Off to the Races Betting Game and Multi-Client Chat Log Server

Team 4M - (Maxy, Makiah, Mykhaylo, Minhaz)

Design Document - Revised Version: 3.0

Authors:

Maxy Tolosa (pft5290@rit.edu)

Ma'kiah Holliday (mmh3885@rit.edu)

Mykhaylo Berezyuk (mb3230@rit.edu)

Nuzhat Minhaz (nxm1137@rit.edu)

Table of Contents

Revision History

Name	Date	Reason for Change	Version
Minhaz	March 4th	Began editing	1.0
Entire Team	March 6th	Decided on Game and wrote description	1.1
Mykhaylo Makiah	March 9th	Potential UML Diagram	1.2
Maxy	March 10th	Updated Classes & Methods Overview	1.3
Minhaz	March 11th	Completed Executive Overview	1.4
Minhaz	March 11th	Completed Application Intention	1.5
Minhaz	March 11th	Completed Audience and Assumptions	1.6
Minhaz	March 11th	Added User Behaviour to be Monitored	1.7
Maxy Mykhaylo Makiah	March 16th	Updated Clients, Protocols, Ports	1.8
Maxy Mykhaylo Ma' kiah	March 17th	Updating Clients, Protocols, Ports	1.9
Entire Team	March 20th	Updated Gantt Chart	2.0
Entire Team	March 29th	Final Changes for Design Document submission 1	2.1
Maxy Mykhaylo Makiah	April 4th	Brought previous typed information on the Client, Protocol, and Classes/Methods back from the original document	2.2.0
Makiah Mykhaylo	April 6th	Uploaded UML Diagram	2.2.1
Maxy	April 8th	Uploaded the Outline of the Project and Gantt Chart	2.3
Mykhaylo	April 9th	Added additional information in Clients, Protocols, Ports	2.4
Maxy	April 9th	Created a table demonstrating the Client and Server Communication in Protocols	2.5
Maxy	April 9th	Uploaded the Client GUI outline	2.6
Mykhaylo	April 13th	Added information on where and how data get sent over the network with the protocols.	2.7
Ma' kiah	April 13th	Uploaded UML Diagram	2.8
Minhaz	April 13th	Update Punch List	2.9
Minhaz	April 13th	Update Unresolved Issues, minor overall changes before final submission	3.0

Executive Overview

Welcome to Off to the Races: a racing game where users can bet on which horse they think will win in rounds of 3, while interacting with each other! This game will display a race of a set number of horses that run at different speeds each time a round is begun.

Before the round is begun, users will be allowed to join the game and are each given a certain amount of money as credit to start ($ 100). They are allowed to enter a monetary amount that they want to bet on which horse they think will win. After they've entered their betting amount, they will not be allowed to change it once the race begins.

While the game is in progress and/or before it begins, users will be able to interact with each other using a multi-client chat server, for many purposes, be it consulting to decide on amounts they want to bet, or even rejoice over winning once the round is over! This will be done using a multi-client chat log server where users will be able to see each others' responses and enter text simultaneously, as means of communication.

Summary:

1. Users enter the game and each have $100 on their account to begin.
2. Users have access to the chat interface to communicate with each other.
3. Users bet on racing horses to see which one will win the first round.
4. The winner will get their wager amount added to their account. Users who lost will get their wager amount deducted from their account.
5. Users can bet again for round 2, and then repeat the process for round 3.
6. User with the most amount of money when they finish all 3 rounds wins.

Application Intentions:

This game of betting on randomized outcomes might appear entertaining and simply an engaging game, however, similar game tasks are often used by many companies to monitor the behaviour of users or potential employees as to how much or how little they are willing to risk when taking chances, and whether they are making judgement based on instincts or calculations. Many companies nowadays are adding a layer during their hiring process for interns/employees where applicants are asked to play a series of games where the outcomes are used to place these recruits in specific departments where employers think they will flourish based on the outcomes of the games (not based on whether they won or lost, rather the decisions and choices they made during the games).

Examples (Factors further explained under Audience for this Design Document):

1) If someone bets a lesser amount in the second round because they lost a higher amount of money in the first round, it means they are willing to risk less owing to chances of losing their valuable money.

2) If a user bets the same or similar amounts of money all three times, it is an indicator that they play it safe under risky conditions.

3) If a user bets a higher amount of money the second time because they won the first time, it shows that they are motivated by incentive and are willing to take risks based on past experiences.

Therefore, decisions made during this simple game is a strong indicator of when a user is taking chances under high-risk, or medium-risk, or low-risk conditions, and whether they are following instincts and gut-feelings or logic and reasoning based on numbers. The chat log is an added layer to see what they're thinking as the game is progressing, and to see whether communication allows said users to make more informed decisions.

Overall, it is intended to be a fun game with randomized outcomes to test your luck, and be connected to other users while playing it, but the implications and underlying behavioural tendencies that are displayed by users who play this game are essential to understanding one's psychology when under risky conditions, degree of motivation through incentive or loss based on experience, and communication as a factor behind their decision-making process.

Audience for this Design Document:

Companies as successful as JP Morgan Chase & Co. are using Pymetrics Games as an added layer before they hire their recruits (not only business, but also technology programs) and place them in departments, and it has shown an improvement in modeling the efficiency of division of tasks within the organization, all while prioritizing the existing skills and personality traits of an applicant rather than penalizing them for responding a certain way.

Our targeted audience for the inner workings of this game includes companies that are looking to hire new employees, which is essentially any company that is looking to grow in size, and require a method of testing this aspect of an employee's behaviour if its related to what the company intends for this recruit to accomplish after they have been hired.

This Design Document is for companies that are filtering employees for positions that require the skills that can be monitored using this game. Organizations such as but not limited to:

- Companies that are hiring for technology departments to understand whether the recruit will make changes to the technology they work on based on past experiences;
- Companies that are hiring for business-related departments such as investment, accounting, etc, to understand whether the recruit is able to logically analyze numbers that may be crucial to the future of the organization;
- Companies that are hiring for human resource positions to see if the employee takes uncalculated risks by hiring people based on instincts or by using logical reasoning behind their decision during the recruitment process;

User's Behavioural Factors that can be Observed Using Off to the Races Game + Multi Client Chat Server:

1) Risk Preference for Ambiguous Risks:
By noting how much the user bets on the first round. This will help employers understand whether the potential recruit is more likely to be cautious, or willing to take more ambiguous risks without prior experience.

2) Risk Preference for Low Risks:
By noting how much the user lost or gained in the first round and if the user risks a relatively lower amount in the second round, showcases that the user is analytical of how much they are willing to lose under low risk conditions (low risk because a third round is still there). Shows that the user does a good job at learning from mistakes, and is cautious of the outcome.

3) Risk Preference for Medium Risks:
By noting if a user bets more amount of money in the second round despite losing money in the first round. Shows that the user is still confident and motivated by incentive and is not flustered by making mistakes, shows positive attitude. Note: there is a need to understand whether the user is doing so randomly or by genuinely being motivated by chance of winning the second round. Could be monitored if they share their thoughts via chat log server with other users.

4) Risk-taking under High Risk conditions:
By noting if the user is still betting a high amount of money on the third round despite losing first and second round.

5) Risk-taking under Medium Risk conditions:
By noting if the user is betting a low of money on the third round after having won both of the first and second rounds.

6) Risk-taking under Low Risk conditions:
By noting if the user is betting a low of money on the third round after having won both of the first and second rounds.

7) Other Behavioural Aspects that can potentially be observed:

- How communication can affect user's decisions (via interaction on the chat)
- How monetary incentives / rewards can affect user's decisions (based on wager amount)
- How user reflects on mistakes (not flustered by mistakes, changes behaviour based on mistake made, etc)

Assumptions Made for this Project:

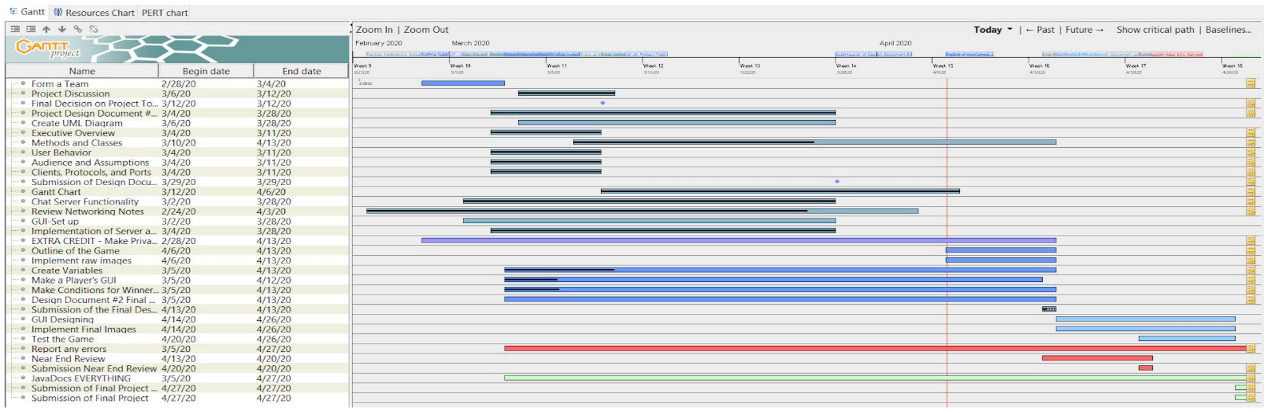
- User must have access to Internet for communication over chat log server
- If companies are hiring through this, the user playing the game must be notified that it is part of the hiring process
- User must be made aware that there is no right or wrong way of doing this, to make sure the monitored behaviour of these users are accurate and honest;
- Accessibility factors:

  - User should be able to read (minimum education) ;

  - User should be able to visually comprehend the outcome or have someone assisting them if they are blind;

  - User should understand basic Math (betting on monetary amount)

Gantt Chart:



Name	Begin date	End date
Form a Team	2/28/20	3/4/20
Project Discussion	3/6/20	3/12/20
Final Decision on Project To...	3/12/20	3/12/20
Project Design Document #...	3/4/20	3/28/20
Create UML Diagram	3/6/20	3/28/20
Executive Overview	3/4/20	3/11/20
Methods and Classes	3/10/20	4/13/20
User Behavior	3/4/20	3/11/20
Audience and Assumptions	3/4/20	3/11/20
Clients, Protocols, and Ports	3/4/20	3/11/20
Submission of Design Docu...	3/29/20	3/29/20
Gantt Chart	3/12/20	4/6/20
Chat Server Functionality	3/2/20	3/28/20
Review Networking Notes	2/24/20	4/3/20
GUI-Set up	3/2/20	3/28/20
Implementation of Server a...	3/4/20	3/28/20
EXTRA CREDIT - Make Priva...	2/28/20	4/13/20
Outline of the Game	4/6/20	4/13/20
Implement raw images	4/6/20	4/13/20
Create Variables	3/5/20	4/13/20
Make a Player's GUI	3/5/20	4/12/20
Make Conditions for Winner...	3/5/20	4/13/20
Design Document #2 Final ...	3/5/20	4/13/20
Submission of the Final Des...	4/13/20	4/13/20
GUI Designing	4/14/20	4/26/20
Implement Final Images	4/14/20	4/26/20
Test the Game	4/20/20	4/26/20
Report any errors	3/5/20	4/27/20
Near End Review	4/13/20	4/20/20
Submission Near End Review	4/20/20	4/20/20
JavaDocs EVERYTHING	3/5/20	4/27/20
Submission of Final Project ...	4/27/20	4/27/20
Submission of Final Project	4/27/20	4/27/20

Gantt Task Descriptions:

M	Member in Charge	Tasks	Completion %	Date
M		Formed a Team	100%	Mar 4th
	Team	Project Discussion		Mar 6th
M	Team	Final Decision on Project Topic	100%	Mar 12th
	Ma' kiah	Created UML Diagram	98%	Mar 10th
	Minhaz	Executive Overview in DD#1	100%	Mar 11th
	Maxy	Classes and Methods	60%	Mar 11th
	Minhaz	User Behavior	100%	Mar 11th
	Minhaz	Audience and Assumptions	100%	Mar 16th
	Mykhaylo	Clients, Protocols, Ports	100%	Mar 17th
M	Minhaz	Submission of Design Document #1	100%	Mar 20th
	Maxy	Gantt Chart	100%	Mar 29th
	Team	Chat Server Functionality	98%	April 4th
	Team	Reviewed Networking Notes	100%	April 6th
	Ma' kiah	Established GUI-Setup Sketch	100%	April 8th
	Mykhaylo Maxy	Implemented Client and Server functions into the game	99%	April 9th
		EXTRA CREDIT - Provided private messages implemented into the public chat similar to Minecraft chat mechanism	-	
	Maxy	Created the outline of the game	100%	April 8th

	Mykhaylo	Implemented the raw images for testing purposes	50%	April 13th
	Team	Created variables to use for the game	100%	April 6th
		Created a client's own GUI	50%	April 12th
	Maxy	Made conditions for both the winners and losers in each round and in the final round.	60%	April 13th
M	Minhaz	Submission of Design Document #2	100%	April 13th
	Team	Designed the final touches in the GUI	0%	April 26th
	Mykhaylo	Implemented final images for the game	0%	April 26th
	Team	Tested the game and reported for errors	50%	April 27th
M	Team	Submission of Near End Review	0%	April 20th
	Team	Put sufficient JavaDocs + comments	5%	April 26th
M	Team	Submission of the Final Project Presentation	0%	April 27th
M	Team	Submission of the Final Project with files	0%	April 27th

**M - Milestones

Class and Method Overview:

Off To The Races! Set Up

- Contains Port number constants, used by Client and Server
- Contains Options and About messages in Menu Bar
- Contains Threads and Inner Classes for each Client
- Contains Options to change GUI layout color for individual client and not globally
- Contains Booleans and statements conditions for the winners and the losers in each round and in the final round
- Contains Messages Dialogs asking clients how much they will bet and what horse will win in each round

Client

- Main Method
    - Establishes a new connection to the Server
- Client Constructor
    - Creates GUI Set up along with the chat tab
    - Thread will begin as soon as the connection is made with the server
- ColorOptions
    - Offers client options to change the layout of the GUI chat box

   - Contains various of colors implemented from Java Library (Drawing Jar/Javaxswing)

Server

- Main Method
   - Creates a server socket
   - Waits for client's connection to accept
- Server Constructor
   - Adds each client to the Array List for data purposes
   - Includes the process of sending messages in clients' interaction in the chat.
- BeginGame Method
   - Contains the statements when number of clients have reached the capacity and can begin the game

---

Off To The Races! UML:

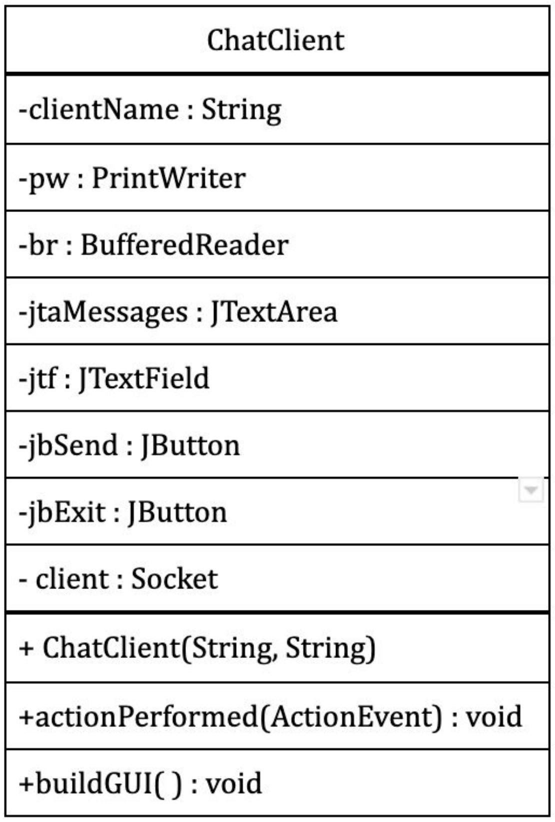


Figure 1: ChatClient class UML diagram

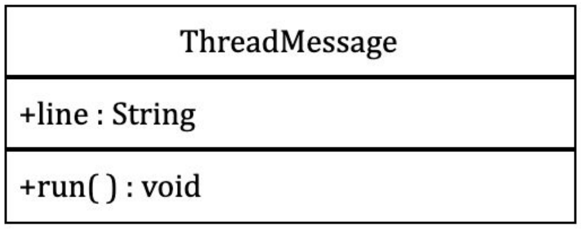


Figure 2: Threaded inner class UML diagram in ChatClient class.

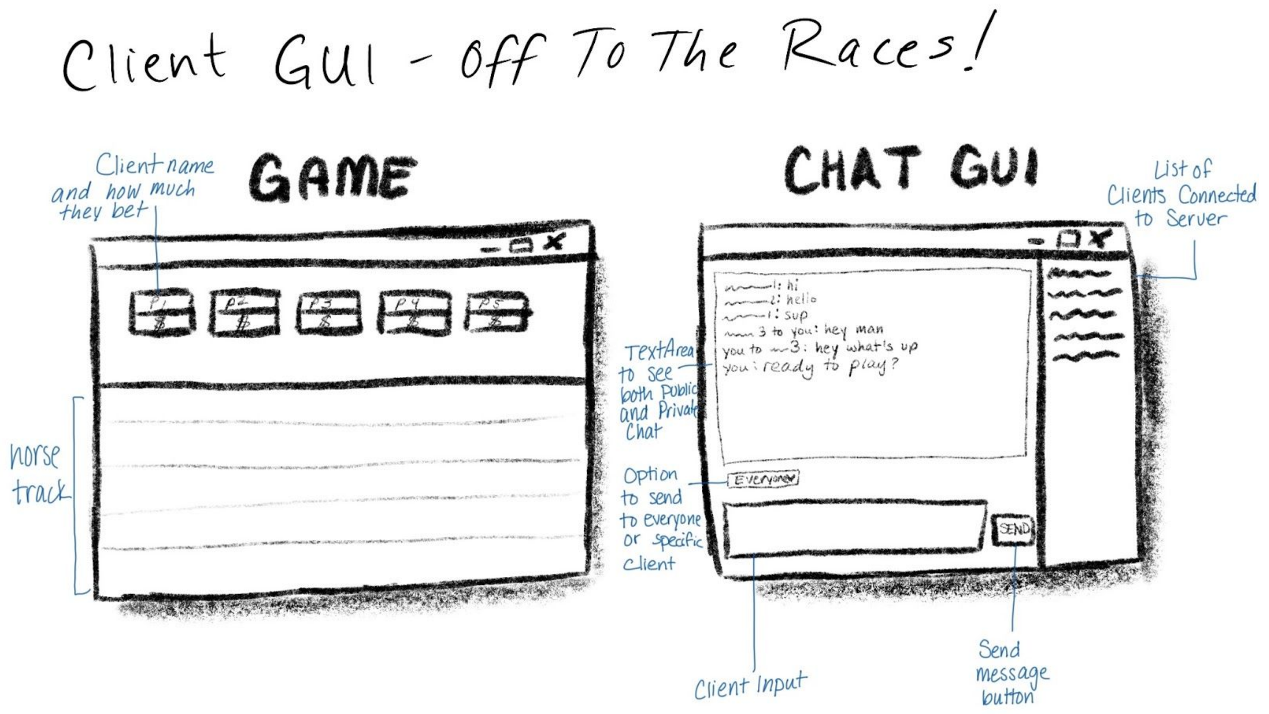
ChatServer
+users : Vector<String>
+clients : Vector<ThreadedServer>
+ChatServer( )
+ beginServer( ) : void
+broadcast(String,String) : void

Figure 3: ChatServer class UML diagram

ThreadServer
+br : BufferedReader
+pw : PrintWriter
+name : String
+ThreadServer(Socket)
+getUserName( ) : String
+run( ) : void
+sendMessage(String, String) : void

Figure 4: Threaded inner class UML diagram in ChatServer class.

Client GUI:


Client GUI - Off To The Races!
GAME
Client name and how much they bet
P1
P2
P3
P4
P5
horse track
CHAT GUI
List of Clients Connected to Server
TextArea to see both Public and Private Chat
Option to send to everyone or specific client
Everyone
SEND
Client Input
Send message button

1: hi
2: hello
1: sup
3 to you: hey man
you to 3: hey what's up
you: ready to play?

Protocols:

Networking connections & Protocols

Client connection to server: The server will have one unique IP address for the clients to connect to. The client will connect to the server with the server IP and port.

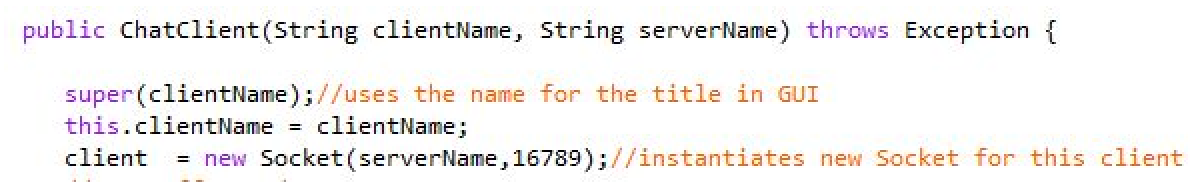
Port number(s): We are using the port number 16789. This is just a single port number that can be changed anytime. We can use any port number from 1024 to 49151 that are open and available.

The reason we need both IP and port because IP is the address of the server and port is for the application that can run on the network. The client is connecting to the server via socket connection which is IP address + port number. The data goes from the client by sending packets over the network. First the client must know the address of the server, which is given in this project. Then the client will attempt to send data through with the TCP protocol which will ensure the reliability of the packets. But the packet must go somewhere when it reaches the IP. This means the client must know an open and correct port for the application they are using. The port is also given in this project. When the packet reaches the given port and it is open, the data has been successfully transmitted.

Protocol interface code for both client and server: The client and server is using the TCP/IP protocol.

The protocol we're using is TCP/IP. TCP is a protocol that sends data in packets that can also easily re-route data if it appears to be disconnecting, damaged, or even lost. The IP is the Internet Protocol that is used to address and routing data. In this project we are utilizing both TCP and IP in order to have this client and server running.

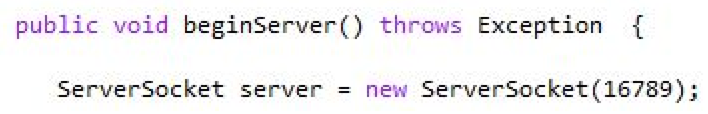
Client side example code:



```
public ChatClient(String clientName, String serverName) throws Exception {

    super(clientName);//uses the name for the title in GUI
    this.clientName = clientName;
    client  = new Socket(serverName,16789);//instantiates new Socket for this client
```

Server side example code:



```
public void beginServer() throws Exception  {

    ServerSocket server = new ServerSocket(16789);
```

Communication class

Chat interaction between Clients and Server

The server begins and waits for clients to connect. Once connected clients can enter their username and this name is sent to the server. The server replies with a welcome message including the clients chosen username. This message is visible to the client upon initial connection. The clients are now able to send and receive messages from the server. The server reads the client messages and updates the chat text area for all clients. The server will specifically read for the word "end" sent by the client which allows the client to leave the chat. The client prints a "*user* left the chat" message and removes the client. This messaging is viewable by all clients.

Game interaction between Clients and Server

The client joins the game through the connection made between the client and server. The server manages the clients added to the game. If the amount of users in the game is not fulfilled, the client will receive a message from the server informing them the game is ready. However, if the max number of clients has been reached the client will receive a message saying the game is full. Clients that receive the full game message connections will be terminated. Once clients reach the end of the game their connection will also be terminated.

Client	Communication	Server
		Start-Up
Client connection	Connection, no data ->	Waits for client to connect Accepts connection
Client sends a message	message is sent to server ->	Server reads the message
		Server will take the message and sends it to the public (other clients or specific client)
Client receives the public/private message.	<- Server will take data and sends it into the JTextArea (chat) for all or one client(s) to see	Server sends the read message to other client(s)
Client joins the game	Connection, added client to arraylist in party ->	Server takes account for the client joining the game. Accepts connection

Client(s) receives a message that the game has a full party.	<- Server sends the message to the clients that the game is ready.	If max of clients is reached, the server will send a message to the clients.
Client(s) receives the closing message and will leave the connection.	<- Server sends an ending message to force close the game and go back to the title.	Server says it is game over after reaching the last round, forcing players to leave the game.

Data Used:

Currently, we are in process of using hand drawn images and Photoshop for the images but are not implemented into the program yet. No other additional files, URLs or network connections will be used externally for this project. All charts and images have been made by members of our team.

Punch List Used:

To Do:

- Code for the implementation of the game needs to be modified
- Game GUI needs to be developed
- Functionality for the game GUI needs to be implemented into the chat/server program
- Final images for the game need to be created/selected
- Test the game and chat functionalities

Done:

- Client/Server chat completed
- Design document completed

Unresolved Issues:

Currently, no issues have arisen from testing the initial code of the program. Some minor issues regarding GUI to be implemented to enhance user experience and ease of game to make sure instructions are not too confusing for the user.