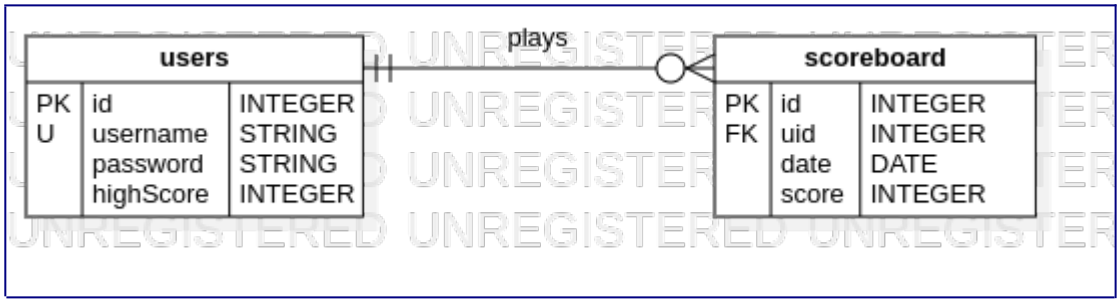
Milestone 2

Brief Project Description

The project aims to create a game with one player or multiple players and aliens with different types.As the game starts there will be aliens that player should attack. Some of the aliens can also attack to player. The attack mechanism is autofiring. There are 5 levels and their difficulty increases as level gets higher. At the last level there is a giant alien called COVID-19.

Database Design



Our database has two tables, namely users and scoreboard. Users table is responsible for holding user data, which are user id, username, password and high score. The other table is scoreboard. In this table, each gameplay is recorded with its own id, its player's id, player's score during that gameplay and the date of the gameplay.

Users have one-to-many relationship with gameplays. A user can have many games played. A gameplay can only have one user, as gameplays are recorded with their winners only or gameplays are recorded when the player dies during the single player levels.

One can register by using unique username and a password, and login by their registered username and password. Once a game is played, the relative user id is assigned to the game. The functionality of having the leader board from the last 7 days and from the last 30 days is added as well. With this functionality, one can simply see the highest scores played in 7 days or 30 days.

Aliens and the Player

An interface called Entity is implemented to give an interface for both the aliens and the player, as these entities have plenty of common features: All of them move on the grid, shoot, have hit points and die when all of their hit points are consumed. The implementers of the Entity interface are Alien and Player classes. Alien class is then further extended by subclasses of different alient types. We called them SARS, MERS, H5N1 and as a giant alien COVID-19. These have different properties in terms of their HP, their shooting and movement capabilities.

H5N1: Simplest alien in our implementation. It has only 2 hit points, it cannot shoot or move.

SARS: The second simplest alien we implemented. It has 5 hit points and it can move in two dimensions. It can shoot.

MERS: It has 10 hit points and it can move in two dimensions. We plan to make MERS shoot a little more frequently than SARS as well.

COVID-19: No surprises, this will be our final boss. It has 100 hit points. We are yet to decide its moving pattern and how it shoots, as this will be in Milestone 4.

Gameplay

1st Level: There will be H5N1s on the top 3 rows of the grid. Player is only responsible for killing them, no risk for getting shot.

2nd Level: In addition to the 1st level's configuration, there will be 1 SARS located right below the H5N1s. SARS will move and shoot.

3rd Level: In addition to the 2nd level, there will be a MARS located one row lower than SARS. Now, the user has to escape from both SARS and MERS and at the same time tries to shoot them.

4rd Level: In this level, there will be 3 SARS' and 3 MERS' located on the grid, each of them are located one row apart from each other, occupying the top 6 rows of the grid.

Grid

A grid is used in order to track the coordinates of the aliens and the player. It is a 12x12 square shaped grid with coordinate system [x,y] where x denotes the width and y denotes the height. Lower left corner is [0,0].

The Grid class is implemented as a Singleton class, as every movement happens on the same grid. When the user is leveled up, aliens of new level are placed on the grid.

The shooting mechanism is also easier using a grid. Basically, when aliens shoot, their missile appears on the cell right below of them. That is, if an alien on the cell [3,4] shoots down, its missile appears on [3,5]. If, on the other hand, the player from [5,0] shoots, its missile will appear on [5,1]. Both alien missiles and missiles shot by the player advance on the grid by 1 cell at a time. While alien missiles go down on the y-axis at a time, missiler shot by the player go up on the y-axis. After each advance of missiles, it is checked whether it hit an alien or the player. If so, the alien's/the player's hit points are decreased by one.

An alien's or the player's hit points reaching zero means that the entity is dead. If an alien is dead, it is removed from the grid. Information regarding on how many aliens are left on the grid with their current hit points and the player's remaining hit points will be provided to the client through API calls. It will be client side's responsibility to check whether all aliens are or the player is dead. If so, it will make a levelUp or saveGamePlay, updateHighScore and finishGame request, respectively.

Besides, a cheating mechanism's back end is also implemented during this phase. A controller can send the cheat command, and the current level is passed automatically by killing all aliens.

Unit & API Testing

We have tried to test every possible case as we could think of for both testing types. Here you can see the Postman collection share link:
https://www.getpostman.com/collections/6dbba829b2af040402ff

Issues

We wrote unit tests for DB connectivity and functionality on repositories for User and ScoreBoard classes. Those tests pass only once with a unique username since there is a unique tag on username. We do not know whether it was a good idea to run unit test on the database but since we wrote them, we submit them as well.

Further Work & Improvements

1)Due to the fact that we had little time for this phase, unfortunately we could not implement a proper authentication system. It will be our first job to replace it with a proper authentication system.

2)We need to do some refactorings and further debugging for eliminating unsure and shady situations.