CS 102	Spring 2021/22	Project Group	1I
Instructor:	Aynur Dayanık		
Assistant:	Mousa Azari		

Criteria	TA/Grader	Instructor
Presentation		
Overall		

Group I

ATLA

Muhammad Shayan Usman 22101343, Zahaab Khawaja 22101038, Dilara Yilmaz 22101646, Özgür İkidağ 22102315, Mohammed Alhaidari 22101490

Detailed Design

(Version 1)

24/04/2022

1. Introduction

2 player card turn-based game based on musical gameplay. The user will have a basic profile e.g., cast/class, health, stamina, and shield. The player gets a set number of cards (8). They can use special attacks and the entire board can get affected. Cards can have offensive, defensive or gaining health/stamina. Each player can cast a spell or attack on others.

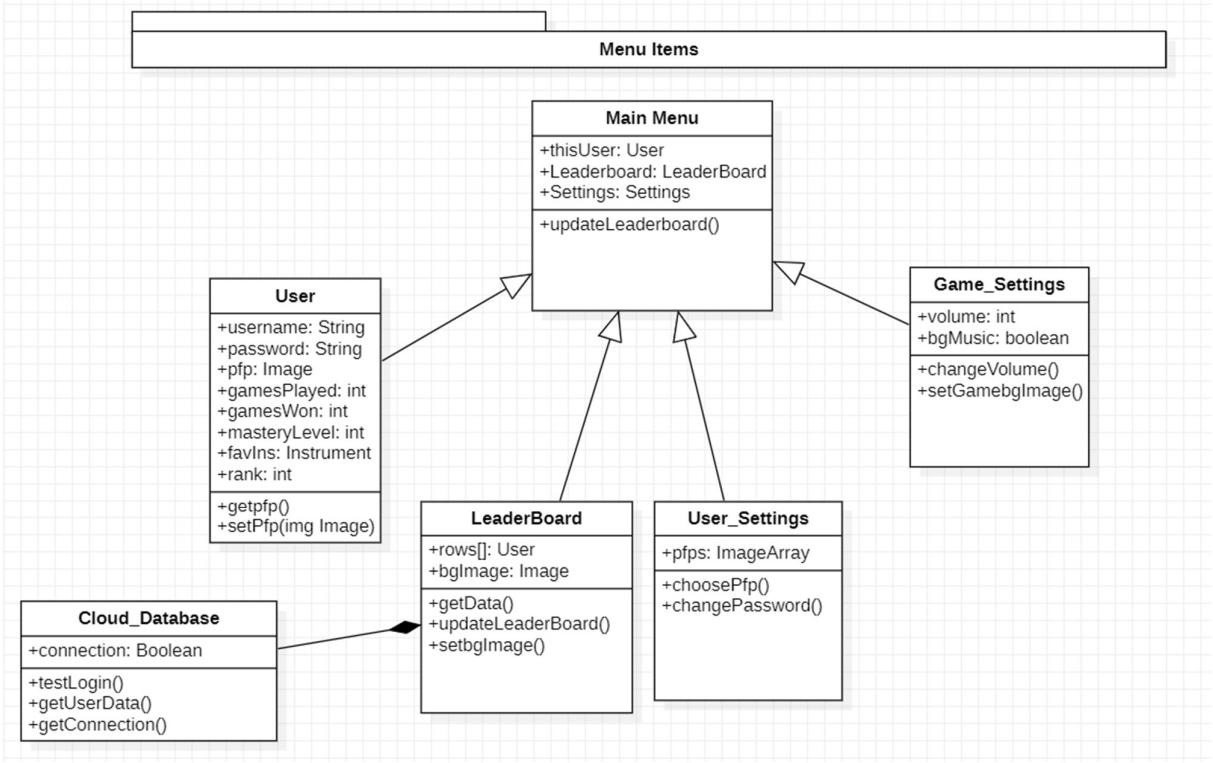
- Attack: Health damage
- Defensive (Shield generate)

We will also add multiplayer to allow players on the same network to play together. It will be turn-based so that the game revolves from player to player.

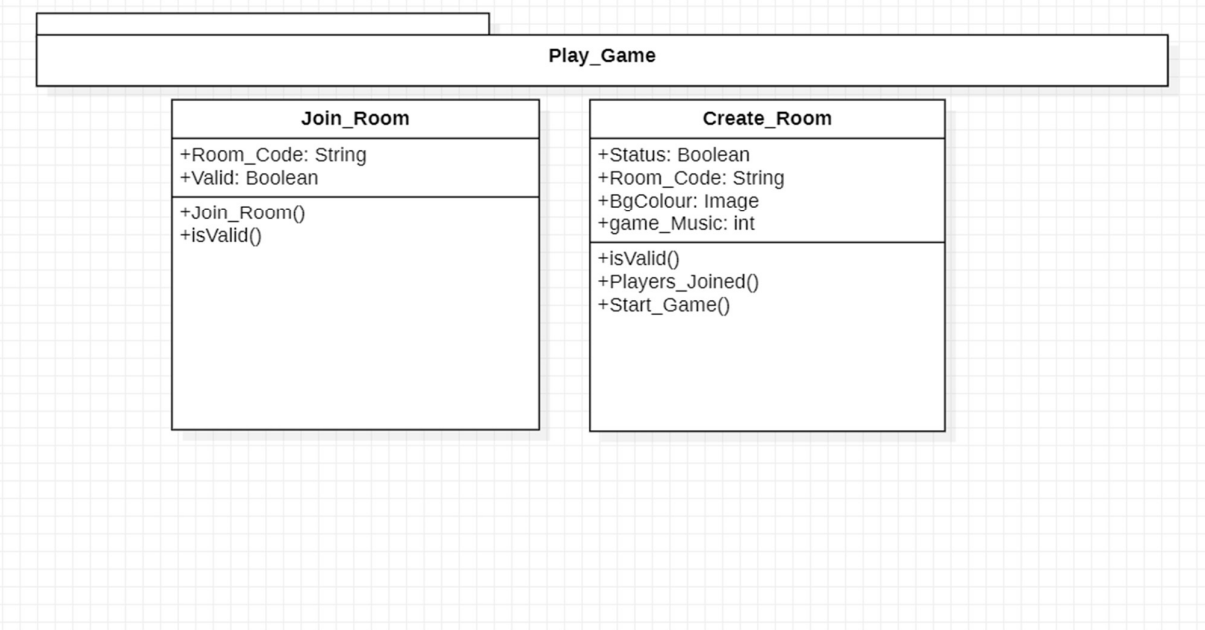
2. Details

2.1. UML Diagrams

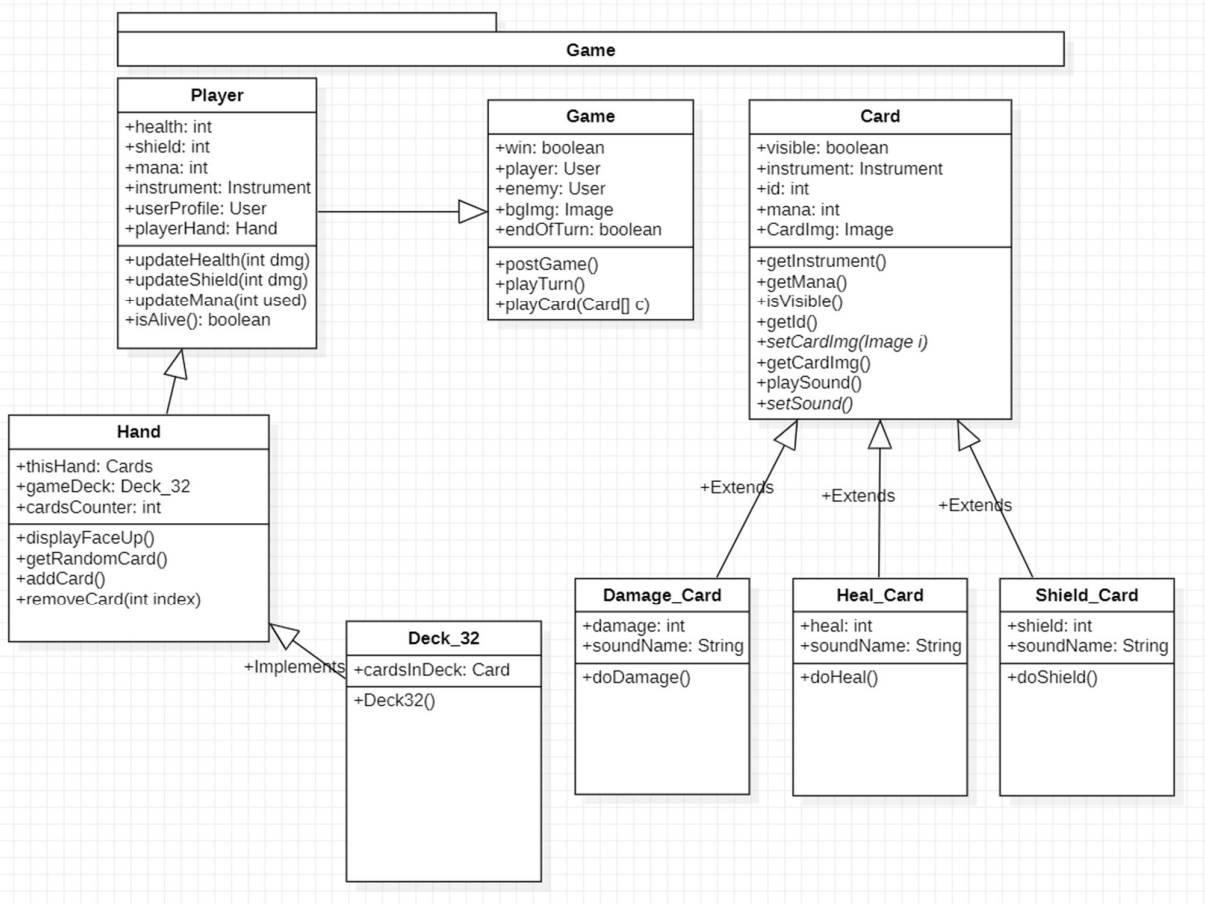
The UML has been split into three main parts, the main menu items, the play game (game lobbies) and the main game.



The menu items will consist of a main menu class which will store all the necessary information and have the necessary methods for the navigation of the main menu. There will also be a User class which will store all the user's information like their login details, profile picture and game stats. There will be a User settings class that will allow the user to change the settings of their profile. There will also be a game settings class that will allow the user to control things like the volume and background music.



The play game will consist of two classes. The join room class will check the inputted code and will send the player into the lobby. The create room class will allow the host to change game settings and to start the game.



The game will consist of an abstract class for the cards. There will be concrete classes for each subtype of card. A deck class will generate a deck of 32 cards that will be used to assign the player's hand. The hand class will have methods that will remove a specified card or will add a random card from the deck. The player class will have all the player's game stats like health,

stamina and shield. The game class will have instance variables and methods that allow the game to be played turn by turn.

2.2. Multiplayer

We will be using the Server-To-Client network to implement the multiplayer part of the game. This will make use of the java.net package and use its Socket and ServerSocket classes. “A socket is defined as one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.” [1]

Each game will be played on a server and the player will wait for another player to join. After every game, the server will be refreshed. Since the game is turn-based, there will be one server and two clients. The server will be listening to the client. After the client makes the part, it will listen to the other user. The turn will be decided by the software.

2.3. Web Application

We plan to turn our game into a webapp by using spring boot. By using spring boot, we will be able to display our java app as a web app. Spring boot will also help us create the UI for the website. Rather than creating HTML code and a CSS code to allow our web app to have a design, spring boot will help us generate all the necessary code for these features. Along with spring boot, we will be using vaadin to allow us to display the UI of the card game.

2.4. Database Storage

We plan to implement a database to our game that will store the clients’ data such as but not limited to the name of the client, the client’s statistics (i.e. games won and lost), and the user profile such as the most played instrument. We will be saving the data from the HTML form and connecting it to a database. [2]This data will help give a user profile that can be seen and edited before and after the game. We will probably choose to utilize some sort of open-source relational database management system. Once the information is stored, we can access the database and use it to showcase it on the main page. We plan on using Google Firebase cloud database. It is a cloud-hosted NoSQL database that will allow us to store all the necessary information and game stats that are required for our implementation.[3]

2.5. Distribution of Work

The distribution of the workload for the implementation of the classes is as follows:

Muhammad Shayan Usman 22101343 - Game mechanics, Multiplayer

Zahaab Khawaja 22101038 - Web app & GUI, Multiplayer

Dilara Yilmaz 22101646 - Play_Game methods

---

[1] *What Is a Socket? (The Java™ Tutorials > Custom Networking > All About Sockets)*, https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html. Accessed 7 March 2022.

[2] Deshpande, Ashish. “How to Save Data from an HTML Form to a Database.” *frevvo*, 1 September 2021, https://www.frevvo.com/blog/save-data-from-html-form-to-database/. Accessed 7 March 2022.

[3]Firebase, G. (n.d.). Firebase Realtime Database | Store and sync data in real time. Firebase. https://firebase.google.com/products/realtime-database Accessed 24 April 2022

Özgür İkidağ 22102315 - UI card and UI arena design

Mohammed Alhaidari 22101490 - Database & play game classes, Multiplayer

3. Summary and Conclusion

To conclude, we will all be working on separate classes that can be seen above in the UML diagram. We will implement the multiplayer feature of our game using the java.net package. To turn our code into a web application, we will be using spring boot and vaadin to create the web app and the UI elements. To create a database, we will be using google firebase which will allow us to store all the necessary stats for our game.