**Design Part 1 (draft 2):**

Table

Description automatically generated

I would like to note that I have decided not to include the login system initially in my plan. This was because I determined that the number of features that required a user to be logged in was low. Therefore the benefit is small when considering the significant added complexity. So I have decided that my initial plan will not include a login. I will then add detail about the potential structure of a login system later.

**Part A: Breaking down the problem into smaller parts suitable for computational solution justifying any decisions made.**

**And Part B: Explain and justify the structure of the solution**

Due to the lack of a login system the initial proposed user interface is rather simplistic. This is an advantage as it will be easier to program and easier for the user to use. I can potentially add to it later.

The problem that this simplified version of my solution is trying to tackle: is to simply and easily allow my users to play chess.

In order to allow the user to play chess I have decomposed the problem in this way.

The problem consist of:

* Getting the users input for the pregame configurations
* Providing a way to get input moves and give output moves in order to facilitate a chess game
* Infirming the user when a game ends and how it has ended.

Here is a mind map of how I decomposed this problem into separate pages and decomposed each page into a set of different inputs and outputs that had to be gathered. This method aims to use decomposition to break the problem into a collection of pages and widgets and thinking ahead to determine the nature and requirements of each of these widgets / subcomponents.

Pages needed:

A picture containing diagram

Description automatically generated

Breakdown of each page (decomposed by the various inputs and outputs that the page is responsible for).

A picture containing diagram

Description automatically generated

Text

Description automatically generated  
Graphical user interface, text, application

Description automatically generated

Many decisions were in the decomposition of the problem into various inputs and outputs. here are my justifications:

* For one I decided to reduce the initial problem size by removing the login and user part form my design. This could be added in later but I would first like to complete this design without. This greatly reduced complexity and reduced the number of webpages needed.
* For another I decided that I should include default options for many inputs in the pre-game form to ensure that no matter what the user does a valid input will be given (e.g. using radio buttons). This will reduce work needed for validation and make my program more error resistant.
* I also decided to approach receiving input moves and giving output moves in the way I did as this is a standard among chess apps and websites. Ensuring my program meets the convention will makes it feel more intuitive to those who have used other similar programs before and will reduce the learning curve. This was inspired by the research completed in my analysis step on websites such as chess.com.

I have also included a breakdown of the modules that will be needed in the client side JavaScript

Teams

Description automatically generated with medium confidence

I should explain why the frontend has so few modules and why it doesn’t need a way to understand a chess game such as where a bishop can move or if a given game state is check mate.

The reason for this is that I intend to process all that information on the backend and leave only validation to the frontend.

API socket connection to backend is a module that connects to the backend and request information using methods such as is\_game\_over(). It then uses another module to handle the response, in this case a small submodule responsible for causing the appropriate popup to occur to notify the user that the game is over and to state the result.

Here is a diagram of the API connection between my frontend and backend

Timeline

Description automatically generated with medium confidence

It shows how the front end client will be equipped with an array of moves that are legal and an array of moves that are legal and end the game + how the game ends.

Here is a flow chart to act as another way to show the logical flow of handling input to produce output in the frontend

A picture containing polygon

Description automatically generated

The point of this is to explain that the frontend is relatively dumb as it is unable to make inferences about the game ion its own. However by using one request per user move the server can send the client all the data it needs to validate the users next move. This is done by sending a new array of all the legal moves the user can make for future validation along with the server’s move.

This means that the main responsibility of the frontend (and therefore the purpose of most of its modules) is to validate the users move and process it into a request for the server as well as to process the servers move and turn it into the appropriate output for the user (e.g. know when to display game over) This reduces the complexity of the frontend greatly as it reduces the number of methods and classes needed. For instance no method is needed to determine if a game is over and no class is needed to represent the moving patterns of a knight.

I have used decomposition and thinking ahead to identify the various pages needed and the inputs and outputs that each should provide. Using this I have created on paper some designs for the various pages that would make up the user interface

The scans include arrows with annotations of the various features, some features in brackets are there to show where such a feature could fit if implemented in future.

My user interface will be made up of these pages:

* A menu page / title page
* A pre-game configurations form
* The main page in which a chess game is played
* A popup menu providing information about how the user has won.

A picture containing text, whiteboard

Description automatically generated Diagram

Description automatically generated Diagram

Description automatically generated A piece of paper with writing

Description automatically generated with medium confidence

I will outline and explain some of the key features and decisions made when creating each page.

Note the aesthetics such as the background have been abstracted in order to reduce the problem. Implementations of these user interfaces will not have plain monochrome backgrounds and features.

I also choose to centre align all my pages in order to catch the users attention (they are most likely to look in the centre)

I would also like to mention here that while I intend to make my website work for multiple resolutions, these sketches are for the 820x1180 resolution of an iPad Air in portrait

The title / menu page:

This page will initially be a simply page with only a title, a chess related image and a button to begin playing vs the computer. As more features are added it will be expanded to become a home page and menu for the website. Examples are included.

The pregame config form page:

This page will be what the user sees after they click to begin a game vs the computer. I will use radio buttons with default options to ensure that the user cannot provide an invalid input for fields like difficulty and starting colour. I will need to add validation to the times length inputs (as an example). This would likely include a presence check, type check (integer) and a range check (e.g. 15<= time <120).

The chess game play page:

As earlier stated the method of inputting and outputting is standardised to make the program easier to use.

Outputs like previous moves and pieces taken will auto update after each move. How the timer works will depend on implementation. It will only appear in the top corner if requested in the game configuration.

The game over pop-up:

The main decision here was how the information should be conveyed. I decided against a different webpage or a popup square that obscures the chess game. This was because I thought that users would appreciate still being able to see the state of the current chess board. The current implementation will work by using the same webpage as the chess game play page. When the game is over some components like timers, whose turn and the pieces taken will be hidden and another widget that announces that the game is over unhidden. I may have the pieces taken component not disappear but this would mean that all of the widgets would not fit on a mobile phone screen without scrolling down. This may not be a bad thing if there is enough space left as users may not mind scrolling down to see all the previous moves. This decision depends on how the widgets fit together once the GUI is created and what feels natural.

Here is a menu flow diagram to show how the website is composed of these webpages to form the website.

Diagram

Description automatically generated

**Part C describe usability features to be used in the solution:**

I have fully read this article which introduced me to the breakdown of usability into these components. I have integrated its tips with my project.

<http://www.wqusability.com/articles/more-than-ease-of-use.html>

**Effective**

For a user interface (UI) to be effective it must meet the expectations of its users with regard to the utility it offers and also how effectively an average user could use this site to meet their goals.

I have determined in my analysis step that my target users are casual chess players of all ages. As such I have tried to meet their needs specifically be adding features like take backs (if implemented) and an easy difficulty.

To help meet users of all ages I intend to utilise my discussions with my grandparents and my research. I realised that many older people enjoy playing games like scrabble and chess on an app or website. They also commonly use iPads and tablets as they are bigger and so require less finger dexterity. Here is a breakdown of hardware usage by age: (<https://www.statista.com/statistics>)

Proportion of people in different age brackets who use a tablet to go online:

<https://www.statista.com/statistics/300257/tablet-pc-usage-to-go-online-by-age-uk/>

Chart, bar chart

Description automatically generated

52% of people aged 65+ have a tablet that they use to access the internet. I found this surprising and useful. For this reason one of the other ways in which I aim to ensure that all my target users are able to effectively make use of this product is by ensuring that it has specialist layout that are compatible with large resolutions of laptops and PCs (resolutions vary) as well as tablets (baseline could be iPad Air which is 820px \* 1180px) and mobile phones (a baseline could be the iPhone SE with a resolution of 375px \* 667px). (My source for resolutions is the chrome web browser in its dev-tools section). This way I can ensure that my website can be accessed by commuters on a mobile, and by people of all ages at home with laptops, PCs and tablets.

Due to the target users I am prioritising some features over others. For one example I am prioritising crating a range of difficulty settings over making a mobile chess clock component in order to best meet my casual user’s needs.

**Efficient**

I would like to divide efficiency into two distinct parts: user interface efficiency and overall program time efficiency.

I aim to make the user interface efficient as I believe that this goes hand in hand with being easy to learn. For example I have minimised the number of clicks that the user will need to make in order to begin playing chess. They can click play, change 1 or 2 configurations by clicking radio buttons and begin the game unhindered. They only need to make 2 intuitive clicks in order to move a piece. Additionally conformation popups are kept to a minimum.

The program has also had multiple considerations factored in to maximise efficiency. For instance I am using web sockets for the super-low latency (as well as the highly practical full duplex connection and ability to implement user vs user games is needed). I am also factoring in time complexity with all of the algorithms I employ. For instance I am using pruning and caching to improve the efficiency of my minimax variant algorithm. Additionally the computer is only given a limited time to think (for example 5 seconds max, if needed, difficulty dependant) as the user may become bored if the computer took as long as an average user to make a move. This allows the user to always be playing the game.

I may in future add keyboard shortcuts but I don’t currently deem it necessary or a priority.

**Error tolerant:**

With regard to error tolerance I have used some of the tips provided in the article to passively improve my application error tolerance without more validation and input sanitation. For example I decided to implement the difficulty input field in the program config form using radio buttons with medium as a default. This means that it is impossible to select an invalid difficulty or not provide a difficulty. I also took the tip of making everything easily reversible by adding a ‘back to config’ hyperlink in the chess game page so that difficulty etc. can be easily changed without going back to the main menu and restarting.

With some fields like the duration of the timer if a timer is preferred by the user I will need validation as mentioned earlier. I will use a presence check, a type check to ensure the input is of type integer and a range check such as timer in range 15<= t<= 120. When the submit button on the form is clicked, if this validation fails I will use the industry standard and give the box with invalid input a red outline along with red text to indicate the input requirements.

With regards to moves made by the user on my chess widget. I will talk about it in more depth later but validation is needed. For instance I must ensure that the first square that the user clicks on contains one of their pieces, then I must ensure that the second square is one that the user could move to. I will highlight the legal squares a piece, one selected, can move to with a green dot to eliminate ambiguity.

This validation will be quiet failing as if the user clicks on an invalid first square nothing will happen and if the user clicks on an invalid second square then the piece will only become unselected. No error messages will be used as I believe that this way is most intuitive due to convention.

**Easy to learn**

One of the reasons that this user interface is easy to learn is that is follows the convention of other chess programs with regards to how the user inputs a move. For example the same method, including highlighting legal moves, is used on Lichess and chess.com which I examined as part of my research. I was impressed by how intuitive it felt when I experimented with these websites.

Additionally by using a chess program convention in my interface design I will reduce the learning curve for new users that wish to start using my program. In this way I am providing an interface in the way that the user expects it. This interface also maps well onto both PC and mobile touch screen device.

**Engaging**

I have aimed to make my user interface pleasant and satisfying to use by not making it too complicated. I have tried to reduce the level of noise and needless complexity on the screen and aimed to logically block together information for the user. For instance I put the submit button at the end of the form as users complete forms top to bottom and are less likely to read them properly if the button is at the top. I also used this principle of distinct blocks on information in my chess game play page as information such as previous moves and pieces left was grouped together with clearly allocated, non-overlapping space. They were also at the bottom as they are less important than the chess board and the concede button which should catch the users full attention.

I also plan to make my website visually appealing. I have already in the plan included an image in the home page and centred all the content to look appealing. In the final version I intend to have a different background colours for the webpages (this detail was abstracted away in my plans and so left white). I also want to give the chess board more of a wooden look or even better give the user the ability to alter the look in the pre-game configuration or their account setting. Here is an example of the wooden look I would like to add.



Future plans to maximise how engaging the website is include:

* Adding sound effects for moving a piece and taking an opponents piece (inspired by interview and Lichess research)
* Adding proper animation to show the piece actually moving from one square to another rather than just appearing in the selected square. This would help make the computer’s move more clear as it would show what piece had moves and where as a moving object is more catching to the eye.
* The image on the title slide could be replaced with a chess board showing a rapid replay of a series of example matches on loop.