|  |
| --- |
| Final Report |
| Major Project |

|  |
| --- |
| Xinghua Wei – A00978597  8-8-2021 |

Table of Contents

[1. Student Background 2](#_Toc53664363)

[1.1. Education 2](#_Toc53664364)

[1.2. Work Experience 3](#_Toc53664365)

[2. Project Description 3](#_Toc53664366)

[Rules of the game: **Error! Bookmark not defined.**](#_Toc53664367)

[3. Problem Statement and Background 4](#_Toc53664368)

[4. Scope and Depth 9](#_Toc53664369)

[User Interface 10](#_Toc53664370)

[Functional Requirements 10](#_Toc53664371)

[Non-Functional Requirements 10](#_Toc53664372)

[Out of Scope 10](#_Toc53664373)

[5. Test Plan 11](#_Toc53664374)

[6. Methodology 15](#_Toc53664375)

[7. System/Software Architecture Diagram 26](#_Toc53664376)

[8. Innovation 29](#_Toc53664377)

[9. Complexity 30](#_Toc53664378)

[10. Technical Challenges 32](#_Toc53664379)

[11. Development Schedule and Milestones **Error! Bookmark not defined.**](#_Toc53664380)

[12. Deliverables 33](#_Toc53664381)

[13. Conclusion and Expertise Development 33](#_Toc53664382)

[14. References 34](#_Toc53664383)

[15. Change Log 34](#_Toc53664384)

# 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.

## Work Experience

In 2019 summer, I interned at ScopeMedia, a studio that delivers products for e-commerce websites. My main job there was to help another developer testing his API calls.

# 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.

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.
  + Players will need to register and be verified using two-factor authentication.

# Problem Statement and 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.

# 4. Chosen Solution

## 4.1 Navy Chess Game General Rules

The chart below shows the settings of each chess units 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |

## 4.2 AI Algorithm Explain

#### 3.3.1 MiniMax

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. 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.

Shape

Description automatically generated

**Step 1:**

In the first step, the algorithm will generate the entire game tree and get each terminal node's values. Suppose Max takes the first turn. It will choose the best value from the bottom to the top for each node because the miniMax is a backtracking algorithm.

* For D: max(-1:4) = 4
* For E: max(2:6) =6
* For F: max(-3,-5) = -3
* For G: max(0,7) = 7

**Step 2:**

Then it is Min's turn. Min will try to get the minimum possible score.

* For B: min(4:6) = 4
* For C: min(-3,7) = -3

**Step 3:**

Now it is Max's turn. Max will again get the maximum possible score.

* For A: max(-3,4) = 4

Therefore, in this current position, Max will choose to get a value of 4, which means that Max will move accordingly.

## 4.2.1 Alpha-Beta Pruning

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.

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. The Alpha-beta algorithm will apply the same methodology to the other nodes.

There will be more than 4 layers and more than 2 nodes to compare for the real algorithm in the game. Each chess unit will be assigned a value according to its rank level. For example, a general has a value of 10, and a major have a value of 6. The AI will be the Max in the example, and it will try to get the maximum possible score around the board to make its move.

## 4.2.2 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.

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.

## 4.2.3 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.

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.

# Scope and Depth

The list of activities the game is expected to have are:

## User Interface

* Display a game background map for both players setting up their chess
* Display 50 chess units in total for both players setting up their strategy
* Display buttons for surrender, join, invite, exit, etc.
* Display a live chat panel on the side of the game map.

## Functional Requirements

* Chess units are movable by the correct player on the right path of the map
* Higher-ranked chess units should destroy Lower-ranked chess units
* Players take turns to perform their moves
* Map space should be correctly allocated for each chess unit
* A player communicates with another player through live chat
* A player generate invitation links to friends to invite the second player to join the game
* AI can attack, move chess units and be able to win or lose the game.
* Players should be able to see the game rules by pressing the H button during a game.
* Player invitation use two-factor authentication via an external application.

## Non-Functional Requirements

* In each turn, players should not exceed 30 seconds.
* A game introduction for new players to understand the rules of the game
* Communication between players should be encrypted except for two players
* Frequently used functions should be tested for stability and usability
* Each invitation links from a player lost invitation ability after 30 minutes
* The game will be developed by Python without any game engine
* The game should be available for both Windows platforms
* The game should be implemented with Python, Python socket and Python Tkinter.
* Players should be able to change AI mode. For example, choose an easy AI or a hard AI

## Out of Scope

* Upload the game online to make it a website available game

# Test Plan

Manual testings, including unit testing, will be done throughout the development of the application. Acceptance testing will also be done after finishing unit testing.

## 5.1 Manual Testing & Unit Testing

**Manual Testing**: Manual testing is the testing in which test cases are executed manually using any automated tool. The purpose of manual testing is to identify any existing bugs and defects in the application.

Manual Testing will be done for modules that are difficult to test, such as controller modules.

* List testing features in the modules
* List types of input the features can accept
* List expect result

**Unit Testing**: Unit testing will be done for a small, easy to test portion of the project and verifies its behaviour independently from other parts. For unit testing, the Python library "unittest" will be used.

* List codes I want to test
* Initialize a piece of code
* Apply unit testing code for the code
* Compare results

## 5.2 Acceptance Testing

Acceptance Testing will be done after unit testing of the project. Acceptance testing needs to verify that the application passes the manual tests list below. Additional tests may be conducted and added as need.

### 5.2.1 Players enter the match

Players enter a game match with correct IP addresses and ports.

* One player starts a match and opened a port
* The other player joins the match with the correct IP address and port
* Pass criteria:
  + One player should be able to create a game room and visually see that the game is running.
  + The second player should join the room with the correct IP address and port.

### 5.2.2 Display players' information

Players enter their players' names before joining a match

* Players join a match
* Players' names display on the correct side
* Pass criteria:
  + Both players should be able to see their player information on the game interface.

### 5.2.3 Players move their unit by turns

Players move their units to occupy a place on the map.

* Units can be moved correctly according to players' actions
* Units show up on the target position
* Pass criteria:
  + One player takes his or her turn to perform a move
  + Once the player finished his or her move, that player should not move any units until the other player completes his or her activity.

### 5.2.4 Players attack opponents' units

Players use their units to attack the opponent's units.

* Players move one of their units to the target unit's position
* The target unit is eliminated if the attacker's unit has a higher rank. Otherwise, the attacker's unit is eliminated.
* Pass criteria:
  + One player chooses a unit and moves the unit towards an enemy's unit
  + If the attacking unit's rank is higher than the victim's rank, the victim unit should be removed from the map. Otherwise, the attacking unit should be removed.

### 5.2.5 Players lose a game

Players lose a game if opponents' units eliminate their base.

* One of the opponents' units attacks and occupies a player' base
* The player loses the game
* Pass criteria:
  + The opponent's unit attacks one player's base. The player should lose the game.

### 5.2.6 Players win a game

Players win a game if they eliminate opponents' bases.

* One of the units attacks and occupies an opponent' base
* The player wins the game
* Pass criteria:
  + One player's unit attacks the opponent's base. The player should win the game.

### 5.2.7 Players communicate with each other

Players chat with their opponents using the chat function during a match.

* Players type and click send to send messages
* Opponents receive the messages
* Pass criteria:
  + One player types in the chat section and click send to send messages
  + The opponent should receive the messages and be able to send messages back

### 5.2.8 Players leave a game room

Players leave a game room before, during or after a match.

* Players click the leave the game room button
* Another window pops up to confirm players' actions
* Players leave the game room
* Pass criteria:
  + One player click the leave the game room button
  + The player clicks the confirm button when the confirm to leave alert shows up
  + The player should be able to leave the room

### 5.2.5 AI tests

The following tests are AI Accepting test cases:

|  |  |  |
| --- | --- | --- |
| Test Case | Test Description | Input |
| AI Moves a unit | AI should move a unit based on the game rules whenever the human player finishs his turn | Move Command |
| AI attacks with a unit | AI should move a unit towards the human player's unit to perform an attack | Attack Command |
| AI set up its unit | AI should be able to set up all its 50 units on the game map based on the game rules | AI set up commands |
| AI notifies the human player | AI should automatically notify the human player with a message to inform the human player that the player is playing against an AI through the chat section | A string message |
| AI wins the human player | AI should automatically notify the human player with a message to inform the human player that the AI beats the human player through the chat section | A string message |
| AI loses the human player | AI should automatically notify the human player with a message to inform the human player that the player beats the AI through the chat section | A string message |

# Methodology

## 6.1 Methodologies & Approach

The software development methodology for this project is waterfall methodology. Each task of this project will be executed serially because there is only one developer for this project. The waterfall methodology provides the capability to implement a minimum viable product quickly.

Diagram

Description automatically generated

Figure 1Waterfall Methodology Lifecycle

* **First stage**: Requirements collection
  + Expectations and goals of the project will be collected in this stage.
* **Second stag**e: Design
  + Finalizing the core structure of the product
  + Prepare a schedule for the developer
* **Third stage**: Building
  + Developing the product
* **Fourth stage**: Testing
  + Manual Testing
  + Unit Testing
  + Acceptance Testing
* **Fifth stage**: Finishing
  + Validate the project whether the project meets the requirements
  + Create a final report of the project

Waterfall methodology is chosen because it is useful for projects that prioritize quality rather than the cost of time duration to control the quality of the project better. Also, the methodology's stability makes project management easier since only one developer will monitor the entire project. I could actively monitor the progress at any stage of the development. The waterfall methodology reduces project complexity since all phases occur without any overlap. Additionally, it provides me with flexibility when I need to redesign any part of my project's functionalities.

First, after finish collecting the requirements for the project, the project enters the research phase. I don't have much experience working with 2D Python, AI algorithms, and two-factor authentication; some basic experiments with the related tools are required before starting the actual development process. This phase will take up to a minimum of 1 week to a maximum of 2 weeks.

With the tools and plans ready, the project can enter the development phase. I will start implementing the project and test after finishing the entire project. This phase could take up to 3 months.

## 6.2 Technologies

For project management, the following tools and technologies:

* **Microsoft Excel**: use to design a schedule of the progress of each task, each stage and upcoming tasks and stages.
  + Microsoft Project is a project management software to help me develop a schedule, assign resources to tasks, and track progress.

For software design, the following tools and technologies:

* **Draw.io** to create diagrams like UML diagrams, system diagrams.
  + Draw.io is a free diagram editor.
* **Proto.io** to design UI for the project
  + Proto.io is an application to prototype apps for interfaces.

For software development, the following tools and technologies:

* **Python**: The whole project will be implemented based on Python 3.8. Python is an interpreted, high-level and general-purpose programming language. Because Python has many well-developed third-party libraries, a straightforward structure, it is suitable for this project.
* **Python Tkinter**: The Python Tkinter will be used to create user interfaces. Tkinter for Python is a GUI widget. Tkinter is a blend of Python programming language and the Tkinter library. It is suitable for developing graphical applications (Herrmann, 2020).
* **MiniMax**: MiniMax algorithm with Alpha-beta pruning will be used for the AI part of the game
* **Python unittest**: Unit testing will be done by using the Python unittest. The Python unittest unit testing framework is inspired by JUnit, and it supports test automation, unit testing.
* **Github**: Github will be used for version control and source code management.
* **Tencent SMTP Service:** Tencent allows users to use their QQ account for its SMTP service. Users can check their email and send messages using emails. This could also be used for the two-factor authentication of the project.

# Detail Designs and Chosen Solutions

## 7.1 Game UI and Functionality

### 7.1.1 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.

### 7.1.2 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

### 7.1.3 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.



### 7.1.4 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.

## 7.2 AI

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

## 7.3 Chat function

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

## 7.4 Two-factor Authentication

## 7.4.1 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.



## 7.4.2 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.



# System/Software Architecture Diagram

## 7.1 Software UML Diagram

The following diagram is a software architecture diagram that visually represents how players interact with one another and how the software modules interact with one another.

The blocks in the diagram include:

* **Players**: Players take actions on playing the game, communicating with one another. A player versus player mode requires at least two players in a match.
* **Physical**: Data encryption will be done in the physical layer. In end-to-end encryption, the data is encrypted on the sender's system or device, and only the recipient is able to decrypt it. Nobody in the transport layer could read it or tamper with it.
  + The server would be the player who created a game room
  + The client would be the player who joins a game room
* **Application**: It will deal with network protocols, interact with them and present data change to players.
  + Player versus player: In this mode, players can match against another player based on the game rules.
  + Player versus Computer: In this mode, players can match against AI based on the game rules.

Diagram

Description automatically generated

Figure 2 Software UML Diagram

## 7.2 Software Class Diagram

The following diagram is a software class diagram that visually represents how players interact with one another and how the software modules interact with one another.

* **UI Controller**: UI controller is also the central controller that integrates, controls, and calls every class function. It also generates UI/UX components:
  + Generate players' units
  + Generate a map
  + Generate action buttons
  + Generate chat panel
* **Players**: Player class contains functions that record players' names, IP addresses and ports.
* **Match**: Match class contains functions:
  + Win: Validate if one player wins the game
  + Lose: Validate if one player loses the game
  + Leave: Leave the game and the game room
  + Join: Join a game room and a match
* **Units**: Units class contains functions:
  + Move: Units could move to target positions
  + Attack: Units could attack an opponent's units
  + Be Destroyed: Units could be destroyed by opponents' based on game rules
  + Check Game Rules: Validate if a unit's current action is processable.
* **AI**: AI class contains the minimax algorithm that a computer could move units, attack units fast and accurate based on game rules.
  + Get the largest value of chess unit of the two sides of a unit
  + Calculate the score of the current board
  + AI seeks to maximize the board score
  + Player seeks to minimize the board score
  + Move: Computer understands units could move to target positions
  + Attack: Computer understands units could attack an opponent's units

Diagram

Description automatically generated

Figure 3 Software Class Diagram

# Innovation

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

**Lagging**: The game also needs to solve the lagging problem. For example, a player who is 50 pings should wait for a player with 100 pings.

**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.

**Art design**: I want each unit in the game to feel like that unit has a height instead of just a 2D image. so, the art design is somehow important for this game. This will make the game feels more interactive.

Overall, this game requires students to have proper art design, socket, Python, Python GUI, and algorithm skills to finish the game development. A diploma student should not be able to do such comprehensive work within 350 hours.

# Technical Challenges

For this game, I would use TCP/IP communication skills, socket skills I learned from the Btech program to handle player connections and set up encrypted data transmission.

**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. I need to research methods and tools to perform a proper test.

**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.

# Deliverables

The deliverables for this project are as follow:

* **Navy Chess Game Client**: A playable digitalized Navy Chess game
  + Game UI
  + Playable Game
  + Chat function with end-to-end encryption Communication
  + AI capability
* **Final Report**: Final report for the major project
* **User Guide**: User Guide to explain game rules and how to use the game client

# Conclusion and Expertise Development

In conclusion, the digitalized Navy Chess game is a huge challenge for me. I will develop a fully functional digitalized 2D Navy Chess game for the Windows and Linux platforms. I need to research and develop 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 will help enhance 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.

# 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

### Version 0.0:

* Nothing