| | |
|---|---|
| **Name** | Shivalik Pandita and Pratik Pujari |
| **UID** | 2020300049 and 2020300054 |
| **Project Name** | CrossFit |
| **Experiment No.** | 02 |
| **Aim** | To create a use case diagram, use case description, and a class diagram for our Flutter application, CrossFit |
| **Problem Statement:** | CrossFit is an application that is designed to help people in reaching their fitness goals. This app provides features like calorie management to track your day-to-day calorie goals, and customized meal plans according to your fitness and calorie intake. Along with that the app also provides a workout planner keeping in mind the fitness goals of the user. We have also tried third-party app integration. |

# Theory:

## What is a use case diagram?

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

## Purpose of Use Case Diagram

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

# Why Make Use Case Diagrams?

Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration.
They provide a good high-level analysis from outside the system. Use case diagrams to specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

### System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

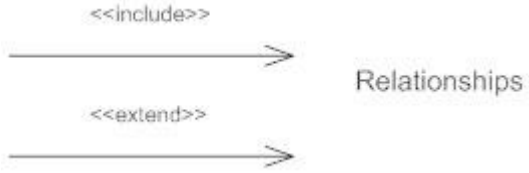

### Use Case

Draw use cases using ovals. Label the ovals with verbs that represent the system's functions.


### Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.


**Relationships**
Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.




## How use case diagrams work

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles played by the actors within and around the system.
- The relationships between and among the actors and the use cases

## Use Case Description

The Use Case Description defines the information of a particular use case which helps one understand the purpose behind that use case. Also, the use case description is written for every use case and it has one or more flow of events which can be classified as normal flow, alternate flow, and exceptional flow. Thus, the use case description helps in understanding the use cases much better.

A typical use case description may include the following:

- *Pre-conditions:* the conditions that must hold for the use case to begin.
- *Post-conditions:* the conditions that must hold once the use case has been completed. ● *Normal Flow:* the most frequent scenario or scenarios of the use case.

Alternate and/or exception flows—the scenarios that are less frequent or other than nominal. The exception flows may reference extension points and generally represent flows that are not directly in support of the goals of the primary flow.

Other information may augment the basic use case description to further elaborate the interaction between the actors and the subject.

The given below is the sample use case description:

| Use Case Name | | [Name of the use case] |
|---|---|---|
| Actors | | [An actor is a person or other entity external to the system being specified who interacts with the system and performs use cases to accomplish tasks] |
| Preconditions | | [Activities that must take place, or any conditions that must be true, before the use case can be started] |
| Normal Flow | Description | [User actions and system responses that will take place during execution of the use case under normal, expected conditions.] |
| | Postconditions | [State of the system at the conclusion of the use case execution with a normal flow (nominal)] |
| Alternative flows and exceptions | | [Major alternative flows or exceptions that may occur in the flow of event] |
| Non functional requirements | | [All non-functional requirement: e.g., dependability (safety, reliability, etc.), performance, ergonomic] |

# What is a class diagram in UML?

The Unified Modeling Language (UML) can help you model systems in various ways. One of the more popular types in UML is the class diagram. Popular among software engineers to document software architecture, class diagrams are a type of structure diagram because they describe what must be present in the system being modeled. No matter your level of familiarity with UML or class diagrams, our UML software is designed to be simple and easy to use.

UML was set up as a standardized model to describe an object-oriented programming approach. Since classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

## Benefits of class diagrams

Class diagrams offer a number of benefits for any organization. Use UML class diagrams to

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation-independent description of types used in a system that are later passed between its components.

## Components of Class Diagrams

- **Upper section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.

- **Middle section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

## Relationships in Class Diagrams

**Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class.
For example, The Current Account, Saving Account, and Credit Account are the generalized form of Bank accounts.

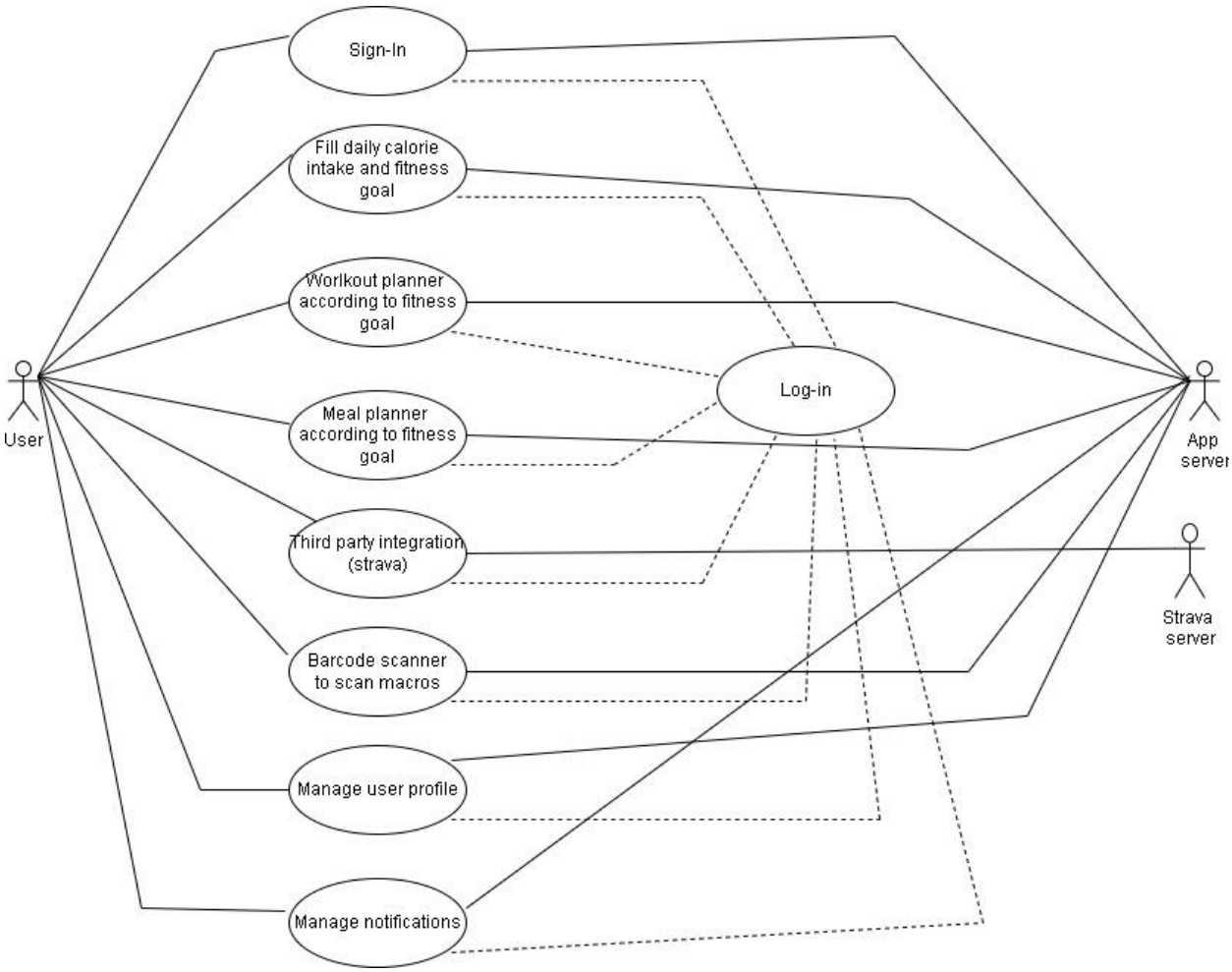
**Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.
For example, a department is associated with the college.

**Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship. In the following example, Student_Name is dependent on the Student_Id.

**An aggregation** is a subset of association, which represents a relationship. It is more specific than an association. It defines a part-whole or part-of relationship. In this kind of relationship, the child class can exist independently of its parent class. The company encompasses a number of employees, and even if one employee resigns, the company still exists.

**Composition:** It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship. A contact book consists of multiple contacts, and if you delete the contact book, all the contacts will be lost.

# Use Case Diagram: CrossFit



## Use Case Description: CrossFit

### Feature 1: Calorie Tracker

| Calorie Tracker Use case Description | |
|---|---|
| Brief Description | Using the calorie tracker user can set a calorie goal and track his progress over the day. |
| Actors | User |
| Precondition | The actor must be logged into his account with stable connection. |
| Basic flow of Events | **Normal flow:**<br>1. User logs in to the account.<br>2. The user chooses either custom input for calorie goal or recommended calorie intake.<br>3. The user adds the meals he has consumed.<br>4. Accordingly the calorie count will be maintained and calorie progress will be displayed. |
| Extensions: Alternate flows and Exceptions | **Alternate flow:**<br>No alternate flow<br><br>**Exception:**<br>a.) The calories of meal consumed by user may differ as feeded in system and in actual depending on factors like oil used in cooking, spices used.<br>b.) User inputs a custom calorie goal that is unrealistic hence the system shows a popping screen and does not proceed further until realistic goal is set. |
| Post Conditions | User can keep track of his calories and see if he has some pending meals or not. |
| Requirements | The actor must have a stable internet connection and must be logged in to his account. |

**Feature 2: Meal planner**

| Meal Planner Use case Description (Add, View, Edit, Delete Custom Meals) | |
|---|---|
| Brief Description | User's current data, and from the given calorie intake of per day, Spoonacular API will generate meal plans for days/week |
| Actors | User |
| Precondition | The actor must be logged in and must have access to internet |
| Basic flow of Events | **Normal flow:**<br>1. User logs in to the account.<br>2. The backend system will authenticate the user after he/she enters the email and password correctly<br>3. The user is navigated to the home page after the verification.<br>4. User navigates to the My meals section and chooses the daily calorie intake and which kind of meal planner is to be generated.<br>5. The user clicks on the generate button and waits for the output.<br>6. Application generates a list of meal plans suitable for the user's need |
| Extensions: Alternate flows and Exceptions | **Alternate flow:**<br>a. User generates a meal plan<br>a.1 The applications show "No meal plans found" due to lack of information<br>b. User clicks the meal plan as default<br>b.1 The application shows "Meal planner changed to default" and redirects the user to the meal planner page.<br><br>**Exception:**<br>a. User doesn't input regarding meal planning logic<br>3.a.1 The application will prompt the user to enter the required details to proceed |

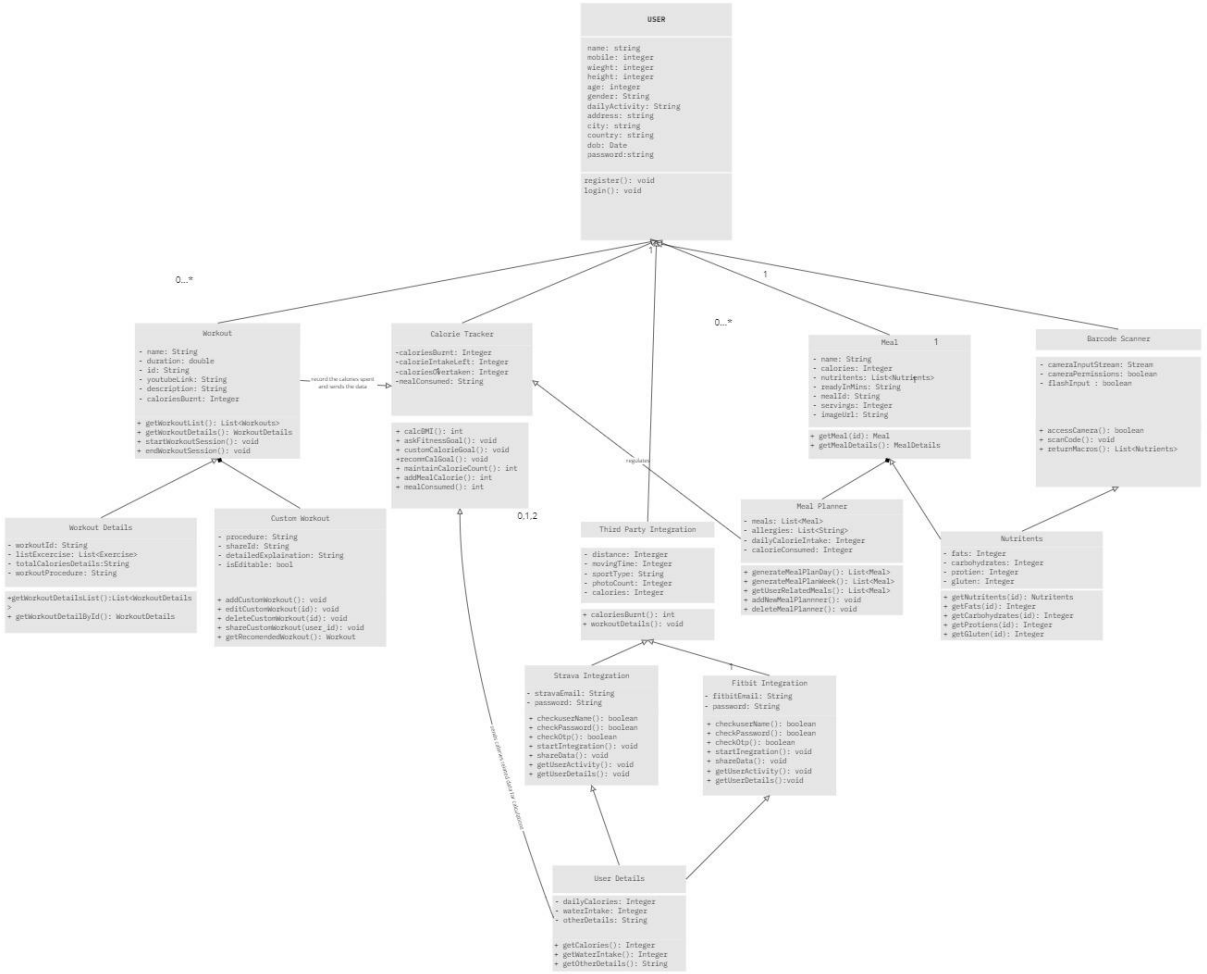
| | |
|---|---|
| | b. The user adds a meal planner with a name that already exists<br>b.1 The application shows the user that a meal plan already exists with that name<br>c. The user tries to delete the meal plan, but the server doesn't delete it correctly<br>c.1 The application will show an internal server error message |
| Post Conditions | List of newly generated meal plans will be displayed |
| Requirements | The actor must have a good internet connection, a valid registered CrossFit email account, and required permissions need to be granted. |

**Feature 3: Workout planner**

| Using Workout Planner Use case Description | |
|---|---|
| Brief Description | Using the workout planner, user can plan out his workout schedule i accordance to his fitness goals. |
| Actors | User |
| Precondition | The actor must be logged into his account with stable connection. |
| Basic flow of Events | **Normal flow:**<br>1.User logs into his account.<br>2.User goes to workout planner and starts the playlist.<br>2a.User can follow the recommended planner or go on to create a custom one.<br>2b.User can further add, view and delete his created sessions.<br>3.The time begins to elapse. |

| | |
|---|---|
| | 4.As user completes an exercise of the routine, he checks it off and proceeds on to next one.<br>5.After completion of the routine a message is popped up displaying user's success. |
| Extensions: Alternate flows and Exceptions | **Alternate flow:**<br>a. User searches a workout<br>a.1 The applications show "No workout sessions found" due to lack of data or incorrect workout name<br>b. User tries to end a workout session after waking the application(if the application is in sleep mode)<br>b.1 The applications will show "Workout already ended" and navigate to the main home page.<br><br>**Exception:**<br>a. User tries to delete a custom workout session and can reach the server due to some issues<br>a.1 The application responds with an error message. |
| Post Conditions | Users can keep track of their workout sessions. |
| Requirements | The actor must have a stable internet connection, logged in and access to the database. |

# Class Diagram: CrossFit



## Conclusion

We learned about use case diagrams, use case descriptions, and class diagrams, which provide one an idea of what the system would look like. The added use cases are not introduced in a sequential manner, and the use case diagram helps to visualize how the project will look. The system for which a use case diagram was created is the subject of the use case description. The class diagram aids in providing a static representation of the system. We were thus able to build the UML use case diagram, use case description, and class diagram in the provided project called "CrossFit" after understanding the three concepts.

## References


https://www.researchgate.net/publication/322991847_role_of_use_case_diagram_in_software_development

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

https://www.techtarget.com/whatis/definition/use-case-diagram

https://www.lucidchart.com/pages/uml-class-diagram