- **Sprint summary**

  In this sprint, we have added new features beside refactoring and reviewing the old ones.

  Regarding Backend, we have implemented:

  - **Achievement models:**
    - daily_challange_achievements
    - games_achievement
    - level_achievement
    - wins_achievement
    - xp_achievement
  - **add backend-layer**
    - facade: responsible for the communication between different GUIs and backend logic
    - getter: get instance of db_manager and keep access token , get random quote and get leaderboard
    - updater: save changes made in player data to database
  - **modify player model**
    - add achievement list and methods
  - **modify db_manager**
    - achievement methods
    - leaderboard methods
    - fix some issues

  Regarding Frontend, we have implemented:

  - refactor old pages
  - achievements_page
  - leaderboard
  - add achievements button to profile page
  - add leaderboard button to gamespace page

        - add new pages to navigation
        - fix loading progress in landing page
        - add exit button to start page
        - add game gui and integrate it with pyqt

- **Requirements**
    - Functional
        - User is able to :
            - view achievements
            - choose and start game
            - view leaderboard
            - change profile avatar
            - change account name
            - view his stats
            - earn xp and weekly xp from games against computer
        - Game should support 3 levels of difficulties
        - Game should support two modes:
            - player vs player
            - computer vs player
        - xp earned from a game is relative to player's score
    - Non functional
        - landing page loads within 3~5 seconds
        - game is responsive to user actions within 200~400 ms
        - fetching from database happens in reasonable time (~ms)
        - game gui is easy to figure out within 2~3 minutes
        - optimize communication with database

- **Use case diagrams**
  - Game use case link
  - profile use case link
- **Sequence diagrams**
  - gamespace flow link
  - play game link

- **Testing**
  - refactor and re-organize old tests
  - test singleton of database
  - add testing of achievements and check if it is saved in database
  - test player model functionalities
- **Changes In design**
  - We didn’t make a folder for each player with his completed achievements. We made achievements saved in the database table with a boolean variable checked , this variable = 0 if he didn’t complete this achievement otherwise 1.
  - User information isn't in his folder , They are in a database table called user info , It contains all users with their information .
  - We made an avatars folder but not a folder for each player. Avatars folder contains every user's current avatar saved in it , Every user avatar is named by user name.

- **Changes In code**
  - Being able to deal with multiple classes asking to have facade functions at the same time . We made facade singleton as we need only one instance of it and also the facade class assigns the player object if we had more than one facade so more than one will change player object.
  - We made the database can only be accessed by the DB manager which is made singleton . We need only 1 DB manager .
  - We made a table in the database for achievements which are added to each player once he signs up .
  - Once a player signs in or up all its achievements will be loaded in his object .
  - We linked game code to our application -> We integrated game gui with pyQt and made GUI interfaces to link game and application .
  - We made different types of achievements classes ; each one will be notified whenever change occurs in the user object . every achievement type will only care about the thing which it is responsible for .
  - Saving in the database only at the end when the user clicks on exit to decrease response time when dealing with the user otherwise if the user asked for showing his achievements for example it will be viewed from the player object .
  - we linked between the achievement page , leaderboard page , game pages and other pages we made in the previous sprint.

- When a user changes his avatar from his computer this avatar is copied from the path he wants to the Avatars folder and named by his name . This new path is the path added at his tuble in the database table. So if the user deleted this photo from his computer path it doesn’t mean it should be removed from his account .
- When a user is first signed up with no avatar there is a default avatar made for this case .
- We added the functionality of getting random quote at the beginning but in the backend only we didn’t call it from frontend yet.
- We added the functionality of viewing user stats whenever he enters his profile Gui page . And linked this functionality with the backend.
- We made the progress bar in game space changes by changing user xp .
- We made level number changes as soon as he finished the xp needed for this level .
- Every level needed xp is calculated by function in level number .
- We added functionality of changing username if it is a valid name and unique in the database.
- User xp changes whenever he plays a game with computer according to his score and also he weekly xp is affected it is also changed .
- Adding functionality of viewing leaderboard by getting top 10 users with respect to their weekly xp . and linking it with the frontend.
- Adding 2 modes for game -> playing with computer and playing with another player .
- Adding 3 levels of difficulties to the game : Easy , Medium and Hard .

- Current schema

User_info

name	password	salt	gender	avatar	level	xp	weekly_xp	wins	games	daily_ch

Quotes

Id	quote

game_achievements

id	description	type	goal

player_achievements

player_name	achievement_id	checked

- Jira link