School of Computing
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

Final Report

Remote Chess Playing

Anthony Moran

Submitted in accordance with the requirements for the degree of
BSc Computer Science and Mathematics

2022/2023

COMP3931 Individual Project

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Final Report	PDF file	Uploaded to Minerva (DD/MM/YY)
Link to Online Repository	URL	
Link to Video Demo	URL	

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

Summary

My main objective is to build a web service that allows users to play chess over a network. *My current solution is using the peer-to-peer architecture, I need to research whether it is possible to port my solution to users over a wide area network*.
Users will be able to register an account, and become friends with other users on the platform.
The server will be able to keep a record of users' games and the "elo"/skill of each user.
A user can be a part of multiple games with different people simultaneously.

Acknowledgements

<The page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by other to the project.>

Note that it is not acceptable to solicit assistance on 'proof reading' which is defined as the "the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the test"; see
`https://www.leeds.ac.uk/secretariat/documents/proof_reading_policy.pdf`

Contents

Chapter 1

Introduction and Background Research

1.1 Introduction

Chess is a board game that has existed since the 7th century [**?**]. The game is played by two people, each one in control of an army of equal strength; it is up to the player's logical reasoning and deduction in order to conquer the board. For a long time, chess could only be played in person, or perhaps through the post. Internet Chess Club was founded by Danny Sleator in 1992 [**?**], and he lead a small team of programmers to develop the first dedicated chess server. This was the introduction to playing chess over the internet and it allowed people to play chess together, regardless of the distance between them. It wasn't until 1995, where the first web based chess server was launched by Caissa [**?**], which featured a graphical user interface. This most likely contributed to a higher adoption of playing chess online because it gave users a friendlier interface, which was more intuitive and approachable than what was previously available. Since then, many similar services have been created such as <list examples> and it goes to show that there is quite some variety in the way this service is implemented.

1.2 Literature review

1.3 Libraries and Tools

1.3.1 Python

Flask / Flask Restful
Keep brief because it is not in the final solution
Initially chosen due to comfortablility and experience
Used polling
python modules here

1.3.2 Web Languages: HTML, CSS and Javascript

websockets

1.3.3 Github Pages

1.3.4 Heroku

1.4 Background Experience

Distributed Systems' - Webservice coursework and how it assisted in developing a web app building for system vs building for user

Chapter 2

Methods

2.1 Version Control

All of the code developed for this project has been managed by git and pushed to a github respository with can be accessed by the link:

https://github.com/Anthony-Moran/Final-Year-Project

The history will show the various stages of this project and how we started off with a blank workspace and ended up with this product now. When implementing the online features, the commit messages became less detailed because the changes were extremely minor and changes were being made rapidly for debugging the transition from offline to online.
Offline and online version, mention it here but explain more detail will be given in the backend section.

2.2 Back End

Processing logic on server side to reduce computation on client side and also means there is a single point of truth

2.2.1 Chess Logic

Mention transition from old chess engine to python-chess. The old version was going to be used when the project started off as machine learning but in the end it wasn't fully implemented so now we are using the built in chess module.
Board is represented using FEN.

2.2.2 Game Management

Storing a tuple of game state and connected users in a dictionary with join key
Generating join keys

2.2.3 Python Server

Offline server, explain how to enact

2.2.4 Github Pages

Online server

2.3 Communication

Network speed - applicable to online version
Consider security
Bad join keys redirect users to home menu
Challenege with matching the backend board with the frontend representation: orientation of pieces vs canvas coordinates

2.3.1 Websockets

Code was adapted from https://websockets.readthedocs.io/en/stable/
Flawed with short life time and disconnect easily
Pro is that they use bidirectional channel, so polling is no longer required
Iphone will disconnect if locked/leaves safari (added cheap fix to allow reconnection but it's not special to the user, anyone can connect in their place)
Refreshing with one user would cause the game to be deleted because the server would temporarily see that there are no users in the game. And after the page refreshed the user wouldn't be able to join because the game is now deleted. Added a small buffer time to keep the game alive, to combat this.

Closed after win because no more messages need to be exchanged.

2.3.2 Heroku

Platform as a service (distirbuted systems?)
Server sleeps after a period of not being used. Causes next request to take a few seconds longer than usual

2.4 Front End

2.4.1 The Welcome Page

Option to start a new game
Alternatively you can enter a code and join an existing game
Red message appears under user input if they entered an invalid code. Catches the eye and red subconsciously indicates error.

2.4.2 The Chess Page

Nav bar to return to main menu
Text at the top to show join code and link
Also prompts on who's turn it is and whether user is in check or if the game has finished with the result.
board is drawn with canvas
reference sprite sheet
requires javascript

interaction through mouse click events : converts mouse position to row and column using modolos (check spelling) and integer division
only parts of the board that need updating are updated: only drawing pieces that have moved or been taken, more resource efficient than redrawing the whole board.

2.4.3 Responsive Design

(CSS) Prioritises mobile devices and forces computers to adjust the layout. Computers typically have faster processors than mobile devices.
computer users can resize the window and the design adapts:
https://web.archive.org/web/20220714020647/https://bencentra.com/code/2015/02/27/optimizing-window-resize.html (debouncing?)

Chapter 3

Results

Test everything works.
Was the solution limited to peer-to-peer or was it able to expand beyond that?

3.1 Alpha Testing

Use this fen: 5b2/8/8/8/8/n7/PP6/K7 b to quickly show a stalemate (move the bishop down right one space)
In general, use different fens to show efficient testing
Castling
En passant
Promotion
Edge cases with connections

3.2 User Testing and Feedback

Discuss feedback from users

Chapter 4

Discussion

4.1 Conclusions

4.2 Features to be added in the future

Ability to resign
Show move history or allow pgn to be downloaded
Add ability to spectate

Appendix A

Self-appraisal

<This appendix should contain everything covered by the 'self-appraisal' criterion in the mark scheme. Although there is no length limit for this section, 2—4 pages will normally be sufficient. The format of this section is not prescribed, but you may like to organise your discussion into the following sections and subsections.>

A.1 Critical self-evaluation

Code could have been refactored better into a more modular structure with clearer responsiblities for each file (an attempt was made with the two javascript files making one handle the connection and the other to handle the board)

A.2 Personal reflection and lessons learned

How to use websockets to make real time applications

A.3 Legal, social, ethical and professional issues

<Refer to each of these issues in turn. If one or more is not relevant to your project, you should still explain *why* you think it was not relevant.>

A.3.1 Legal issues

A.3.2 Social issues

A.3.3 Ethical issues

A.3.4 Professional issues

Appendix B

External Material

<This appendix should provide a brief record of materials used in the solution that are not the student's own work. Such materials might be pieces of codes made available from a research group/company or from the internet, datasets prepared by external users or any preliminary materials/drafts/notes provided by a supervisor. It should be clear what was used as ready-made components and what was developed as part of the project. This appendix should be included even if no external materials were used, in which case a statement to that effect is all that is required.>