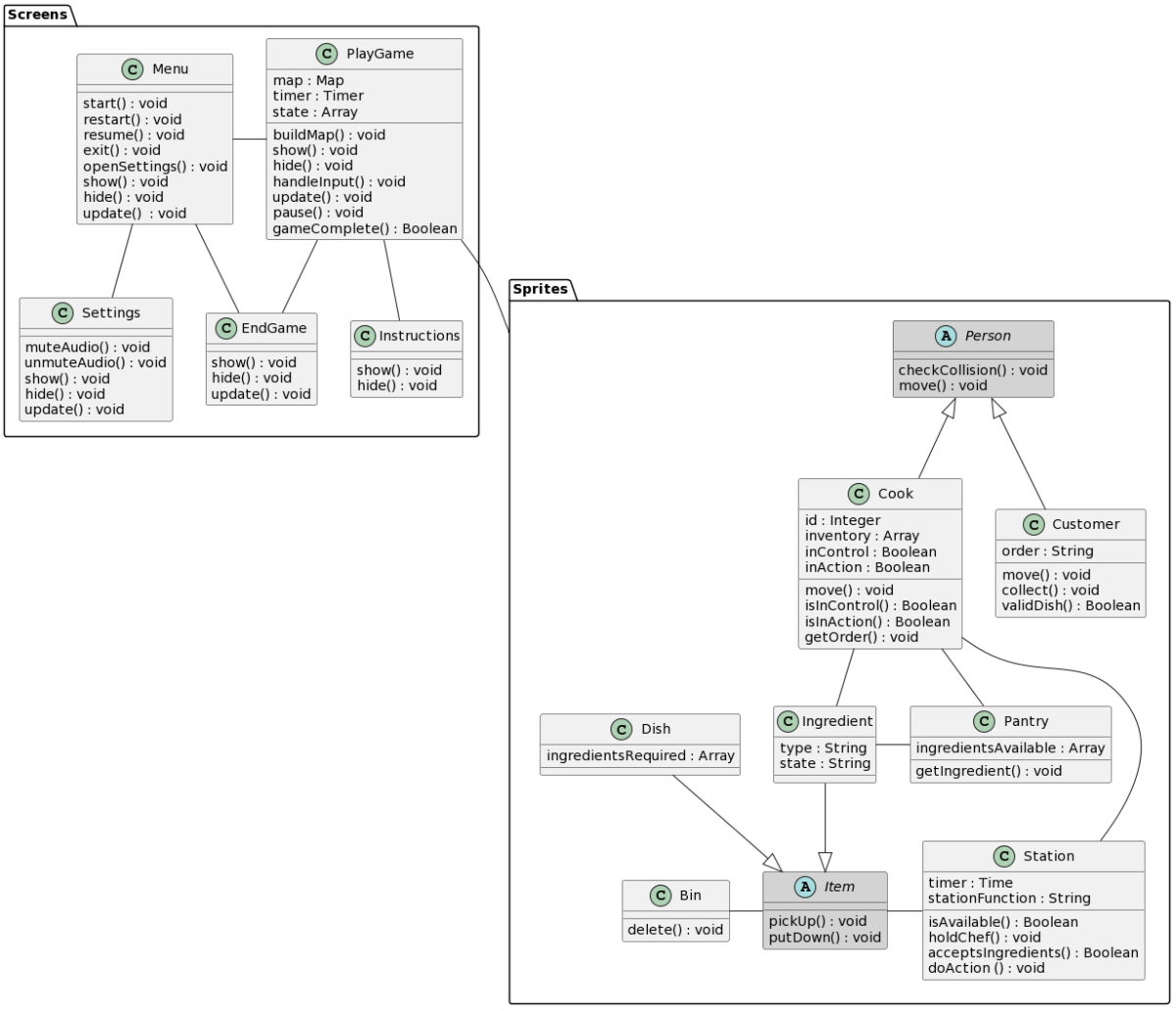
Architecture
Group 30 Triple 10
Team Members:
Kelvin Chen, Amy Cross, Amber Gange, Robin Graham, Riko Puusepp, Labib Zabaneh

Structural Diagrams

We created a class diagram, with UML Class Diagram using plantUML, to consider different components of the program and model the various classes and their relationships. We could decide from each class their main attributes and methods and whether any could be grouped into packages or relate to a more generalised abstract class. The Class diagram used for the game implementation is shown below.

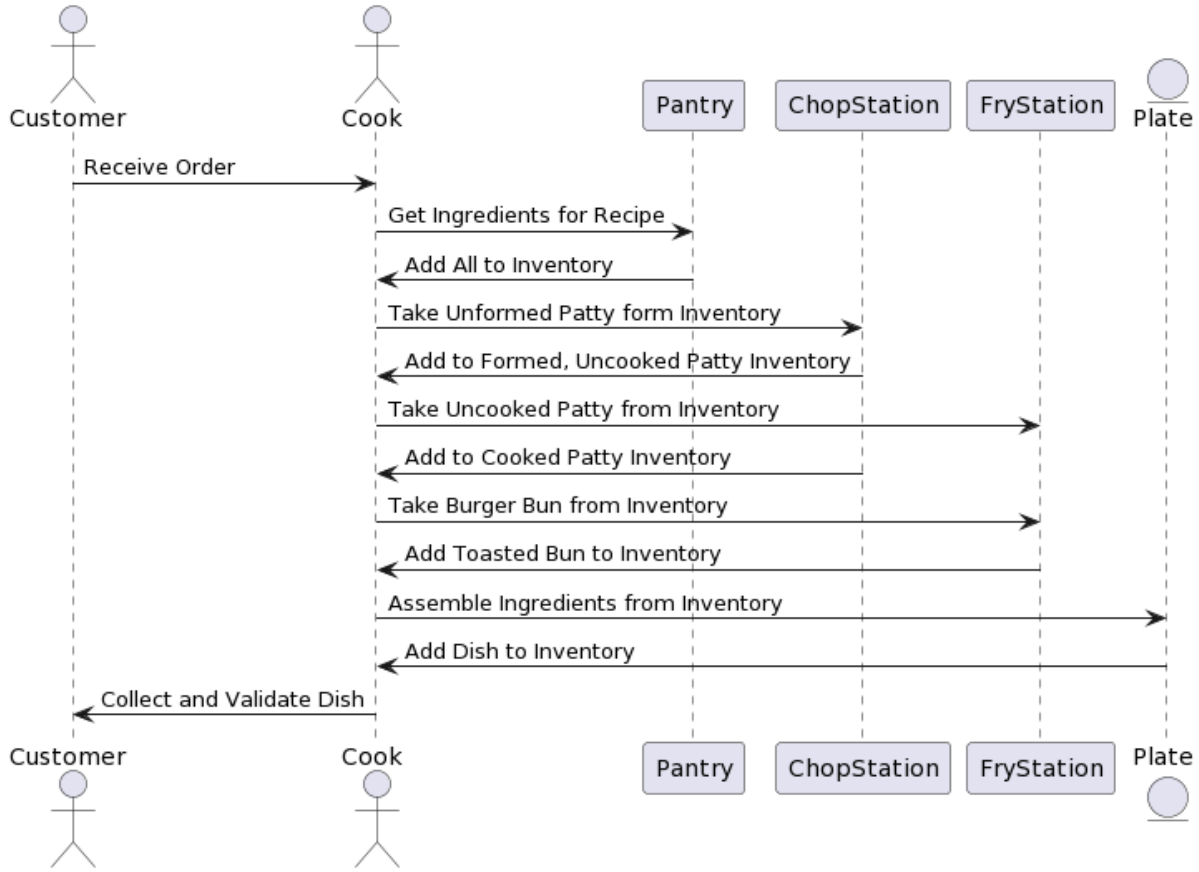


We decided to have two separate folders for the screens and sprites to separate the classes the user interacts with in the game and the more structural classes that form the background. Doing this keeps the diagram easier to follow when implementing and separating workload. Following implementation, we noticed that several of the classes we initially had designed in our class diagram had to expand to include more methods and attributes. Furthermore, we found some of the classes we first created needed to be more precise and so had to be separated into different but still connected classes. Some of the classes directly relate to user requirements, e.g. class menu and requirement FR_MENU and some requirements are represented through methods/ attributes in a class (UR_GAME_MAP for the map in playGame)

Behavioural Diagrams

Along with our class diagram, it was helpful to consider how the user would interact with this software. As such, we created a sequence diagram to show how objects would exchange messages during a given scenario and an activity diagram to map out the steps taken for the user to complete an action. It was created using UML Sequence Diagrams on PlantUML.

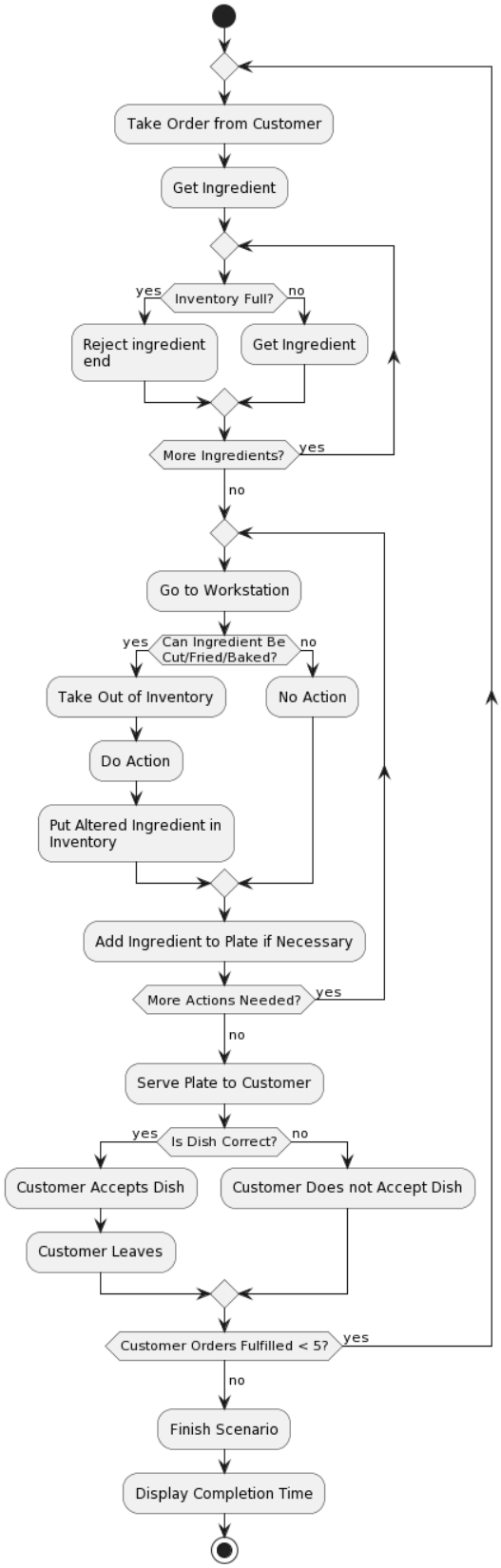
Sequence Diagram



Following this diagram, we could expand on each action to show what steps must be taken in the game. For example, when adding ingredients to inventory, the system must decide whether this is allowed depending on whether the inventory is full. For this, there is an activity diagram, as detailed on the next page.
In this diagram, we separated the stations to clarify which station each action is to take place in and described what each activity contains. The design hasn't changed for this diagram, as the product brief described the actions, so they cannot change during development. Many actions in the sequence diagram directly relate to our defined requirements. So the first action defined, where a chef receives the order from a

customer, directly relates to UR_CUSTOMER_ORDER. Specific requirements also relate to the areas illustrated in the sequence diagram, like Pantry matching with UR_PANTRY.

Activity Diagram



We also created an Activity Diagram using UML (plant UML) to show the steps a user takes in the game, from taking an order to displaying the completion. This diagram includes decisions.

The diagram shows a brief overview of the actions taken between the customer first making an order and when they receive and validate the dish and its steps based on the sequence diagram. We have used this diagram to build up more detail in each game process to ensure we follow the requirements we have outlined and that they are correctly implemented. In the sequence diagram, the user simply got the ingredients from the pantry and was put into their inventory, which has been expanded to include a check that the inventory is not full. Some tasks have been generalised. For example, all workstation tasks have been combined to reduce overlap and confusion.

Requirements and how they relate to the Activity Diagram:

1. Receive Order → FR_ORDER
2. Get Ingredients for Recipe → FR_PANTRY
3. Interactions with Workstations → FR_MULTI_ITEM, FR_INGREDIENTS, FR_STEP_VALIDATION, FR_COOK_BUSY
4. Collect and Validate Dish → FR_DISH_VALIDATION