System Architecture

Monster Fighter 30K

Document version number *1.0*

Written by *Team 5*

Point of contact *Barbara Seidl (seidl@finf.uni-hannover.de)*

2023.05.12.

Group members

Name	Role	Main contributions
Barbara Seidl	Team leader, Developer, Product	Backend
Kay Hartmann	Developer, Product Owner	Frontend
Yana Mavlenko	Developer, Product Owner	Game Design and Animations

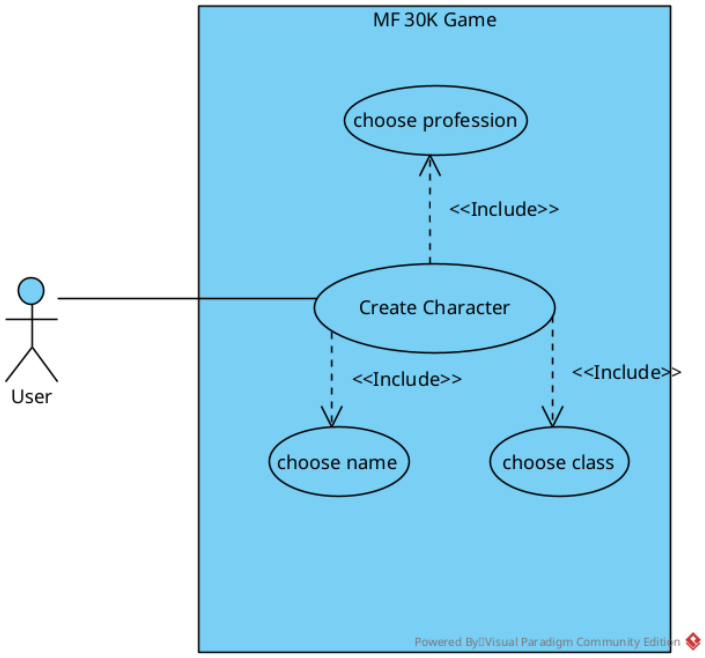
Document history

Revision	Date	Name	Comment
1	14.06.	Updated Class Diagrams	Class Diagrams and Database design

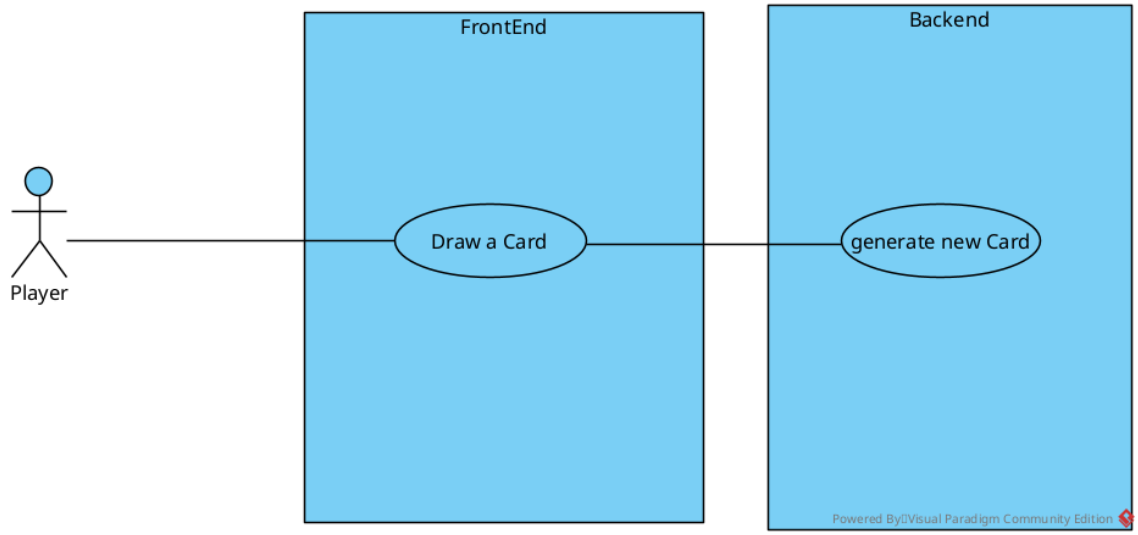
1. Object-Oriented Analysis

Present the results of your object-oriented analysis steps here. For each diagram, write also text to explain the key points of it. Keep these updated during the project.

1.1. Use-case diagrams

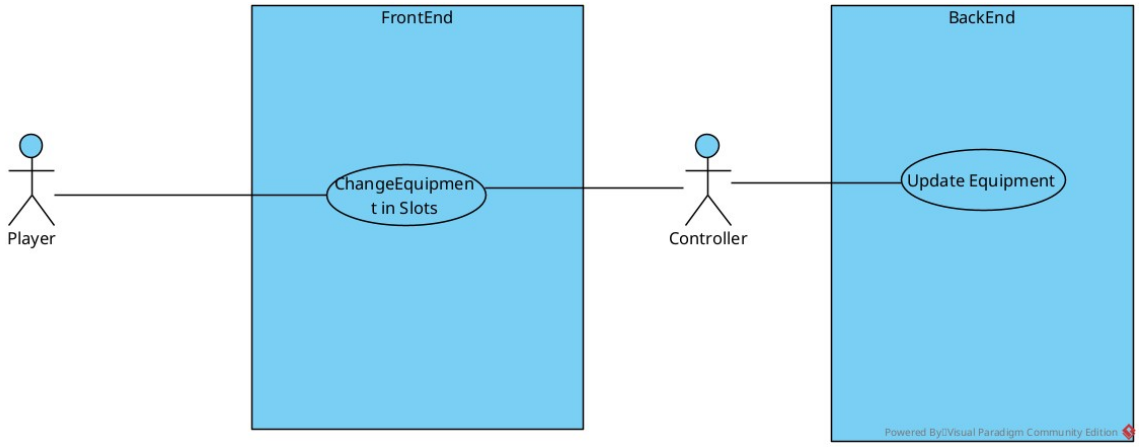


Initially any player needs to create a character to play the game. They choose their name (which is not allowed to be taken already) and chooses their initial race and profession.



Whenever a player draws a card, a new card is generated by the backend.

Update Equipment:



1.2. Class diagrams (initial)

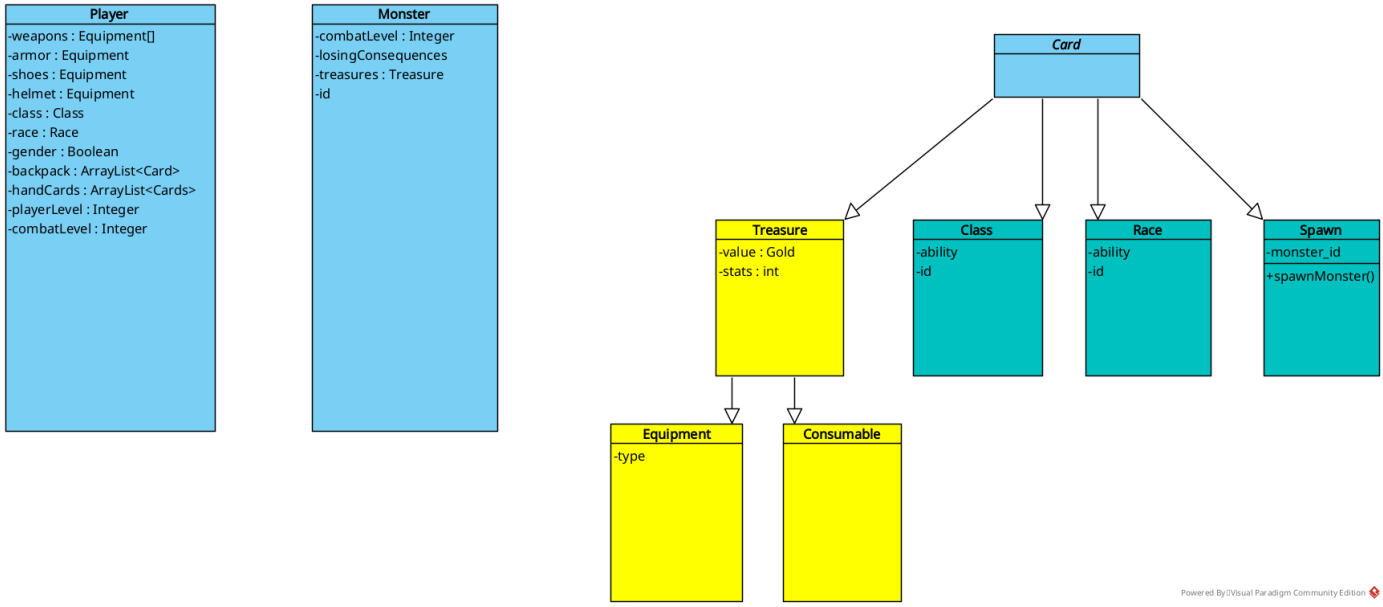
Player: The player class holds all relevant information regarding the player that is needed for the game.

Monster: The adversary to the player. For Combat combatlevel of player and monster are compared.

Card: An abstract class serving as a superclass to all further cards. All cards can be drawn during the game.

Auxiliary Cards: These are Race, Profession and Spawning Cards. Spawning Cards spawn monsters. All auxiliary cards can be drawn during the game.

Treasure Cards: After defeating a monster, a certain amount of treasure cards is drawn. These include Equipment and Consumables (One-Time-Use-Buffs) that can be used by the player, to temporarily boost their combat level.



1.3. State diagrams

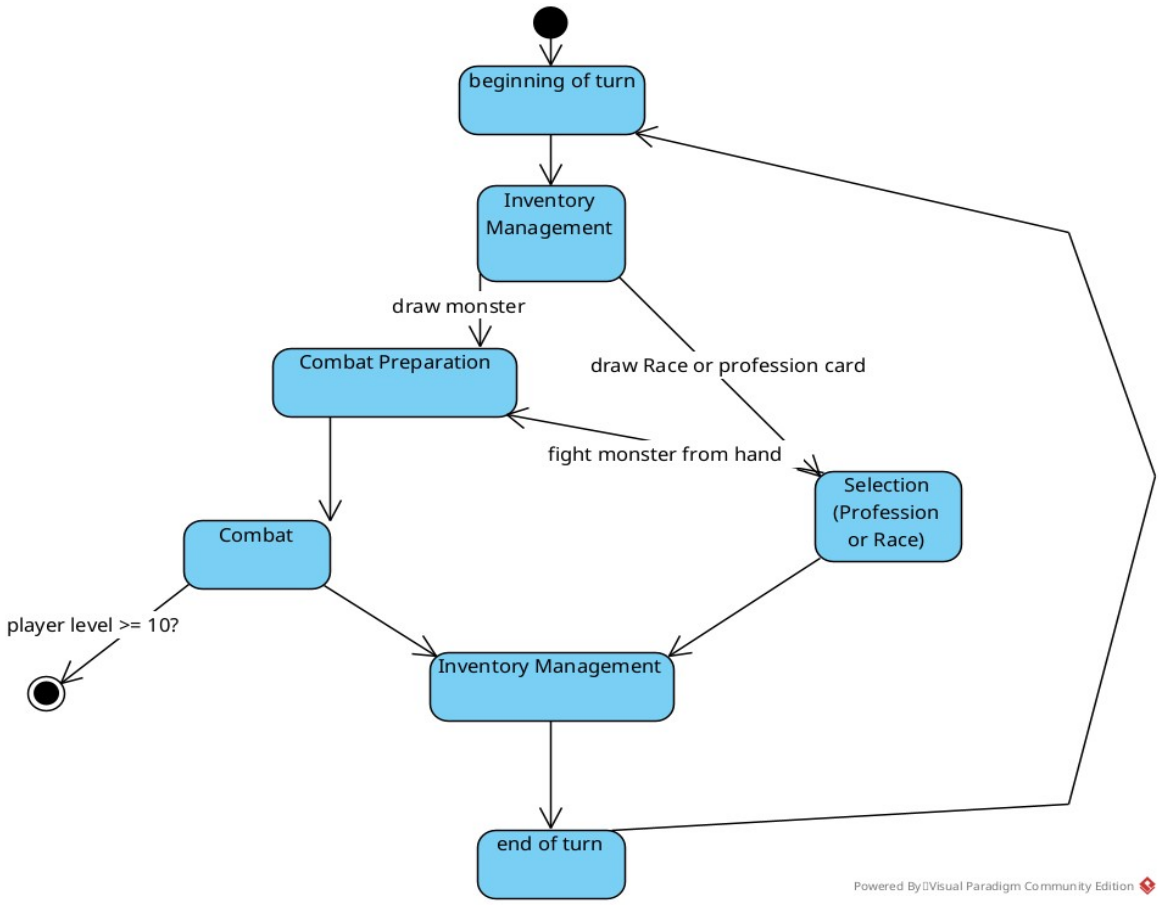
The game has four main states: Inventory management, where the player can equip cards from their hand or from their backpack, put cards from the hand in the backpack (and vice versa) or sell items for gold.

After that the player draws a card, which leads either to a combat preparation state, if the drawn card is a monster, or to a selection state, if the drawn card is a race or profession card.

In the combat preparation state the player can apply consumable items from their hand to the game. These can either be applied to the monster, to receive more treasures (or in the multiplayer variant to let other players lose) or to the player side, to help win against a monster.

After combat or selection, the player can manage their inventory again, before ending the turn.

The game ends the moment at least one player reaches player level 10. Levels can be gained by defeating monsters (one level per monster).



2. Object-Oriented Design

Present the results of your object-oriented design steps here. These diagrams constitute the overall architecture of your project. Keep these updated during the project. For each diagram, write also text to explain the key points of it.

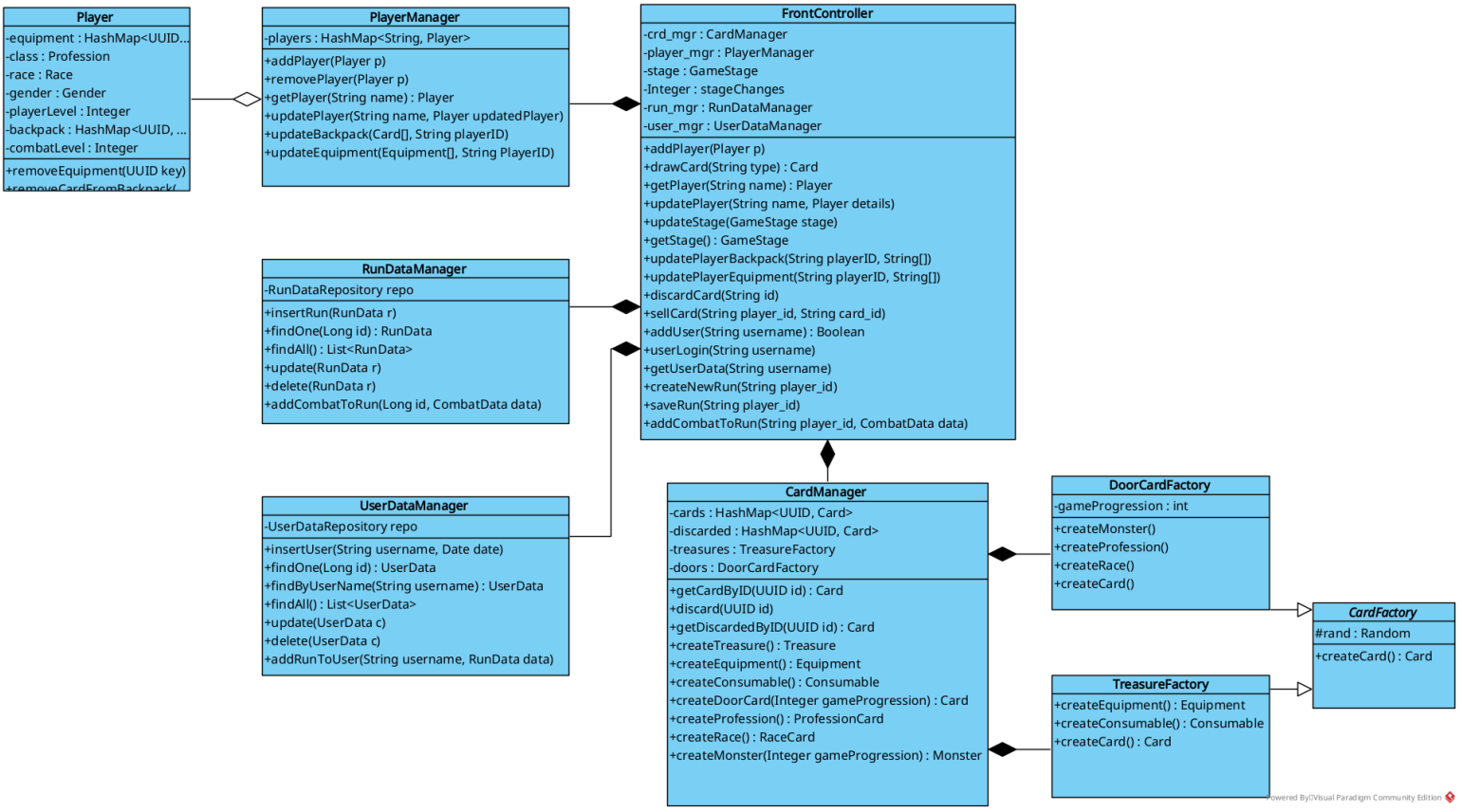
2.1. Class diagrams (complete)

Player: The player class holds all relevant information regarding the player that is needed for the game: The adversary to the player. For Combat combat level of player and monster are compared.

Card: An abstract class serving as a superclass to all further cards. All cards can be drawn during the game. Each Card features its type (e.g. “Consumable”, “Equipment”, etc.) as a String, that is set at construction.

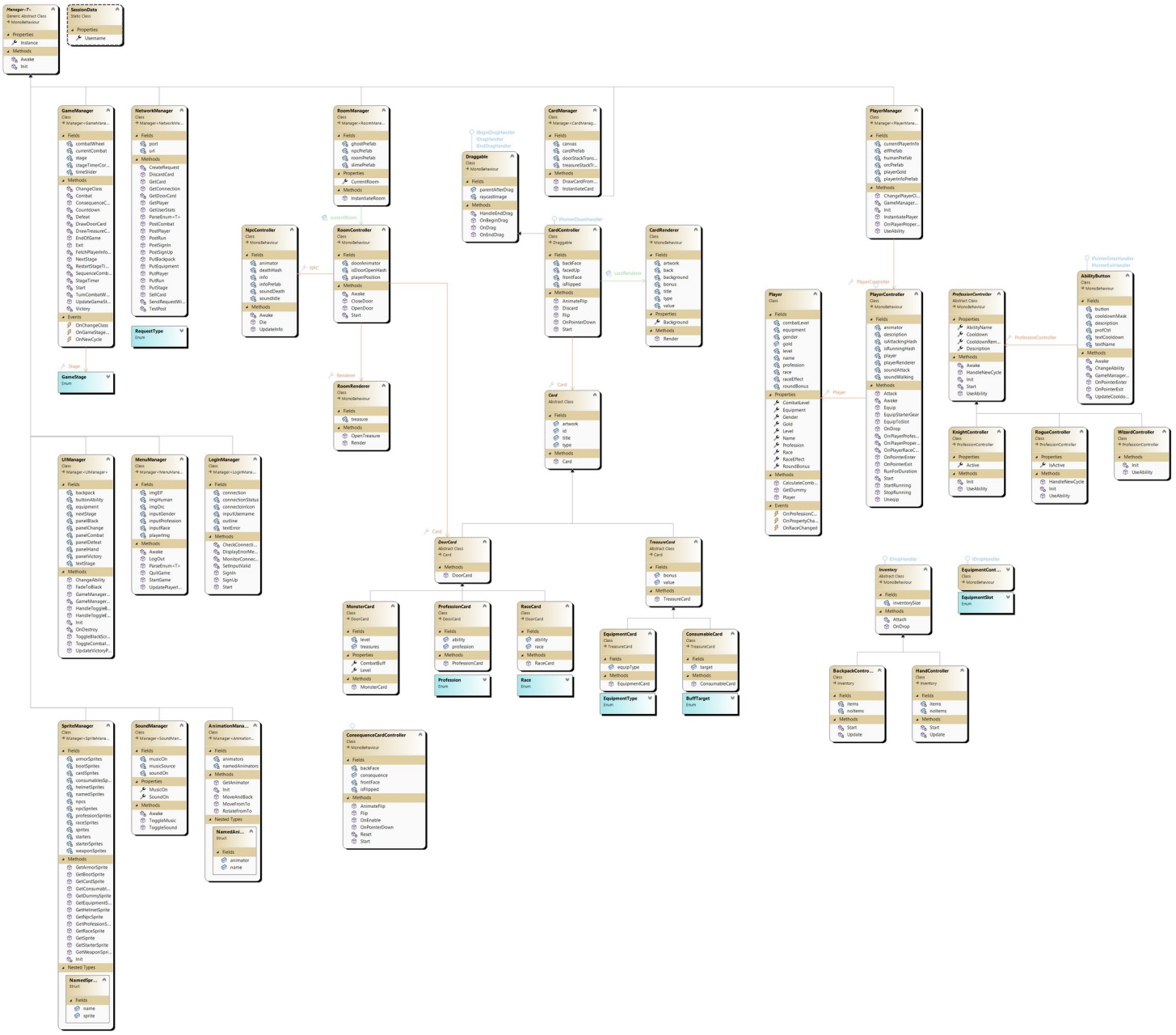
Auxiliary Cards: These are Race, Profession and Monster cards. They are drawn after inventory preparation phase at random.

Treasure Cards: After defeating a monster, a certain amount of treasure cards is drawn. These include Equipment and Consumables (One-Time-Use-Buffs) that can be used by the player, to temporarily boost their combat level. (Or in the multiplayer variant, they can be used to boost the monster's combat level, to let another player lose.)



The Front Controller is responsible for handling and delegating all requests made by the frontend. The PlayerManager manages all Player objects and the according backpacks and equipment of the players. The CardManager is responsible for creating cards and keeping track of all cards currently in game.

Run- and UserDataManager are the Interfaces to the Database.



This is the class diagram for the front end.

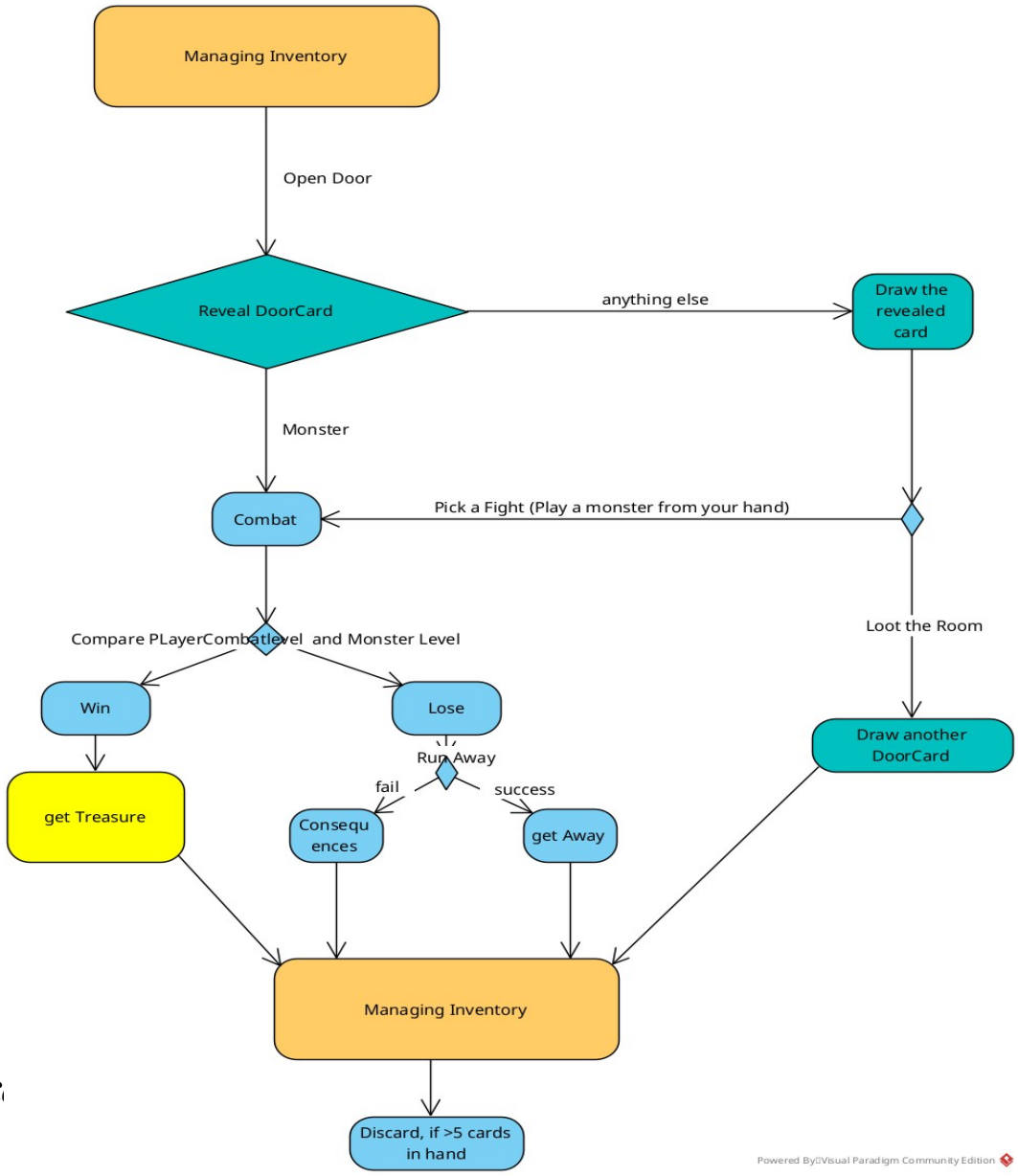
2.2. Activity diagrams

Turn process:

The player can first manage their inventory (change their equipment, sell items). Then they draw a random card (“they kick in the door”) which can either be a monster, race or profession card. If it is a monster, they have to fight it. Otherwise they can just draw the card. If they don’t fight a monster, they can either fight a monster from their handcards, or just draw another hidden card from the stack.

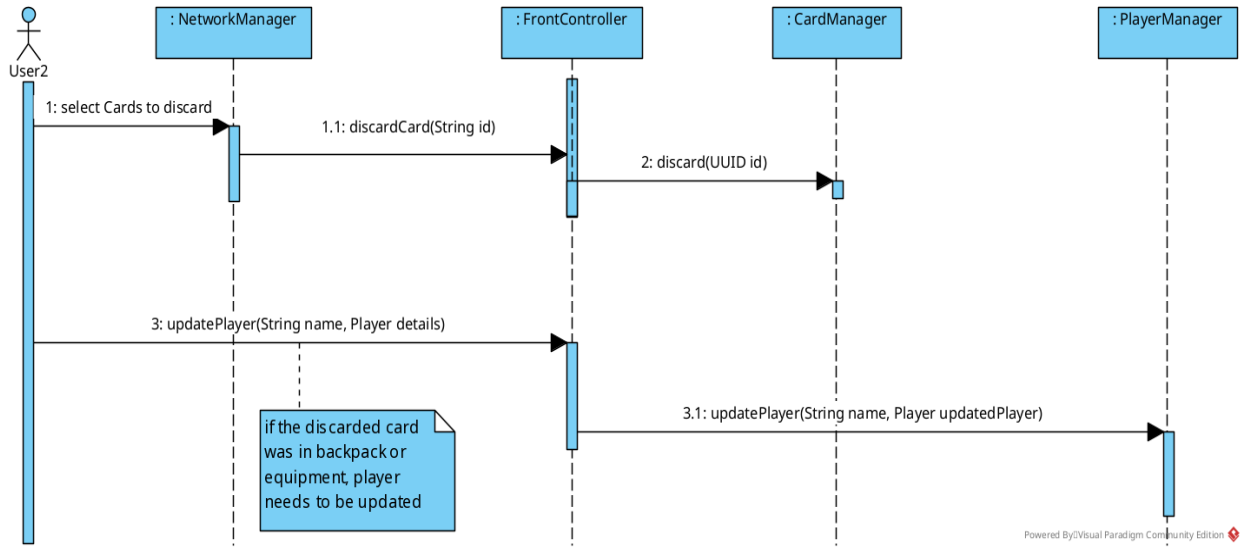
If the player fights a monster and wins, they draw one, two or three treasure cards, depending on the monster level. If they lose, they can run away and have some chance of succeeding. If they don’t succeed, they face some kind of losing consequences, again depending on the monster.

After that, they can manage their inventory again and end their turn.

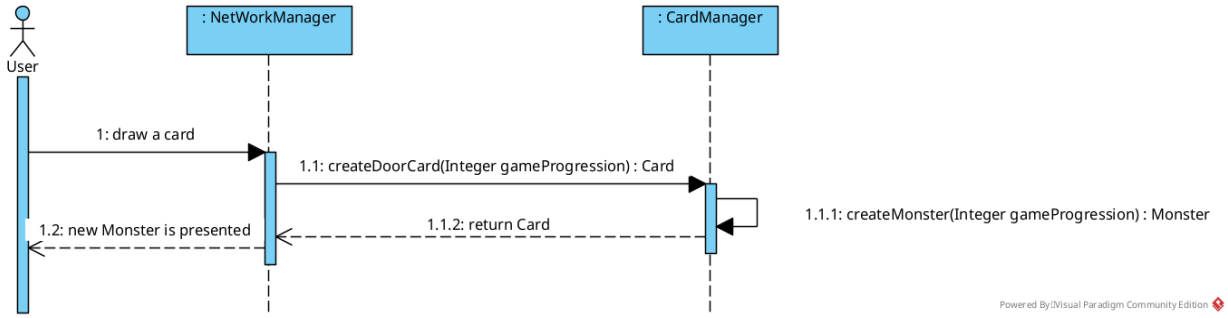


2.3. Sequence diagrams

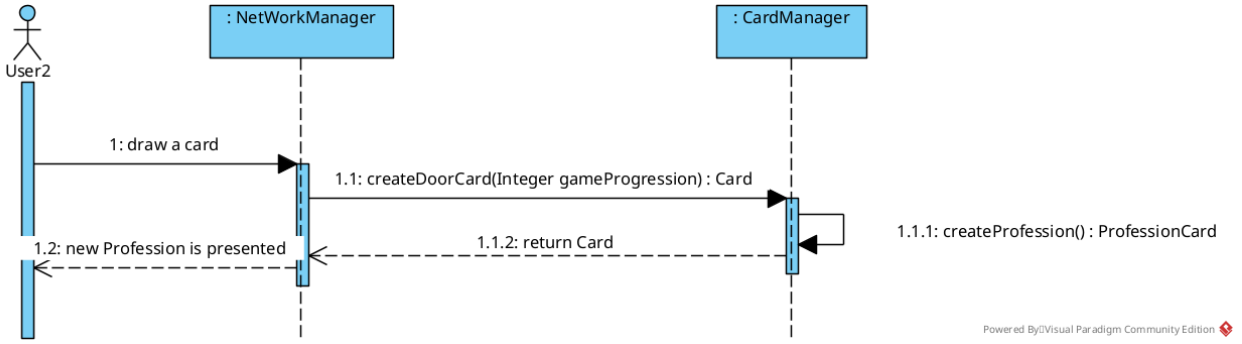
Discarding Cards:

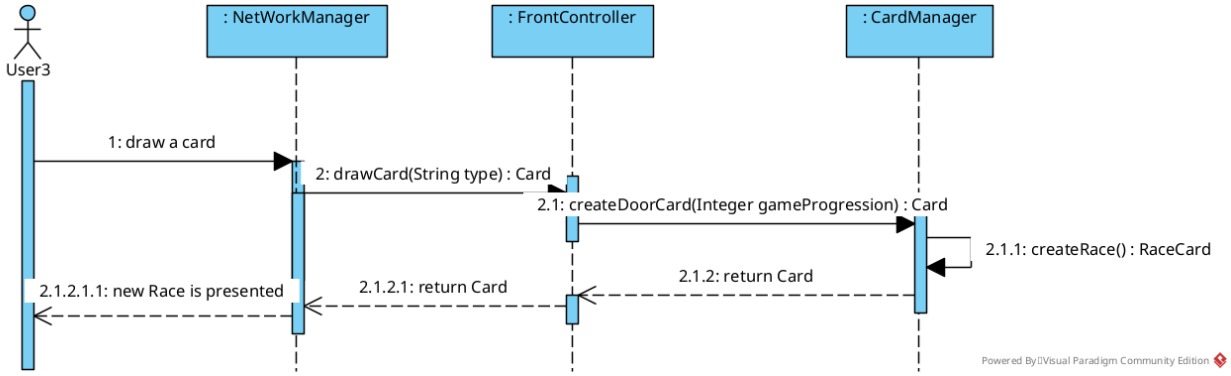


Drawing a card (its a monster):

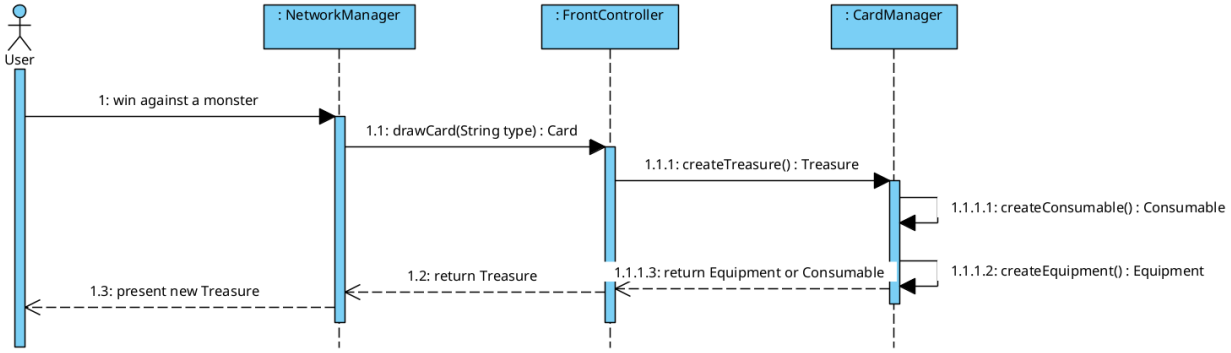


Drawing a card (its a Profession/Race):

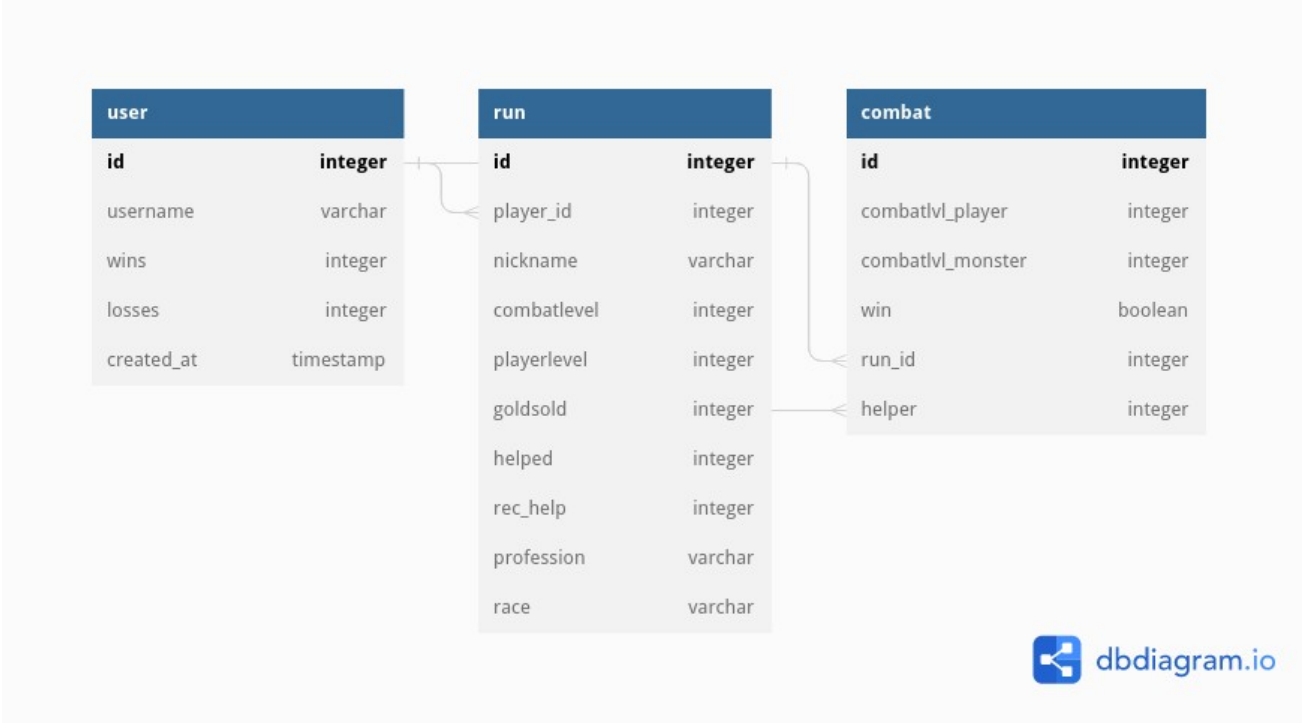




Drawing a treasure card after winning against a monster:



2.4. Database architecture



3. Design patterns

Factory Method:

The cards in the game are created via a factory method.

Singleton:

All controllers in the game (both frontend and backend) are Singletons.

Facade:

Additionally to being a singleton, the frontcontroller in the backend is also a facade, hiding the more complex sub-managers and offering a unified point of access to the frontend.

Model Controllers:

Arguably, the controllers could also be seen as Model – (View) – Controllers, as they control the backend and frontend models, but they lack a view. So we are not sure if this counts.

4. API documentation

Our entire documentation can be found here:
https://documenter.getpostman.com/view/26862483/2s93sf3rkg