- Sprint summary

  In this sprint we intended to choose some basic functionalities of the system to get familiar with the new technologies. We have worked in both the backend and frontend of the project.

  Regarding Backend, we have implemented:
    - Sign up
    - Sign in
    - Database model
    - Player model

  Regarding Backend, we have implemented:
    - Landing page
    - Starting window
    - Sign in page
    - Sign up page
    - Gamespace page
    - Profile page

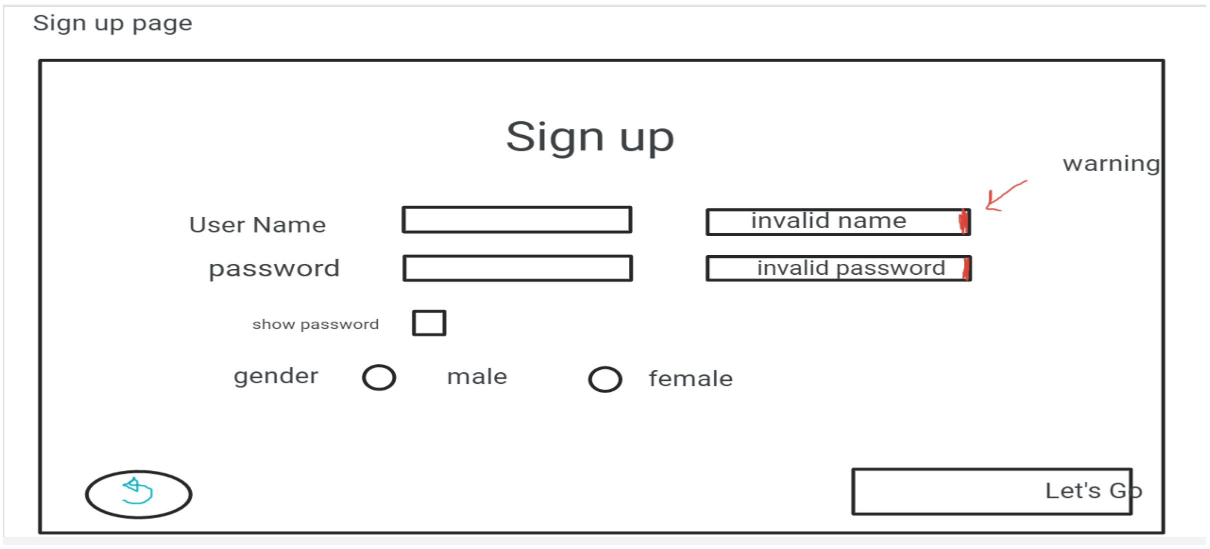
  Also, we have created part of a relational data model using sql.

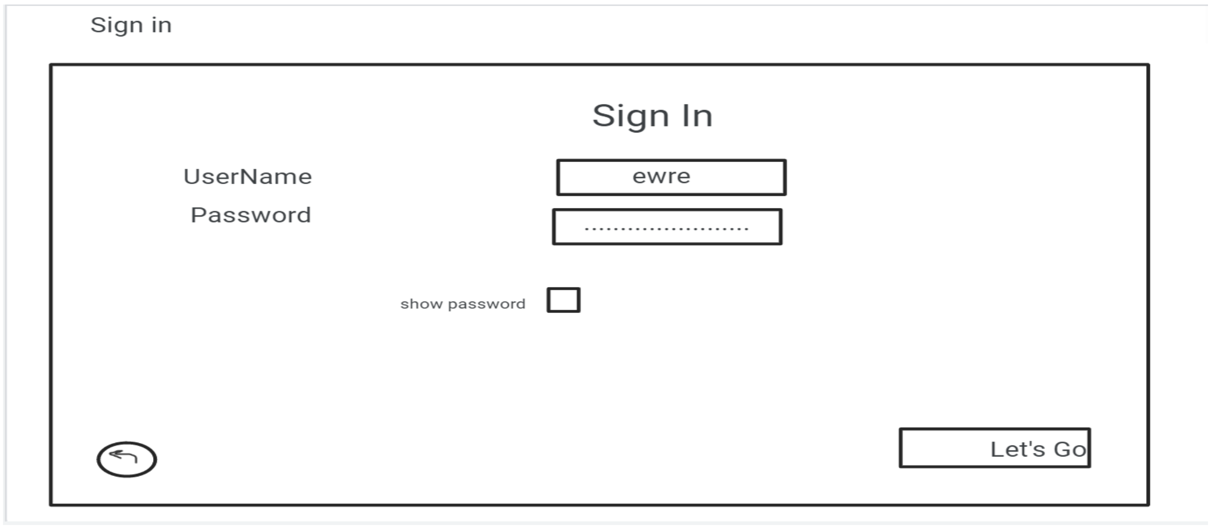
- Code structure

  Project has two directories :
    - Code : which is divided into sub directories to group related classes together
      - Gui: contains files of gui
      - Auth proxy : responsible for authenticate users

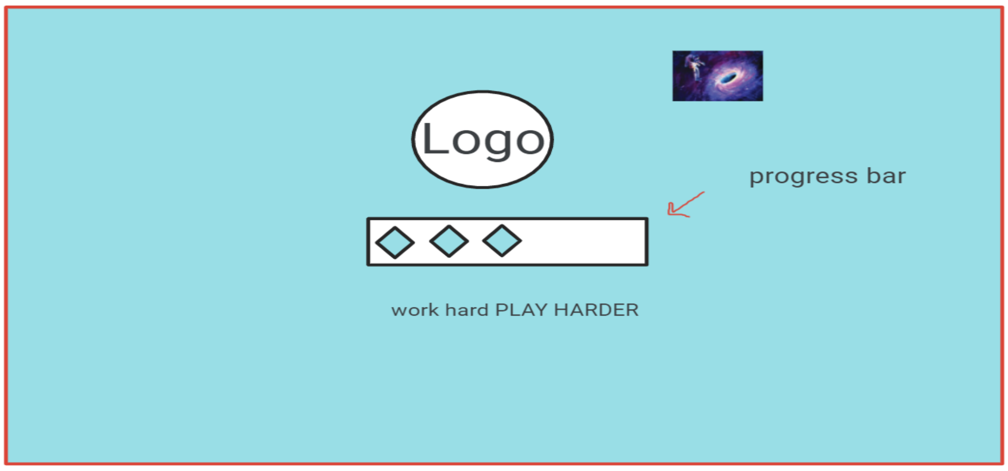
    - Database : responsible for communication between application and database layer
    - Models : contains the general models which will be used
    - Storage : which contains all icons, images and sounds that we use in the application
- Design: which has qt designer files of different Gui pages

- Initial Prototype
  To have a better vision of the gui and choosing its theme we've developed a low fidelity prototype at the beginning of this sprint

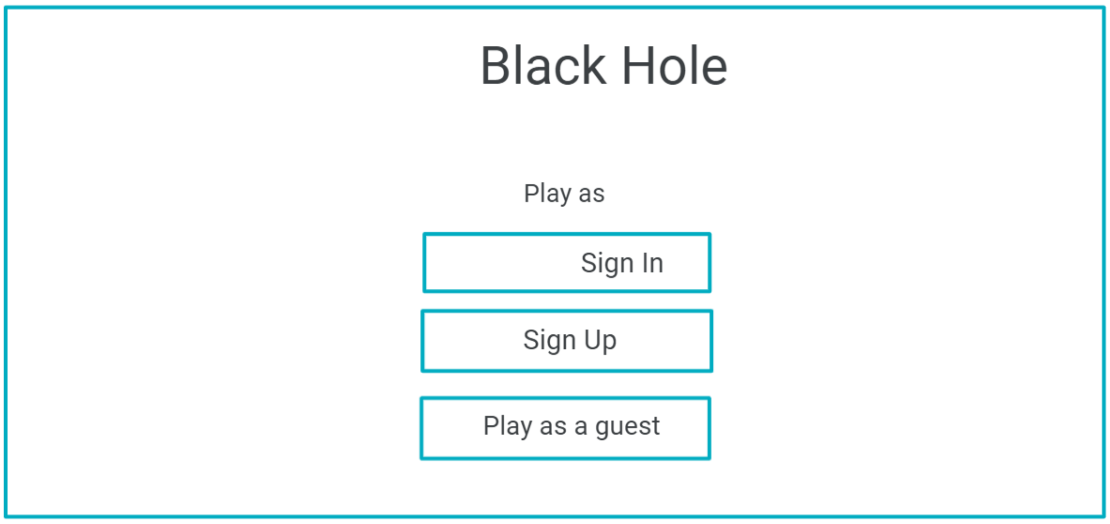




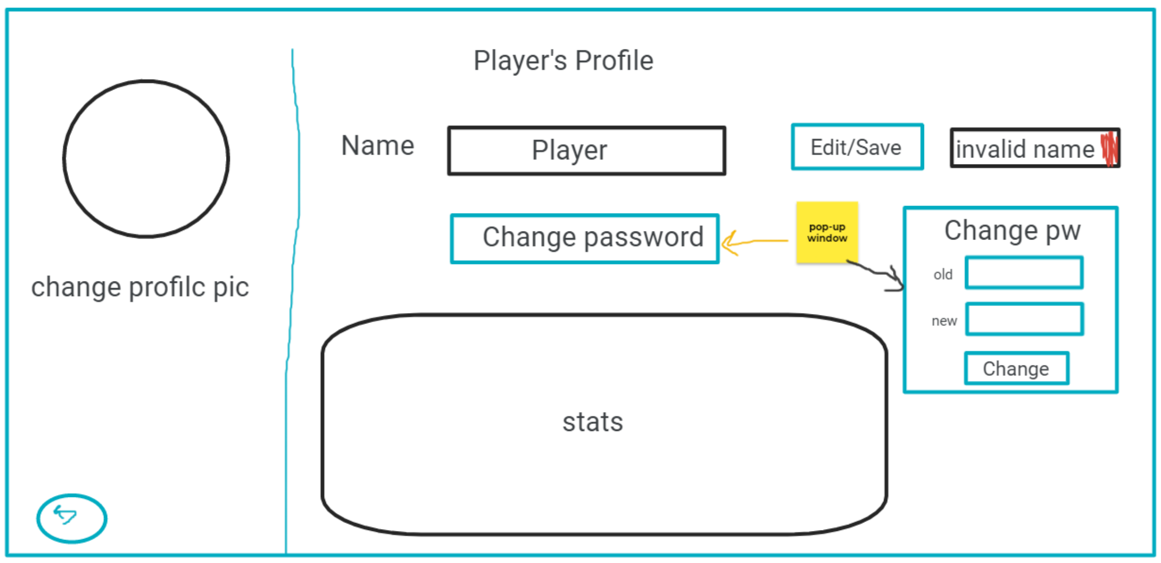
landing page



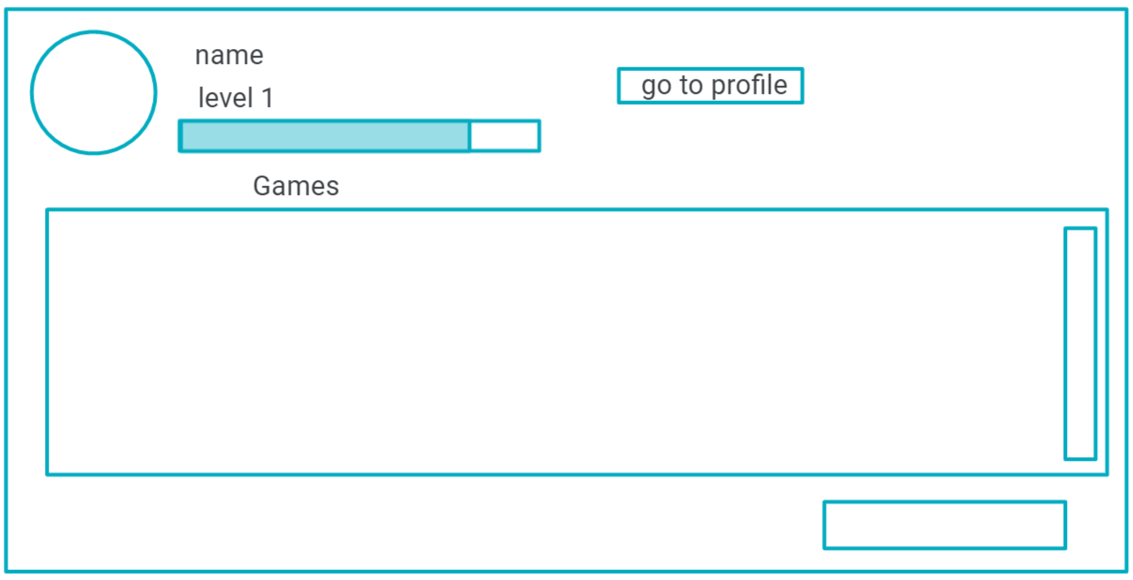
start page



profile



gamespace



- Requirements
  - Functional’
    - User is able to sign in using his username and password .
    - User is able to sign up using gender , username and password.
    - Registered users and guest users have different classes.
  - Non functional
    - User information should be secured only the one who accesses database is database manager with his name and id ->Security
    - Response time is reduced in this system as it interacts with users -> we reduced interaction with the database.
    - Simple interface which is easy to figure out.
- Sequence diagram
  - signup signin: Lucidchart

- User Stories

Sign In Story
User will request to sign in to the system as long as he has a username and password . He types his name and password and clicks sign in . if the username isn’t in the database the system will show him an error message showing that username is invalid . If the username is valid but the password is invalid so also system shows him an error message saying that he entered invalid password . if the user name and password are correct so he can get to the system .

Sign Up Story
User will request to sign up to the system . He types his username he wants , password he requires and selects his gender . When he finishes that he will click on sign up . if the password he chose is less than 8 characters the system will show him an error message showing that password is invalid . if a user typed username which already exists in the system so the system will show him an error showing that name is invalid . if both username and password are valid so he can enter the system .

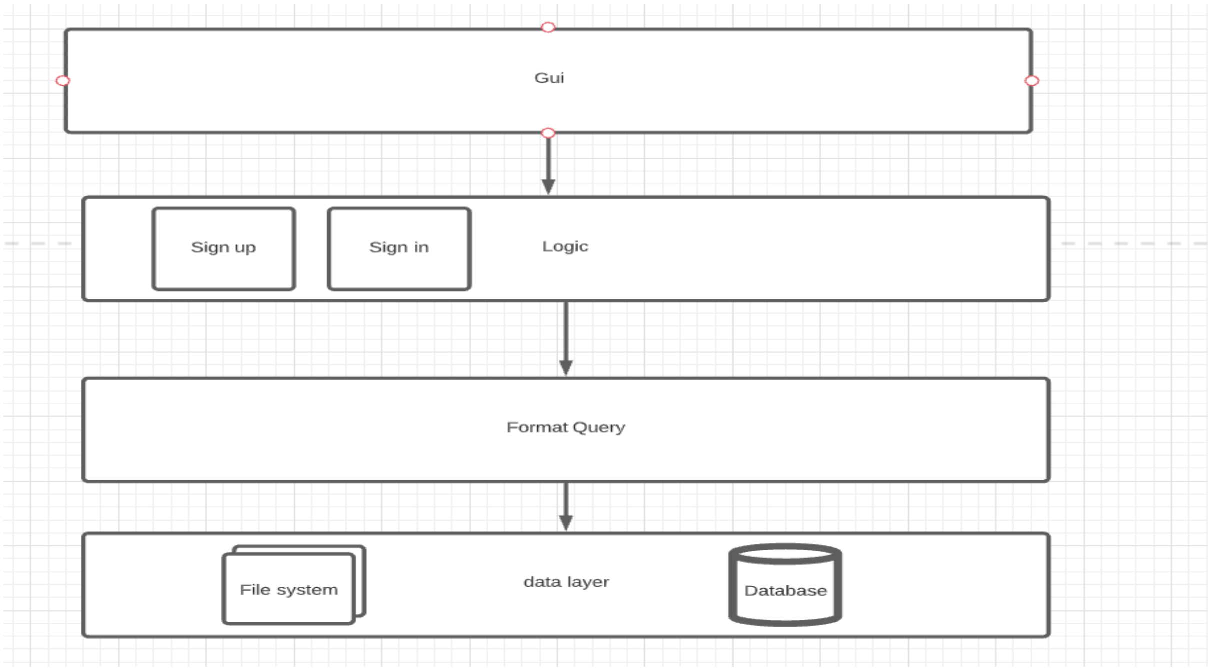
- Testing

  We have developed 2 unit testing for this phase:
  - DB_unit_test: it tests the different behaviour of the application when it encounters valid input , wrong input , none input , empty input or input of other types.

    It also tests that dbmanger must be one instance in all the application and has limited access from other classes

    - Auth_unit_test: test the functionality of sign in and sign up

- Changes
- In code
    - We started to separate applications into different layers



    - We removed player factory and made user player extend guest player as it has functionalities of guest player in addition to other functionalities such as get achievements , get weekly xp, get daily challenges , get number of wins , get number of played games and get rank .
    - We used the Facade design pattern to link Frontend with Backend so that they can only communicate through facade.
    - We made the class sign in , sign up as proxy classes they ask database for query by the database manager and they have methods to validate name and password

    In case of sign in the sign in class asks database to get user with this name if found the database will returns it else it will return empty tuple the sign in class will validate if the data got from database is empty tuple or not if it is empty tuple so user entered invalid name else it compares the password got in the tuble with user entered password if they are equal so allow user to access system else show him error message .
    In case of sign up : the sign up class asks the database to add user using name , password , gender if name existed before in the database so it will not allow user to enter system as this user it shows him error message showing that he entered invalid name . if everything is correct so it allows him to enter the system as this user.

- In database
    - In the design phase , we decided to use file system as storage for the data of our application but it turns out that using a database for some of the data would be more appropriate and save time. A sql database is used to store (in this phase):
        - Information of users
        - List of available quotes
    - We also use file system to store the icons , backgrounds and sounds used in the application

- Current schema

User_info

name	password	salt	gender	avatar	level	xp	weekly_xp	wins	Games daily_ch

Quotes

Id	quote

- Trello link