# Team Mercury Milestone 0

## Functional Requirements:

### Player

1. Player should be able to turn tank token.
2. Player should be able to move tank token.
3. Player should be able to fire bullets fire bullets.
4. Player should be able to configure and save tank token (skin).
5. Player should be able to login to the server.
6. Player should be able able to shake the device to fire.
7. Player should be able to see their account balance and items in their garage.

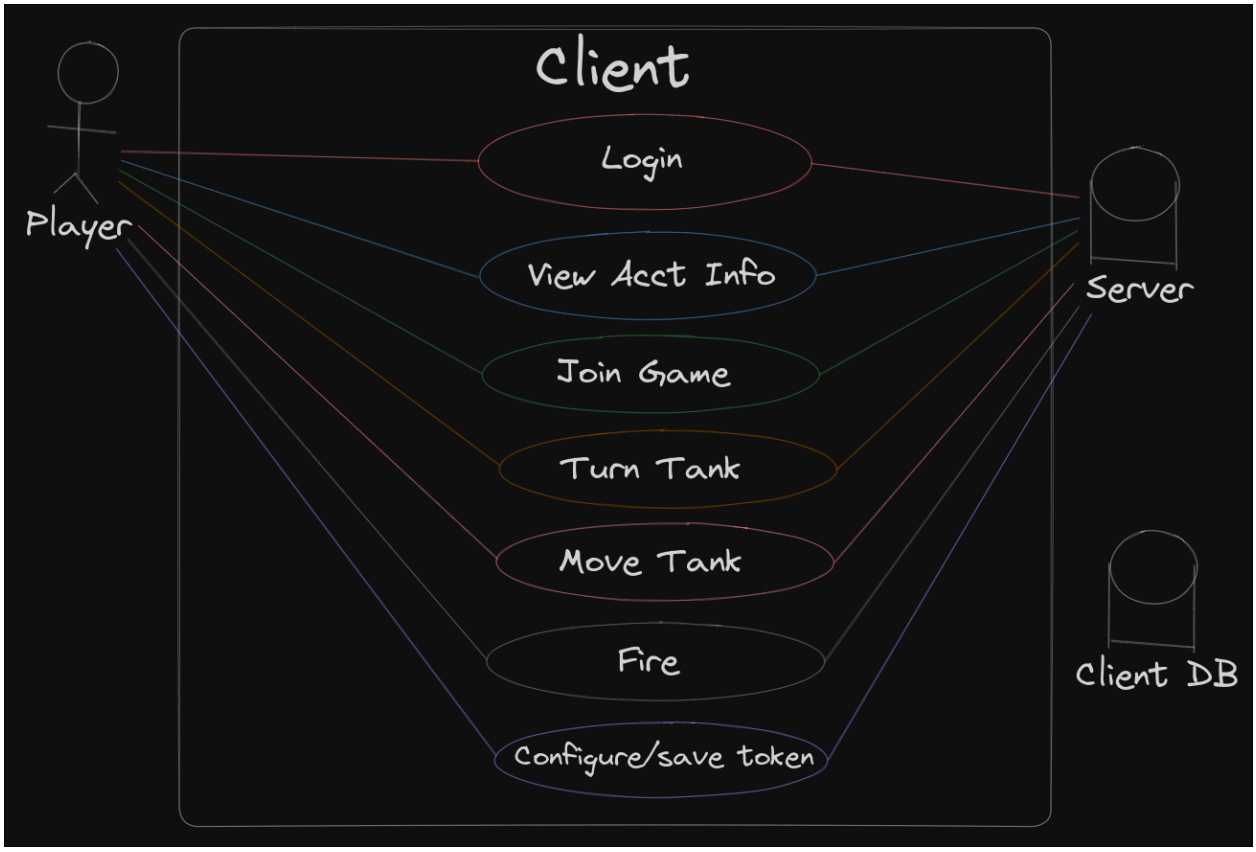
### client

1. Client should have entities (Tank, wall or bullet).
2. Client should keep track of account information.Join
3. Client should provide interface to user.
4. Client should show game state.
5. Client should support player login.
6. Client should store timestamped game state.
7. Client should be able to replay saved game state at configurable speeds.

### server

1. Server should manage battlefield.
2. Server should manage game time.
3. Server should enforce rules.
4. Server should manage player accounts.
5. Server should keep track of entity properties (improvements, terrain types, and item).
6. Server should allow tank to move once every half second.
7. Server should allow tank to fire once every half second.
8. Server should allow tank fire only two bullets at a time.
9. Server should allow tank to make one 90 degree turn per step.
10. Server should allow tank to move forwards and backwards.
11. Server keep track of the player's bank account.
12. Server should ensure bank can store tanks and parts that the player acquires.

Use-case Diagram:



# Main Success Scenario:

## View Bank & Garage

### Preconditions

1) user logged in

### Main Success Scenario

1) User navigates to Bank/Garage
2) Client requests user data from server
3) Server retrieves data from the server database
4) Server returns the user data to the client
5) Client displays the user's bank and garage information

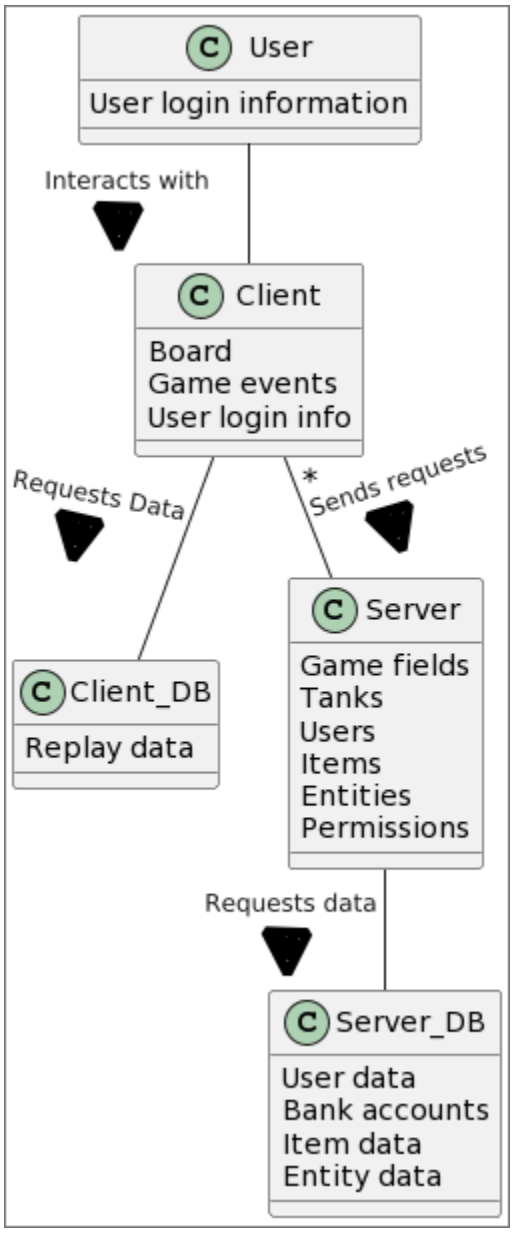
### Extensions

N/A

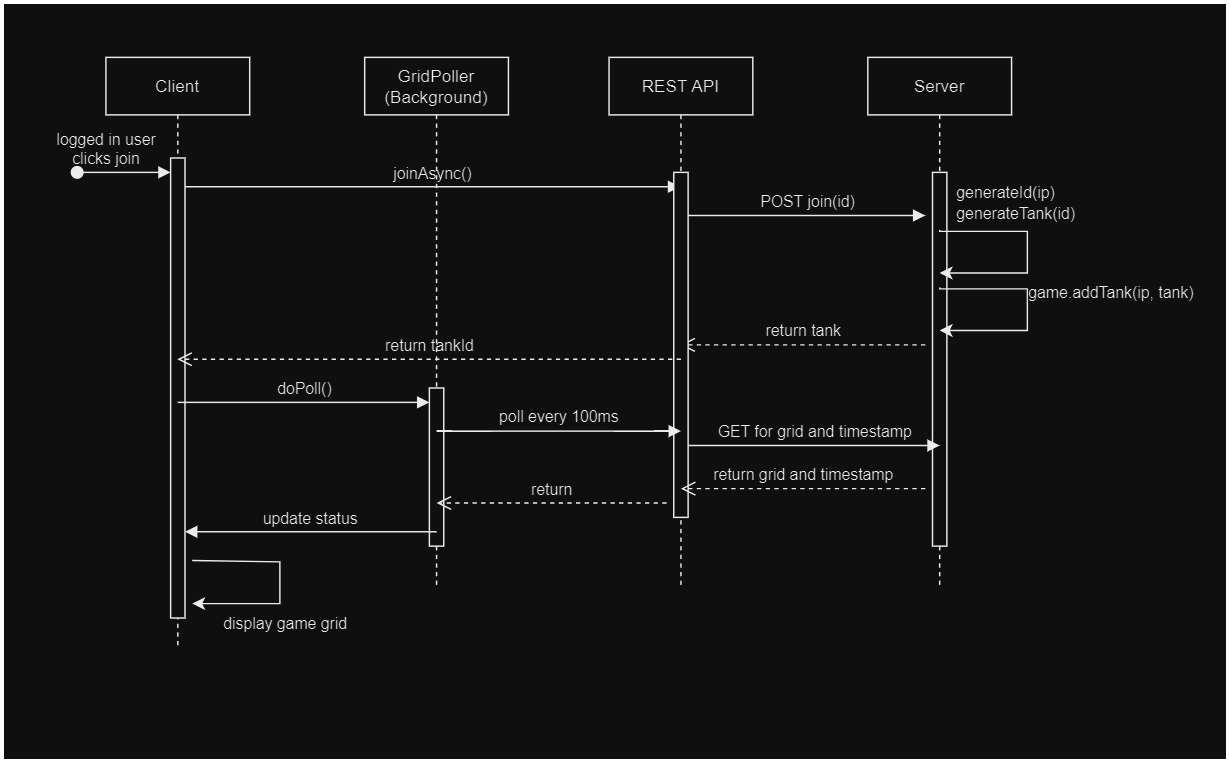
### Postconditions

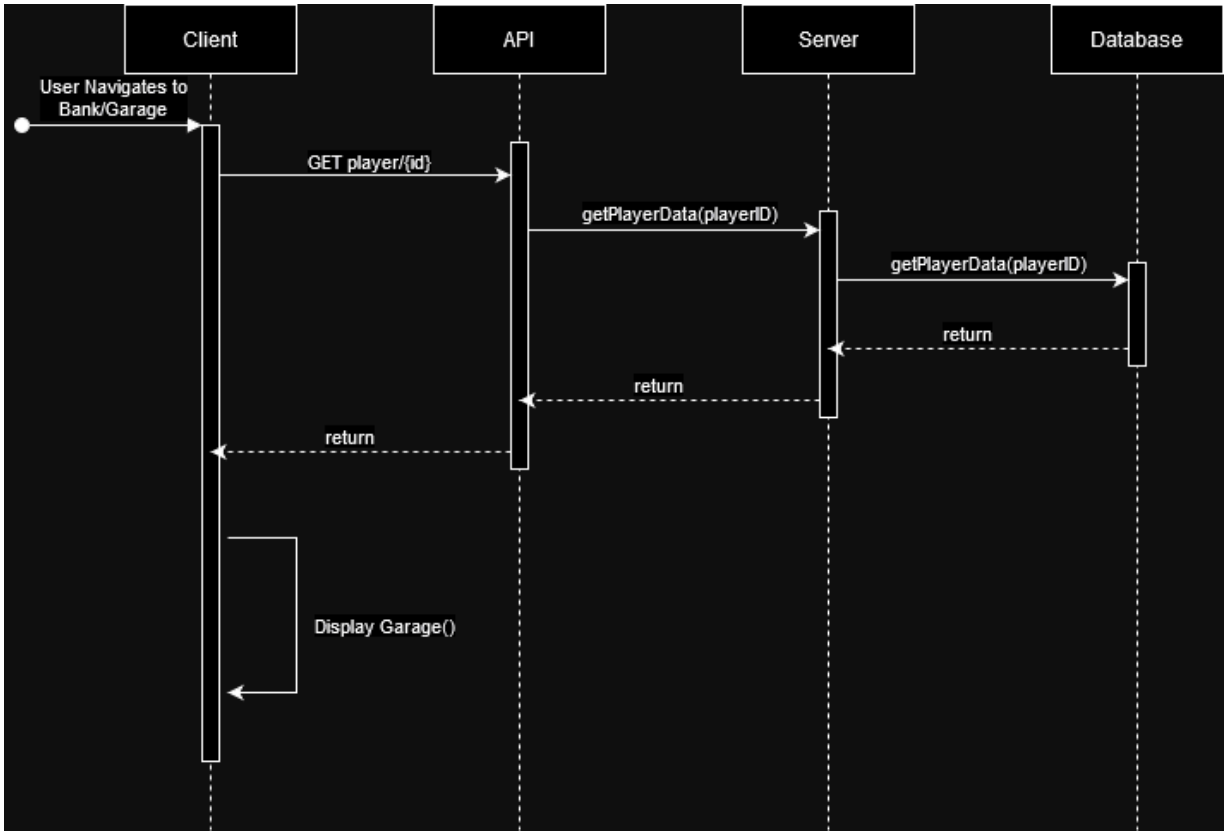
N/A

# Domain Model:

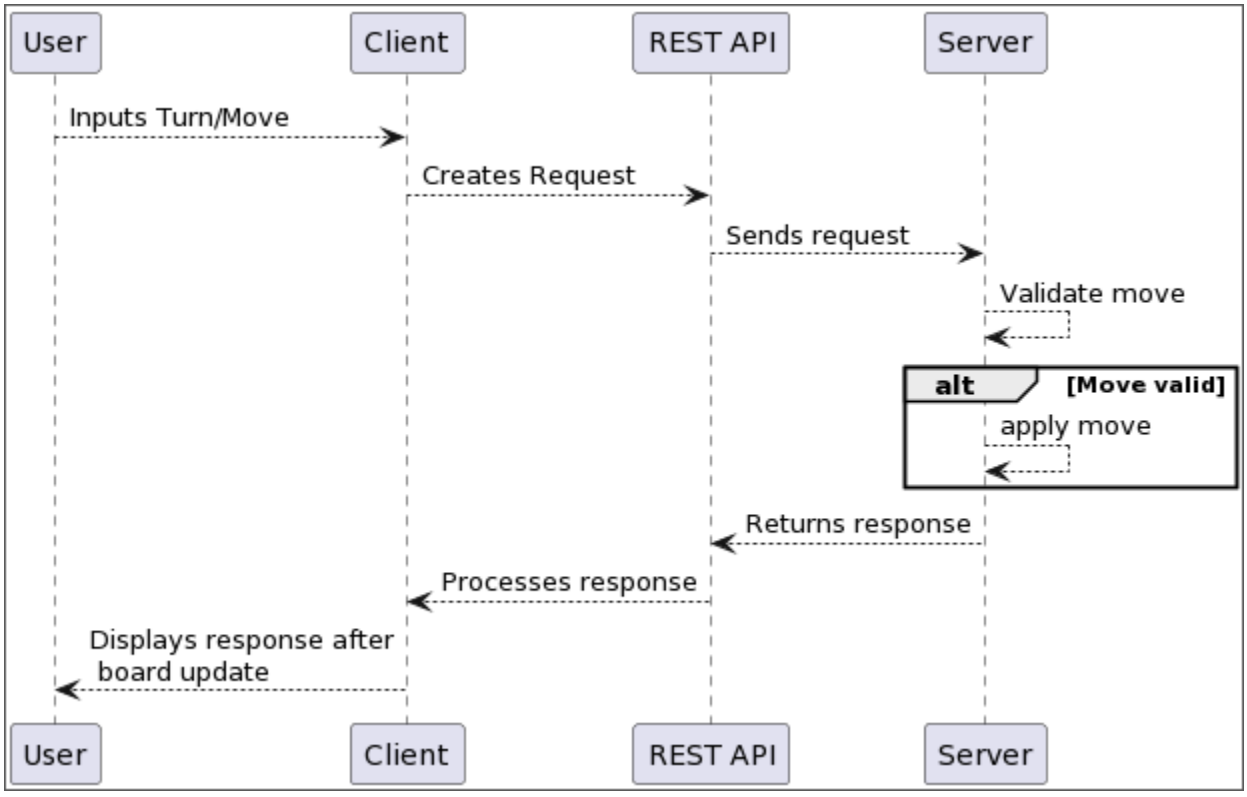


# UML Sequence Diagrams:

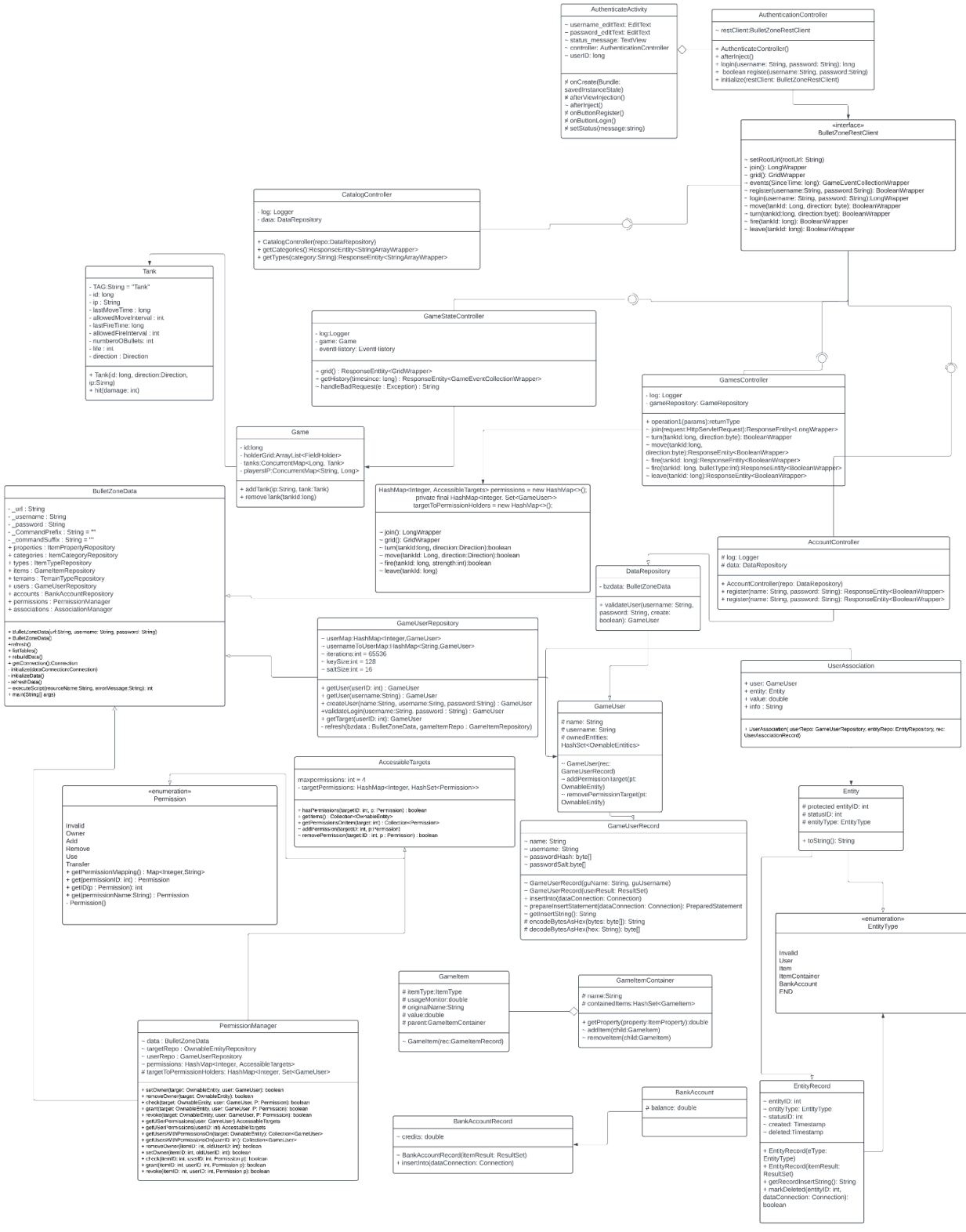


# Turning & Moving UML Sequence Diagram:



# UML Class Diagram:



# Identified Patterns:

The project will utilize multiple patterns in order to properly organize code into appropriate abstractions and make the development process easier. The poller will be a singleton, because there should only ever be one at a time. Any information that only needs to exist in one place but be accessed in multiple places can also be kept as a part of a singleton. This object can also subscribe to events to make sure that its information is updated appropriately. The Model View Controller (MVC) pattern is also used to better organize the different parts of the program. Any input that comes in from the user will be a part of the "controller," and will make changes to the "model." Anything that displays this information from the model back to the user is a part of the "view." The game board and its multiple terrain types, improvements, and items per cell can be handled using a builder pattern. An observer pattern is used to connect anything that posts to the event bus with its subscribers. If a client's interactions prove to be too complex, a facade might be necessary in order to simplify them.