|  |
| --- |
| Navy Chess Game Final Report |
| COMP 8046 |

|  |
| --- |
| Xinghua Wei – A00978597  10-5-2021 |

Table of Contents

[1. Introduction 4](#_Toc83231982)

[1.1. Student Background 4](#_Toc83231983)

[1.2. Project Description 5](#_Toc83231984)

[1.2.1. Essential Problems 6](#_Toc83231985)

[1.2.2. Goals and Objectives 6](#_Toc83231986)

[2. Body 7](#_Toc83231987)

[2.1. Background 7](#_Toc83231988)

[2.2. Project Statement 7](#_Toc83231989)

[2.3. Possible Alternative Solutions 8](#_Toc83231990)

[2.4. Chosen Solution 9](#_Toc83231991)

[2.5. Details of Design and Development 11](#_Toc83231992)

[2.5.1 Game Rules Setup 11](#_Toc83231993)

[2.5.2 MiniMax Algorithm Design Idea 12](#_Toc83231994)

[2.5.3 Alpha-Beta Pruning Design Idea 13](#_Toc83231995)

[2.5.4 End-to-End Encrpytion 16](#_Toc83231996)

[2.5.5 Two-Factor Authenticaiton 17](#_Toc83231997)

[2.6. Detail Designs 17](#_Toc83231998)

[2.6.1 Chess Unit 17](#_Toc83231999)

[2.6.2 Map Objects 18](#_Toc83232000)

[2.6.3 Map Board 20](#_Toc83232001)

[2.6.4 Units Functionalities 22](#_Toc83232002)

[2.6.5 AI Design 25](#_Toc83232003)

[2.6.6 Chat and Encrpytion Funcition 27](#_Toc83232004)

[2.6.7 SMTP Server 29](#_Toc83232005)

[2.6.8 Player Register and Log in 30](#_Toc83232006)

[2.7. Diagrams 31](#_Toc83232007)

[2.7.1 Map Objects Class Diagram 31](#_Toc83232008)

[2.7.2 Chess Units Class Diagram 32](#_Toc83232009)

[2.8. Testing Details and Results 33](#_Toc83232010)

[2.9. Implications of Implementation 47](#_Toc83232011)

[2.10. Innovation 49](#_Toc83232012)

[2.11. Complexity 51](#_Toc83232013)

[2.12. Research in New Technologies 52](#_Toc83232014)

[2.13. Future Enhancements 53](#_Toc83232015)

[2.14. Timeline and Milestones 54](#_Toc83232016)

[3. Conclusion 55](#_Toc83232017)

[3.1. Lessons Learned 56](#_Toc83232018)

[3.2. Closing Remarks 57](#_Toc83232019)

[4. Appendix 57](#_Toc83232020)

[4.1. Approved Proposal 57](#_Toc83232021)

[4.2. Project Supervisor Approvals 82](#_Toc83232022)

[5. References 83](#_Toc83232023)

[6. Change Log 83](#_Toc83232024)

# Introduction

## Student Background

I am a student enrolled in the BCIT CST Btech program major in the network security application development option. I graduated from the BCIT CST diploma program Database option.

During the BCIT CST diploma program, I had one year of programming experience with JAVA and some programming experience with C, C++, JavaScript, HTML5 and MySQL. I have accomplished a few website applications, such as a website to encourage users to save food and a website that supports selling products online. I also had an opportunity to create an Android mobile application that indicates crime in the current user area.

During the ongoing network security application development option, I learned some basic ideas and functionalities behind TCP/IP. Also, I learned how to create simple TCP/IP communication applications, a simple firewall using Shell Script. Now I am learning ideas and methods to do ethical hacking and hiding data while transmitting data. Project Description

Education

British Columbia Institute of Technology

Program: Computer Systems Technology 2016-2018

* Graduate from CST Database option

Program: Computer Network Security Application Development Ongoing

Skills

* Network Security

Understand basic ideas about network security, network applications using Python, Shell Script on the Linux platform. Understand basic socket, TCP/IP communication using Python to implement some IPv4 client/server applications.

* Website

Implement and design websites with HTML5 and JavaScript. I have completed five team projects successfully. Also, I have programming experience using Bootstrap.

* Database

I have half-year experience with Oracle, T-SQL, MySQL, Visual studio, and data mining with different software such as Hive and R studio.

## Project Description

I propose to create a multiplayer chess game call Navy Chess, a two-player board game. There are no existing games on any platforms since it is an old-fashion board game to my current knowledge and research. I want to create a digital version of Navy Chess for Windows platforms where users can download it and connect to other players via the players' IP address and port using Python socket.

Besides the game itself, a live chat function with encrypt data transmit underneath that will provide a safer and more reliable gaming environment. To let the signal player enjoy the game, Two different AI levels for this game with Alpha-beta pruning and Minimax allow the signal player to start a match without waiting for another player. In terms of player connections, there will be a two-factor authentication applied to the game that whenever play A wants to connect to player B, there will be a request to authenticate it via an external email system.

### Essential Problems

For this project, I tried to digitalize a board chess game into a multiplayer game and there are some essential problems I try to solve:

* There are 50 chess units in the game. The variables of each chess object are essential since they need to move, attack, hide and be destroyed.
* I need to solve how the Alpha-beta pruning and Minimax algorithms can fit this chess game model.
* I need to set up a third-party SMTP service to build a two-factor authentication process based on that.
* I need to embed RSA encryption to the chat function.

### Goals and Objectives

Overall, this project has four main goals:

* To develop a fully functional digitalized 2D Navy Chess game for the Windows platform using Python.
* To research and develop a single-player mode with Alpha-beta pruning and Minimax.
* To develop a secure live chat with a data transmitting function to enhance the gamer's gaming environment by using end-to-end encryption.
* Two-factor authentication function to secure player register, login and communicate.

# Body

## Background

Navy Chess is a two-player board game, and it was derived from another board game SanJun Chess. To my current knowledge and research, both games are no longer online on any platform since it is an old-fashion board game, and most people treat it as antique.

The main combat units include aircraft carriers, battleships, torpedoes (same as all types of ships), mine blasters (lightning strike ships), and submarines (the only ships that can defeat all types of ships are defeated by lightning strike ships). The secondary combat units include three missile boats (which can knock down Aircraft carriers), torpedo boats (which can knock down submarines), mines (to protect military flags at sea), and three minesweepers (which can knock down mines). In addition, because there are two anti-submarine aircraft groups (anti-submarine aircraft and bombers) walking on the chessboard with black dotted lines (courses), Navy chess has produced the concept of three-dimensional space combat because of the submarine, anti-submarine aircraft, and bomber group.

## Project Statement

The problem is that the Navy Chess Game people can play their child games online without searching for such games in local stores. It has several modes suck as dark mode, normal multiplayer mode, and singal player mode, which solves the different needs of different players. The data encryption and the Two-factor authentication functions solve the potential user credential loss and could enlighten any children who could play the game with email verification.

## Possible Alternative Solutions

There are some possible alternative solutions I can use for this project.

* I could use Java to build this program and GUI. Java is a high-level, object-oriented programing language, and it is generally faster compared to Python. Object-oriented means that it is suitable for a project that needs to create dozens of objects. Besides, Java can create GUI applications, but a little bit more complex than Python, in my opinion.
* I could use JavaScript to build this program and GUI. JavaScript is also a high-level, object-oriented language suitable for building applications that need a GUI.
* I could build this program by Android Studio. Android Studio is the official integrated development environment for Google's Android operating system. It is easy to create GUI by Android Studio.
* Instead of Minimax and the Alpha-beta pruning, I could build a database and use the method called Opening Move Database because many combinations of the first few initial moves on the board are known to establish a good position. The program can use this database at the beginning of play. Instead of performing the usual local search, it simply checks whether the moves performed so far fit into a given opening strategy, and then plays the appropriate response.
* I could also use machine learning to train the AI to play. Machine learning is the study of computer algorithms that can automatically improve through experience by using data.
* I could use DES, AES or other ciphers like substitution, transposition. Ciphers like substitution are easier to achieve. DES and AES are similar to RSA, and AES is the most commonly used encryption method.
* I could use Qt for Python GUI design. Qt for Python is another popular GUI toolkit. It is extracted from the library of Qt. PyQt is the product of the combination of the Python language and the Qt library. PyQt is coming with the Qt Builder. We will use it to get the python code from the Qt Creator. With the support of a qt designer, we can build a GUI, and then we can get python code for that GUI. PyQt supports all platforms, including Windows, macOS and UNIX. PyQt can be used to create stylish GUIs, a modern and portable python framework.
* I could use another third-party SMTP service like Google, Amazon. SMTP, Simple Mail Transfer Protocol, is the industry standard to send emails on the internet. It uses proper authentication, which increases the chances that your emails will actually get delivered to the users' inbox.

## Chosen Solution

Although there are many alternatives I could use in terms of language choice and technologies, I still want to use the current setup.

* Python is an interpreted high-level programming language. It is designed to have more excellent readability, and it is also object-oriented. Python is easier to read and use than Java, and Python has tons of libraries that I do not need to start from scratches, such as cipher library and standard GUI library. JavaScript is not very suitable for build windows applications compared to Python. It works better if I want to build a web application. Android Studio is designed to build mobile applications, and my purpose is to build a desktop application. Besides, I am more familiar with Python than Java or other languages. Therefore, I choose Python to build this project.
* I choose to use Minimax and Alpha-beta pruning because Minimax is the most popular searching algorithm for chess games. Minimax is suitable for chess games like this project which does not need to search too deep. Opening Move Database requires a large database that better have hundreds or thousands of openings and following movement stored, and it does not worth the time to build such large database for this new game manually. Machine learning is actually a good alternative choice, but I have no experience building such a learning model for an AI, and there is a high possibility for me not able to finish this part of the program in this project. Therefore, Minimax is achievable, robust and feasible to use in a new chess game.
* For the data encryption, I choose to use RSA encryption. First of all, ciphers like substitution are too easy to decode. Also, I do not want to use symmetric encryption, which uses the same secret key to encrypt and decrypt information. Asymmetric encryption uses a pair of keys to encrypt information, and it is more secure than symmetric encryption. AES, Advanced Encryption Standard, is an asymmetric encryption algorithm that encrypts fixed blocks of data. Although AES is more popular, RSA works better when transmitting data between two separate endpoints, and it fits the game.
* I choose to use Tencent SMTP service mainly because I have no use of Tencent account, so I do not need to worry about account security when setting up the SMTP service. Also, the Tencent SMTP service is easier to use than it only requires an account. Unlike Google SMTP, which requires allowing "less secure" apps access in order for my SMTP settings to work.

## Details of Design and Development

### 2.5.1 Game Rules Setup

The chart below shows the settings of each chess unit of the game.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Defence** | **Attack** | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Aircraft carriers** | **Battleship** | **cruiser** | **Destroyers** | **Strikers** | **Mine blasters** | **Submarines** | **Torpedo** | **Mine** | **Flagship** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Aircraft carriers** | Equal | Defence Win | DefenceWin | DefenceWin | DefenceWin | DefenceWin | AttackWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Battleship** | AttackWin | Equal | DefenceWin | DefenceWin | DefenceWin | DefenceWin | AttackWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **cruiser** | AttackWin | AttackWin | Equal | DefenceWin | DefenceWin | DefenceWin | AttackWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Destroyers** | AttackWin | AttackWin | AttackWin | Equal | DefenceWin | DefenceWin | AttackWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Strikers** | AttackWin | AttackWin | AttackWin | AttackWin | Equal | DefenceWin | DefenceWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Mine blasters** | AttackWin | AttackWin | AttackWin | AttackWin | AttackWin | Equal | AttackWin | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Submarines** | DefenceWin | DefenceWin | DefenceWin | DefenceWin | AttackWin | DefenceWin | Equal | Equal | — | DefenceWin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Torpedo** | Equal | Equal | Equal | Equal | Equal | Equal | Equal | Equal | — | Equal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Mine** | Equal | Equal | Equal | Equal | Equal | AttackWin | Equal | Equal | — | Equal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |
| **Flagship** | AttackWin | AttackWin | AttackWin | AttackWin | AttackWin | AttackWin | AttackWin | Equal | — | Equal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |

### 2.5.2 MiniMax Algorithm Design Idea

The miniMax algorithm will be used for this game. It is a search algorithm that allows the program to look ahead at possible future positions before deciding what move it wants to make in the current position. Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. It is widely used in two player turn-based games like chess games.

In Minimax, the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible, while the minimizer tries to do the opposite and get the lowest score possible. Every board state has a value associated with it. In a given state, if the maximizer has the upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper hand in that board state, it will tend to be negative. To determine the value of the board, I give different values to different types of chess units. For example, the Flagship has the highest value on board, and a small ship only has a value of 10. AI, the maximizer, will try to calculate the maximum onboard value based on chess values on board to decide the next movement. The image blows give an example of different chess units with different values. The "self.value" variable will be used to maximize board values by AI.

Text

Description automatically generated Text

Description automatically generated

But MiniMax itself is kind of slow. The searching and deciding process could be optimized by Alpha-beta pruning.

### 2.5.3 Alpha-Beta Pruning Design Idea

Alpha-beta pruning is a modified version of the miniMax algorithm. It is an optimization technique that will fast the miniMax process. With Alpha-beta pruning, the program will not need to check each node of the game to compute the correct decision. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree, which need not be searched because a better move is already available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely Alpha and Beta.

For example, there are two players that one is called Max, and the other is called Min. Each terminal node represents a chess unit, and each node has a value. The red node represents Max, and the blue node represents Min. Max will try to get the Maximum score, where Min will try to get the minimum possible score. Also, the Alpha will be added and means negative infinite. Beta means positive infinite.

Shape

Description automatically generated

**Step 1:**

Max will start the move from node A that Alpha is negative infinity, and Beta is positive infinity. The values will be passed down to nodes B and D.

At node D, the value of node H and I are compared. Max will get the maximum value:

* For D: max(-1:4) = 4

**Step 2:**

It is Min's turn that Min will compare the beta value with the current node B value at node B. Min will get the minimum value:

For B: min(4, positive infinite) = 4

**Step 3:**

At node E, Max will take its turn, and the value of Alpha will change. The current value of Alpha will be compared with 5,

* For alpha: max(-∞, 5) = 5,

Therefore, at node E, Alpha is 5 and Beta is 3, where Alpha is larger than Beta. So the right successor of node E will be pruned, and the algorithm will not traverse it, and the value at node E will be 5.

This is an example of how a node will be pruned. It will still use the chess unit value, such as that Flagships have a value of 2000 and a small ship has a value of 10. Then the Alpha-beta algorithm will apply the same methodology to the other nodes. In the real game, Alpha represents the highest value that the maximizer can presently promise at that level or higher, whereas Beta represents the best value that the minimizer can currently guarantee at that level or higher. The Alpha will be set to a big negative number, whereas the Beta will be set to a large positive value in this project. The AI will discover the highest score on the board, and if the following movement can create more scores than the highest score on the board, the AI will record it. The AI will then assign the Alpha the greatest value between the alpha value and the current best score on board. This node will be the optimum movement to act if the Alpha score has now surpassed the Beta score.

### 2.5.4 End-to-End Encrpytion

End to end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

There are two types of encryption, one is symmetric encryption, and the other is asymmetric encryption. I will use asymmetric encryption for this project. In Asymmetric encryption, two types of keys are used, one public key and one private key. The sender and the receiver both have the public key and the private key. The Public Keys are available to both sender and receiver to share public keys before communication starts. The sender uses the receiver's public key to encrypt the messages to be sent, and the receiver uses both its public and private keys to decrypt the messages. The private key to the receiver is available only with the receiver and no one else.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

### 2.5.5 Two-Factor Authenticaiton

Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

The client-side player will need to use a valid email account to connect to the server-side player for this project. First, the server will use a proper SMTP service provided by a third-party organization. Then, one or two players will go through a setup process that first configures his account to send his email a code by text. At the end of the setup, they will have a randomly generated authorization code to their email. Once done, the player will login with the registered email and receive an authentication code in the email when he wants to join the game.

## Detail Designs

### 2.6.1 Chess Unit

There are 50 chess units in a single game. Each unit has its own purpose and value. Therefore, each chess unit will be created as different classes. Every chess unit will be assigned an "order" variable representing its rank value; a "value" variable representing how much score can one player earn if the player eliminates the unit; a "side" variable to different if a specific unit is at player A's side or player B's side. Same units on different sides will show in different colours. For example, the image below shows what is inside a Flag class:

Text

Description automatically generated

A flag is a target to win the game. The opponent will win the game if he destroys my flag. The flag cannot move or attack, so its order is 0. It can directly let a player win the game, and it has a maximum value among all the chess units, so its value is 2000. 2000 score value in a single game is unbeatable; therefore, the opponent will win. The sides variable shows that if a flag is at A's side, it will appear in orange colour. Otherwise, it will appear in blue at B's side.

### 2.6.2 Map Objects

The game map overall is a 5 \* 12 grid. It will be separated into two 6 \* 5 grids for each player. Each side will contain 25 chess units and 5 camps. Chess units are designed to a class object, Post class, which is generally separated into two steps. For example, drawSkeleton function and reverseDraw will draw player A's side and Player's B side. That is for the convenience of drawing different groups of chess units on the board.

Text

Description automatically generated

Camps are designed for players to protect any movable chess units from being destroyed by the enemy. Camps are designed to a class object as well, and they inherit the chess units class, which is Post class. It will also draw all camps on players A's side and B's side separately.

A picture containing calendar

Description automatically generated

Also, the headquarters class is created for flagships to stay to represent a target to win the game. It also inherits the Post class and draws headquarters separately.

Text

Description automatically generated

### 2.6.3 Map Board

To draw the map board that units can move through the board, I choose Python Tkinter to accomplish that. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's standard GUI. Also, the Tkinter package is the standard Python interface to the Tk GUI toolkit, and it is available on most Unix platforms and Windows systems.

There are 5 vertical grids and 12 horizontal grids; therefore, a 5 \* 12 grids in total, including 50 chess units, 10 camps and 4 headquarters. Headquarters will also be a location to put flagship and chess units at the beginning of the game, so, visually, 60 objects are on the map board. The image below shows the game board and 60 objects.

Diagram

Description automatically generated

To draw all the objects, I will create all objects layer by layer. For example, I will draw the first layer at the bottom. Therefore, a Post object will be created first, then a Headquarters object, followed by another Post object, the second Headquarters object, and the third Post object. The image below shows that each layer contains 5 objects and will be store in a list.

Text

Description automatically generated

This means the first layer of players A's side is created, as in the image below.

A screenshot of a computer

Description automatically generated with medium confidence

After drawing all the objects, objects need to be connected with lines. The lines are created with create\_line function from the Tkinter library. It allows me to draw lines between two coordinates. For example,  allows me to draw a wider line from location (50, 110) to location (50, 690). Overall, the board is designed to be a 600 \* 800 space. After drawing the lines, objects will appear to be connected as in the image below.



### 2.6.4 Units Functionalities

To make sure some chess units are in the proper location, I will create a check function to check if some chess units are in the correct location. For example, flagships must be put in one of the headquarters. Therefore, in this 5 \* 12 grids board, I need to check the soundings of the headquarters and the flagships in its layer. The Flagship has an order of 0, so I will check the headquarters if they contain a rank order of 0. If not, an error message will be showed. The image below gives an example of flagships location checking.

Text

Description automatically generated

The image below shows that I was trying to put one of the flagships out of the headquarters (Movement show as red rectangles), and an error message appeared.

Diagram, schematic

Description automatically generated

To track players' mouse click activity, I have a mousePressed function to react to different clicks. I will divide the click button into different 2D areas. For example, the landing page will show several different buttons. Each button will have a react area. The image below shows an example. If the click happens in the (200,500) and (400,580) area, which creates a rectangle area, a certain mode flag will be triggered.

Text

Description automatically generated

To move a chess unit, players need to click a chess unit first. Once a player clicks a rectangle area of a chess unit, the chess will be highlighted in a green rectangle. Then the getLocation function will be called first to record its location on the game board. If the move does not violate the game rules, the player can select a second spot to move to. The second spot will also be recorded its location first; then, the first selected chess unit will replace the second selected chess unit location.

Text

Description automatically generated

The image above shows that a chess unit is selected, and the select count will be set to 0. After selecting the second spot, the select count will be set to 1. If the movement passes the game rule check (pass the isLegal function check), the movement will take action.

Text

Description automatically generated

If the chess unit attacks (move to) an opponent's chess unit, the program will check if the game-winning condition is met or not first. That is, if player A's chess unit is at the location of player B's flagship location, player A wins. The following image explains that if a chess unit appears at the location of the opponent's chess unit of the rank order of 0, that player wins. If the winning condition is not met, call the contact function for regular chess unit contact.

Text

Description automatically generated

Text

Description automatically generated

The image above shows that each chess unit has a rank order, and an order 9 unit will destroy an order 8 unit, as I mentioned before. This program checks the order of two contacted units and limits the lower order one, and the winning chess unit will replace the location of the lost one.

### 2.6.5 AI Design

For this project, the Minimax algorithm and the alpha-beta pruning will be used for AI to allow users to play single-player mode. The miniMax algorithm will be used for this game. It is a search algorithm that allows the program to look ahead at possible future positions before deciding what move it wants to make in the current position.

For example, there are two players that one is called Max, and the other is called Min. AI will seek to maximize the board score where player move seeks to minimize the board score. To achieve that, the program will need to calculate board scores. As I mentioned before, each chess unit has a value that the higher-ordered unit will have a higher score value.

Text

Description automatically generated

The above image shows that the board score will be added if a player has taken a unit. If the player lost a unit, the score would be minus according to the unit value.

Also, to use the algorithms, I need to get the largest unit value on the current board. The image below shows that the program will scan the entire board and find the largest existing unit value because that will be the top priority target for the AI.

Text

Description automatically generated

Alpha-beta pruning is a modified version of the miniMax algorithm. It is an optimization technique that will fast the miniMax process. To use this algorithm, I need to let it find the more promising subtree on the board. Thus, the "more promising" subtree will be the node with the largest onboard value unit. I also need an alpha value and a beta value. Alpha is the best value that the maximizer currently can guarantee at that level or above, and Beta is the best value that the minimizer currently can guarantee at that level or above. In this project, the Alpha will be set to -100000, and the Beta will be set to 100000. The AI will find the best score on board; if the next movement can generate more scores than the current best score on board, AI will record this movement. Then the AI will find the largest value between the alpha value and the current best score on board and assign it to the Alpha. If the Alpha is now greater than the beta score, this node will be the best movement to act.

Text

Description automatically generated

There is also a depth value for the algorithm, and I use that to split the single-player mode into easy and hard modes. The difference is that the easy mode will set the max depth to 2, while the hard mode will be 4.

A screen shot of a computer screen

Description automatically generated with low confidence

### 2.6.6 Chat and Encrpytion Funcition

The chat function allows players to communicate with each other over a channel. The message will be encrypted with RSA encryption. RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys, i.e. Public Key and Private Key. As the name describes, the public key is given to everyone, and the private key is kept private.

A pair of RSA keys are generated using the standard RSA library in Python. In this project, it will generate a 1024 bits key.



Then the public key will be encoded in UTF-8 and send to the client side. If the client-side receives the message and finds out the message's first component is a public key, it will decode and verify it and send its public key to the server.

A screenshot of a computer

Description automatically generated with medium confidence

After the server gets the client's public key, it will decode and verify it and add the client's key to every message sent to the client side.

Text

Description automatically generated

Below is the diagram of the encryption process.

Diagram

Description automatically generated

### 2.6.7 SMTP Server

I choose to use a third-party SMTP service to achieve the functionality of the two-factor authentication. I choose to use the Tencent SMTP server since it is more convenient to send emails. I need to register a valid Tencent QQ account, a valid SMTP server address, and a valid SMTP port to set up the service.

The image below shows all elements to set up the SMTP server: a Tencent account and password, server address and server port.

Text

Description automatically generated

Then I need to create a random verification code for each request. I choose to use the Python Random library to generate a six-digit verification code for each request.



Then the verification code will be embedded into the body part of each email and send to the player's emails.



### 2.6.8 Player Register and Log in

Players can register and log in to the game via two-factor authentication. I created a JSON format file to store player login information to the local JSON file. As the image shown below, players will have a valid email address, passwords and verification code stored in the local file. If a player chooses to register in the first step, the SMTP server will send a verification email with a verification code inside.

A screenshot of a computer

Description automatically generated with medium confidence

Once a player is registered, he will have a JSON file with his email and password for logging into the game. Every time the player tries to log in, the program will check whether the user entered password matches the email and password in the JSON file.



The image below explains the process.

Diagram

Description automatically generated

## Diagrams

### 2.7.1 Map Objects Class Diagram

Class diagrams is the diagram in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects.

Diagram

Description automatically generated

### 2.7.2 Chess Units Class Diagram

Below image will be a example of class diagram of a chess unit. Each chess unit will have same variable names but different values.

Diagram

Description automatically generated

## Testing Details and Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 1 | Verify landing page display properly | The landing page should display 3 buttons: Single Player, Multiplayer, Dark Mode | Pass |

Result:

Graphical user interface, diagram, text, application, chat or text message

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 2 | Verify landing page transfer to helper page | Players should enter and see the helper manual after pressing the "H" button on the landing page | Pass |

Result:

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 3 | Verify helper page transfer to the next helper page | Players should enter and see the second helper manual page after pressing the "N" button on the first | Pass |

Result:

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 4 | Verify helper page transfer back to the landing page | Players should enter and see the landing page after pressing the "R" button in the second help manual. | Pass |

Result:

A picture containing logo

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 5 | Verify transfer from landing page to single-player mode select page | Players should enter and see the single-player page and see easy and hard mode choices | Pass |

Result:

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 6 | Verify single-player units set up | Players should see the game map, chess units and should be able to move chess units | Pass |

Result:

A picture containing text, sky, clock

Description automatically generated Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 7 | Verify single-player units set up rules | Players should see the game map, chess units and should be able to move chess units only to the allowed positions | Pass |

Result:

A picture containing text, sign

Description automatically generated

A picture containing diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 8 | Verify single player start button | Players should see the game start and be able to start attacking opponent units | Pass |

Result:

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 9 | Verify single-player opponent setup | Players should see the opponent chess unit after starting the game | Pass |

Result:

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 10 | Verify single-player timer | Players should see the timer after starting the game | Pass |

Result:

A screenshot of a computer

Description automatically generated with low confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 11 | Verify single player that AI can win the game | Players should lose the game if the AI destroys their FlagShips. | Pass |

Result:



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 12 | Verify single player server | User should see the server start and see server IP and port open | Pass |

Result:

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 13 | Verify single-player server-client connection | User should see the server start and connect to the server using the showing IP address and port | Pass |

Result:

Text

Description automatically generated

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 14 | Verify single-player user login | The user should enter the correct registered password and email address to login | Pass |

Result:





|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 15 | Verify single-player two-factor authentication code | The user should receive a verification code after entering the correct password and email address | Pass |

Result:

A picture containing shape

Description automatically generated



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 16 | Verify player register function | Players should be able to register accounts following the instructions showing on the terminal | Pass |

Result:

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 17 | Verify multiplayer player connection | The second player should enter the multiplayer game | Pass |

Result:

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 18 | Verify multiplayer player ready button | Both players should see the ready button and will only start the game when both players click the ready button | Pass |

Result:

Diagram

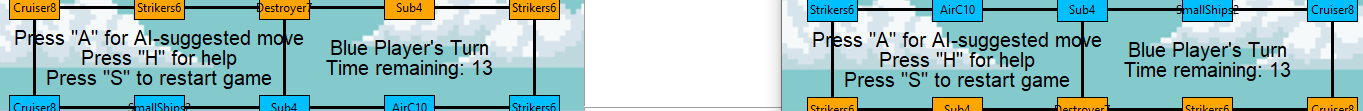
Description automatically generated

Diagram

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 19 | Verify multiplayer player timer | After starting the game, both players should see a timer | Pass |

Result:



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 20 | Verify multiplayer AI suggestions | Both players should be able to see AI suggestions when pressing A button | Pass |

Result:

A picture containing text, clock, sign

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 21 | Verify multiplayer player movements | Both players should be able to attack, move their units | Pass |

Result:

A picture containing text, clock, screenshot

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 22 | Verify multiplayer player helper function | Both players should be able to check the help manual anytime after pressing the H button in the game | Pass |

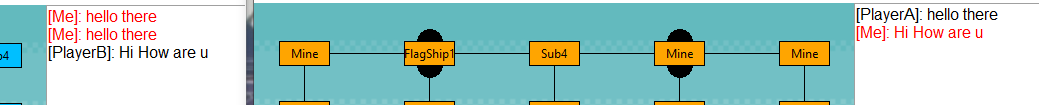
Result:

Graphical user interface, application, Word

Description automatically generated

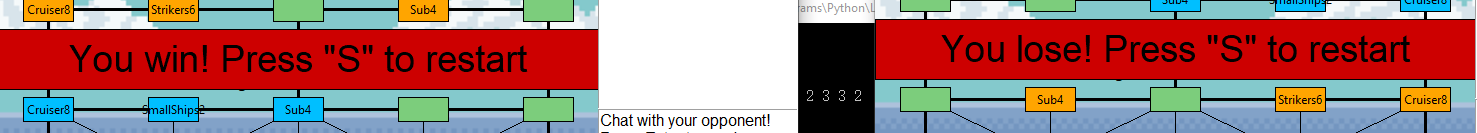
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 23 | Verify multiplayer chat function | Both players should be able to communicate through the chat window on their right | Pass |

Result:



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 24 | Verify multiplayer win and lose | If one player wins the game, the other should lose the game | Pass |

Result:



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expect Result | Pass/Fail |
| 25 | Verify data encryption | Using wireshark monitoring data transfer, it should be encrypted messages instead of plaintext | Pass |

Result:

Graphical user interface, text, Word

Description automatically generated

## Implications of Implementation

In this project, I digitalize a board chess game using Python. For the graphic part, I used Python Tkinter, which is a GUI toolkit that allows me to draw maps, buttons, chess units. For the client-server part of the project, I used the UDP and basic socket library in Python. This allows me to connect players to the server, and UDP ensures better performance in multiplayer game mode. By using UDP, the program can synchronize both players' chess board movements in less than 1.5 seconds.

As for the single-player part, AI is built by MiniaMax and Alpha-beta pruning algorithms. They are search algorithms that allow AI to find the best possible movement. For easy single-player mode, MiniMax with Alpha-beta pruning allows AI to make a move within 1.5 seconds. Considering the algorithm depth setting for easy mode is only 2, a player will sometimes see AI move its units back and forth in some rows even the players give AI a chance to win. Unlike easy mode, the algorithm depth setting for hard mode is 4. AI will make its next move in less than 2 seconds, but AI will always aim to the most important target if AI has a chance. For example, in easy mode, AI will take about 20 turns to win the game even it can win the game in 14 turns. AI will take only 14 to 15 turns in hard mode if AI can win the game in 14 turns. In the image below, it can be clearly seen that the AI's Battleship is aiming directly at my Flagships without any hesitation in hard mode.

Diagram, schematic

Description automatically generated

The two-factor authentication is achieved by using the SMTP service provided by Tencent, which allows me to send and receive verification codes. Although I use Tencent's SMTP service, players can still get their verification code in less than 10 seconds.

## Innovation

**Play Mode**: The current existing game in the Google Play store is four players mode. There will be two players modes for this game.

**Chat**: There will be a chat function that players can interact with their opponents during the game.

**Encrypted data transfer**: I will also add a live chat function for the game so players can communicate. But data transfer during the live chat will be encrypted by end-to-end encryption.

End-to-end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

**AI**: There will be a single-player mode for the game that players can match with AI. The AI algorithm will use miniMax with heuristics and alpha-beta pruning.

The MiniMax algorithm is a recursive algorithm usually used in decision-making games such as chess games. MiniMax uses recursion to search through the game tree. If two players play a game with the MiniMax algorithm, both the players will fight their opponent player to get the maximum benefit where their opponent gets the minimum benefit.

The heuristic algorithm is to find solutions among all possible solutions. However, it does not guarantee that the best solution will be found; therefore, it can be considered not accurate algorithms. The heuristic algorithm usually finds a solution close to the best one, and it finds it fast.

Alpha-beta pruning is an optimization technique for the minimax algorithm. Because there are 50 chess units for a game, alpha-beta pruning reduces the computation time. This allows the AI to search much faster and even dig deeper into the game tree that it also allows me to implement an easy AI mode and a hard AI mode. Alpha-beta will cut off branches in the game tree that will not be searched because there are better moves that exist.

**Two-factor Authentication**: Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

For this project, the client-side player will need to use a valid Gmail account to connect to the server-side player. If one player enables Two-Factor Authentication for Gmail, the player goes through a setup process that first configures his account to send his phone a code by text or voice. At the end of the setup, he will have an opportunity to add Google Authenticator as an alternative code generator. He will use Google Authenticator to enter a 32-character secret string that he will use to add configuration to this account in the authenticator. Once done, the player will receive an authentication code in the Google Authenticator when he wants to join the game.

## Complexity

**Synchronize issue**: this game needs to solve the synchronize problem. How the game can handle each command, such as move the unit, attack the opponent and sync all the steps so players can be in the same stage, instead of one player already wins the game and the other player is still moving his units. UDP is really fast, and if the program is not properly set up, there could be a situation that one player hasn't finished the movement, but the other player can start moving his units.

**Algorithm**: Because the game contains about 50 units per match, I need to create an algorithm that allows each unit to move, attack as players want. Also, an AI algorithm for the game requires me to research and study how to do AI for a game. It is hard to decide how the algorithm can fit the game.

**Programming skills**: I need to improve my Python skills more because BCIT never really taught us about this language. I would not use Unity for this game but building the game from scratch. So, I need to learn how to GUI programming based on Python. Also, I need to search for how to create links to make the player invitation function work.

**Testing**: Testing is also a big barrier I need to conquer. Because I have no experience testing a multiplayer game build by Python, and I need to test every part of the project.

**Time**: This game requires a student to have proper art design, socket, Python, Python GUI, AI algorithm skills to finish the game within 350 hours. Skills like art design, GUI, AI are fairly new to me. So, after researching and learning some of the skills, I may need to extend my schedule or limit my scope.

**Encryption**: End-to-end encryption is a new subject for me because my experience with encryption is to encrypt data with only one layer of protection, such as RSA encryption or AES encryption. End-to-end requires a secure channel with a combination of encryption methods, such as RSA with AES and SHA. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

**2FA**: Apply two-factor authentication to the game is a challenge. I need to combine the authentication with a TCP socket, Python and an external mobile application. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

## Research in New Technologies

There are few technical challenges for this project. One of which is understanding how Alpha-beta pruning and Minimax can fit the game. Although there are many chess games that already use the algorithms, those games do not have the ability to suicide chess units on purpose. I treat the whole chess map as 12 times 5 list, and the program needs to calculate the highest onboard score every time on AI's turn. The second challenge is how to fit chess units variables into the searching algorithms. Since chess units can suicide in this game, I give every unit a rank order, a health value and a score value. The algorithms will calculate the best possible move best on the highest possible score value it can earn and use the highest possible rank order units to attack. Another challenge is how to add a two-factor authentication function to the game. I need to connect the two-factor authentication process to the client-server connections. Before doing that, I need to research which SMTP service is convenient and robust.

## Future Enhancements

I have tested and verified that every function in this game can perform normally. However, the server is only designed to have maximum 2 people gaming at the same time. In the future, this game could have a game lobby to allow players to create their individual game room that dozens of people can game at the same time.

Second, players need to log in to their registered account every time they join the game. The game could have a function to remember user credentials if players do not want to enter email and passwords on their computer. Besides, the two-factor authentication process can be transformed to a third-party mobile application instead of using email to receive verification codes every time.

## Timeline and Milestones

Chart, waterfall chart

Description automatically generated

Table

Description automatically generated with medium confidence

# Conclusion

In conclusion, the digitalized Navy Chess game is a huge challenge for me. I developed a fully functional digitalized 2D Navy Chess game for the Windows and Linux platforms. I researched and developed AI algorithms for the game so players can match with AI. Chat function and end-to-end encryption are essential for the game's security aspect to prevent potential data leaks and intercept during transmission. Also, players can send a link to their friends so that players can invite another player to his match room.

The Navy Chess game project helped me enhancing many of my skills in terms of software development:

* Software architecture design
* Software API design
* Software testing
* 2D art design
* Project time and schedule management
* Project scope control
* Python programming, Python GUI programming, Python Socket programming
* Networking programming, network security programming
* Decision-making algorithms

My overall skills in programming, algorithms, managing, networking and designing will be improved as I overcome the technical challenges and achieve the innovation part of the project.

## Lessons Learned

The list below shows the knowledge I can get upon the completion of the project and how the acquired knowledge contributes to my expertise development.

* I learned how to use a third-party SMTP service. Third-party SMTP service is very useful. It can accomplish two-factor authentication, but it can also help achieve functions like sending notification emails.
* I learned how to do GUI using Python Tkinter. Tkinter is a really useful Python GUI toolkit. Besides, Tkinter is Python's standard library, and it is easy to use. I will be able to build more Python applications with GUI.
* I learned how to embed encryption into UDP translations. UDP is generally fast but unreliable compare to TCP. But encrypting plaintext or data before sending them out can largely increase the security level of UDP transmission. As a network security option student, knowing more about encryption will be helpful to my further career since suitable encryption will avoid lots of data security problems like the data leak. Besides, one of the applications will be written in Python. As Python is a popular language for network security applications, it is helpful to build different kinds of network apps in the future if we expertise in Python. Moreover, we will use Python encryption libraries in this project. It will be a good practice because knowing these libraries will be useful for us to build our own encryption library.

## Closing Remarks

[Include some concluding remarks regarding the project.]

# Appendix

## Approved Proposal

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart, treemap chart

Description automatically generated

Diagram

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

Diagram

Description automatically generated with medium confidence

Text, letter

Description automatically generated

Text

Description automatically generated

Text, application

Description automatically generated

Table

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated with medium confidence

A picture containing table

Description automatically generated

Diagram

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, chart, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Project Supervisor Approvals

Text

Description automatically generated with medium confidence

# References

Herrmann, M. (n.d.). PyQt5 tutorial 2020: Create a GUI with Python and Qt. Retrieved October 28, 2020, from https://build-system.fman.io/pyqt5-tutorial

# Change Log

* [List all changes made from previous versions, referencing the page numbers where changes were made. Delete this section if not applicable.]