# Design Rationale REQ1

## Design Rationale

### Implementation & Why:

- Created Destination Class:
    - Stores information regarding a destination to travel to through a gate
        - X, Y coordinates
        - GameMap to travel to
        - Name of the map
    - Has standard getters so that Gate can access Destiantion's field
    - Adheres to Single Responsibility Principle as Gate doesn't have to store all information about each GameMap and which x,y coordinate to store at.
- Modified Gate:
    - Stores a list of Destinations (destinationList)
    - Modified AllowableActions()
        - Iterates through each Destination in the destinationList and adds a TravelAction to that Destination based off its fields
- Modified Maps
    - Added Overgrown Sanctuary as a map into the Maps class

- Modified Void
    - Modified tick()
        - Checks if the actor in the void has Ability.VOID_IMMUNITY, if not only then the actor's unconscious method is called
- Modified Abxervyer:
    - Instead of taking two maps to create the gate, now the Abxervyer stores the gate it drops after being killed. Reduces dependency on maps and adheres to Single Responsibility Principle as Abxervyer doesn't have to initialise the gate.

- Created LivingBranch Class
    - Added the method addDropableItem() in the constructor so the enemy will have 50% chance to drop a Bloodberry
    - Modified AllowableActions()
        - Removed the wandering behaviour so the living branch can't wander around.
- Created EldentreeGuardian class
    - Added the method addDropableItem() in the constructor so the enemy will have 25% chance to drop Healing Vial and 15% chance to drop Refreshing Flask.
    - Modified AllowableActions()
        - Added following behaviour to the Eldentree Guardian enemy.

- Modified Application
    - When creating Gate instances in application, instances of destinations are created inside the Gate to work with the new functionality
        - Ex: abandonedGroundMap.at(22, 3).setGround(new Gate(new Destination(burialGroundMap, "Burial Ground",22, 6)));
    - Created gate for boss
    - Added Graveyards, Bushes, and EmptyHuts into Overgrown Sanctuary map

## Pros:

- Destination:
    - Including the name of the map improves user experience. Otherwise they wouldn't know where they are travelling
    - Allows player to be spawned at a specific x,y coordinate when travelling
- Gate:
    - Separating Destination and Gate allows us to add extra Destinations to a Gate as the game goes on, adding extra functionality to the game

- LivingBranch and EldentreeGuardian:
    - Our code shows the Single Responsibility Principle by dividing responsibilities into distinct classes like Maps, EmptyHuts, EldentreeGuardian, LivingBranch, etc.
- Maps:
    - Our code shows the Open-Closed Principle by creating a 'Maps' class and adding the 'OVERGROWN_SANCTUARY' map without modifying existing code, and new maps can be added without changing existing code.

## Cons:

- Destination:
    - Map name must be defined every time a new destination is made. Intuitively, the map name should be stored in the GameMap, however, as we can't edit engine files, adding the map name in Destination is our best resolution

## Extendability:

- Gate:
    - Can take any number of gates through the use of the destination arrayList
- Changes can be made to Destination without affecting Gate
- Destination can be used in other ground objects if needed
- Destination can be used to specify a specific location on other maps, allowing for efficient storage and access of GameMap and coordinates
    - Ex: used in Respawner to identify specific respawn point