ANALYSIS

The problem

When asked to think of a board game, Chess comes to mind. Not only is it the best-selling board game of all time, but it also dates back all the way to 6th century India.

The ancient game is widely said to improve your critical thinking skills, your ability to focus, and playing it has proved to have a beneficial impact on those with attention deficit hyperactivity disorder (ADHD).

The game of chess has been sparsely changed over the last few centuries and yet most people still do not understand every niche rule and strategy that Chess has.

The reason for this lack of thorough understanding is the availability of in-depth Chess tutorials. In most cases, one of two factors come into play:

1. *Tutorials are locked behind subscriptions or one-time payments.*

For most people, paying hard-earned money to learn Chess seems absurd, and so they are limited in what they can learn.

1. *Effective tutorials are difficult to find without being bombarded with advertisements of the aforementioned paid learning methods.*

In this instance, casual or beginner players are unable to spend prolonged amounts of time sifting through advertisement riddled content, and so give up their attempt to improve. Again, limiting what they can learn.

So, the poor availability of effective Chess tutorials is the problem I wish to tackle, by creating Chess software that will allow a user to:

1. *Play simple Chess.*
2. *Play simple Chess over networks.*
3. *Play Chess in a tutorial mode where tips such as the best move to make are highlighted to the player.*
4. *Save in-progress Chess games to a database, and them creating a login so only they can access these saves at a later date.*

My ideal end user

The group I aim to target this software to is those with none to beginner knowledge of chess. Specifically, those who have to balance their time, such as workers or students, who would benefit from the ability to save an in-progress game and come back to it when they are free.

Required research

Required research - The minimax algorithm

In order to begin this project, I have conducted some research I felt was necessary. Notably, I did a lot of research on algorithms, as I realised that I would extensively use algorithms in the game of Chess I will create. One use of them is with the tutorial mode I have proposed, where the game will generate the best move to make in order to give the player an insight into moves, they may have not of thought of. To do this, an algorithm will have to check all possible moves, and by hard coding a ‘capture value’ to each piece, determine which move is the most valuable.

Required research - Alpha-beta pruning

While researching the minimax algorithm I came across Alpha-beta pruning, a search algorithm that is used to reduce the number of nodes that are analysed during the minimax algorithm. This is crucial as the minimax algorithm that I aim to create will have to evaluate many different situations when determining the best possible move to make. By reducing the time taken to complete this action, the game will not only run quicker but also make the computer seem more human. In other words, it will improve the time complexity of the minimax algorithm from **O(bd**) to **O(nd/2)** where n is the number of children per node, and d is the depth of the tree that represents all the possible moves.

Required research - Choosing the right programming language

When choosing Computer Science as an A-Level subject I was sure that I’d be using Python as the programming language. However, I have recently completed an online course on C# and have found that I prefer the .Net framework. In addition, C# tends to run at least 10 times quicker than Python. This would be important due to the complex algorithms that would be executed by my code. Also, for database administration/creation I will use SQL as it is widely used, and I have some experience with the language from the A-Level course.

Required research - Choosing the right IDE

As I will be using C# my main free options for IDEs are either Visual Studio Code or Visual Studio. After comparing the two, I decided that I will use Visual Studio as it has better tools for working with large projects whereas Visual Studio Code is more aimed towards single file projects. Visual Studio Code is a code editor like notepad++ and is not classed as an IDE. Also, I have previous experience with Visual Studio and so that is my definitive choice.

Required research - Windows Presentation Foundation (WPF)

As I intend to use C# the two ways to create user interfaces are to either create a Windows Forms application or a Windows Presentation Foundation (WPF) application. At a work experience placement I recently went to, I created a complex WPF application and so naturally aim to create another for this NEA. Due to this past experience, I already know XAML which also will help in the creation of this software’s UI. Finally, WPF takes a much more modern approach to UI creation and so can be used for more complex systems. For this NEA, a WPF application is the way to go.

Challenges and fixes

Some challenges I may face are as follows:

1. *Algorithmic efficiency. I will be using the minimax algorithm to determine the best move for a player to make. This algorithm is very powerful but complex. I must program the algorithm to differentiate between legal and illegal moves, and so this whole process can make the algorithm sluggish depending on the context of the move.*

To counteract this, I will attempt to reduce the number of moves the algorithm has to search through, using a process called Alpha-beta pruning. More detail on this can be found on the previous page.

1. *Database administration. I’m beginning this project with limited experience in the SQL language and using it to create a relational database.*

In order to prevent myself from struggling with the database side of this project later on, I will practice my SQL with the website www.sql-practice.com.

1. *Creating an easily understandable user interface. As my software will be aimed at those with little experience of the game the interface must be accessible for them. Due to the fact that I am not a beginner at Chess, I may not immediately know how to cater the interface to those who are.*

*To ensure the UI is Chess beginner friendly, I may have to refer to possible end users to critique the interface as I create it.*

Challenges and fixes - Constraints

The main constraints I will face with this project are time and ability. Time could be a constraint as I have just under a year to complete the project. To ensure this does not affect the quality of the project, I will incorporate this project into my existing routine, meaning I will consistently work on it. Secondly, ability could be an issue when it comes to sections of the project that I’m only slightly familiar with. Such as communicating over networks, or when programming the minimax algorithm/alpha-beta pruning process.

Existing systems and their issues

Existing systems and their issues - Chess.com

A screenshot of a game

Description automatically generated

As you can see, Chess.com does have a beginner friendly user interface. Notice the black dots near the selected pawn which display where I could possibly move the piece to.

Also Chess.com allows you to somewhat change the difficulty of the computer that you play against. The difficulty is represented by the number in the top left of the screenshot, and to change the difficulty you select different ‘bots’. See below.

A screenshot of a video game

Description automatically generated

The icons shown here all represent the aforementioned ‘bots’ (different difficulty levels). There are two issues here that I have identified:

1. *The UI is cluttered. A new user to the website could be confused to begin with as to what the icons mean and the difference between them.*

To avoid such an issue in my software, I will make sure my UI is simple and easy to understand. Possibly adding sections that provide help such as answers to frequently asked questions.

1. *Most of the icons representing difficulty are locked behind Chess.com’s premium plans. These monthly subscriptions can cost between £4 and £10 a month. This issue is one of the main reasons that beginners are limited in how they can improve at Chess.*

To stop players being locked away from certain parts of the experience I am going to make the software completely free for users. As somebody who would not be able to afford a £10 a month subscription for Chess, I understand the need for accessible Chess resources.

Existing systems and their issues - YouTube tutorials

A screenshot of a video game

Description automatically generated

I am in no way trying to slander the creators of the videos shown above, however I see a few issues with the first three results when searching ‘Chess tutorial’.

The first result is an advertisement of a paid Chess course. After a short visit to the linked website, I found that his course is £10 a month. This is not economically viable for the group that form my projected end users.

Next, the second result is a video by GothamChess, a highly decorated Chess YouTuber. The video is quite a good tutorial for the basics of Chess, however it contains the second part of the problem I wish to solve. He spends the video advertising his Chess course. I spent over 10 minutes trying to sign up to this course, and even after that time had elapsed, I was still oblivious to how much the course costs.

Two account verification emails later I was still in the dark. So, like all of my projected end users would, I gave up with GothamChess’ course.

Finally, the third result is a playlist of 300+ videos which would immediately put any beginner off watching this online tutorial despite the videos being great tutorials from what I saw.

Similarly to my solution for Chess.com’s problem (the cost) I will make my final product free to play. To avoid users being put off by overwhelming factors of a tutorial, such as the playlist with 300+ videos, I will prioritise simplicity when creating the user interface.

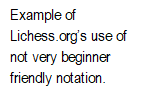
Existing systems and their issues - Lichess.org

Rivalling Chess.com, Lichess.org is a website that offers a free Chess tutorial service. The website claims to be advertisement free “now and forever” (Lichess’s description on the Apple App Store).

A screenshot of a computer

Description automatically generated

The preceding image is of Lichess.org’s main webpage. I find this display hard to understand even as an intermediate Chess player. Take one of the central boxes for example (shown below), the meaning of the 1+0 notation is not clear to a beginner. This notation means each player has a total of one minute to make their moves, with zero time increment (no time is added to a players clock when they make their move). This notation was created by the Chess community to describe different ways of playing.

A black and white sign with white text

Description automatically generated

While Lichess.org is great in that it is completely free of charge, advertisement free, and open source, the website is tailored towards seasoned Chess players. This will make it difficult for any new player to learn the game from Lichess.org.

My software however will have a beginner friendly display, and possibly a section that will allow the user to learn the different types of notations for chess games. This way once a user has used my software they are not only better at chess, but also more knowledgeable about the Chess community.

Interviews

The methods and their disadvantages

To garner the opinions of my potential end user I’d need to conduct an interview. These opinions would help me to better understand the needs of my end users and what features they approve/disapprove of.

Before creating the questions that would make up my interview I had to decide between a face-to-face interview or a survey. The two main objectives of the interview I will create are:

1. To get information from multiple people, preferably as many as possible.
2. To get information from people of differing experience with chess. Although my software is aimed at beginners I am still interested in the opinions of more advanced players.

To fulfil both objectives I decided that a survey would be the superior option. This way, I could get a large number of people’s opinions without conducting many time-consuming face-to-face interviews.

Also, research shows that when people are interviewed in person their responses can be distorted unknowingly by the interviewer, known as interviewer bias. According to a report from the University Of North Carolina:

“Without formal training interviewers may inadvertently make inappropriate selection choices by misinterpreting the information they receive from selection interviews.”

As I don’t have formal training in interviewing people, I worry that could lead to this bias affecting the results of my research. This provides another reason to use a survey instead.

Despite this, the method of surveying to attain information also can lead to bias, known aptly as survey bias. However, survey bias is knowingly caused by the interviewer in the form of leading questions, according to Voxco:

“Leading questions is the most typical example of a biased survey question. They lead the respondents towards a certain answer.”

As a result, survey bias will distort the results of my research less than interviewer bias because survey bias is controllable. Now confident in my decision to use a survey instead of an interview, I will ensure to avoid using leading questions in my survey to prevent any bias.

To create this survey, I have decided to use Microsoft Forms as they provide excellent tools for both forming surveys and collecting the responses to them.

The composition of the survey

The survey was formed of 13 questions in total. The first section involved simple questions related to the respondents age, Chess ability, and how frequently they played Chess. As I had used Microsoft Forms, I would be able to

The results of the survey

To get results from my survey, I was able to unleash it upon students at Bourne Grammar School. This allowed me to complete my first objective for this research as this survey was responded to by many students, allowing for a larger set of information.

In total, I received between 175 and 185 responses at the time of writing. This was an unexpectedly large number however it will further help to me to tune my software to suit the end user. Some of the more important questions and their responses are shown below.

Have you ever played Chess? A white background with black text

Description automatically generated

Here we can see an overwhelming 82% of respondents have previously played Chess. In my opinion this could be due to the vast number of responses, but also the fact that only students who attend a Grammar school were able to provide their opinions, and these types of schools tend to be more academic and therefore have a higher proportion of students who play Chess than other schools.

A screenshot of a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

These two questions were important for me as it allowed me to see if the respondents were made up of differing types of Chess experience, my second objective for this research.

As you can see, there is a large variation in the frequency of Chess playing from the respondents, however the majority of them would classify themselves as beginners. This enforces my reasoning to target my software towards beginners.

While not necessarily important to my research, I also asked the respondents if they preferred to play as white or black (make the first move or the second). Interestingly out of the 54 people who classed themselves as intermediate Chess players, the majority answered white for that question. According to [Wikipedia](https://en.wikipedia.org/wiki/First-move_advantage_in_chess#:~:text=In%20chess%2C%20there%20is%20a,to%20win%20with%20perfect%20play.):

“There is a consensus… that who makes the first move has an inherent advantage”

A screenshot of a chat

Description automatically generated

As seen in the responses to these questions about online Chess tutorials, more than half of the respondents that said they play Chess have used an online tool to improve at Chess. I asked this question as I was curious which of and how the existing systems that I covered earlier in this section were used by people. Unsurprisingly Chess.com was the most used online tool. Considering the Chess.com engine had 11 million daily active users as of April 2023, I expected this majority.

A white background with a circle

Description automatically generated with medium confidence

As seen above, Lichess.com had only one user out of the respondents, which could be due to its unfriendly UI. I say this as a huge majority of respondents who say that have played Chess before also answered that they would not be able to understand Lichess’ notation. Because of this I will not use said notation and instead include a section explaining this, to make my software accessible to all users (specifically beginners).

A screenshot of a computer

Description automatically generated

A pie chart with text below

Description automatically generated

Finally, I asked three questions related to the proposed features of my software. I previously planned to make it advertisement free, include a simple UI, and free of charge to differentiate my project from the existing systems: YouTube tutorials, Lichess.com, and Chess.com (respectively).

Concluding the research, I had gained a lot of useful information and realised the extent to which people already used online Chess tutorials. From this feedback I decided that I needed to change one thing about my plan for the project, to definitely include a section in my software that explains advanced Chess notation. This is due to the fact that only 16% of respondents could decipher it.

My objectives

**1** **To allow the user to simply play the game of chess against a computer.**

* 1. Create a minimax algorithm to deduce the most valuable move the computer could make
     1. Create a tree data structure containing every possible move.
     2. Create an algorithm to assign each possible move a value that represents the impact of said move.
     3. Create the Alpha-beta pruning search algorithm to decrease the number of nodes that are searched by the minimax algorithm.
     4. Finally create the minimax algorithm that will search the tree for the move with the highest value assigned to it.
  2. Create a user-friendly UI to show the Chess game.
     1. Use WPF to display an image of a Chess board that will represent the playing board.
     2. Also use WPF to display images of Chess pieces that will represent the playing pieces.
     3. Also use WPF to display all possible moves by changing the colour of tiles that a piece can be placed on.
     4. Also use WPF to display a move by changing the position of the image that represents the piece being moved.
  3. Allow the user to move pieces.
     1. Allow the user to select a piece to move.
     2. Show the user all possible moves (1.2.3).
     3. Allow the user to unselect a piece and instead select a different piece and display that piece’s possible moves (1.2.3).
     4. Finally allow the user to select a move and then move the piece to the desired position (1.2.4).
  4. Create an algorithm that analyses for check/checkmate.
     1. To analyse for check, before the user or computer makes a move traverse through the tree of all possible moves of the opponent and if any involve taking the current player’s king, restrict possible moves to those that protect the king.
     2. To analyse for checkmate the process in 1.4.1 is used, along with analysing the possible moves on the next go. See the flowcharts in the modelling section for more details.

Objectives continued

1. **Allow the user to play Chess over networks.**
   1. Create a WebSocket server that will receive and send data.
      1. Configure the server to receive data (a player’s move) and send said data to the other player.
      2. Configure the server to send and receive data that represents a change in game state (E.g. checkmate, resignation)
   2. Allow both clients to connect to the created WebSocket server and send data to/receive data from it
2. **Allow the user to play Chess against a computer in a tutorial mode.**
   1. Use an algorithm like (1.1) where the best moves a player can make is found.
      1. The algorithm will find the best move a player can make (which move has the most value) and return the tile coordinate to move to, and the piece to move.
      2. Change the colour of the returned piece and the tile that the algorithm recommends moving to.
      3. Also allow the user to make their own move, they are not forced to play the recommended move.
   2. Throughout each game with tutorial mode activated, provide hints or extra useful information.
      1. As player and computer make their moves, record the algebraic notation of the moves (E.g. Rg4, rook to g4) and display these to the user to help them understand this widely used notation.
3. **Allow the user to save unfinished games of chess against the computer to be able to continue at a later date.**
   1. Allow the user to create and name a folder which will be used to store unfinished games that they save.
   2. Allow the user to stop an in-progress game and create a game save that stores the position of all the pieces, and who has the next move.
   3. Allow the user to select unfinished games and continue them. To do this, the software would recreate the previous state of the game using the information stored with the game save, such as the position of pieces.

Objectives continued again

1. **Allow the user to create a Build Your Chess account and use this to access their game saves.**
   1. Allow the user to sign up for an account.
      1. Allow the user to enter a username and password to use as their account credentials. Use a double entry field to ensure the user does not mistakenly enter a password they didn’t mean to.
      2. Create an identification number for the user (UserID).
      3. Create a hash of the entered password using the MD5 hash function and store it into a SQL database alongside the UserID.
      4. Also store the user’s username into the database for login purposes.
   2. Allow the user to log in to their account.
      1. Allow the user to enter their username and password into the system.
      2. Create a hash of the entered password using the MD5 hash function. Compare this hash to the one stored alongside the entered username.
         1. If the hash of the entered password matches that of the stored hash, allow the user entry to the account requested.
         2. If the hash of the entered password does not match the stored hash, do not allow the user entry to the account requested.

Modelling

Data dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Data Type | Validation | Sample Data |
| UserID | String | =8 digits | 12349826 |
| Username | String | >5 digits  <12 digits | “ChessMaster123” |
| PasswordHash | String | =32 digits | “938c2cc0dcc05f2b68  c4287040cfcf71”  (MD5 returns a hash of 128 bytes, shown as 32 hex values) |
| Other kinds of ID numbers | String | =8 digits | “12381264” |
| NumberOfGameSaves | Integer | None | 12 |
| GameSaveName | String | >5 digits  <12 digits | “MyChessGame” |
| PiecePositions | Array | None | None |
| NextTurn | String | =”Black”  Or  =”White” | “Black” |
| PlayerMove | String | None | “Rg4” (Rook to g4) |
| IsInCheck | Bool | None | True |
| IsInDoubleCheck | Bool | None | False |
| KingIsStuck | Bool | None | True  (True if king cannot safely move to any surrounding tile) |
| IsInCheckmate | Bool | None | True |
| TileIsSafe | Bool | None | True  (True for a tile if no opponent piece can move here) |
| IsTutorialModeActive | Bool | None | False |

Input, process, storage, output tables (for using the account system)

IPSO table 1: Signing up for your Build Your Chess account

|  |  |
| --- | --- |
| IPSO | Information |
| Input | User information:  -Desired username  -Desired password (using a double-entry method) |
| Process | -Create a unique user identification number (UserID) to use in the SQL database |
| Process | -Pass entered password into a function that checks password strength. If it is not strong enough, ask user to try again |
| Process | -Apply a salting function to add characters to the given password to secure the hash. |
| Process | -Create a hash of the entered password and the added salt using the MD5 hash function (PasswordHash) |
| Store | -UserID  -Username  -PasswordHash |
| Output | -Successful sign-up. User may now access the Build Your Chess software |

IPSO table 2: Logging into your Build Your Chess account

|  |  |
| --- | --- |
| IPSO | Information |
| Input | User information:  -Username  -Password |
| Process | -Create a hash of the entered password using the MD5 hash function |
| Process | Compare the created hash with the hash stored to the entered username. If hashes match, allow access. If not then ask user to try again |
| Store | No items to store |
| Output | -Successful log in |

Entity Relationship Diagram (for saving unfinished games)

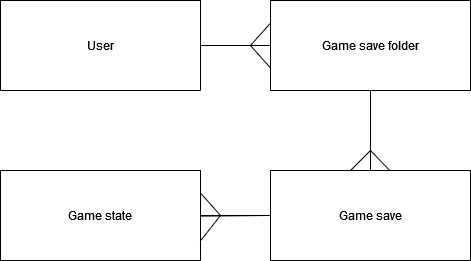
ERD diagram 1: Showing the relationship between a user, the folder they save unfinished Chess games to, the unfinished games themselves, and the game state of those games

User – The person using the Chess software.

Game save folder – A collection of game saves that the user can create.

Game save – An unfinished Chess game that can be saved either internally onto the user’s storage, or externally in a SQL database.

Game state – The actual information that represents a game save, such as the position of all the pieces.



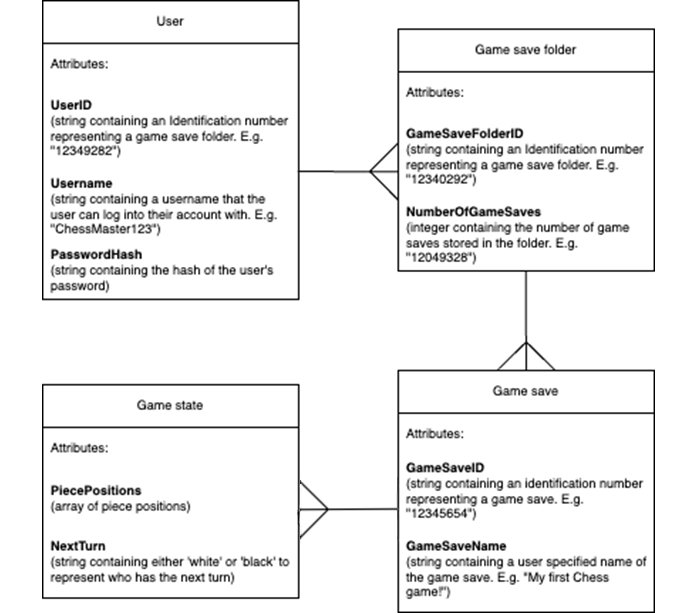
The user may create many game save folders, however each game save folder belongs to one user. Their relationship is one-to-many.

Each game save folder may contain many game saves, however each game save can only be stored in one game save folder. Their relationship is one-to-many.

Each game save can only represent one game of chess, however it can represent multiple stages of said game (user can choose to either overwrite a previous save or create a new one when saving a game). As a result, their relationship is a one-to-many.

Entity Attribute Model (for saving unfinished games)

EAM diagram 1: The relationship between and attributes of User, Game save folder, Game save, and Game state



System flowcharts representing algorithms (for Chess logic)

Flowchart 1: The algorithm that analyses possible moves for check(mate) (single check)

A diagram of a flowchart

Description automatically generated

Flowchart 2: The algorithm that analyses possible moves for check(mate) (double check)

A diagram of a flowchart

Description automatically generated

Data volume diagram

|  |  |  |
| --- | --- | --- |
| Data object | Volume of data | **Sample data** |
| PlayerMove | Average of 0.5 per minute\*  (1 every 2 minutes) | “Rg4” (Rook to g4) |
| PiecePositions | Average of 0.0125 per minute\*\*  (1 every 80 minutes) | A two-dimensional array containing 64 characters |

\*According to multiple Chess forums (such as: <https://www.chess.com/forum/view/general/what-is-the-average-time-you-spend-thinking-about-a-move>) Users spend an average of two minutes to make their move. However, this depends on the stage of the game as moves are usually made much quicker at the beginning of the game, and the time between moves grows as the game progresses.

\*\*According to statistics provided by the ‘Chessgames’ database (which contains 685,801games) the average number of moves per game is 40.04. Using our average time per move, which is one every two minutes, the average game should last around 80 minutes. This can vary wildly depending on the players, and obviously if the game mode has time constraints. As a result, if a user decides to save the game state after finishing, they will be saving data every 80 minutes.