scope number n ** Rm for requirement number m

4- Non-Functional requirements:

Usability

The "Differeat" application prioritizes usability and user experience to create an intuitive and enjoyable cooking experience for students. With a user-friendly interface, clear input processes, personalization options, and responsive design, the app offers a seamless and tailored experience. Efficient performance, visual appeal, contextual help, and social media integration further enhance user engagement. By continuously gathering user feedback and implementing improvements, "Differeat" strives to deliver a user-centric app that empowers students to cook delicious, healthy meals with ease and satisfaction.

Performance

The "Differeat" application places a strong emphasis on performance, aiming to provide users with a seamless and efficient cooking experience. Through responsiveness, minimized loading times, swift search and filtering capabilities, and seamless data synchronization, the app ensures a smooth user journey. Scalability, network efficiency, error handling, and continuous optimization contribute to a reliable and high-performing application. By prioritizing performance optimization, "Differeat" strives to deliver an exceptional user experience, empowering students to access recipes and navigate the app effortlessly and without frustration.

Reliability

Reliability is a cornerstone of the "Differeat" application, establishing trust and dependability for users. Through high uptime, effective error handling, crash recovery, data integrity measures, fault tolerance, and performance monitoring, the app ensures a consistent and stable user experience. The implementation of version control, disaster recovery planning, continuous testing, and user feedback mechanisms further enhance reliability. By prioritizing reliability, "Differeat" fosters user confidence, ensuring uninterrupted access to recipes, protecting user data, and delivering a trustworthy cooking companion that students can rely on.

Supportability

Supportability lies at the core of the "Differeat" application, providing a solid framework for maintenance, user support, and future growth. With **a modular and maintainable design, version control, comprehensive documentation, and effective error reporting, "Differeat" ensures ease of maintenance and updates.** Bug tracking, training materials, and a responsive support system enhance user assistance. Scalability, continuous improvement, and a vibrant user community further contribute to the application's supportability, empowering users with seamless support and fostering an environment of ongoing development and user satisfaction.

Implementation

The implementation phase is crucial for the success of the "Differeat" application, laying the groundwork for a seamless cooking experience. By carefully considering the technology stack, architecture, security measures, and database optimization, the application is designed to meet scalability, performance, and data integrity requirements. Adhering to coding standards, conducting thorough quality assurance, and integrating external services efficiently ensures a solid and reliable codebase. Leveraging performance optimization techniques, cloud deployment, and a continuous integration and deployment pipeline, the application is fine-tuned for optimal performance and ease of updates. Comprehensive technical documentation and knowledge transfer enable efficient maintenance and support. With a focus on implementation, "Differeat" emerges as a robust application, providing users with a trustworthy and effortless cooking companion.

Interface

The interface of the Differeat application takes centre stage, offering a user-focused experience that facilitates effortless cooking. With an intuitive and user-friendly design, responsive layout, and consistent visual aesthetics, Differeat ensures users can navigate the app with ease. The interface prioritizes accessibility, adhering to WCAG 2.1 guidelines, making it inclusive for users with disabilities. Clear and readable text, along with multimedia integration, enhances the overall user experience. Proper error handling and feedback mechanisms help users understand and resolve issues effectively. The interface supports localization and internationalization, catering to a global audience. Usability testing and user feedback play a vital role in continuous improvement, ensuring the interface evolves with user needs. By emphasizing these interface requirements, Differeat creates a seamless and engaging interface that enhances the joy of cooking for all users.

Security And Privacy

Security and privacy are paramount in the design of the Differeat application, aiming to protect user data and foster trust. By implementing robust encryption, secure authentication, and access control mechanisms, Differeat ensures the confidentiality and integrity of user information. Secure coding practices and API protection mitigate common vulnerabilities and safeguard against unauthorized access. Payment processing adheres to industry standards, ensuring the security of financial transactions. Transparent privacy practices, consent management, and data protection measures establish user confidence. Anonymization and aggregation of data protect user privacy while enabling valuable analytics. **Compliance with data protection regulations such as GDPR and CCPA ensures adherence to legal requirements.** Regular security audits and updates maintain a robust security posture. By addressing these security and privacy requirements, Differeat cultivates a safe and private environment, allowing users to confidently engage with the application and focus on their cooking experience.

Packaging

The packaging of the Differeat application plays a vital role in ensuring a seamless deployment process and enhancing the user experience. By addressing various packaging requirements, the application can be efficiently distributed, installed, and updated on different platforms. Compatibility with target platforms, maintaining file integrity, and optimizing package size are essential considerations. The installation process should be user-friendly, providing clear instructions and progress indicators. Versioning and updates enable continuous improvements and bug fixes. Implementing digital signatures enhances security and authenticity. Including licensing and copyright information ensures compliance and protection of intellectual property rights. Packaging should align with the requirements of distribution channels, such as app stores, and facilitate the uninstallation process. Documentation and support resources assist users in installing and using the application. By focusing on these packaging requirements, the Differeat application can provide a smooth and hassle-free deployment experience, ensuring user satisfaction and adoption.

Class Diagram

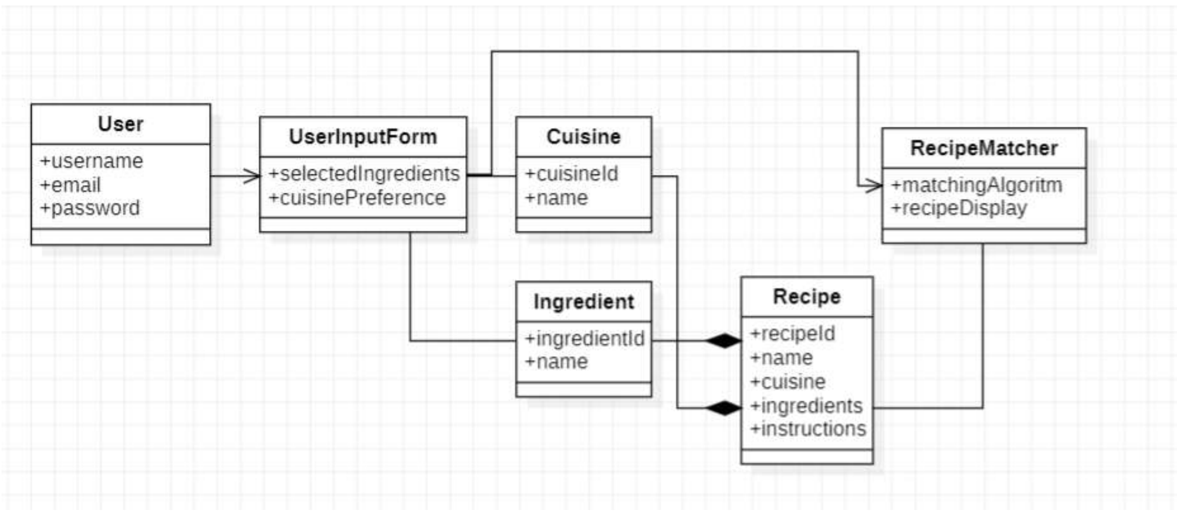
The "User" class represents a user of the application and contains attributes such as userId, username, password, and email. It is associated with the user management functionalities.

The "Ingredient" class represents an ingredient and contains attributes such as ingredientId and name.

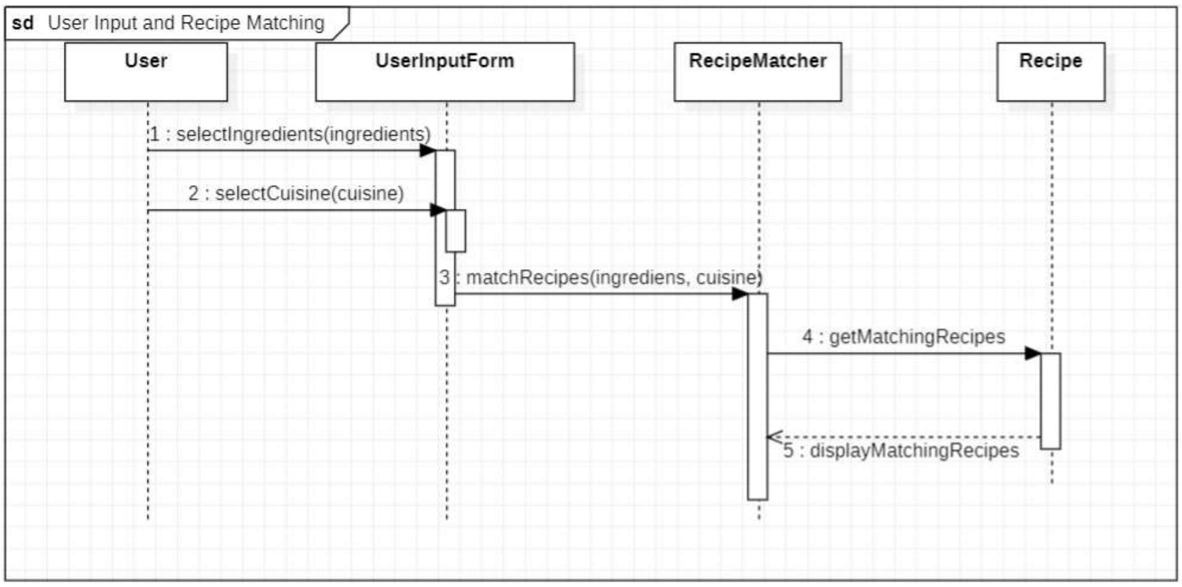
The "Recipe" class represents a recipe and contains attributes such as recipeId, name, cuisine, ingredients, and instructions.

The "UserInputForm" class represents the user's input form and contains attributes such as selectedIngredients and cuisinePreference. It is associated with the user input functionality.

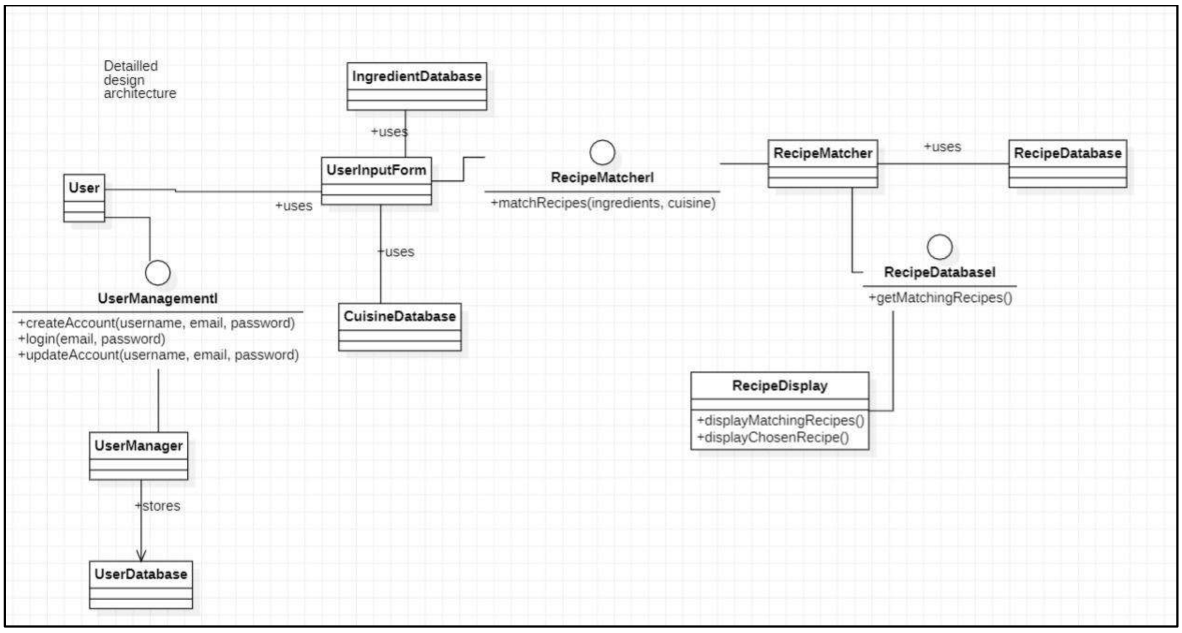
The "RecipeMatcher" class handles the matching algorithm and is associated with the Ingredient and Recipe classes. It uses the selected ingredients and cuisine preference to match recipes. It also handles the display of recipes and is associated with the Recipe class. It displays the top matching recipes along with their details.



Sequence Diagram


sd User Input and Recipe Matching
User
UserInputForm
RecipeMatcher
Recipe
1 : selectIngredients(ingredients)
2 : selectCuisine(cuisine)
3 : matchRecipes(ingrediens, cuisine)
4 : getMatchingRecipes
5 : displayMatchingRecipes

Design Architecture



System models

Objects with he attributes and the relationship between them

- Ingredient: Name, type, and quantity
- Recipe: Name, ingredients, and instructions
- Cuisine: Name and list of typical ingredients
- User: Input

- Application: Recipe name and relevant score (display)

Classes are same as objects ???

The state is the current configuration in which the object is and is defined by the values of its properties (attributes).

- Initializing: This is the initial state of the system, where it is not yet ready to receive user input.
- Idle Time: This state would represent the initial state of the system, where it is waiting for
- Searching for ingredients: The system is searching for available ingredients based on the user input.
- Displaying available ingredients: The system has retrieved the available ingredients and is displaying them to the user.
- Error: This state would represent a state in which an error has occurred, and the system is unable to generate recipe recommendations.
- Selecting preferred cuisine: The user is selecting a preferred cuisine.
- Generating recipe recommendations: The system is generating recipe recommendations based on the available ingredients and preferred cuisine.
- Displaying recipe recommendations: The system has generated the recipe recommendations and is displaying them to the user.
- Selecting a recipe: The user is selecting a recipe from the recommended options.
- Displaying recipe instructions: The system is displaying the recipe instructions for the selected recipe.
- Cooking: The user is following the recipe instructions and cooking the dish.
- Finished: The user has finished cooking the dish and the system is ready to receive new input.

The behaviour of the object determines how it acts and reacts

Ingredient object:

- addQuantity(quantity) - adds the specified quantity of the ingredient to the system

- removeQuantity(quantity) - removes the specified quantity of the ingredient from the system

- getQuantity() - returns the current quantity of the ingredient in the system

- getName() - returns the name of the ingredient

RecipeGenerator object:

- generateRecommendations(criteria) - generates a list of recommended recipes based on the specified criteria

- getRecipeList() - returns the list of recommended recipes

- getRecipeDetails(recipe) - returns additional information about the specified recipe

Recipe object:

- modifyRecipe(modification) - modifies the recipe according to the specified changes

- suggestAlternativeIngredients() - suggests alternative ingredients that can be used in the recipe

- getNutritionalInformation() - returns the nutritional information of the recipe

- getCookingInstructions() - returns the cooking instructions for the recipe

- getIngredients() - returns the list of ingredients needed for the recipe

- getName() - returns the name of the recipe

User object:

- saveRecipe(recipe) - saves the specified recipe to the user's recipe collection

- shareRecipe(recipe, recipients) - shares the specified recipe with the specified recipients

- searchRecipes(criteria) - searches for recipes that match the specified criteria

- generateShoppingList(recipes) - generates a shopping list for the specified recipes

- authenticateUser(username, password) - authenticates the user's login credentials

- updatePreferences(preferences) - updates the user's dietary preferences and restrictions

- getRecipeHistory() - returns the user's cooking history

- getPersonalizedRecommendations() - generates personalized recipe recommendations for the user

Database object:

- retrieveData(dataType) - retrieves the specified data from the database

- updateData(dataType, data) - updates the specified data in the database

- authenticateUser(username, password) - authenticates the user's login credentials

Notification object:

- setReminder(date, message) - sets a reminder for the specified date with the specified message

- sendNotification(recipients, message) - sends the specified message as a notification to the specified recipients

An attribute is a feature of the class, and every attribute must be precisely defined

For the Ingredient class:

- name: the name of the ingredient

- category: the category or type of ingredient

- quantity: the current quantity of the ingredient in the system

- unit: the unit of measurement for the ingredient

For the Recipe class:

- name: the name of the recipe

- cuisine: the cuisine or type of food

- difficulty: the difficulty level of the recipe

- ingredients: a list of the required ingredients and their quantities for the recipe

- instructions: a list of steps to follow to prepare the recipe

- rating: the average rating of the recipe based on user ratings

For the User class:

- username: the user's username or login ID

- password: the user's password

- email: the user's email address

- preferences: the user's dietary preferences or restrictions

- favorite_recipes: a list of the user's favourite recipes

- shopping_list: a list of the ingredients the user needs to buy for their selected recipes

An operation is a transformation that can be applied to an instance of a class

Same as behaviours???

1. Ingredient (superclass)
- Vegetable (subclass)
- Fruit (subclass)
- Meat (subclass)
- Dairy (subclass)
- Grain (subclass)
- Spice (subclass)
- Seafood (subclass)

2. Recipe (superclass)
- Breakfast Recipe (subclass)
- Appetizer Recipe (subclass)
- Lunch Recipe (subclass)
- Dinner Recipe (subclass)
- Dessert Recipe (subclass)

- Vegetarian (subclass)

3. User (superclass)

- Registered User (subclass)
- Premium User (subclass)
- Guest User (subclass)