Evaluation

Following completion of my project, I will evaluate my overall performance throughout the project process. I will also consider changes that would be adopted were I to solve a similar problem in future throughout the evaluation.

Programming Language Used

I primarily used Java and NetBeans IDE 8.2 to complete and create my project. Overall, I think that Java was an excellent choice as a programming language to use. Java has a range of useful features that really helped me throughout my project. For example, the java.net class (which is built into Java) was invaluable as it handled much of the networking requirements for my system – allowing me to effectively transfer data between a server and multiple clients. Java supports multi-threading, which was useful in several areas of my project, particularly when allowing an unspecified number of clients to communicate with a server at any given point of time.

This was achieved by creating a new thread for every single client that connects with the server. Furthermore, threading was used to perform tasks such as updating timers or getting the move of a computer whilst also allowing for the chess board graphics to update. The NetBeans IDE supports a drag and drop GUI builder which enabled me to create good looking visuals very quickly. I would have wasted a lot of time if I were forced to add all components programmatically as I would have no clear idea of how they would look when the code was run.

Java was a very good choice as it is a language that is compiled, which allows code to be run quickly – something which was important to me as some of my code was very computationally intensive. Java offers dynamic arrays (ArrayLists) which were very useful when calculating what move a computer should make. Initially, I had used static arrays and resized them every time I needed to add a new data item to an array which led to a very inefficient system – with simple searches often taking minutes to complete. In order to create computer chess opponents, I needed to make code that was very well optimised in order to search through thousands of board states in a very short space of time. One optimisation I found very useful was the fact that Java supports switch statements. Initially, I was looping through every square on the chess board and performing a series of if statements to determine what type of piece was present on the board. However, Java supports switch statements which are far more efficient than if else statements as the access time for each case is the same, whereas if else statements require all possible states to be evaluated in some instances. This sped up my code massively and is an advantage of using Java, as languages such as python do not explicitly support switch statements.

Java is not a perfect programming language however, as my development and implementation of an artificial neural network was made difficult by the fact that Java does not have great support for optimised deep learning libraries. I coded my own artificial neural network from scratch which went very well, however it would have been easier to make use of an extremely optimised and professional library such as TensorFlow – which is written in highly optimised C++ code. To use such a library, I would have had to make my project in python. A lower-level C based language such as C++ may have made my chess code more efficient (and hence faster). Doing this would have allowed my chess program to search with a greater search depth, which would have enhanced the strength of the computer programming. C++ would also have been useful when training my neural network as the code would have been more efficient. It would have enabled me to make more use of code parallelisation and the GPU. Although this would be

somewhat negated by the fact that my laptop does not have a GPU or many cores. I chose not to use C++ as the faster code would not justify spending lots of time programmatically creating a GUI. Furthermore, I feel that it would have been a waste of time to learn a new programming language when I am already so familiar with Java.

Java supports a feature known as Javadoc which makes it easy to annotate and view annotation of Java code. This made my program much easier to understand and read – and hence easier to maintain and rectify any potential errors that may occur. As Java is such a popular programming language, there is a lot of support online regarding quirks and common errors that may occur. For example, I created an object from which a thread accesses an object variable and decides whether or not to terminate based on this variable. However, the system was not performing as expected despite my code being logically sound. I used a built-in feature of NetBeans – the variable watch – and it confirmed that my logic was correct but somehow the code was not performing as expected. I added in a print statement to confirm the values stored at a given line of code in the thread. However, this print statement made my code work. Through an online forum post, I found out that object variables are not synchronised, and that the Java virtual machine treats them as constants within a loop. Hence, I made use of synchronised statements to rectify this issue. This sort of support available for very specific quirks is what makes Java so appealing.

I also found the NetBeans IDE to be very useful as its syntax highlighting and powerful debugging tools made my code easy to read and debug. NetBeans is also integral to the drag and drop GUI designer which was so useful to design a good-looking system.

Overall, I feel confident that Java was the best choice for me personally as it is a language that provided the best overall trade-off between ease of use and speed of execution.

Comparison to Commercially Available Systems

When comparing my system to other commercially available system it is important to recognise that my system is fairly unique. However, lots of features are emulated by other websites and applications. I have not found any applications that are not websites that support online and offline play. Despite this, there are certainly many pieces of chess software which do many things similarly.

The most used chess software is chess.com – a website that has over 20 million members. The actual mechanics of playing chess are practically identical to my system (as the actual game of chess is well established). Chess.com provides two primary input methods: clicking on a piece and then clicking on one of the highlighted legal moves or dragging a piece to the desired square. My system only supports the former input method, although I feel that this method is more than sufficient. My implementation of the first method is very visually intuitive and easy to use, and I would argue slightly superior to chess.com, which doesn't highlight the king red when the king is in check (thus lacking visual clarity). Both my system and chess.com's system control very well and are intuitive to use. Like chess.com (and similar sites such as Lichess), my system allows users to play online against other people – supporting both games against friends and random players. Of course, my system is less advanced and is not designed to be used by a large group of people. This is evidenced in my server-side implementation which makes use of a single thread for all processing. This decision was taken as the target audience is very small and hence the increased reliability of a single thread is clearly justified. That being said, my online play is fully functional and functions in a similar fashion to these sites – with users being able to offer

draws and resign. In future iterations, I would consider implementing the ability to make chess moves by dragging and dropping the pieces. However, this is not an urgent addition as beta testing proved that the control system I have for the system currently is very easy and enjoyable to use. If I were to spend more time on the system, I would implement features such as ranked matchmaking (which occurs on chess.com) to make online play fairer. However, given that the system is targeted at a small pool of users, I felt that this feature was not essential. Chess.com has a text chat that allows two players to communicate with each other. This is something I would look to implement in future as it would make the experience of playing online games more social and enjoyable. However, the feature is not essential either as the online play is highly enjoyable without the option to chat.

My offline chess system makes use of five difficulty settings – a fairly standard range of difficulty settings for such an application. These settings range from nearly impossible to beat to very easy to beat and are very typical. I am very pleased with this as I used five different approaches for each difficulty setting and each approach has proven itself to be suitable for the task it was intended to perform. These difficulty settings provide a wide range of difficulty and feel very similar to play against compared with other solutions such as Lichess' computer opponents. Certainly, the level 4 and 5 computers are opponents that I find very challenging to play against. I always lose to the level 5 computer and win about 1 in 3 games against the level 4 computer. Conversely, I win the majority of games against the level 2 and 3 computers. This range of challenge is similar to that found on websites such as Lichess, showing that I have created a range of computer opponents that all operate at suitable difficulty settings.

My system differs from all other major commercial systems in that it offers the ability to manage the fixtures and results of a chess club in addition to actually offering the game of chess. However, when compared to other software such as LeagueRepublic, which allows one to manage some form of league – my system provides the necessary functions for such a system although it is significantly less advanced. That being said, it allows for fixture generation, even if it is far less customisable than other fixture generation software. The one major flaw with my current fixture generation is that it does not handle members joining the club halfway through a season. Unfortunately, there is no way to add users halfway through a season and still maintain the integrity of the league table. However, I would look to borrow features from league generation software by adding various options of how to deal with adding new users. This would include the current way the system functions but would also feature the ability for other plays to join and play halfway through a season. My client (Lewis James) did specify that the system should not handle users that join the club hallway through the season, but I feel that looking into more options for handling members joining hallway through a season would add to the strength of the system.

On balance, I feel that my system performs strongly when compared to other commercially available systems, considering that my system is bespoke whereas the aforementioned systems are not.

Evaluation of the Final System

On the whole I think my system ultimately works very well. The system performs everything required of it and meets my objectives set. I think I have particularly succeeded in creating a good looking chess game that has chess computers that are robust. I think the ability to switch between multiple board styles really adds to the quality of the user experience. I was also able to develop a strong alpha-beta pruned minimax search that used different evaluation functions –

including a neural network- to play chess. From games played against the computer, I know that my system has a wide range of difficulty settings that will suit a wide range of players. I also think my implementation of a server based online chess game actually works really well and leads to a very smooth chess experience that does not feel much different from playing against a person offline. I am very pleased that I decided to add a text file that stores common chess opening moves from high level players, as this really improved the overall quality of my chess computers. My networking code for the server also works very well as it allows an unlimited number of clients to perform tasks. However, all these tasks are executed in the order that they are received in which is very useful as it reduces the likelihood of the server attempting to operate on data that has been deleted. From testing, users reported that the system was very easy to use and understand and that the actual chess functionality worked very well.

My implementation of the chess club section meets my client's requirements and hence I think I have succeeded. However, the system is certainly not perfect. For example, my chess computer is only able to search to a depth of 3. To enhance my chess computer, I would look to develop a more efficient minimax search that uses more advanced move ordering and also faster generation of all legal moves from a board state. This would allow the computer to search more board states, hence making it more likely to make a better move. I would also look to implement a quiescence search, Zobrist hashing and iterative deepening alongside a more optimised code base to further enhance the speed and quality of my chess program. I also feel that the artificial neural network I developed would have benefited from more training time. If I were to retrain the network, I would use only grandmaster chess games, as I found that the CCRL database used contained games with computers that were of a lower quality than expected. I would also have considered training an additional neural network to mimic one of the top chess engines – Stockfish. Unfortunately, I am severely limited by the processing power of my laptop and hence I would probably have had to use a cloud service and a deep learning library (such as TensorFlow) to train such a model.

I feel that my online play works pretty well, but I would in future look to allow some sort of chat functionality between players to make the experience more sociable.

Whilst I feel that the chess club section of my app fulfils my client's requirements, I do think that fixture generation and viewing is imperfect and inflexible. To alter this, I would have to research scheduling methods that are more suitable when it comes to allowing players to join a league competition midway through a season. Although the problem is that any system that allows players to join players midway through a season would have to make a trade-off between inclusivity for the user joining and integrity of the points system. Hence, I would maybe to look to segregate the season into the fixtures generated at the start of the season and also friendly matches against players who joined the league halfway through a season.

Initially, I created a prototype for my system that incorporated most of the major offline features of the final system. This was based of the plan presented in my design. Following feedback from my client, I made minor alterations to the design of the system. I decided that I was going to switch the system focus from allowing use on any platform to being a Windows exclusive system. This decision was made to allow the system to make use of the best chess engine – Stockfish - as the level five computer opponent. I feel that this decision worked out very well as it led to a very strong computer opponent which I feel significantly enhances the quality of my system. It came with no real drawback as my client confirmed that all members of the club had Windows computer systems. One of the other big differences was that users were not to play their club chess fixtures using the system. This feature was requested by the club leader as

he felt that it gave the users much more choice about how they could play their games. He felt that implementing such a feature would possibly lead to confusion when / if the West Cross Chess Club meets in person again. The whole point of the system is to meet the demands of my client, and hence this change is obviously better for the system. If I were to make future improvements to the system, I may look for a way for users to play games using the system or to report game results using the system. Although, this change would have to be discussed with the leader.

On the whole, I did not make that many changes from my design to the actual implementation. However, all the changes I did make served to make the system better. The fact that I made few changes demonstrates a strong performance in the design and investigation process.

Perhaps the strongest indication of the system's suitability and success is my testing (please view my testing section for more detail). I performed a large number of tests that thoroughly tested all aspects of the system. These acceptance test covered every single aim, objective and success criterium. The fact that my system passed every single test demonstrates very strongly that the system meets all of its requirements. These requirements were agreed by my client to be what the system needs to do in order to be successful. Given that all of these requirements have been met, I would say that the system is successful. Furthermore, during beta testing I gave my system to the club leader, five club members and five friends who have nothing to do with the club. All feedback obtained was highly positive, with all users agreeing that the system did everything it needed to and did it well. Lewis James (my client) gave very positive feedback on the system in beta testing. I asked Lewis James "Does the system fully meet your system requirements and perform everything you want it to?", and he responded with "Absolutely, it does everything I want it do and it does it very well.". This extremely positive feedback clearly shows that the system meets all of the agreed success criteria and system functionality. More importantly, it shows that my system does everything my client and target audience want it to (and does it well). Therefore, I think that the actual system is clearly a success.

However, in testing (both alpha and beta) I did find a small number of minor visual bugs. I would fix these in future to enhance the system and make the user experience smoother. However, these bugs were only visual and hence do not actually impact the system functionality in any way. Beta testing also highlighted that users found that the system was very easy and enjoyable to use. This suggests that my form layout and design process was highly effective.

As mentioned, my testing comprehensively showed that my final system meets all aims, objectives and success criteria. Whilst I certainly met the objectives and success criteria for fixture generation, I feel that a stronger solution would have multiple options for how to handle users joining the club midway through the season. I also feel that adding the ability for users to player their club fixtures through the club system would make for a stronger system (although as mentioned, my client specifically requested that this feature would not be included). However, I feel certain that I have met these objectives although were I to spend more time on the system, I would look to enhance the solutions to these objectives in future.

Evaluation of Own Performance

On the whole, I feel that I have produced a really high-quality piece of software that fulfils all of my client's requirements for the system and that meets my design vision as well. Whilst the ultimate result is very strong, I do feel that I planned my project poorly. When executing the design and investigation phase of the project, I failed to consider just how long it would take to

develop my system. I have spent a large amount of time coding my solution to the problem – certainly significantly over 100 hours. In hindsight, I should have set myself a project that was less ambitious but still fulfilled all the coursework criteria. The project I ultimately designed was extremely complex and was significantly more advanced than the coursework required to have the potential to achieve full marks. In short, I failed to recognise that I had too many features that were far too complex for the requirements of the coursework. However, I was ultimately able to produce a complete and strong solution that fulfilled all requirements which I feel indicates my strong level of planning (as I would have been unable to create such a complicated system without a strong and thorough plan and design). I feel that the actual design of the system was very thorough as I altered very little following the post-prototype refinement. This demonstrates that I was able to accurately capture the client's requirements for such a system. And given that my system performs so well, I was clearly able to implement these features (even if they took a lot of time to produce).

Creating the system was made much easier by the fact that I had designed all of the validation details, data structures and algorithms to be used before actually starting coding. I would say that I have become a stronger computer scientist over the process of the coursework – developing high fluency in Java and also learning vital communication and design skills. I would say that I showed my ability to work to a deadline as I was able to meet all internal coursework deadlines. If I were to conduct such a project in future, I would devote more time to assessing the suitability of a project both by its achievability and by how long it would realistically take to complete to a high standard. I feel that I communicated well with my client and target audience – obtaining a detailed description of the system I was to make. This communication clearly worked as very few changes were made to the design in the post-prototype refinement of design phase of the project.

Ultimately, I was able to produce a successful system that meets all client and coursework requirements. It is my view that the system is a very good system and one that is enjoyable and suitable for users. As a result of this, I conclude that I ultimately also performed well throughout the process as my successful creation of such a complicated system was only possible through strong performance at every stage of the coursework. Although with hindsight, I do concede that I designed a system that was too ambitious to achieve in a reasonable period of time. Despite this, I feel I showed my ability to work hard and make the most of my time as I was able to achieve all features of the system – even though it took a lot of time.