Checkpoint 3

An Nguyen, Mark Araza

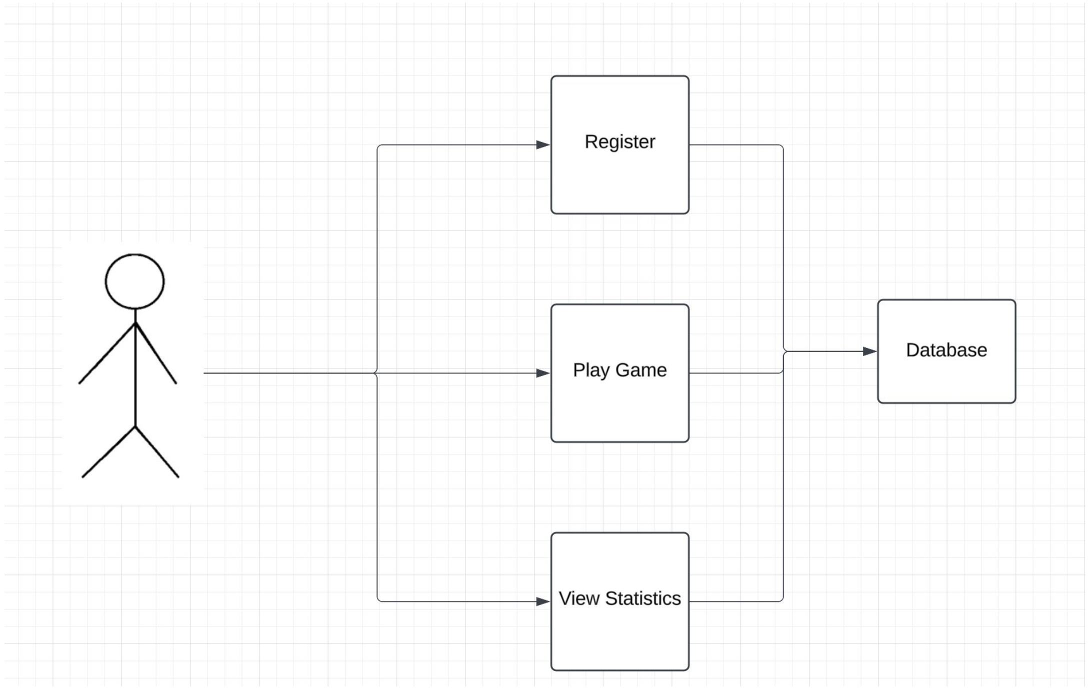
Synopsis

- A turn based auto battler where users put their teams into a pool and fight against stronger teams each turn, working towards achieving 10 wins to win the game, or 3 losses meaning that their run is out. Players can purchase their units from a randomly generated shop each turn using gold that they earn each turn.

Requirements

- Requires an sql database
- Minimally this will be a game played in the terminal through text based input
    - Python with sqlite3
- It would be best if this could be played online, but this depends on time constraints

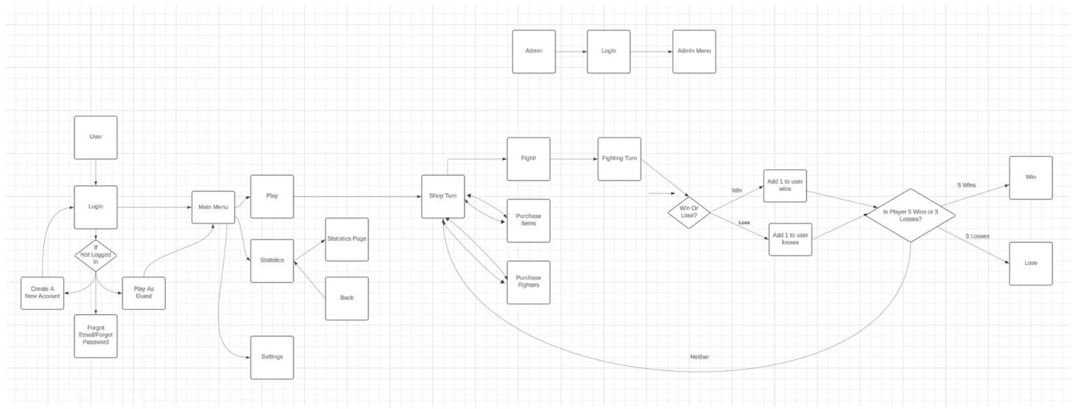
UML Use Case Diagram



Three main use cases

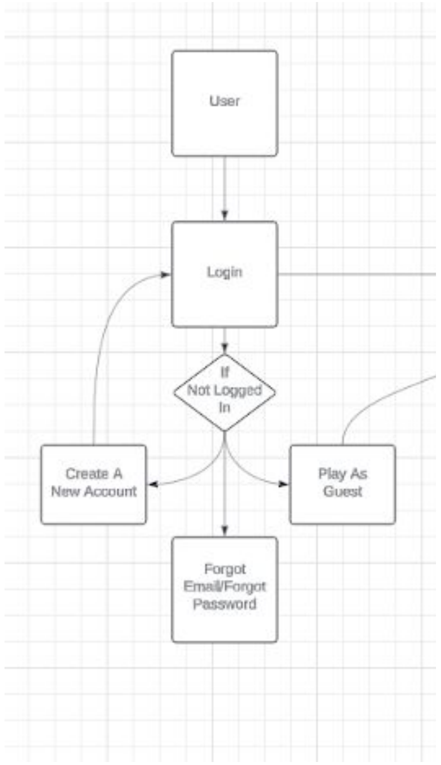
- Registration for game
- Playing game
- Viewing player statistics

UML Use Case Diagram



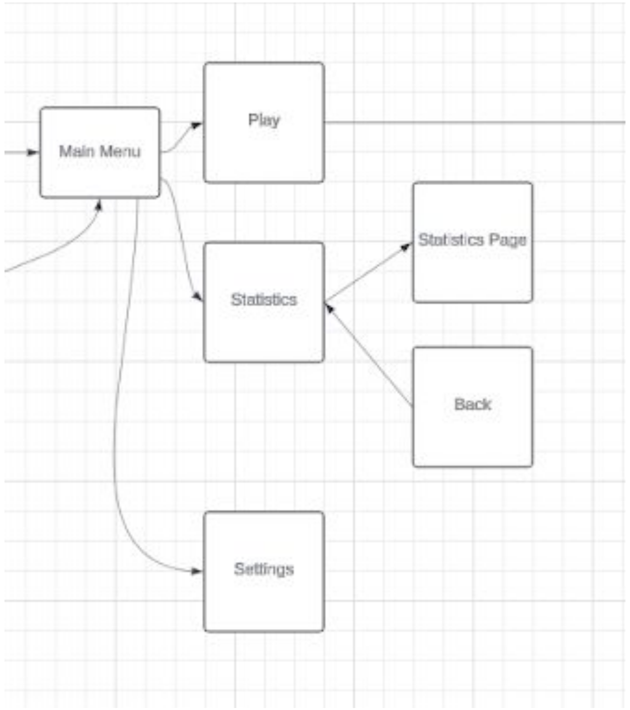
- In this use case, users login or create an account if they don't have one.
- Once playing the game, they can view a settings page, or a statistics page, or join the queue to play a game.
- From there the users buy units and unit modifiers, playing until they get 5 wins or 3 losses, at which they win or lose.

UML Use Case Diagram



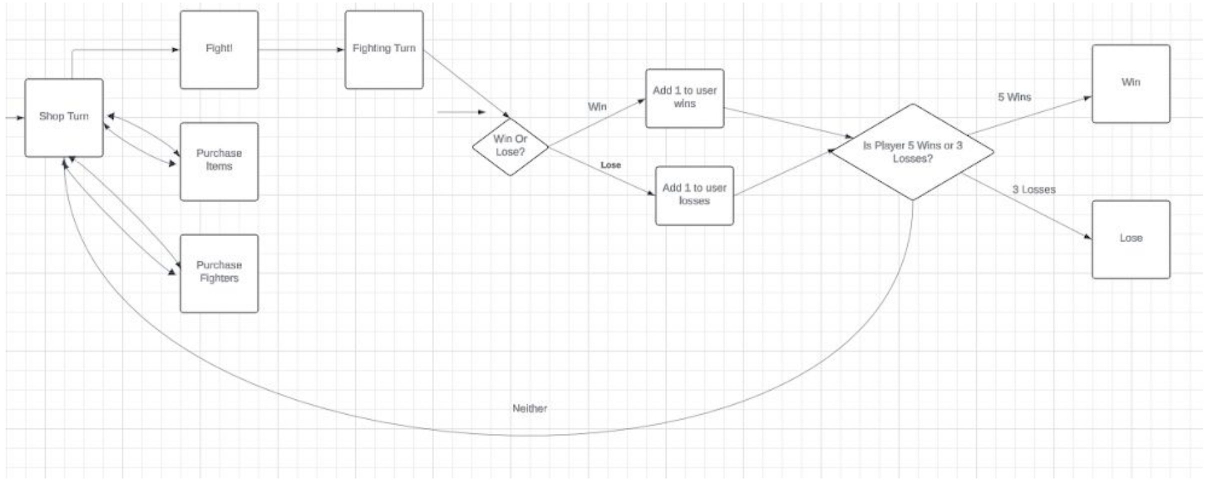
- In this use case, users login or create an account if they don't have one.
- Once playing the game, they can view a settings page, or a statistics page, or join the queue to play a game.
- From there the users buy units and unit modifiers, playing until they get 5 wins or 3 losses, at which they win or lose.

UML Use Case Diagram



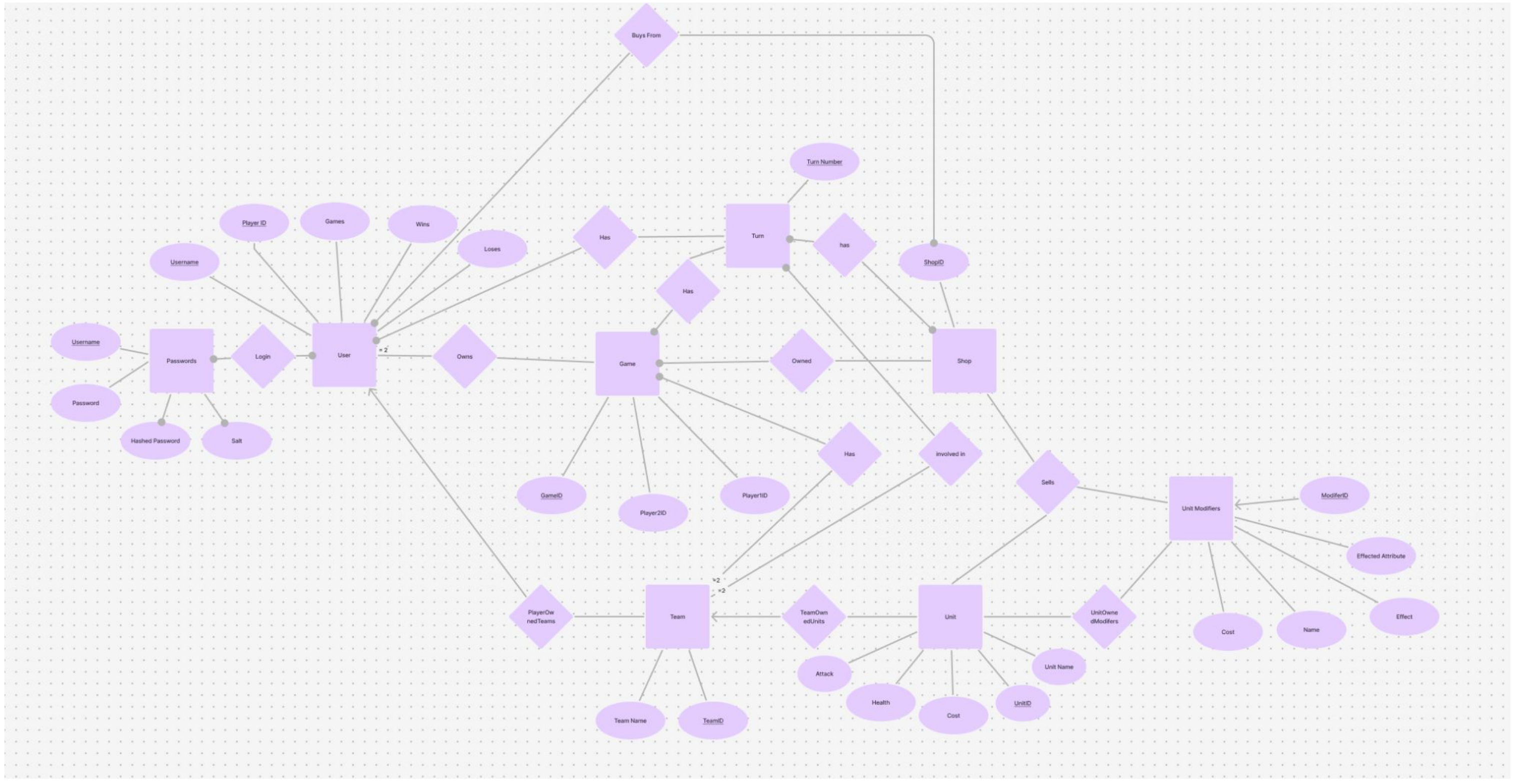
- In this use case, users login or create an account if they don't have one.
- Once playing the game, they can view a settings page, or a statistics page, or join the queue to play a game.
- From there the users buy units and unit modifiers, playing until they get 5 wins or 3 losses, at which they win or lose.

UML Use Case Diagram



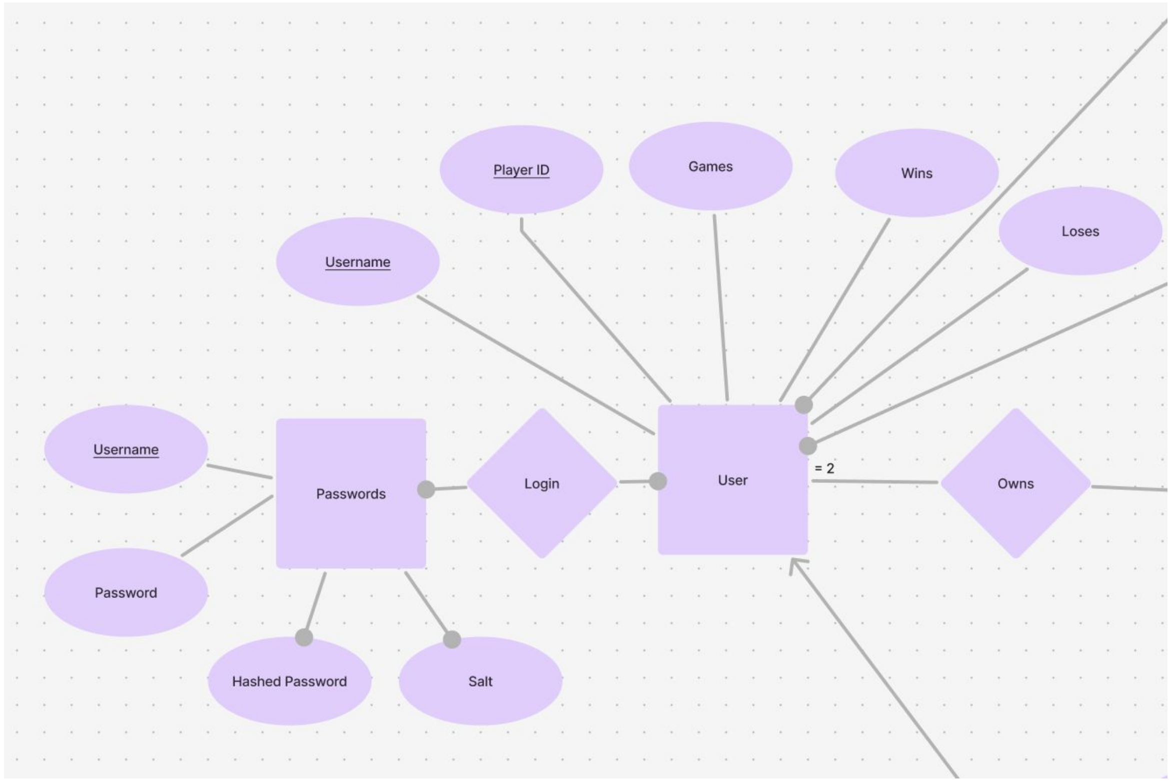
- In this use case, users login or create an account if they don't have one.
- Once playing the game, they can view a settings page, or a statistics page, or join the queue to play a game.
- From there the users buy units and unit modifiers, playing until they get 5 wins or 3 losses, at which they win or lose.

ER Diagram

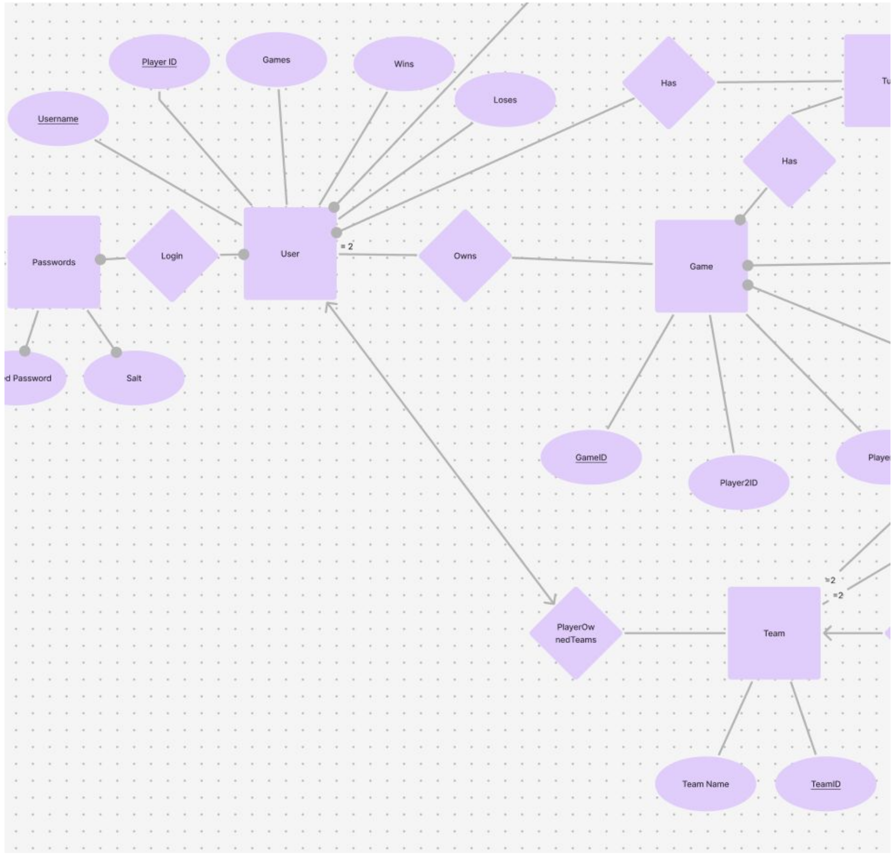


Buys From
Turn Number
Player ID
Games
Wins
Loses
Username
Has
Turn
has
ShopID
Has
Username
Passwords
Login
User
= 2
Owns
Game
Owned
Shop
Password
Hashed Password
Salt
Has
involved in
Sells
GameID
Player2ID
Player1ID
Unit Modifiers
ModiferID
Effected Attribute
=2
=2
PlayerOwnedTeams
Team
TeamOwnedUnits
Unit
UnitOwnedModifers
Cost
Name
Effect
Attack
Unit Name
Health
Cost
UnitID
Team Name
TeamID

ER Diagram

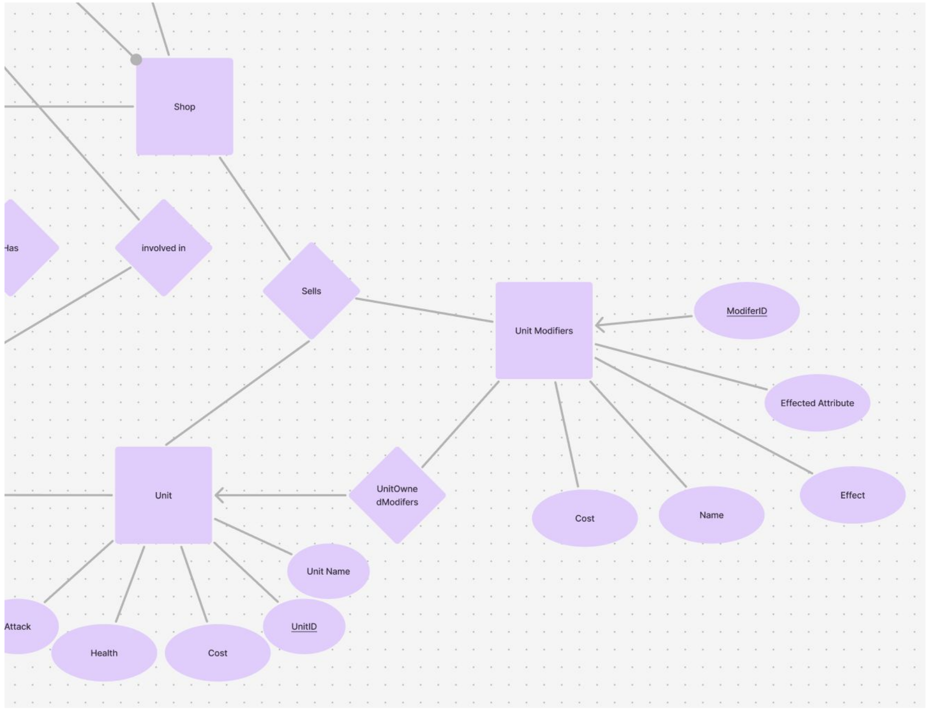

Player ID
Games
Wins
Loses
Username
Username
Passwords
Login
User
= 2
Owns
Password
Hashed Password
Salt

ER diagram

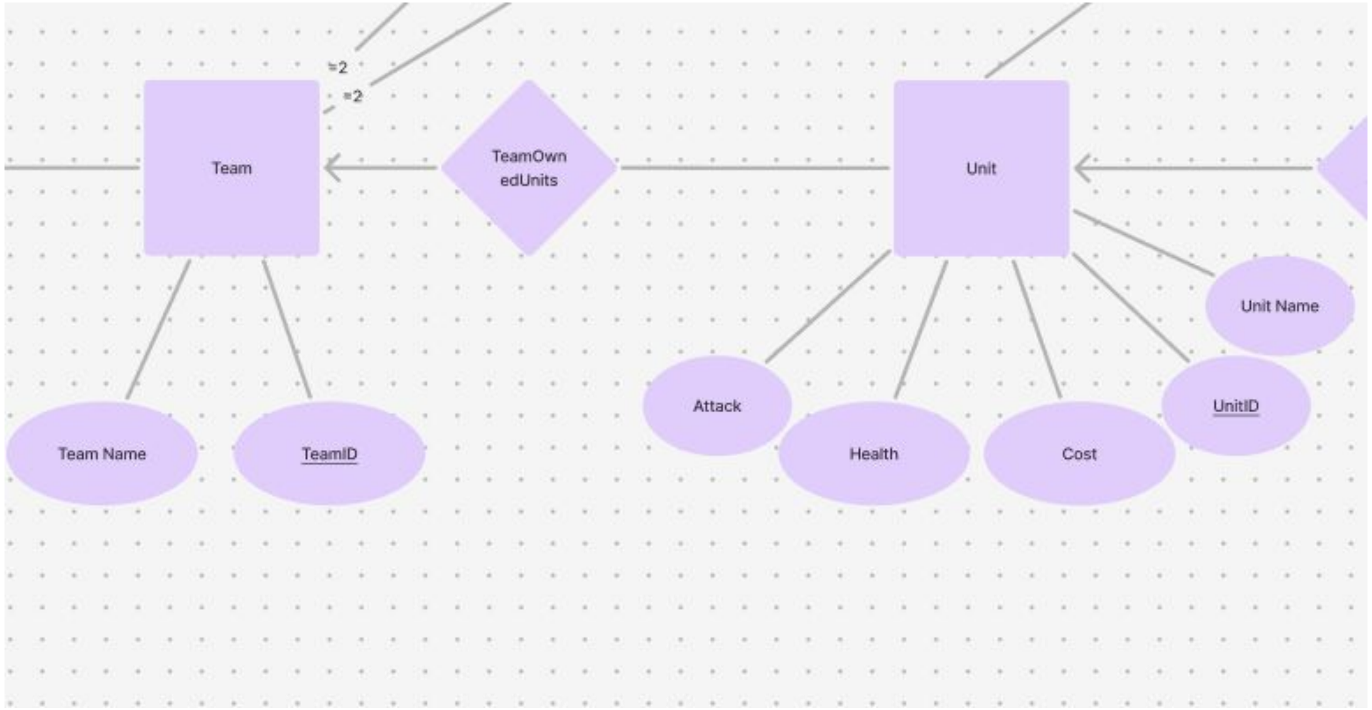


Player ID
Games
Wins
Loses
Username
Has
Tu
Has
Passwords
Login
User
= 2
Owns
Game
d Password
Salt
GameID
Player2ID
Playe
=2
=2
PlayerOwnedTeams
Team
Team Name
TeamID

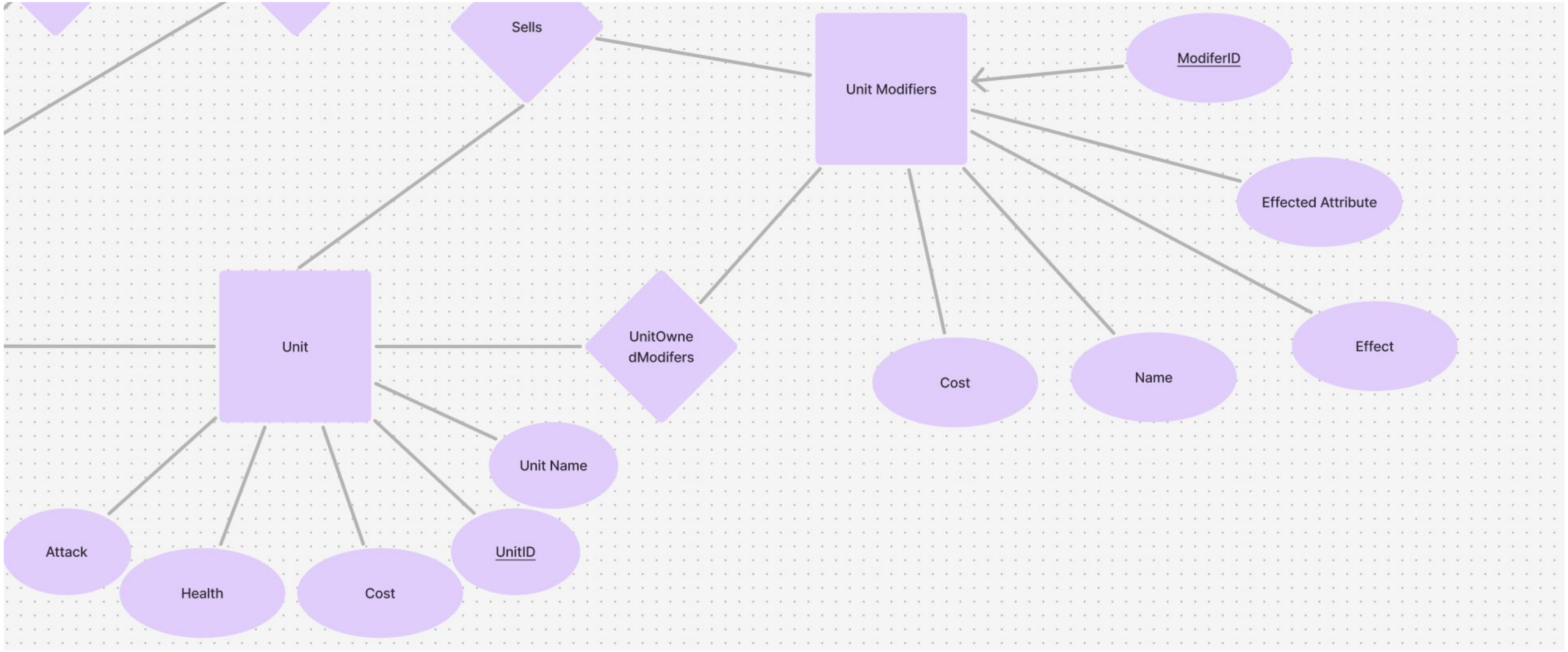
ER Diagram


Shop
Has
involved in
Sells
Unit Modifiers
ModiferID
Effected Attribute
Effect
Name
Cost
UnitOwnedModifers
Unit
Unit Name
UnitID
Attack
Health
Cost

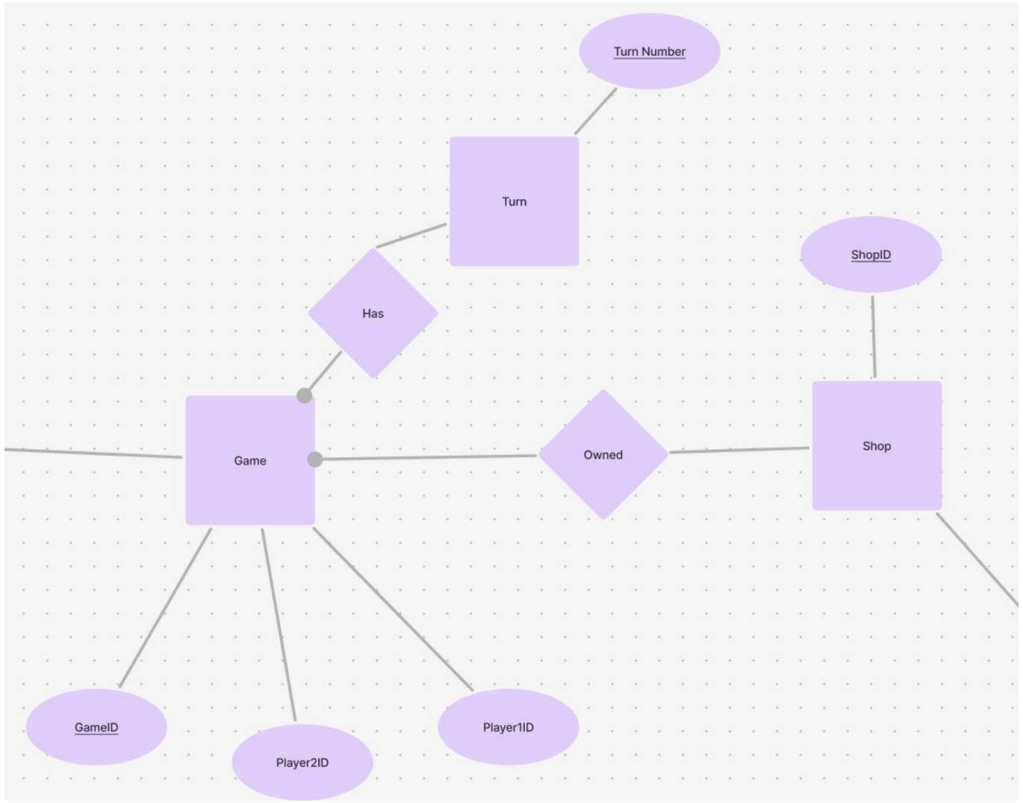
ER Diagram


=2
=2
Team
TeamOwnedUnits
Unit
Unit Name
Attack
Health
Cost
UnitID
Team Name
TeamID

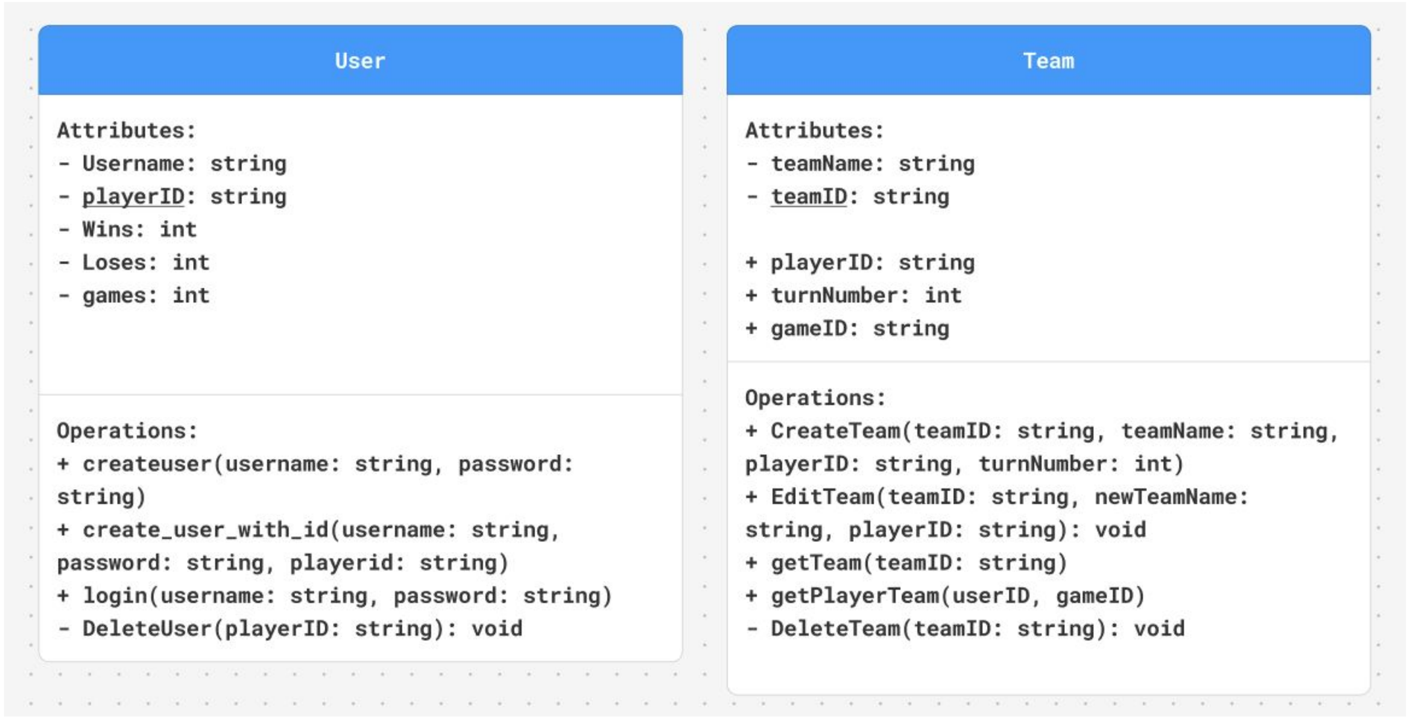
ER Diagram


Sells
Unit Modifiers
ModiferID
Effected Attribute
Unit
UnitOwnedModifers
Cost
Name
Effect
Unit Name
Attack
Health
Cost
UnitID

ER Diagram


Turn Number
Turn
ShopID
Has
Game
Owned
Shop
GameID
Player2ID
Player1ID

ER to Relations



User

```
Attributes:
- Username: string
- playerID: string
- Wins: int
- Loses: int
- games: int

Operations:
+ createuser(username: string, password:
string)
+ create_user_with_id(username: string,
password: string, playerid: string)
+ login(username: string, password: string)
- DeleteUser(playerID: string): void
```

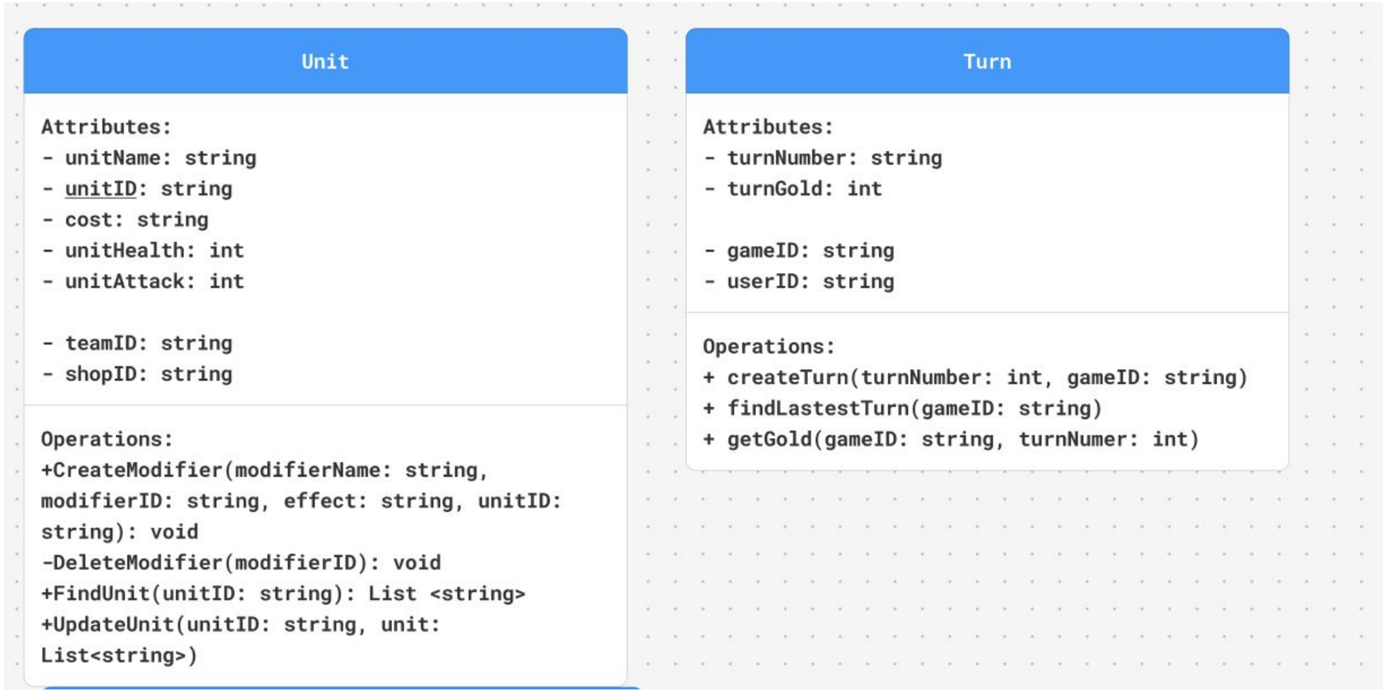
Team

```
Attributes:
- teamName: string
- teamID: string

+ playerID: string
+ turnNumber: int
+ gameID: string

Operations:
+ CreateTeam(teamID: string, teamName: string,
playerID: string, turnNumber: int)
+ EditTeam(teamID: string, newTeamName:
string, playerID: string): void
+ getTeam(teamID: string)
+ getPlayerTeam(userID, gameID)
- DeleteTeam(teamID: string): void
```

ER to Relations



```
Unit

Attributes:
- unitName: string
- unitID: string
- cost: string
- unitHealth: int
- unitAttack: int

- teamID: string
- shopID: string

Operations:
+CreateModifier(modifierName: string,
modifierID: string, effect: string, unitID:
string): void
-DeleteModifier(modifierID): void
+FindUnit(unitID: string): List <string>
+UpdateUnit(unitID: string, unit:
List<string>)
```

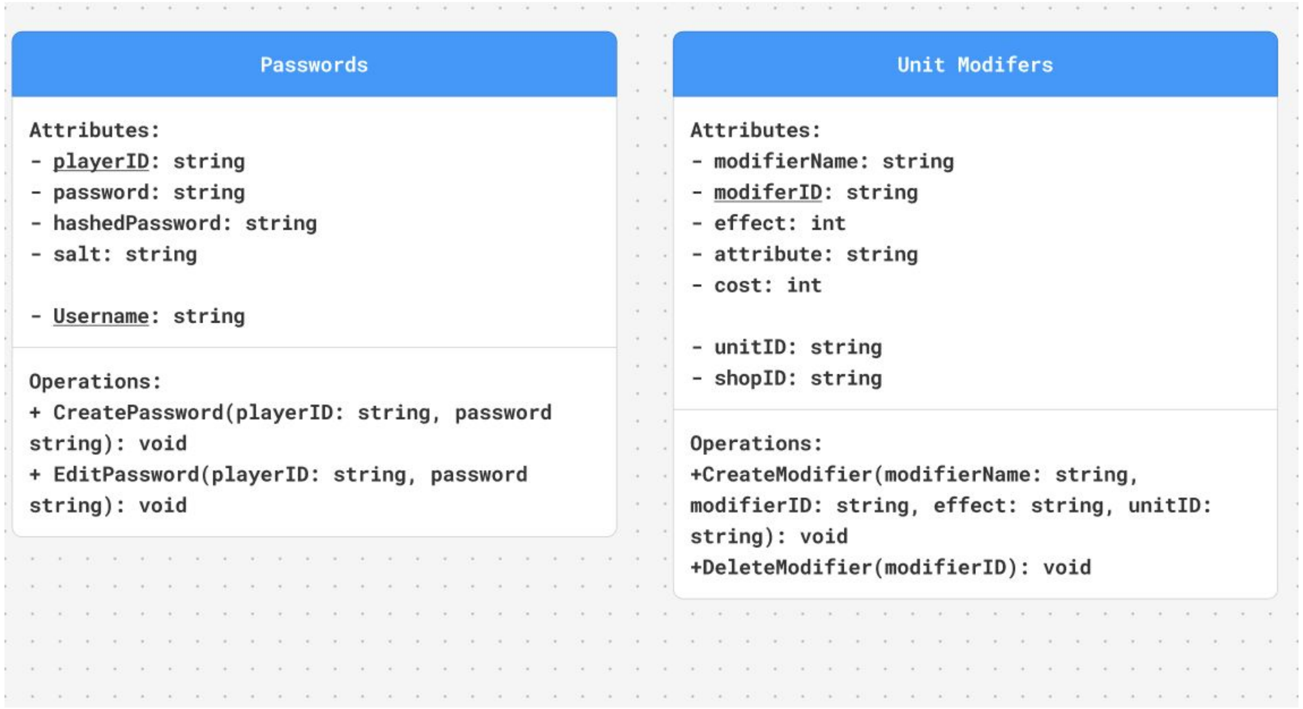
```
Turn

Attributes:
- turnNumber: string
- turnGold: int

- gameID: string
- userID: string

Operations:
+ createTurn(turnNumber: int, gameID: string)
+ findLastestTurn(gameID: string)
+ getGold(gameID: string, turnNumer: int)
```

ER to Relations



Passwords

```
Attributes:
- playerID: string
- password: string
- hashedPassword: string
- salt: string

- Username: string

Operations:
+ CreatePassword(playerID: string, password
string): void
+ EditPassword(playerID: string, password
string): void
```

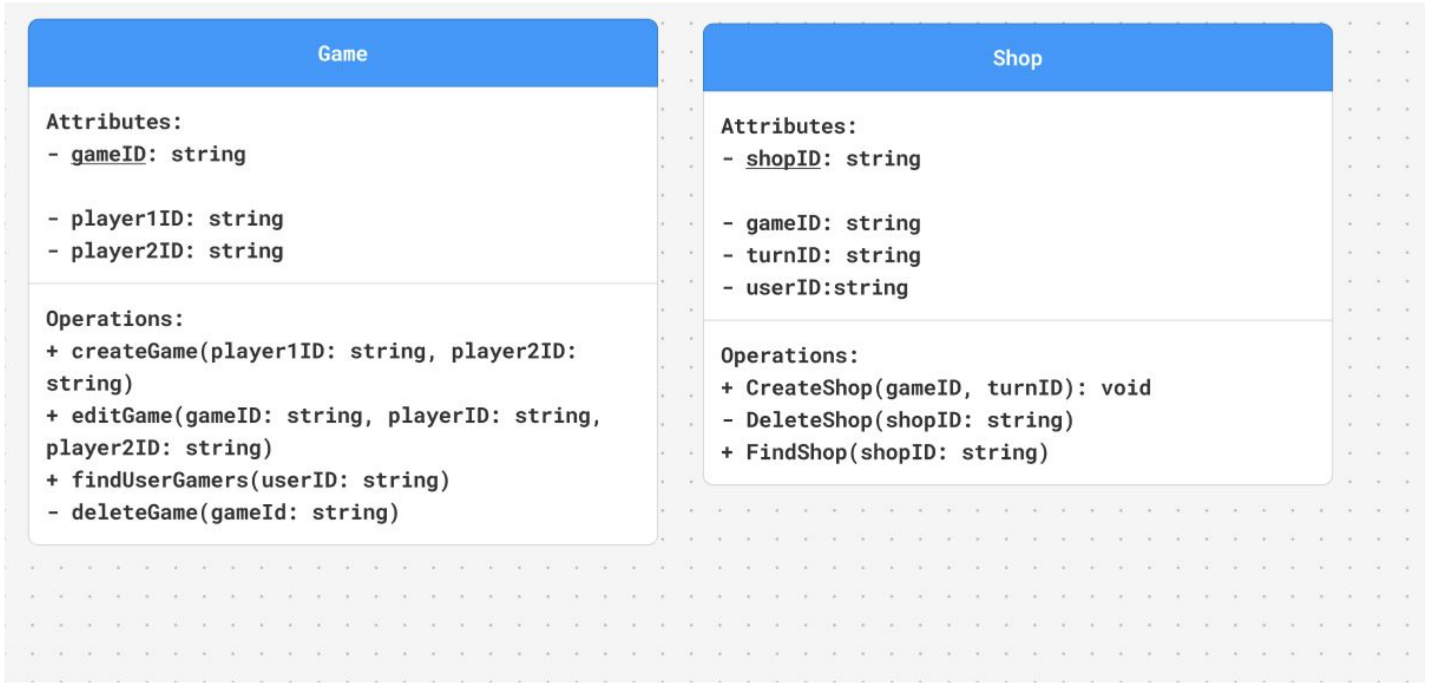
Unit Modifers

```
Attributes:
- modifierName: string
- modiferID: string
- effect: int
- attribute: string
- cost: int

- unitID: string
- shopID: string

Operations:
+CreateModifier(modifierName: string,
modifierID: string, effect: string, unitID:
string): void
+DeleteModifier(modifierID): void
```

ER to Relations



Attributes:
- gameID: string

- player1ID: string
- player2ID: string

Operations:
+ createGame(player1ID: string, player2ID: string)
+ editGame(gameID: string, playerID: string, player2ID: string)
+ findUserGamers(userID: string)
- deleteGame(gameId: string)

Attributes:
- shopID: string

- gameID: string
- turnID: string
- userID:string

Operations:
+ CreateShop(gameID, turnID): void
- DeleteShop(shopID: string)
+ FindShop(shopID: string)