Dracula Theme DraculaTheme.com made a latex theme.

Chess Editor (daChess)

Askaban Dabmentors

April 1, 2021

1 Project Definition

Chess is a classic game that has been played by millions since its conception; be it as a board game or in modern times as a digital game. It is a game that has relatively simple rules that make it a game that has been a standard for testing the computation ability of automata. Although they have become advanced enough where chess is no longer sufficient[1], we feel it is a good start for testing ones' abilities. The goal of this project is to develop a chess application using the Angular framework and to test some concepts that we are interested in, such as User Authentication with Google Identity and the Unified Chess Interface[2] with an added bonus of trying to wrangle with an external web server for multi-player support.

[1] https://openai.com/projects/five/

[2] Current development has this pending, may have to change.

2 Project Requirements

2.1 Constraints

There are two constraints imposed on this project.

2.1.1 Deliverables

The project has the following deliverables that are required for success:

- This Report
- The Program
- The Poster

2.1.2 Time Constraint

This project is to be finished by April 22nd, 2021[3].

2.2 Functional Requirements

The functional requirements of the project include minimum requirements and those which we would like to implement, the latter will be diliminated by a star *:

2.2.1 Primary Requirements

The primary requirements are those which are to always be functional:

- Ability for the user to play a game:
  - Against another user locally.
  - Against an *AI* that will be provided.*
- Ability for the user to track their *statistics* of play over the lifetime of their account including wins, losses, and draws.[4]

[3]Needs to be verified

[4]This does not include syncing which is a secondary requirement.

- Ability for the user to modify the starting layout of the chess pieces to match a configuration of either a previously ongoing game or a game scenario.
  - FEN Processing
  - FEN Interpretation
  - FEN Generation

2.2.2 Secondary Requirements

Secondary requirements are those which will not always be available.

- Ability for the user to create an account to access online features.
  - Ability for the user to sync statistics.
  - Ability for the user to play against another user online.
- Ability for the user to login via their Google Account credentials.

2.3 Usability

2.3.1 User Interface

2.3.2 Performance

2.4 System

Serverside All server-side requirements of the this project were provided by eco.webhosting.co.uk

The requirements of the server include:

- Web Server setup to run Angular applications
- Database running MySQL 10.4.14
- Can handle multiple requests and sessions at a given time.

Clientside The client side requirements are recommended requirements based on our own systems:

Operating System This program does not require a specific Operating System.

Memory The system should have at least 2GB of RAM.

Required Software This program requires a modern GUI web browser, preferably:

- Firefox
- Google Chrome
- Brave

2.5 Database

This project will be using one database to keep track of ongoing games which is using MySQL version 10.4.14.

2.6 Networking

All project artifacts and interconnectivities are being provided by the hosing web server.

2.7 Security

For Google Authentication, we will be leveraging Google OAuth 2.0 API. The other security requirements are inherently provided by the hosting server.

3 Project Specification

3.1 Focus

The focus of this project is to gain experience with Angular and Node.js[5]. Where applicable this project will attempt to utilize open-source projects and existing systems when achievable.

FEN We are using FEN (Forsyth Edwards Notation) to keep track of the positions and moves on the board. This is supported by similar applications and programs that could enhance the application such as Stockfish (w/ UCI) if implemented.

3.2 Platform

This project is developing a web application. The application will be *portable* to devices that have an acceptable screen size.

3.3 Genre

This project is developing a MMO*[6] board game.

3.4 Target Audience

The target audience for this project is for competitive chess players to test moves. This means that the program will need to implement all the chess rules properly and that the systems for processing and interpreting chess moves of import.

3.5 Development Environment

Devin Uses vim(neovim), emacs, Bluefish, angular cli, and Firefox Developer tools.

[5]Devin didn't know about Angular until this project, and only used NPM to download software.

[6]Optional Requirement

3.5.1 Libraries

The project utilizes Angular Framework and Google Identity API (User Authentication), both have their own dependencies which are imported.

3.5.2 Frameworks

This project is developed using the Angular Framework (11.1.2) with TypeScript (and CSS) along with Bootstrap CSS Framework.

4 System Design

4.1 Subsystem Identification

The subsystems of this project are broken up base on their function and dependencies, this is as follows: Editor Subsystem, Chess Subsystem, Board Subsystem, AI Subsystem*, Authentication Subsystem*, Network Play Subsystem*, Statistics Subsystem, and Lobby Subsystem*.

The Editor and Chess subsystems both depend on the Board subsystem, but have functionality that are distinctly different in purpose; latter having rules and turns while the former has has the ability to manipulate the board without such constraints.

The Authentication, Network Play, and Lobby subsystems* are dependent on the web server. There is a distinct flow from authentication to lobby to network play.

Statistics subsystem has its own unique requirements for storing the data that needs to be addressed.

AI Subsystem is optional, and while works with the chess subsystem, is not always there so should be considered as a separate system.

Chess Gameplay

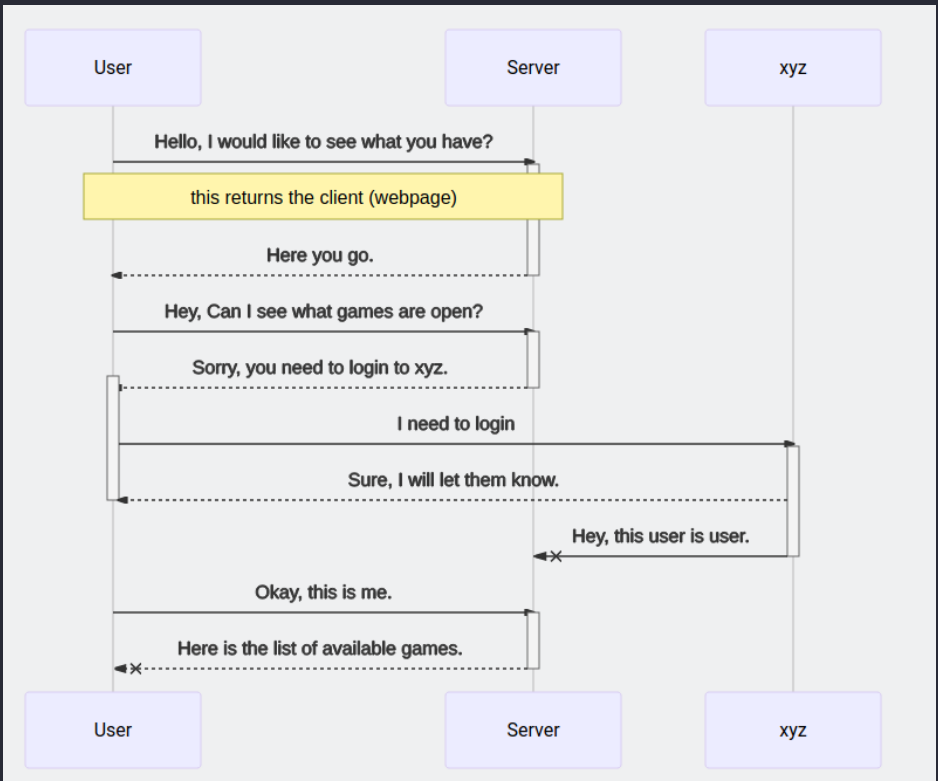
4.2 Subsystem Communication

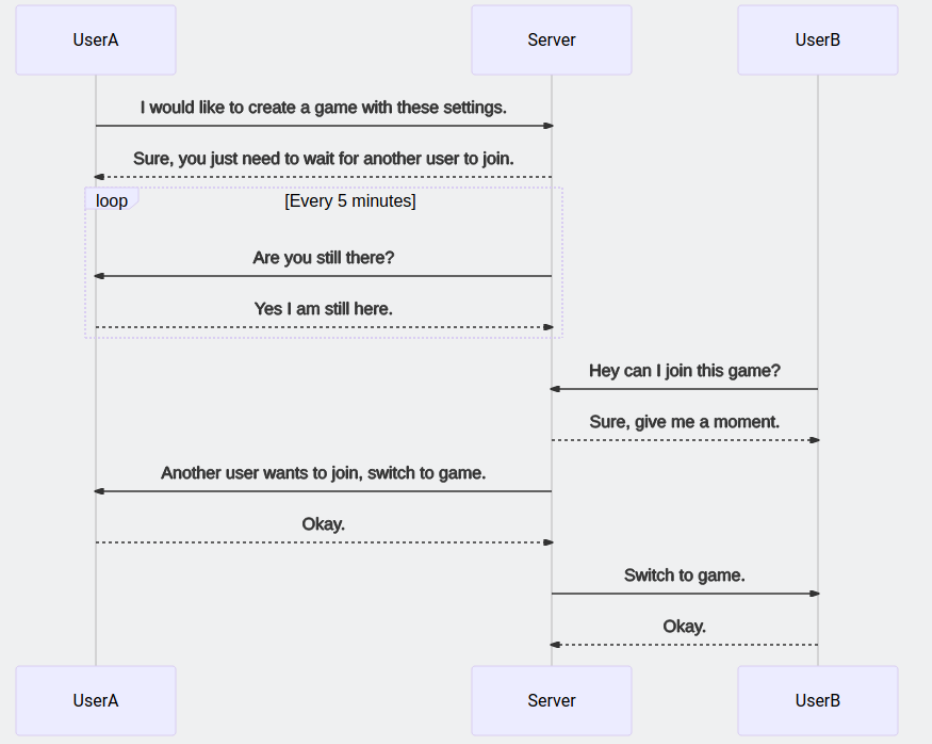
Game Subsystems The game subsystems (AI*, Chess, Editor, Board) have minimal interaction with other subsystems. When they are in use, events from the board are passed to either the chess or editor subsystem to be processed and the outcome is given to the board. The AI subsystem will interact with the chess subsystem when the players turn has ended, which it will be given a FEN string, and return a FEN string with its move.

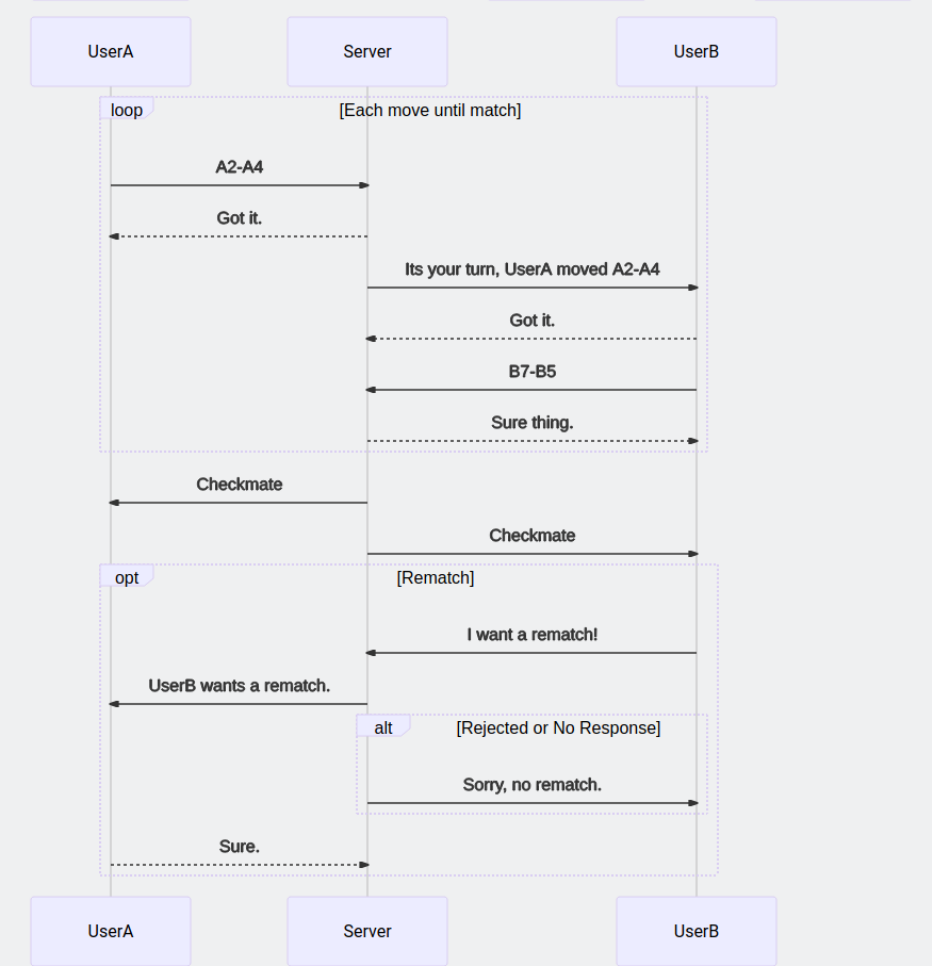
Network Subsystems These subsystems as mentioned earlier, have a flow to them in which authentication leads into the lobby subsystem which leads to the network play subsystem.

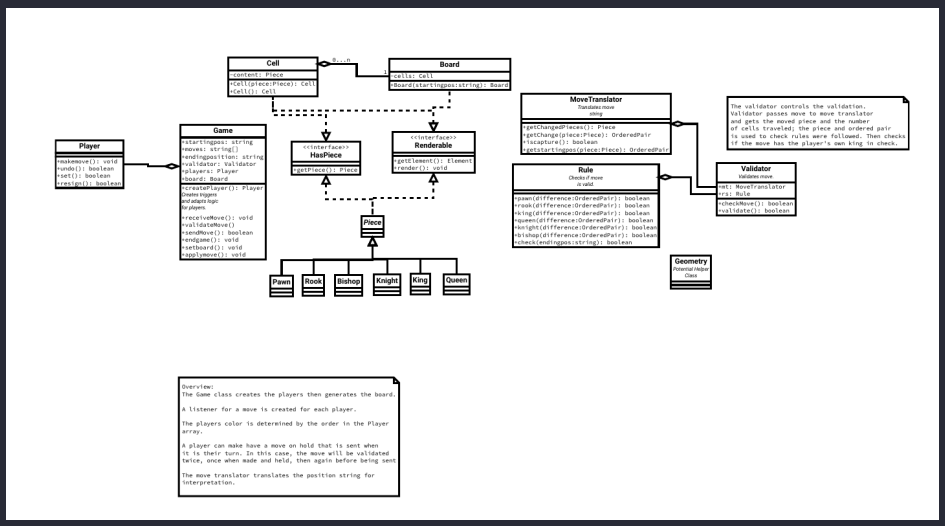
Statistics Subsystem This subsystem is unique in that it has its own unique interactions with the Chess Subsystem and the Authentication subsystem.

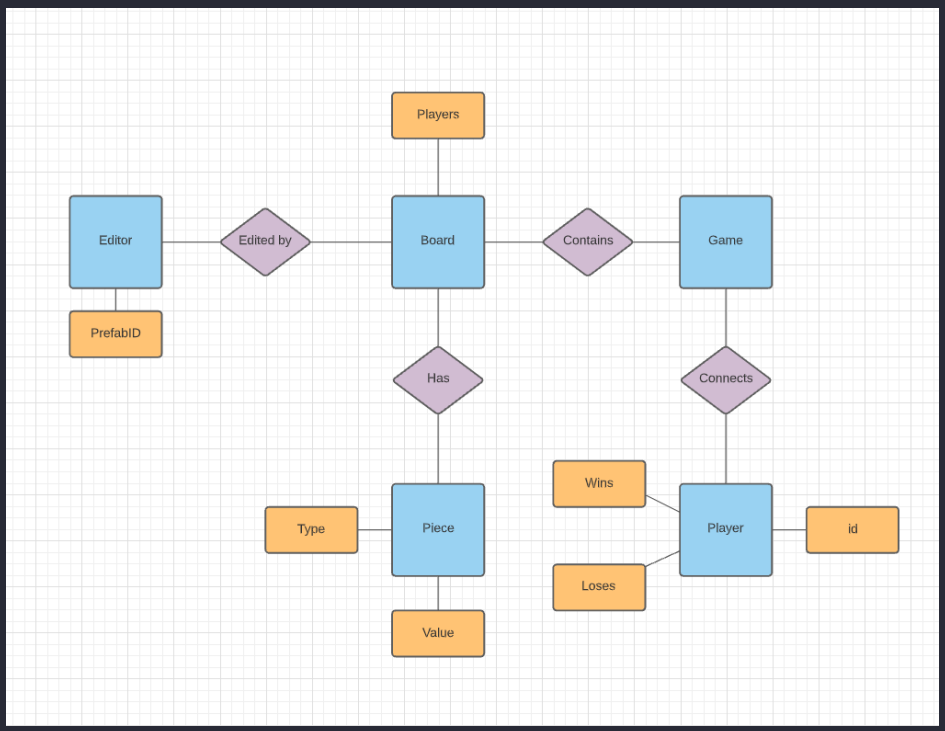
4.3 Diagrams




UserA
Server
UserB
I would like to create a game with these settings.
Sure, you just need to wait for another user to join.
loop
[Every 5 minutes]
Are you still there?
Yes I am still here.
Hey can I join this game?
Sure, give me a moment.
Another user wants to join, switch to game.
Okay.
Switch to game.
Okay.
UserA
Server
UserB


UserA
Server
UserB
loop
[Each move until match]
A2-A4
Got it.
Its your turn, UserA moved A2-A4
Got it.
B7-B5
Sure thing.
Checkmate
Checkmate
opt
[Rematch]
I want a rematch!
UserB wants a rematch.
alt
[Rejected or No Response]
Sorry, no rematch.
Sure.
UserA
Server
UserB





5 System Analysis