SOFTWARE ENGINEERING - CS301

REQUIREMENT ELICITATION REPORT

TEAM: **4 Epicure**

TEAM MEMBERS:

Devarshi Dubey: 21BCS031

Devesh Kumar: 21BCS032

Karthik Avinash: 21BCS052

Md Sameed Yallur: 21BCS068

Sem-4, Section-A

Objective :

- *To perform the Requirement elicitation process in detail using modeling approaches .*
- *Learn Use case, Class and Sequence diagram implementation through Experiential learning of Software engineering course project .*

❖ INTRODUCTION

➔ Project Scope

The 4epicure app aspires to become the ultimate destination for all epicures. Initially limited to India, it has the potential to be extended to other countries, catering to the needs of travelers, students, common people, diet conscious individuals, tourists, and others. The app will provide a centralized platform where users can access all information related to meals, including its segregation based on regions and an A-Z directory of the same.

The app has been designed to offer region-based segregation of meals and recipes, making it easier for users to discover local cuisine. Additionally, through the integration of Google Maps, users can find popular recipes across different states in India. The app will also provide nutritional information for each meal, including calorie counts, protein, carbohydrates, and fat content presented in a graphical format. This information will be particularly useful for health-conscious individuals.

The 4epicure app also features a voice assistant that will provide step-by-step guidance for recipe preparation. Users can search for specific meals or filter results based on dietary restrictions, allergies, or other preferences. They can rate and review meals and recipes and

---

provide comments on recipes created by others. Personalized recommendations will be provided based on the user's search history, ratings, dietary preferences, and available groceries.

The app supports multiple languages to cater to the diverse population of India. Additionally, it will send notifications if a particular meal is scheduled for preparation on a particular day, based on the grocery list updated in the app. This feature helps users keep their grocery items ready for the next day.

➔ **Technologies used:**

The requirement elicitation stage of our 4epicure app involved various techniques for modeling the use-case, class and sequence diagrams for our app. The techniques used included use case analysis, requirements prioritization, functional decomposition, questionnaires and surveys. Use case analysis helped us identify the functional requirements of the system, as well as the processes and steps required to fulfill each use case. Requirements prioritization was employed to identify the most important requirements and ensure that they were addressed first.

Functional decomposition was utilized to break down the system into smaller, more manageable components and to identify functional requirements and ensure that all necessary functionalities were captured. Questionnaires and surveys were used to gather information from stakeholders and end-users, providing valuable insights into the functional and non-functional requirements, as well as user interface requirements.

These techniques enabled us to form an in-depth understanding of the requirements for our 4epicure app and helped us model an efficient and user-friendly solution.

➔ **Functional and non-functional requirements:**

The functional requirements for this app are as follows:

1. Integration with Google Maps to display regional dishes in the user's locality.
2. Voice assistant to read out recipes for convenient cooking.
3. Image recognition feature to detect food items.
4. A recommendation system for personalized user experience

The non-functional requirements for your app are as follows:

1. User-friendly interface for students and travelers.
2. Quick and efficient performance.
3. High level of security for user data.
4. Compatibility with multiple devices and operating systems.
5. Responsiveness and scalability for future growth.

❖ UML DIAGRAMS

UML (Unified Modeling Language) diagrams and use case modeling are important tools in software engineering for representing and analyzing the requirements and design of a system. UML diagrams provide a graphical representation of the components and relationships within a system, allowing for a better understanding of the system's functionality and architecture.

Use case modeling, on the other hand, focuses on identifying and describing the interactions between the system and its users, as well as the sequences of events that take place. By using UML diagrams and use case modeling, software developers can ensure that the system meets the requirements of its users and stakeholders, resulting in a more efficient and effective development process.

➔ More about StarUML Tool used for UML diagrams in this project:

StarUML is an open-source tool for creating UML (Unified Modeling Language) diagrams. It supports various UML diagrams such as class diagrams, activity diagrams, sequence diagrams, use case diagrams and others. With StarUML, we can create and manage large-scale software projects with ease, visualize complex systems and relationships, and share your models with others.

Additionally, it has an intuitive user interface, customizable themes, and a wide range of UML modeling tools, making it a popular choice among software developers and architects. We can also do forward engineering with this tool and it is very helpful for this stage of software development.

Note: We have used an **unregistered version** of the tool.

❖ **Use case diagram:** It represents the interactions between actors and systems to achieve a specific goal.

Level 0 Use Case Diagram:

- Represents the highest level of abstraction of the system's functionality.
- Identifies the actors and the main functionalities of the system, without going into detail.
- The use cases are represented as ellipses, and the actors are represented as stick figures.

Level 1 Use Case Diagram:

- Refines the Level 0 Use Case Diagram by adding more detail to the functionalities of the system.
- It breaks down the main use cases into smaller, more detailed steps.
- Includes additional elements such as preconditions, postconditions, and exceptions.

Both Level 0 and Level 1 Use Case Diagrams are used to represent the functional requirements of a system and help to identify the interactions between actors and systems to achieve specific goals.

Use cases:

- Create Recipes
- View Recipes
- Search a Recipe
- Plan a Meal
- View Profile

Actors:

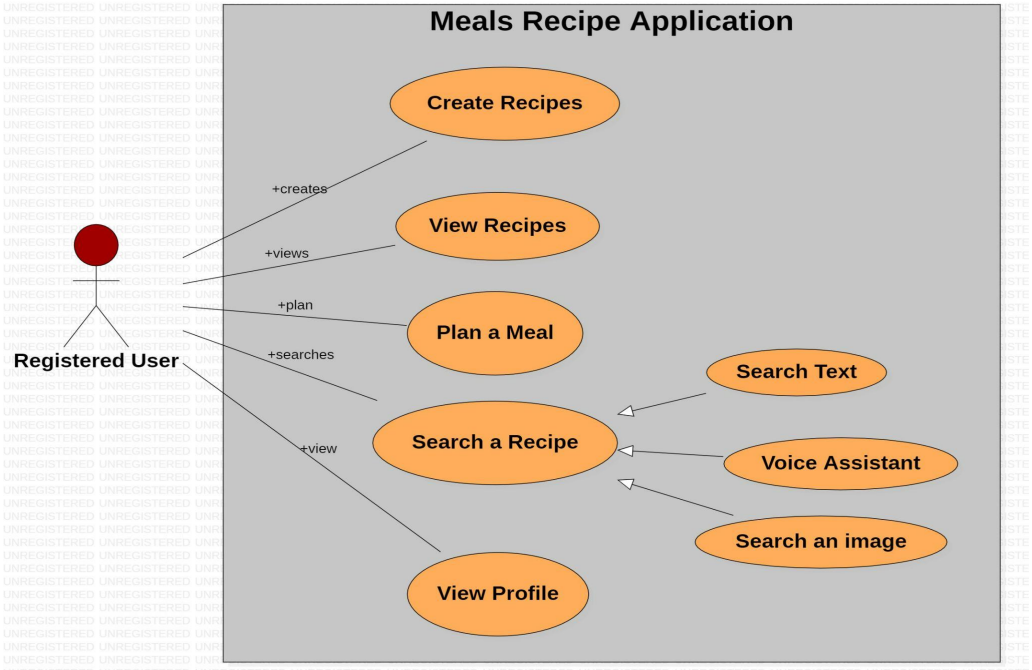
- Actors include the registered User using the application.

Approach:

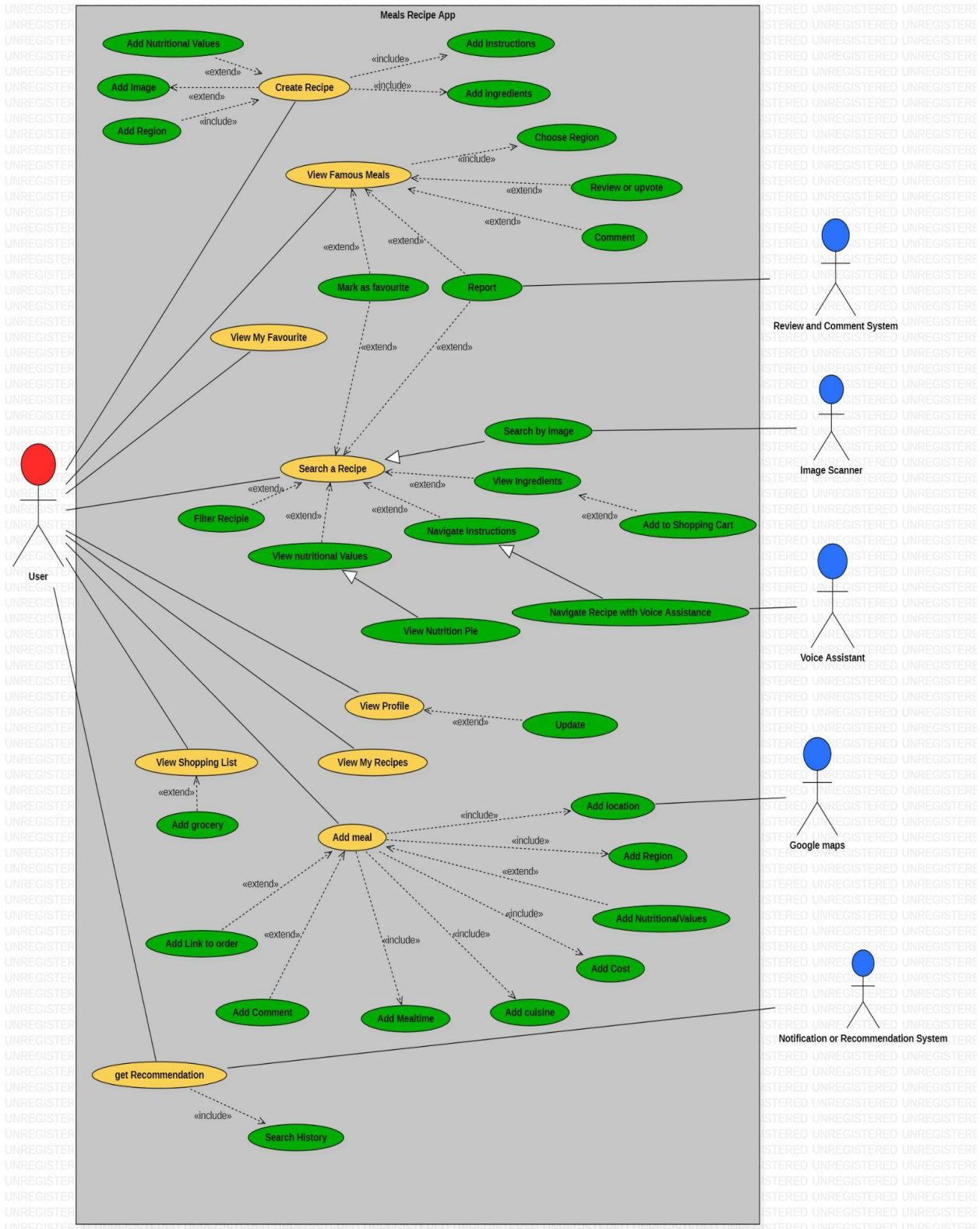
We have considered a **Procedure Driven approach** for identifying use cases. By following a procedure driven approach, use case diagrams ensured consistency, accuracy, and efficiency in the representation of functional requirements of our application. This helped us to minimize misunderstandings and misinterpretations, and ensured that all stakeholders have a clear understanding of the system's intended behavior.

Diagrams:

Level 0 Use Case Diagram:



Level 1 Use Case Diagram:



➔ **Preconditions and postconditions for common use cases:**

1. **Search for meal recipes by ingredient:**
    - **Pre-condition:** The user has entered one or more ingredients to search for and has a valid internet connection.
    - **Post-condition:** The system displays a list of meal recipes that match the ingredients entered by the user, or an error message if the search is unsuccessful due to a lack of internet connection or a server error.
2. **Save meal recipe to favorites:**
    - **Pre-condition:** The user has found a meal recipe that they would like to save and has signed in to their account.
    - **Post-condition:** The meal recipe is saved to the user's list of favorites, or an error message if the save operation is unsuccessful due to an issue with the user's account or the server.
3. **Create recipes:**
    - **Pre-condition:** The user has selected one or more meal recipes to add to their shopping list and has access to the ingredients list required to make that meal.
    - **Post-condition:** The ingredients for the selected meal recipes are added to the user's shopping list, or an error message if the add operation is unsuccessful due to a lack of access to the shopping list feature or a server error.
4. **Edit meal recipe:**
    - **Pre-condition:** The user has selected a meal recipe to edit and has the appropriate permissions to make changes.
    - **Post-condition:** The user has made changes in the instructions or added photos to the meal recipe and saved the updated version, or an error message if the edit operation is unsuccessful due to a lack of permissions or a server error.

Link to view the images:

Use Case Diagram Level 0.jpg

Use Case Diagram Level 1.jpg

❖ **Class diagram**

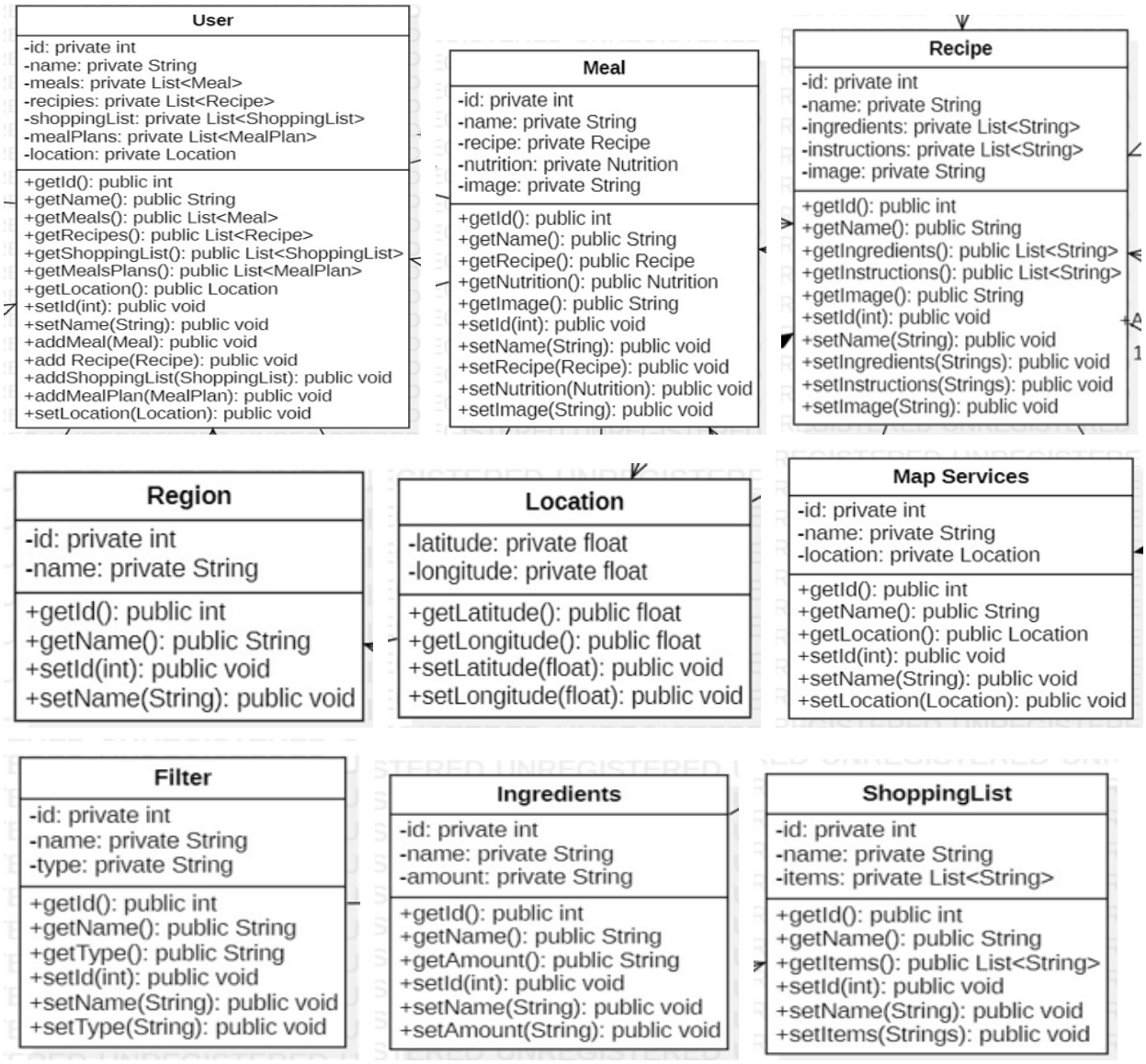
In the requirement elicitation stage of our 4epicure meals and recipes app project, a high-fidelity class diagram will play a crucial role in effectively communicating the design to stakeholders and ensuring that the project meets the requirements of our users. The class diagram will model the system's structure by representing its classes, their attributes, and the relationships between them, providing a visual representation of the innovative new features of our 4epicure app.

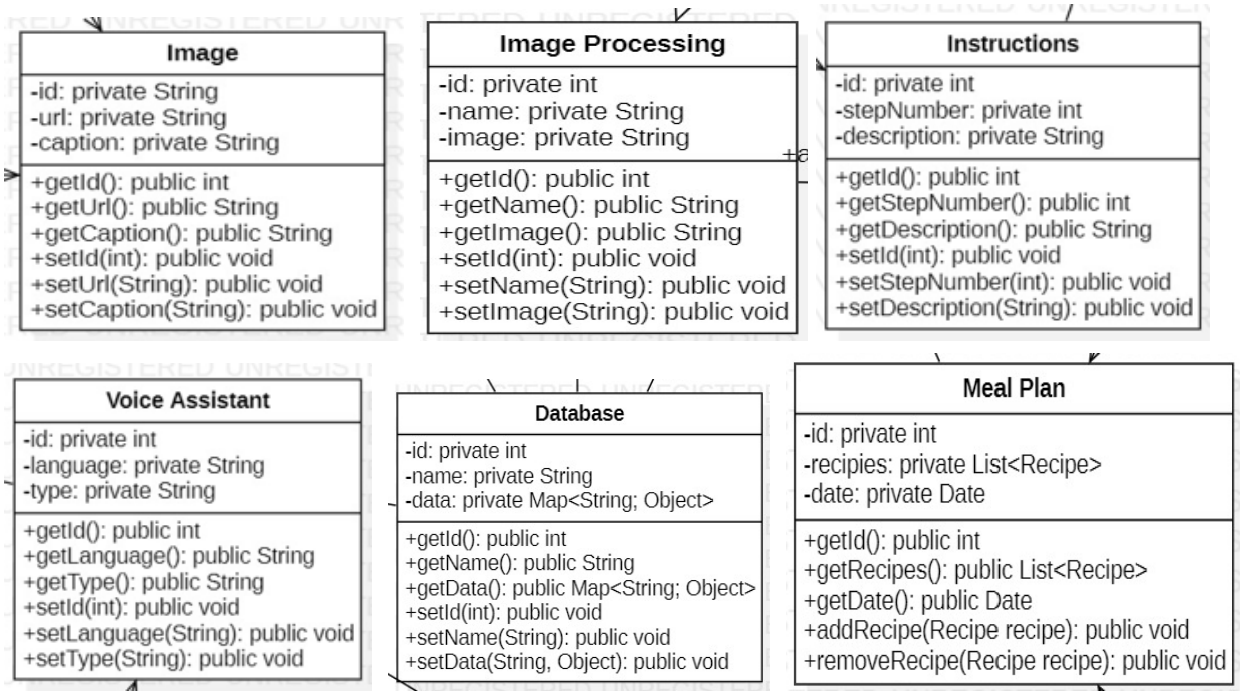
Although a sequence diagram will also be used to represent the dynamic behavior of the system, the emphasis is on the class diagram as it provides a clear understanding of the system's structure and helps to identify potential design issues early in the development process. The use of a high-fidelity class diagram will greatly enhance the overall quality and success of our 4epicure meals and recipes app project.

➔ **Classes, attributes and methods identified from level 1 of use case diagram:**

For the below classes, we have the ***private*** attributes/methods prefixed with " **-** " symbol and the ***public*** attributes/methods prefixed with " + " symbol.

Responsibility Driven Design is used to build this model. We identified all the responsibilities of the software in a nutshell and verified if they are covered by a collection of objects of the classes of the system.





➔ **List of classes:**

- User
- Meal
- Recipe
- Region
- Location
- Map Services
- Filter
- Ingredients
- Shopping List
- Image
- Image Processing
- Instructions
- Voice Assistant
- Database
- Meal Plan

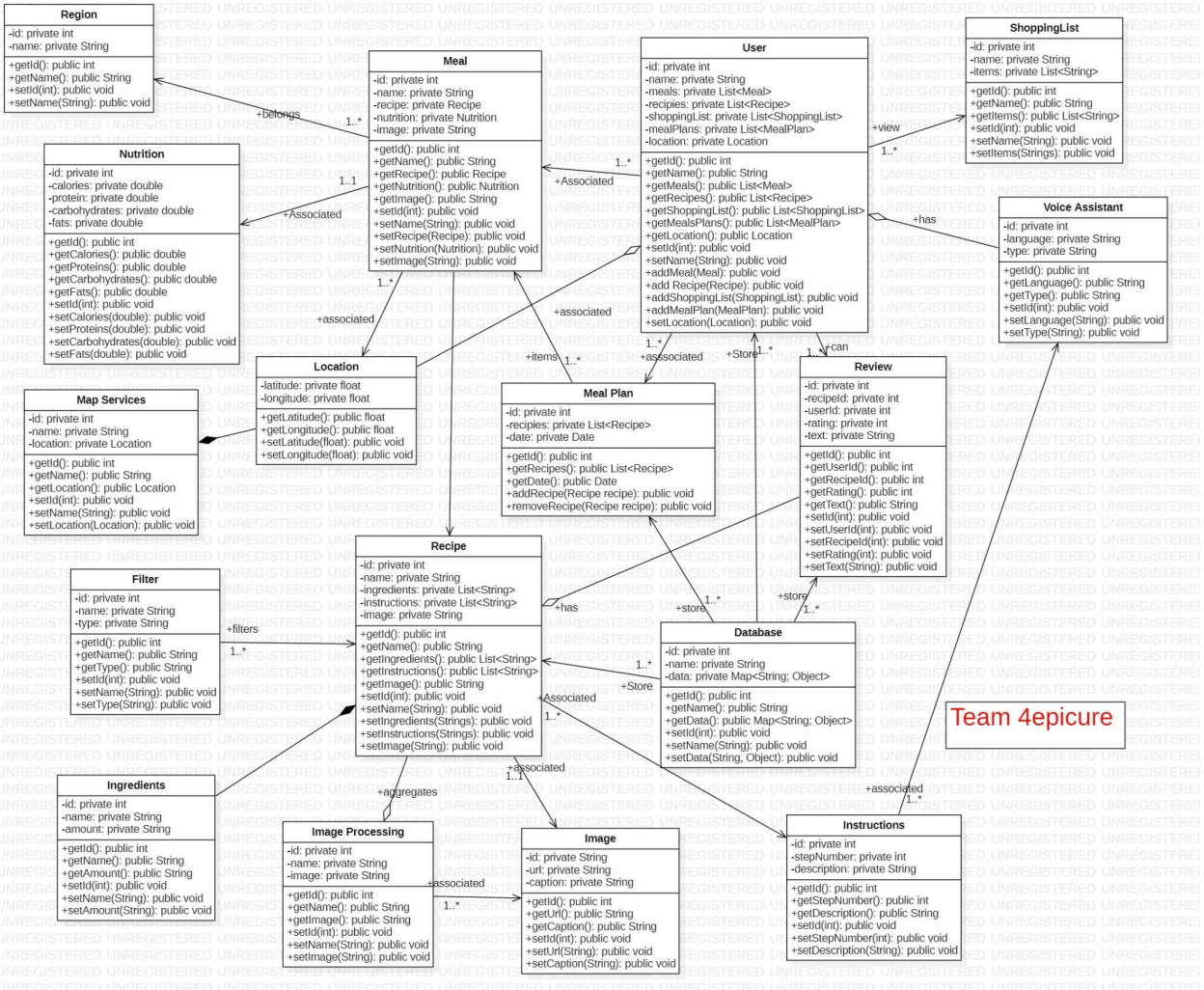
➔ **Relationships between the above classes** (only few classes are mentioned below to show the different types of relations):

- User class:
  - Has an **association** with Meal class as a User object will have many Meal objects associated with it.
  - Has an **association** with Review class as a User object will have many Review objects associated with it.
- ImageProcessing class:
  - Has an **aggregation** with Recipe class as a Recipe object can have an ImageProcessing object for processing its image.
- MapServices class:
  - Has an **aggregation** with Location class as MapServices will use Location object to provide mapping services.

➔ **Types of relationships used in this class diagram:**

1. **Directed Association**: represented with a solid line with an arrow as shown below:
2. **Composition** relation is shown by a solid diamond at end of the line as below:
3. **Aggregation** relation is shown by an empty diamond at end of the line as below:

➔ **Class Diagram:**



Link to view the image:

4epicure_ClassDiagram.jpg

❖ Sequence diagram:

This sequence diagram showcases the flow of interaction between various objects. The diagram covers the user authentication and registration. It proceeds to illustrate the control flow from app dashboard to the search feature. The search can be made through a simple text search or by using an image recognition algorithm. The search results are then selected to view the meal in the map or give a personalized opinion on the dish.

The flow diagram does not cover the full functions of the voice assistant and the shopping cart feature. We have however identified the basic flow of information and we will add further details during the design phase.

Link to view the image:

Sequence Diagram.jpg

The image couldn't fit in the given page width of the report. The link to the google drive containing the image is provided above.

❖ CONCLUSION

In conclusion, the requirement elicitation for the "4epicure" app project has been successful in identifying the key functional and non-functional requirements for the app. The app will display a variety of native regional dishes from nearby localities, providing users with an immersive food experience. The project aims to bridge the gap between local cuisine and modern technology by showcasing the rich cultural heritage of regional dishes. The requirement elicitation phase has helped in defining the scope, user needs, and business objectives of the project. The results of this phase will serve as the foundation for the design and development of the app. In conclusion, the "4epicure" app has the potential to revolutionize the way people discover and experience regional cuisine.

❖ **Note:**

1. **Downloading the tool: (useful to view the UML diagrams of our project uploaded in the github repository)**

   StarUML can be downloaded using the following steps:

   - Go to the official website: https://staruml.io/
   - Click on the "Download" button in the top right corner.
   - Select the version you want to download (Windows, MacOS, or Linux).
   - Once the download is complete, run the installer and follow the on-screen instructions to complete the installation process.

   Note: Before downloading and installing StarUML, make sure your system meets the minimum requirements, which can be found on the official website.

2. **To open the " 4epicure_app.mdj " file in our repository, which contains all the UML diagrams:**

   Download the StarUML tool using the steps provided above. Then download the **"4epicure_UML_diagrams.mdj"** file and open it using the tool.