SENG 300 – Project Plan

Authentication & Profile

Overview

To ensure users have the best possible experience with our gaming platform, we need to make sure that the authentication experience is simple and effective. When they first open the application, they have the option to either login or sign-up. We collect basic information from them such as email, username, and password. All of these values are verified to ensure they are valid entries. These, along with other preferences – such as notification and privacy settings – can be confirmed and edited under the user profile. This is also where the user can edit their friend-list by either adding or removing friends, and blocking players they no longer want to interact with. This document outlines our initial planning of these details.

Core Responsibilities

- ★ **User Authentication**
  - Implement secure login and registration.
    - SignUp
      - Username (Unique)
      - Email
      - Password
      - Confirm Password
      - Date of Birth
      - Are you human? (Emoji with multiple choice)
    - Login
      - Username
      - Password
      - Stay logged in option
      - Contact Us
      - Forgot Password
      - FAQ
  - Support authentication via email/password
  - Potentially enforce security measures like password hashing and account lockout after multiple failed attempts.

- ★ **User Profile HomePage**
  - Enable users to create and edit their profiles.

- Store and display user stats, such as total games played, win/loss record, ranking, and favorite games.
- Provide visibility into other users' profiles for community engagement.
- Allow Friends list to be public or private.

★ Profile Privacy & Settings

- Username
- Password
- Location
- Personal details
  - Contact information (email)
  - Date of Birth
  - Account created on
- Avatar (Choose from options provided)
- Privacy Settings/ Preferences
  - Account visibility (public/private)
  - Friends list (hide/show/edit)
  - Game Inventory
  - Game details (match with friends only/match with anyone)
  - Visibility controls for match history and ranking
- Language
- Help

★ Game History & Statistics

- Track and store match history, including opponents, results, and timestamps.
- Allow users to view past games – (Optional).
- Sync profile stats with the leaderboard system.

★ Friend & Social Features

- Implement a friend system where users can add/remove friends.
- Show online/offline status and current game playing status of friends.
- Allow players to challenge friends to matches.

★ Session & Security Management

- Implement session management (logout, session expiration, remember-me functionality).
- Ensure security best practices, including password principles (alphanumeric, minimum of 8 characters, upper and lowercase, symbols, etc.).
- Provide account recovery options (password reset via email).

★ Integration with Other Teams

- Leaderboard & Matchmaking Team: Sync player statistics, rankings and matchmaking settings.
- Game Logic Team: Retrieve and update match results for profiles.
- Networking Team: Ensure users are correctly validated and can access the network for all game sessions.
- GUI Team: Provide UI elements for profile pages, settings, signup and login.

Timeline

March 6 - Iteration 1 Complete

March 13 - Completion of code for Authentication

March 18 - Completion of code for Editing & Deleting Profile

March 20 - Iteration 2 Complete

March 25 - Completion of code for Friend & Social Features

March 30 - Completion of code for displaying user Game Stats

April 4 - Testing done for Authentication & Profile

April 10 - Completion of integration with other teams

April 10 - Iteration 3 Complete

April 11 - Submit Final Project

Networking

Use Case Diagrams

Shows the details of connecting to the server, player interactions, and updating the state of the games. To be combined with other parts of the project.

Main:

- Players join a game lobby.
- Player sends/receives chat messages.
- Game state updates are synced accross all players.
- Server handles matchmaking and player assignments.

Use Case Description

Descriptions of user interactions with the server. Describing how all the parts of the game connects to the server.

★ Player Actions:

- Logging in and establishing a connection with the game server.
- Sending game actions (eg. moving a chess piece) to the server.
- Receiving updates from the server (eg. opponent's move).
- Chatting with other players in a game session.

★ Server Responsibilities:

- Managing active game sessions.
- Handling real time interactions, including in-game messaging.
- Ensuring that the game state is consistent across all players.
- Managing Matchmaking and queuing players.

Classes

We will have an abstract class "Network Manager". Two classes will extend from this class:

★ GameServerManager

Responsible for:

- Player interactions such as messaging and chat feature in the game.
- Updating the state of the game.

Used for Game Logic and Matchmaking parts of the project.

- Completion date: Before Iteration 2

★ DatabaseManager

Responsible for:

- Connecting to the server to access stored information.

Used for Authentication and Matchmaking & leaderboard parts of the project.

- Completion date: Before Iteration 2

★ NetworkManager

Responsible for:

- Handling server communication, ensuring smooth data transmission between clients and server.

Provides a base structure for subclasses to implement specific networking tasks.

- Completion date: Before Iteration 2

Class Structure Diagram

Illustrating classes that will be combined with other parts of the project. Will illustrate how NetworkManager, GameServerManager, and DatabaseManager interact with other parts of the system.

Includes connections to external components (game logic, GUI, and Authentication modules.)

GUI

Overview

This document outlines the planning details for the GUI team before P2, including use case diagrams, interface designs, and system structure.

Use Case Descriptions

The system features a leaderboard to view top-ranked players that are shown when a game is selected and track personal and friends' rankings. Players get notifications for match invitations and can accept or decline. Users can access personal stats like ELO, win/loss ratio, and compare with friends. The matchmaking system helps players start or join matches with similar ELO ratings. Games like Tic Tac Toe, Chess, and Connect 4 have interactive, turn-based mechanics. A chat system supports in-game communication with message notifications and session-based history. The game menu lets users browse, view descriptions, and join/create matches.

Diagrams

Use Case Diagram
Class Structure Diagram

Interface Design

★ Profile

- Be able to create an account and sign into the account

★ Main Menu

- Provides access to the games and player stats
- Displays notifications for match invites.

★ Leaderboard Display

- Shows top-ranked players for the selected game based on ELO.
- Displays personal ranking.

★ Chat Interface

- Enables in-game communication.
- Provides notifications for match invites.

Tasks & Responsibilities

★ GUI Team

- Implement a way to display the top player rankings based on data from the leaderboard.
- Develop a notification system for match invitations.
- Ensure integration with matchmaking and leaderboard data.
- Work with the leaderboard team to retrieve player stats efficiently.

★ Leaderboard & Matchmaking Teams

- Provide data for top player rankings.
- Ensure ELO updates are correctly reflected in the leaderboard.
- Pass match invitation data to the GUI for notifications.

Completion Date

The GUI implementation mentioned above should be completed before Iteration 2.

Leaderboard

Overview

Our primary goal is to implement both a match-making algorithm that is fair and runs in a timely manner, and making a leaderboard to show the top players.

Core Responsibilities

★ Coding ELO algorithm
- Making an algorithm that returns a signal number that represents the skill of a given player (this will be public)
- This algorithm should be fair to bot player in the match and should account for if there is an ELO difference
- The ELO algorithm will use the following data to get the new ELO for the player
  - Who won the game
  - The number of moves that were played
  - How long the match lasted

★ Making matchmaking using the ELO rating to match players of similar skill levels

★ Coding a leaderboard to show that top players with the highest ELO rating

★ Challenge a player directly with similar ELO that is also online

★ Start a match and wait for a player to join in a lobby

Timeline

March 6 - Iteration 1 complete

March 10 - Game Logic collecting and passing data to Leaderboard

March 13 - Process data that is collected from Game Logic

March 15 - Work with Game Logic to get started on the matchmaking

March 17 - Work with GUI to get the main ranked leaderboard set up

March 20 - Getting the ELO based matchmaking set up more

March 20 - Iteration 2 complete

April 10 - Iteration 3 complete

Game Logic

Overview

Our primary goal is to implement demonstration games for OMG.

Core Responsibilities

★ Coding the game logic
- E.g. designing the algorithms for validating piece interaction, checking for win conditions, and collecting game data (NOT processing).

★ Unit testing for game logic.

★ Helping the GUI team interface with the game.

★ Interfacing between user profile and in-game player controller.

★ Making sure stats are properly tracked, as desired by the leaderboard team.

★ Helping the Network team interface with the game to send/receive game data.

Timeline

March 6 - Iteration 1 complete

March 13 - completion of Tic-Tac-Toe

March 15 - testing done for Tic-Tac-Toe

March 20 - Iteration 2 complete

March 27 - completion of Connect Four

March 29 - testing done for Connect Four

April 3 - completion of Chess

April 5 - testing done for Chess

April 10 - Iteration 3 complete