Milestone 5

Online Chess Application Documentation

CSC 667.01

Team #10

Github Repository: https://github.com/csc667-02-sp19/csc667-sp19-Team10

Steve Rentschler - Team Lead, Front-end Lead
Johnson Wong - Back-end Lead
Leonid Grekhov - Front-end Support
Kurtis Hoang - Back-end Support
Jiannan Li - Back-end Support, GitHub Master

Project Introduction

Our project is an online chess website called TEAM 10 CHESS. First you have to create an account with us to login. Once logged in, the site will direct you to the lobby.

Lobby page: This is where you can join a chess game to play vs another player or create your own game for someone to join. There is a scrollable column for available games and the current games you are playing. To join a game, click on the button next to the players name who created the game in the available games column. When creating a game, you will be able to choose preselected times for each player to make their moves. Once someone has joined the game you created, it will populate in the current games column, where you can rejoin the game. There is a chat window and a leader board that displays the top 10 ranking players along with their rating and wins/losses. There is a button in the leaderboard that will display all player's ranking that you can scroll through. The chat will only display the current chat while you are on the page without refreshing with players that are currently on the lobby page. There is a display at the bottom of the chat that displays how many players are online. When you create a game, you will directly go to the game room. The nav bar has a links to the "How to Play", the "About" page, and the "Profile" page. The About page was never implemented. It was in intended to be an about page of the team members and who they are.

Game room: When playing the game, your name will be displayed at the bottom of the game board along with your current ranking and your opponent's name will be displayed at the top of the board with their current ranking. The player who created the game will always play the white pieces and will move first. When making your move, you will have to drag and drop your pieces, then press the check mark to submit your move, or press the x mark to undo your move. You can only undo your move before you submit it. A time will be displayed for how long you have to make your move. If time runs out before you make your move, you will lose the game. You can offer a draw to the other player, or resign the game when it is your move. When offered a draw, you will have a choice to accept or to not accept from a popup. A players ranking will be calculated by using a true chess ranking system. There are 4 arrow buttons that will go throw the history of the game. On the right side of the board, there is a chat window that you can use to chat with the other player that will be saved even if you leave the room. There is also a history tab on the chat window that will display all of the moves in the game in standard algebraic form. And last but not least, there are two buttons that will switch the board between 2D to 3D. Once in 3D, you can rotate the board by pressing close to the board and dragging the mouse.

Profile page: This is where you can choose your board color and choose your pieces. There are 8 board colors, 2 pieces to choose from for 2D, and 2 pieces to choose from for 3D. To select your board color and pieces, select from the drop down next to them and then click on save settings. Your settings will be stored and when you return to the game room, they will be

displayed. There is also a friends search, where you can search for another player and store them in the column below the search to create a game with.

How to Play page: This page tells you how to play the game of chess and displays all of the legal moves with popup carousels.

Software Stack

Server Provider:	AWS
Database:	MySQL
Web Server:	Nginx
Server Side Language:	Javascript, version: ECMAScript
	Node.js, version: 11.12.0
Frontend:	Bootstrap, version: 3.4.0
	Handlebars, version: 3.0.2
Frontend IDE:	Visual Studio Code, version: 1.33.1
Backend:	Express, version: 4.16.4

Tools used for communication

Discord was the primary communication tool. In Discord, we had several text channels; 667-group-discussion, front-end, back-end, scheduling-thread, git-specific-req-problems, useful-link-front-end, useful-links-back-end, to-do, and issues-bugs. We also used it for calls during team meetings if we needed to. We scheduled all team meetings through the scheduling-thread. We were able to share files, code, and links through it as well.

Tools used for task management

Tasks were assigned on Discord, or in team meetings. Tasks we checked on through discord or in team meetings. Tasks were assigned based on priority and skill sets.

Build Instructions

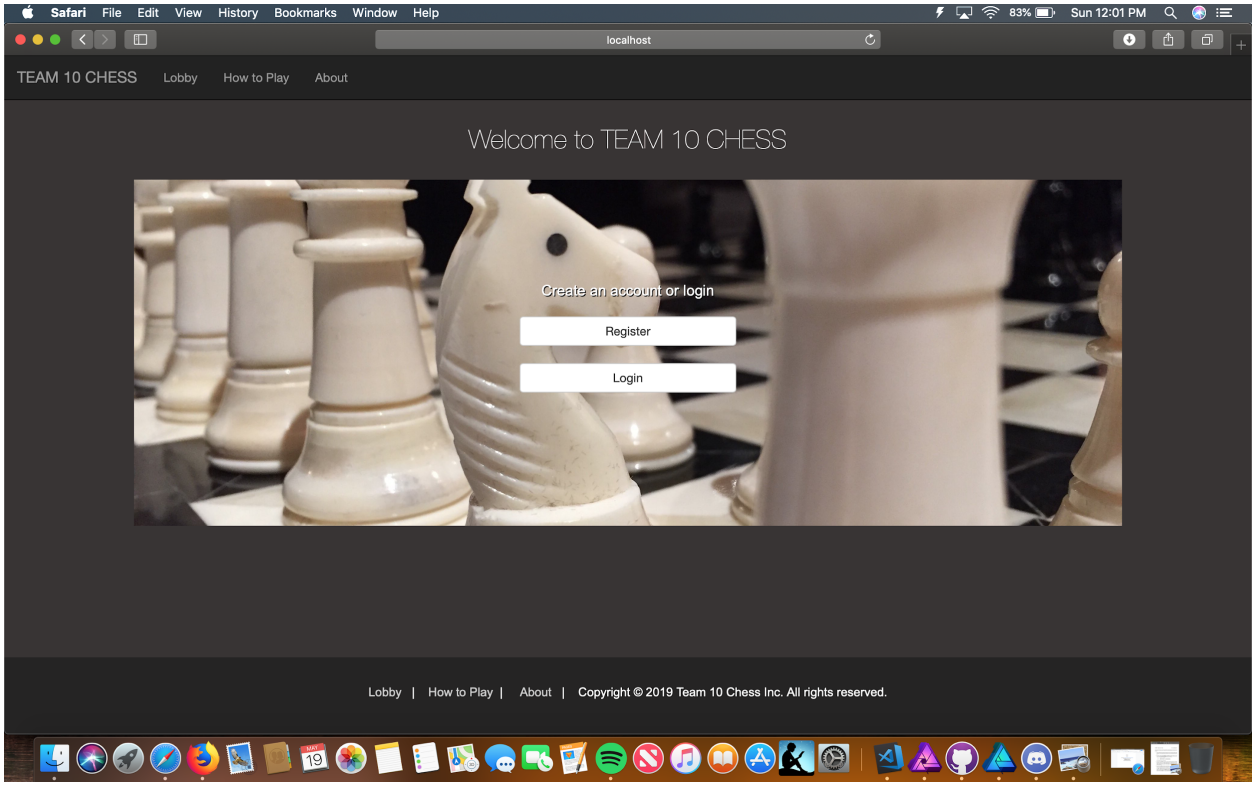
1. Download Node.js by going to https://nodejs.org/en/download/current.
2. In terminal or command prompt, type “git install”.
3. Then, type “git clone https://github.com/csc667-02-sp19/csc667-sp19-Team10.git”.
4. Now, go to the directory that the project is in and type “npm install”.
5. Then, type “npm start” to run the game server.
6. Go to http://localhost:8081.

If the remote server is down:

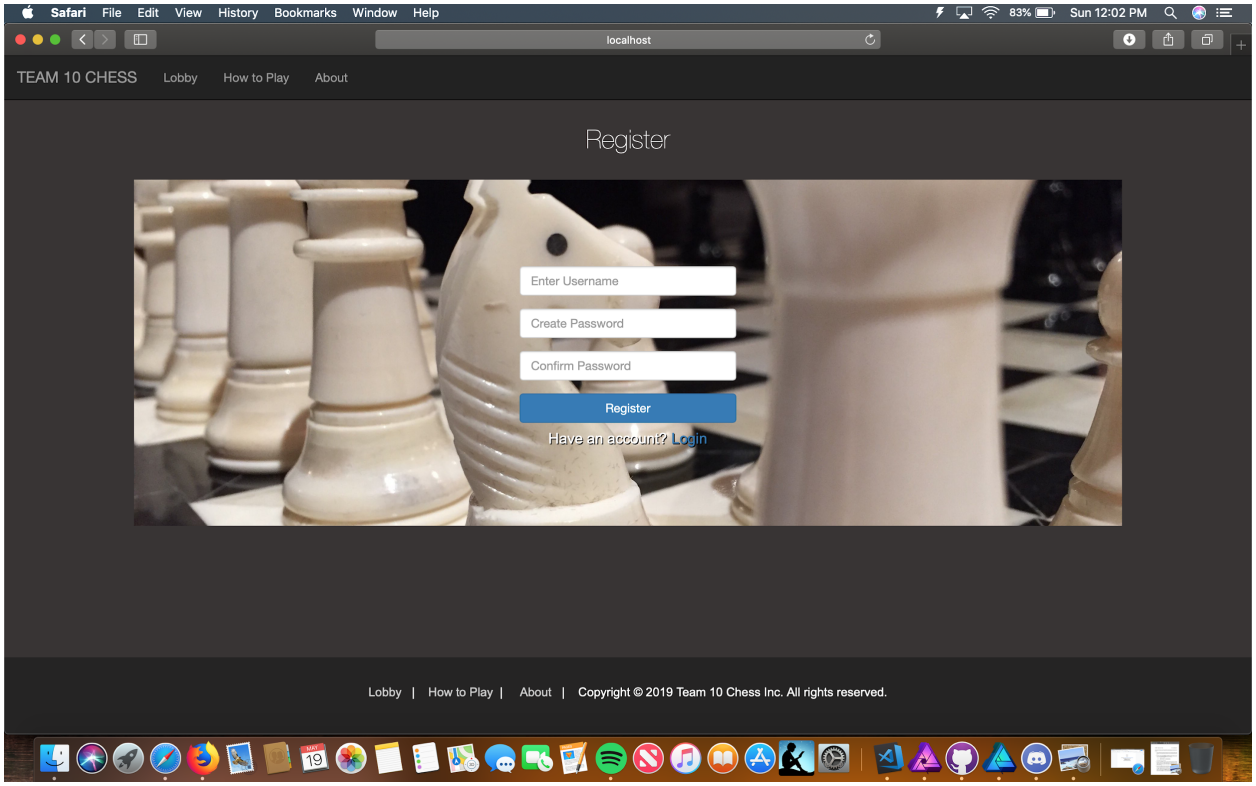
1. Admin need to use ssh key to access the elastic beanstalk server to do the deployment.
2. The original server and application are stored in Github. Admin will need those information to do the deployment.
3. The database information and table information are stored in the ../config/config.json and migration files separately.

Pictures of each page

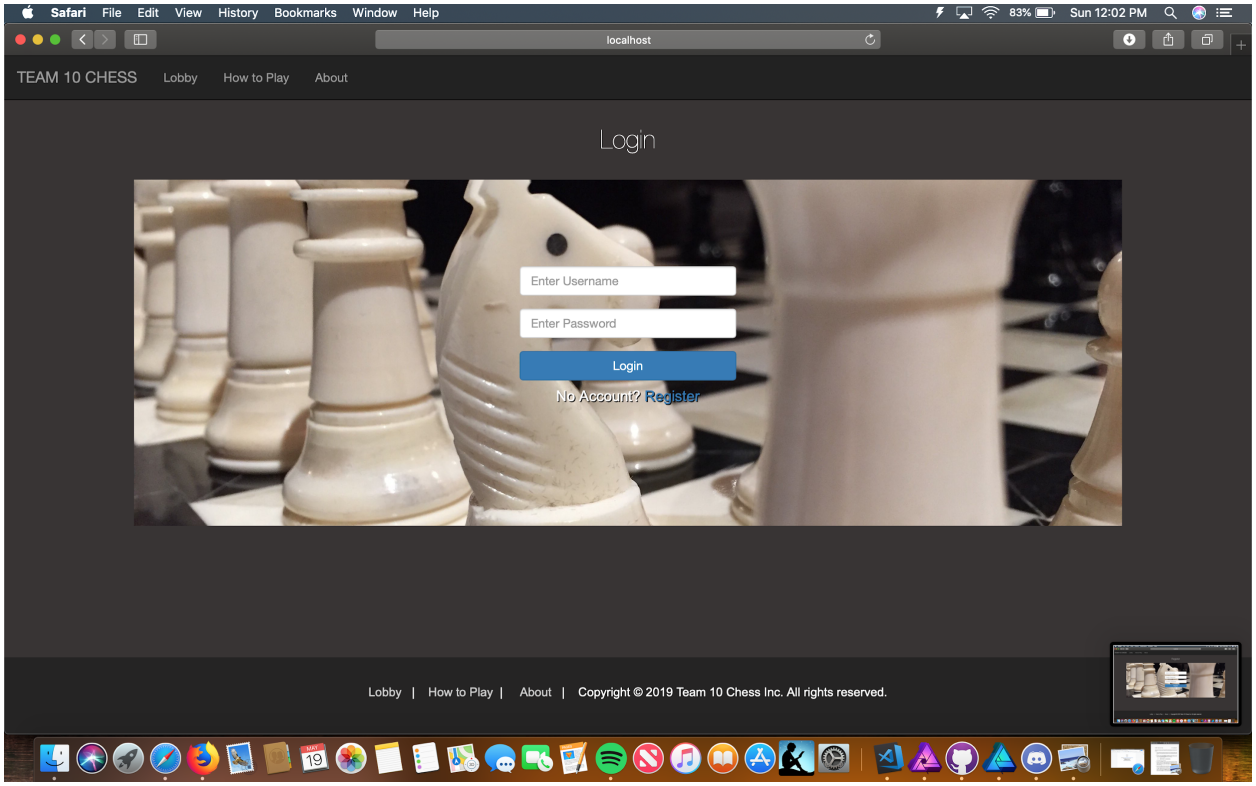
Welcome page



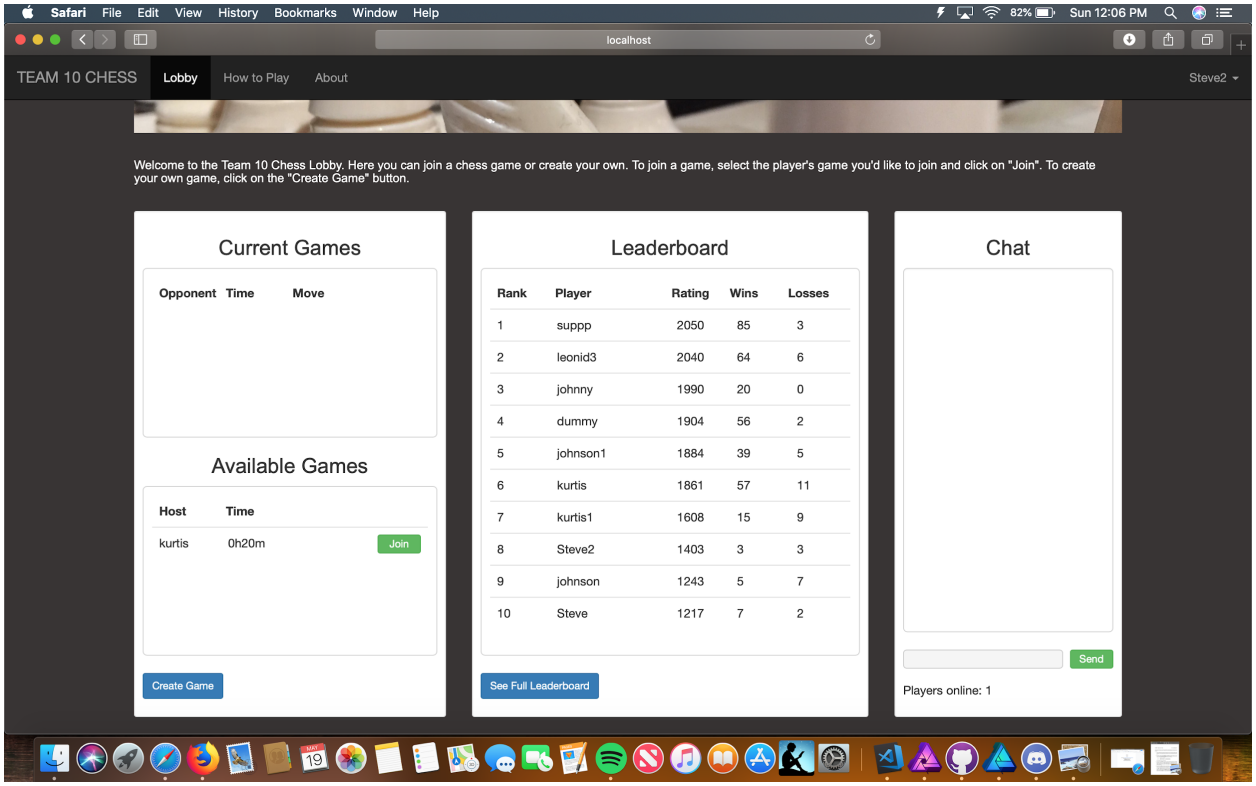
Registration page



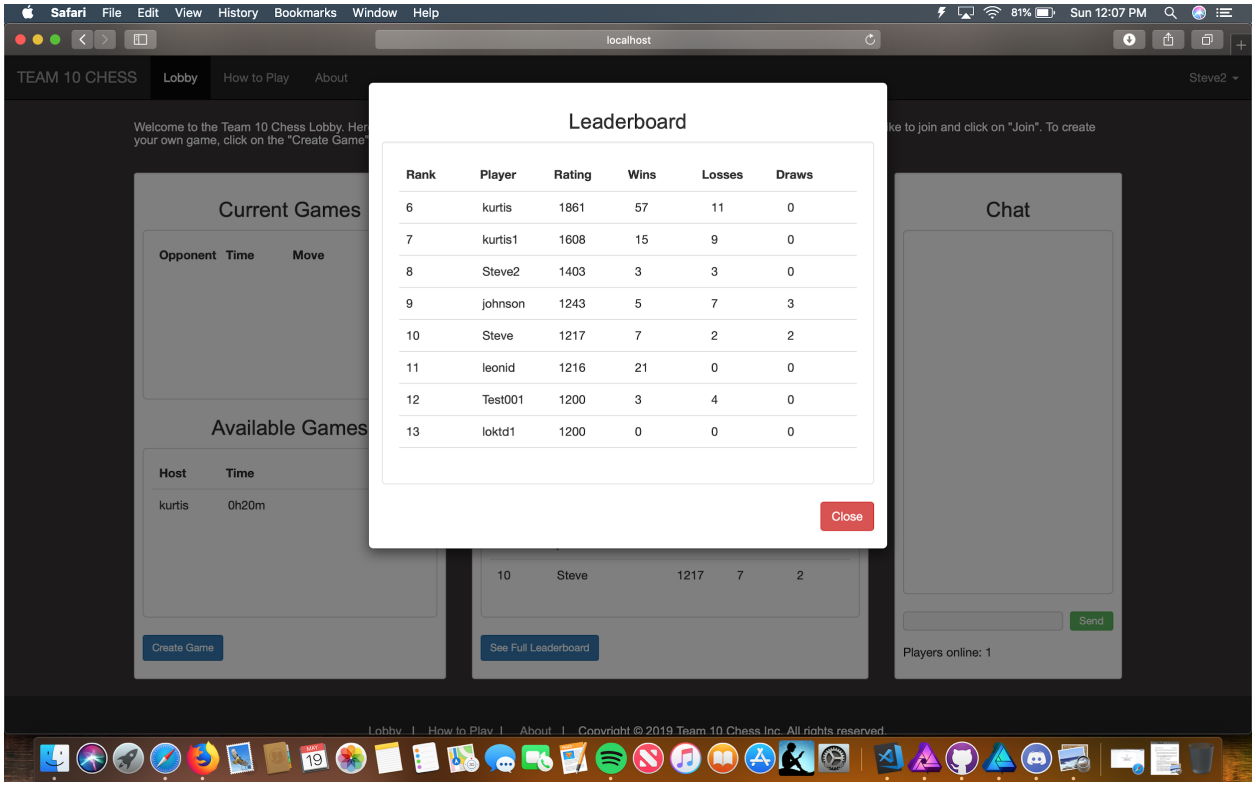
Login page



Lobby page

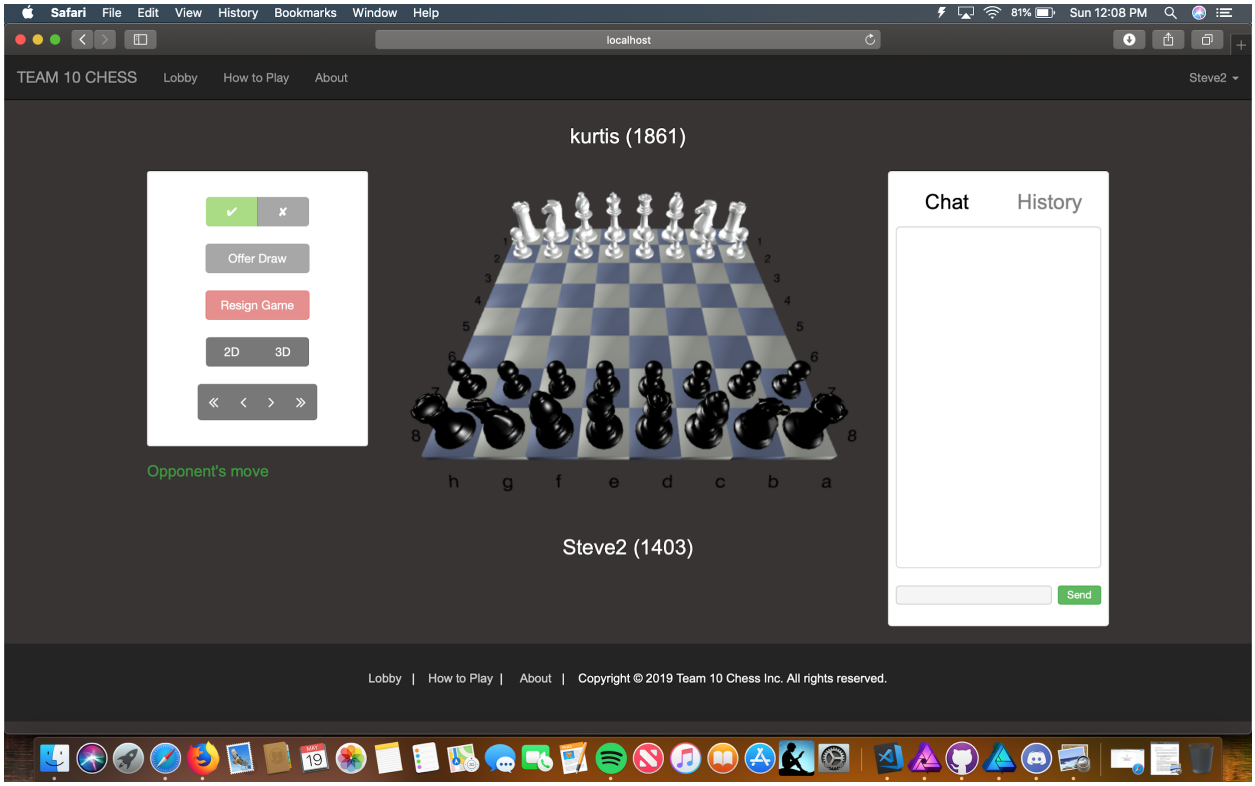


Leaderboard

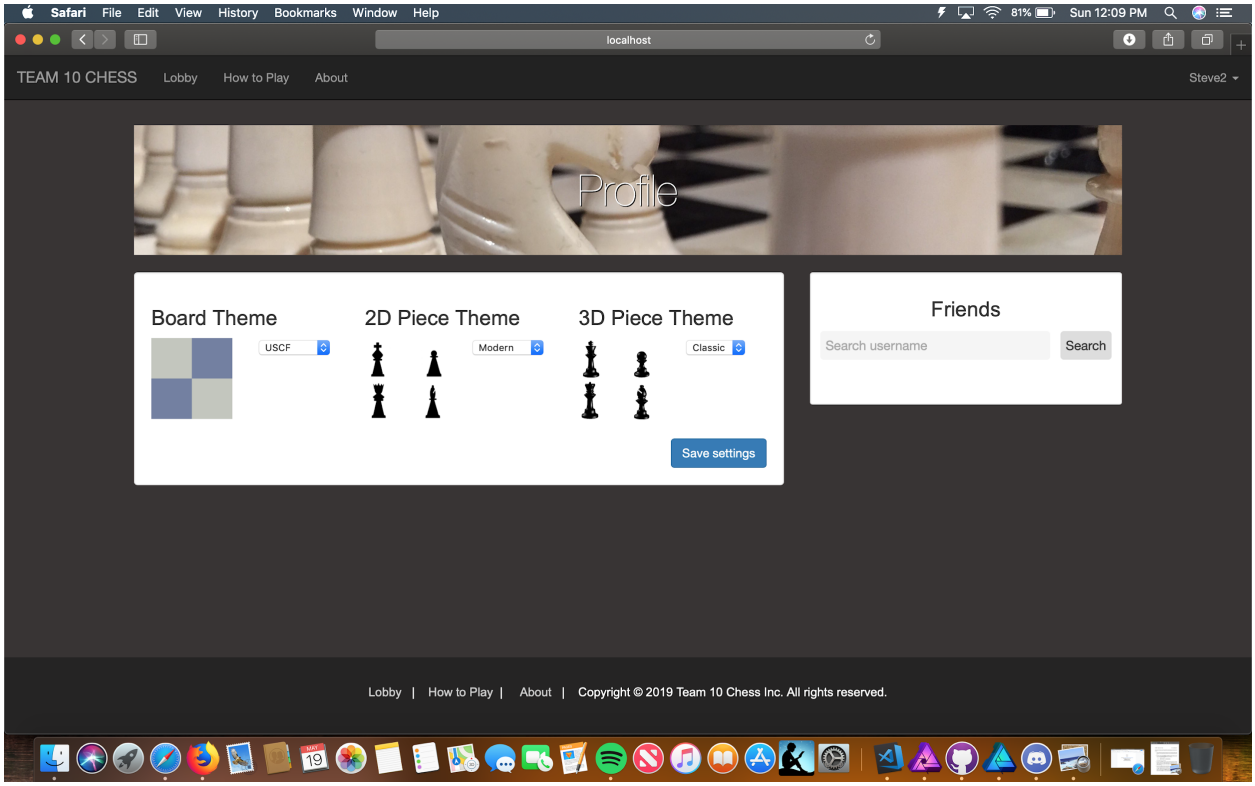


Gameroom (w/ 2D board)

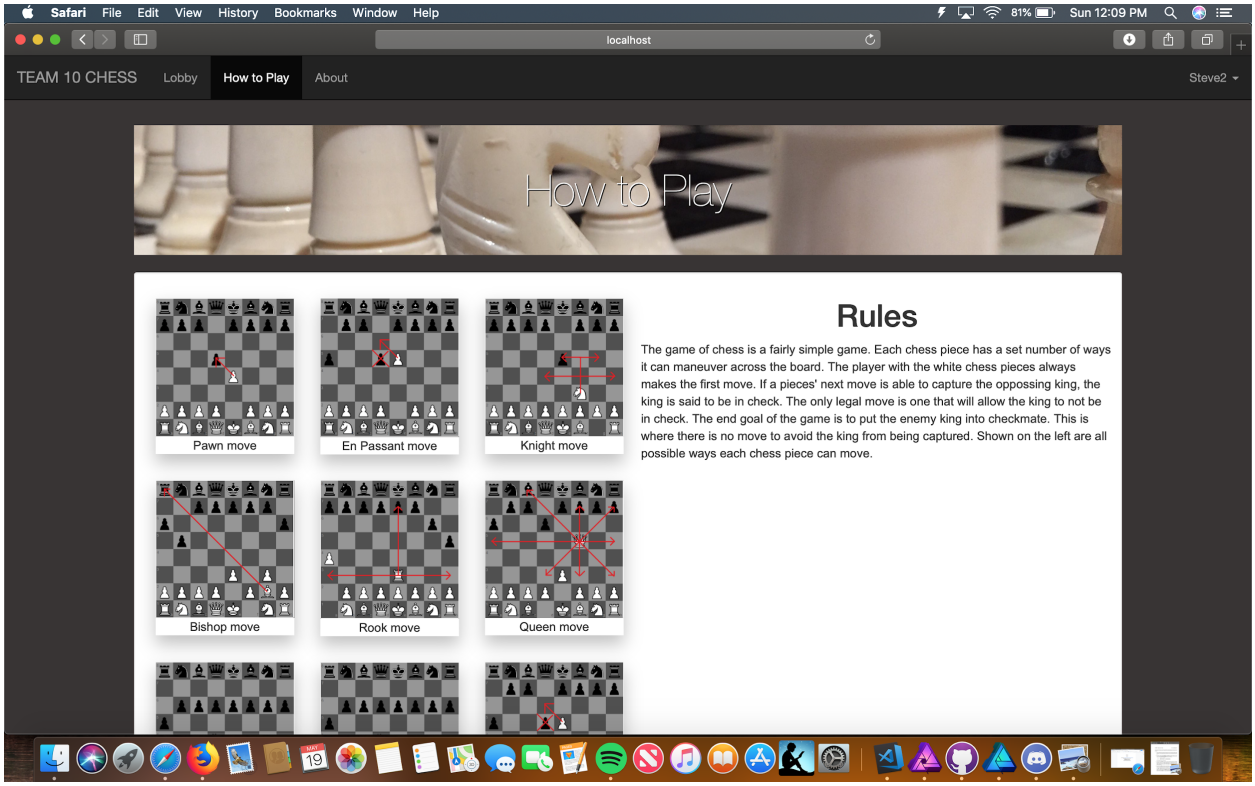
Gameroom (w/ 3D board)



Profile page



How to play page



List of API routes and their descriptions

- "/" - Render welcome page; register or login, forward to lobby if already logged in
- "/lobby" - Render lobby page, must be logged in
- "/howToPlay" - Render how to play page, must be logged in
- "/about" - Render about page, must be logged in
- "/users/login" - Render login page and handle login request and authentication
- "/users/register" - Render register page and handle register request
- "/game/:gameID" - Join user to specific game (gameID) and render gameroom page, must be logged in
- "/profile" - Render profile page, must be logged in
- "/profile/save" - Handle saving profile settings
- "/profile/search" - Handle friend search request
- "/profile/addFriend" - Handle add friend request
- "/profile/removeFriend" - Handle remove friend request

Team Member Contributions

Steve:

- Website flow and design
- Finalized software stack
- Lobby design, layout, and implementation
- Gameroom design, layout ,and implementation
- Welcome/login/register page design, layout, and implementation
- Design, layout, and implementation of profile page
- Implemented Chat/History tab switchability functionality
- History buttons implementation and functionality
- Graphics for Modern chess piece set
- Images for 3D pieces on profile page
- Graphics for Background colors on profile page
- Quality control of Front-End design and overall Website

Johnson:

- Created MySQL tables to store certain game entities
- Implemented functionality of creating a game
- Implemented functionality of joining a game

- Implemented functionality of rejoining a game
- Integrated chessboard.js and chessboard3.js API into game page
- Integrated sockets into chess API's for multiplayer functionality within a game
- Implemented functionality into gameroom buttons (submit move, undo move, offer draw, resign game)
- Implemented rendering board to a specific move using history buttons
- Assisted Kurtis with implementation and debugging of lobby and game chats
- Implemented history tab showing recorded moves
- Implemented time per move
- Implemented functionality of allowing user to customize board theme and piece set in profile page
- Implemented the functionality of displaying user's rank within game
- Debugged user login authentication and registration
- Quality control of back-end design, functionality and overall website

Leo:

- Assisted Steve with implementation of login/register page
- Assisted Steve with implementation of lobby page
- Assisted Steve with implementation of game room page
- Implemented front-end design of the customization of board and chess pieces on the profile page
- Implemented and designed how to play page

Kurtis:

- Helped Johnson with implementation of the login
- Helped Johnson implement the creation of game to the frontend
- Functionality of lobby chat
- Functionality of gameroom chat
- Functionality and implementation of ranking system

Jiannan:

- Initialized database schemas, AWS server settings and Express.js settings
- Tested integration of API's and suggested software stack
- Functionality of login/register
- Functionality of Friends search in profile page and create game between friends
- Dealing with merge conflicts and GitHub branches control
- Pushing code to the server

Project Reflection

In the beginning, we had a very minimal understanding of how our application was going to work, so it was not easy for us to pick a software stack that would work well with our project. As we got a better grasp and understanding of our project, we were able to determine a stack that would mesh well with our application.

Always pushing ourselves to be ahead of the deadlines gave us an edge to fully implement everything we wanted to accomplish with all 3 priorities.

Funny enough, we did not actually incorporate Handlebars until the very end of our project because we simply did not know how it would fit into our application. Familiarizing ourselves with sockets and understanding the mechanisms of it really helped with developing the real-time functionality into our application. For our database system, the back-end team chose MySQL as they had more knowledge about it. Initially, we were writing queries to create tables and manipulate data within them. Luckily, we were able to find a Node package called Sequelize that streamlined the process of managing our MySQL tables, so that we did not have to write queries from scratch.

One of the most challenging issues to debug was the move timer. Once the time to make a move expired, it updated one's win count and loss count inaccurately. After taking a deeper look at our code, we found out the issue did not lie in our code but was that the server hosting our application was running simultaneously with our development machines. There is a piece of code that is executed every second to simulate a timer for each game. When there is no time left to make a move, it will update the status of the game and each player's stats, then alert them. So, if we were to be developing on our local machines and running the game server, it is obvious that it will execute the same code. Therefore, it will update the user's stats once it finds a game in which the time to make a move has expired. To fix the issue, the team was told to comment out that portion of the code that simulates the timer.

Since everyone had their own private life to attend to, working as a team was hard. It was a difficult experience to communicate with one another, but we eventually pulled through and were able to develop the website to our expectations.

Project Conclusion

This project helped us to learn and build a web application using various API's. At the end of our project, we were able to successfully complete the multiplayer chess web application the way we wanted it to be. Our website allows the user to create an account and play and chat with other online users at the same time. We even got to implement our priority 3 list of features in playing the game in 3D, personalizing the user's profile, and creating a friends list. We all learned a lot about collaboration as a team, github use, and how to implement the new API's

during the development of the website. This project would not have been done in time without the collaboration of our front end team and back end team working together.