Project 4: Team # 40

Simrah Shaik

Huy Truong

Maryam Ahmed

Mallika Patil

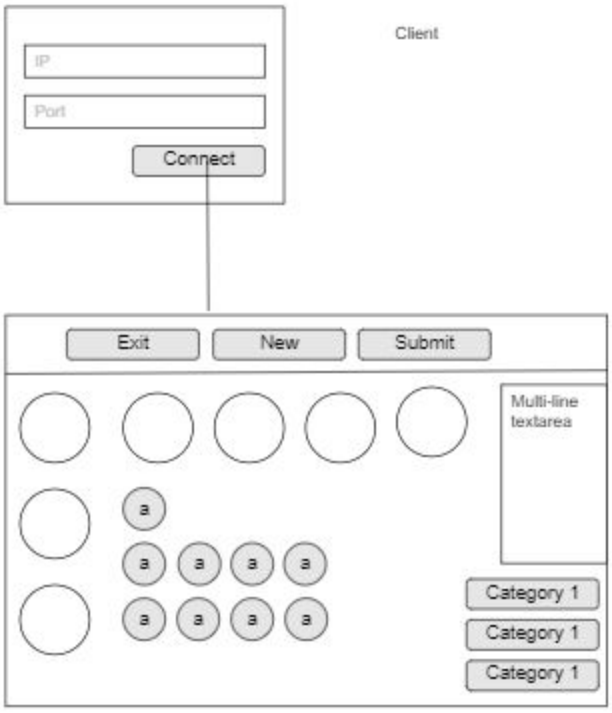
The game is designed so that it closely follows the implementation of Hangman where the player (client) is allowed to make letter guesses and the administrator (in this case the server) lets the player know if that letter is present in the word and where it is present.We decided to implement a letter bank in GUI so that it would make the game more appealing and easier to navigate. You can also pick between Diseases, Mythical Creatures, and Programming Languages. There is also a section that states the number of attempts the player has left so that they can understand the flow of the game better. In the event of communication between the server and the client, there is a text area in the GUI for handling messages from the server.

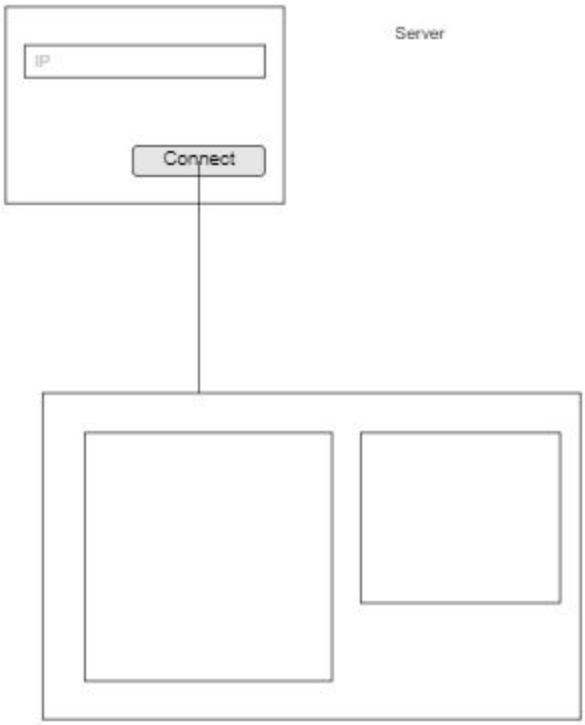
The team chose to collaborate on GitHub as it would be easier to maintain versions of the project and update each other on time. If two members of the team were working on components of the project that were closely related, then they would collaborate separately so that any errors or issues that arise during the development would be handled quickly and accordingly without having to disrupt the work of the team members not associated with that part. We utilized Scene Builder in the making of the GUI to allow for flexibility in the separation of design and logic.

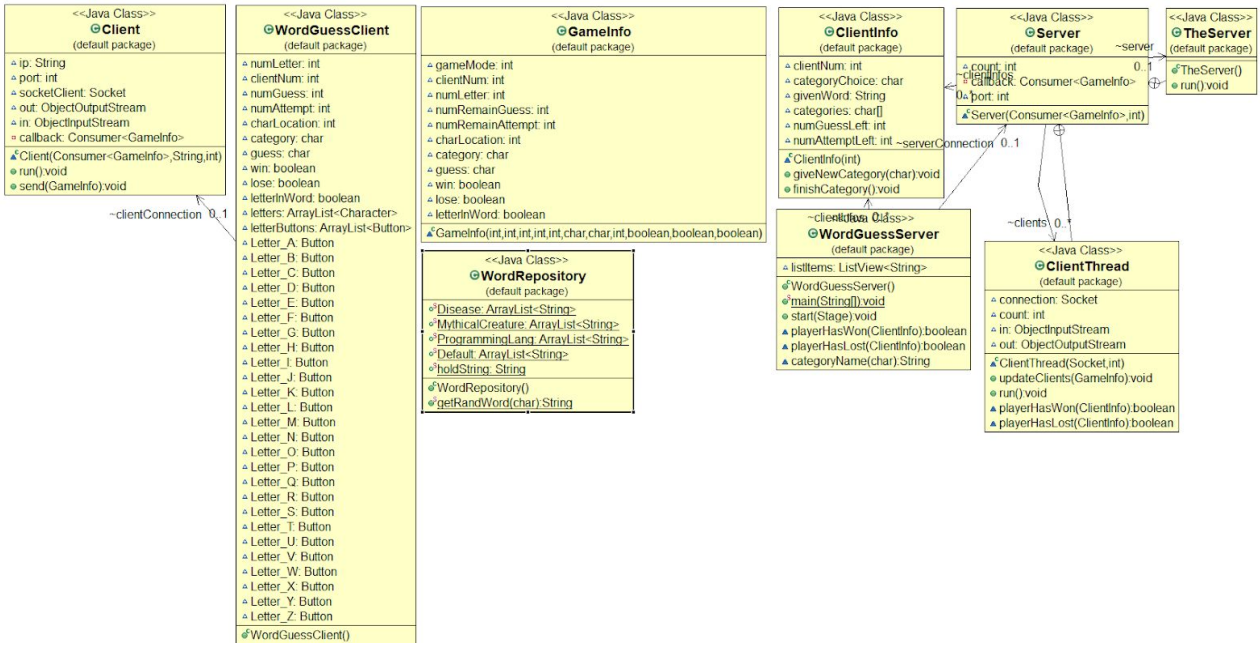
For the individual contributions of each team member, Mallika worked mainly on the graphical user interface and making sure to follow design principles that we utilized in class so that the client can understand and easily follow the progress of the game. Maryam worked on incremental testing of the program and ensuring proper control flow. Huy worked on connectivity between the client and the server and . Simrah worked on designing the logic of the game and how interactions would be handled between the client and the server.

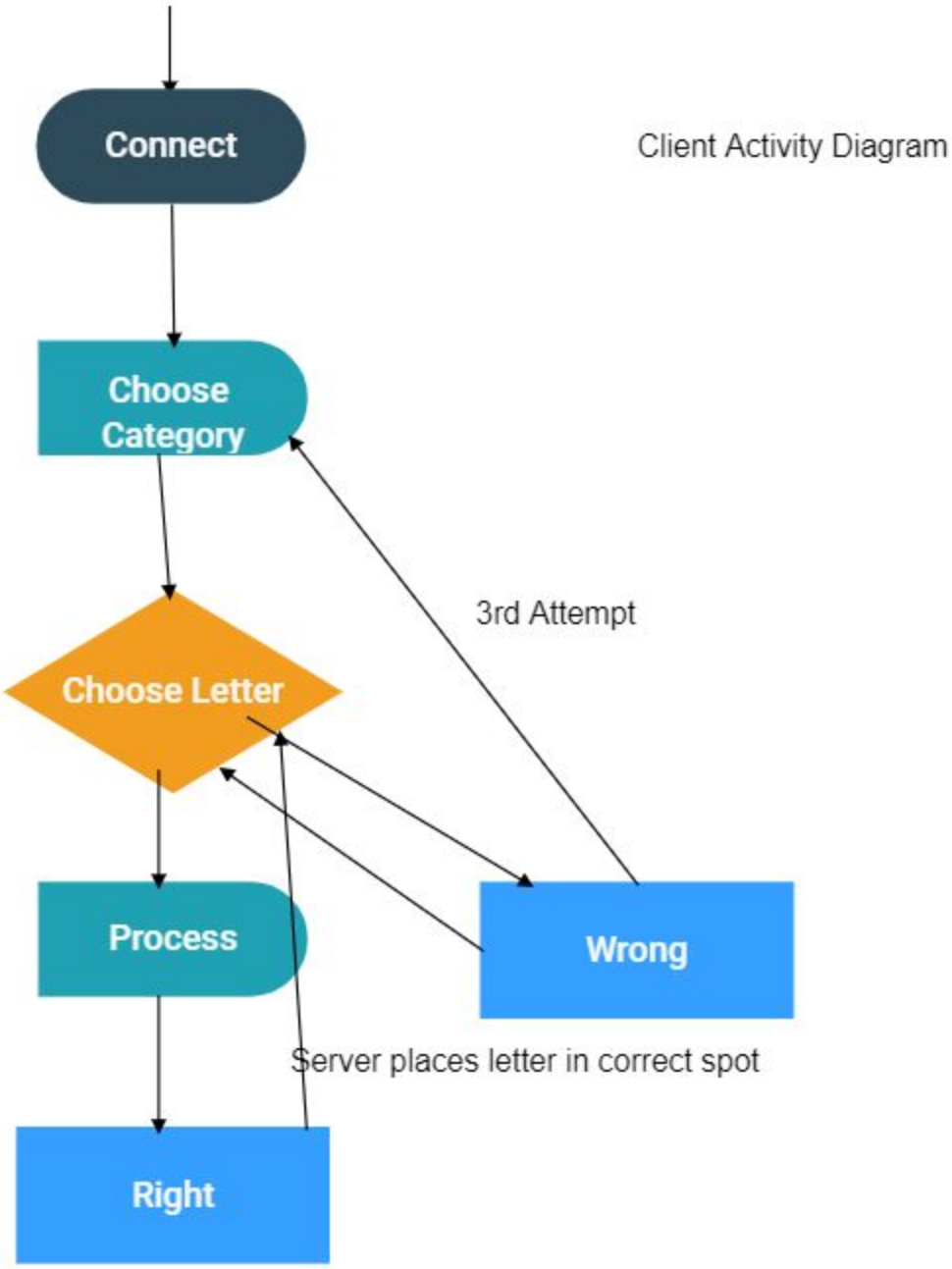
Some of the issues that the team ran into during development began with the initial design of the program itself. It was difficult to figure out some of the logistics concerning the

flow of the game, and it took a significant amount of time to get that working. Another issue was overlapping between team members. Sometimes we would have to wait for one member to finish a certain component before we could make additions/changes to the program.


Client
IP
Port
Connect
Exit
New
Submit
Multi-line
textarea
a
a
a
a
a
a
a
a
a
Category 1
Category 1
Category 1


Server
IP
Connect


<<Java Class>>
Client
(default package)
ip: String
port: int
socketClient: Socket
out: ObjectOutputStream
in: ObjectInputStream
callback: Consumer<GameInfo>
Client(Consumer<GameInfo>,String,int)
run():void
send(GameInfo):void
~clientConnection 0..1
<<Java Class>>
WordGuessClient
(default package)
numLetter: int
clientNum: int
numGuess: int
numAttempt: int
charLocation: int
category: char
guess: char
win: boolean
lose: boolean
letterInWord: boolean
letters: ArrayList<Character>
letterButtons: ArrayList<Button>
Letter_A: Button
Letter_B: Button
Letter_C: Button
Letter_D: Button
Letter_E: Button
Letter_F: Button
Letter_G: Button
Letter_H: Button
Letter_I: Button
Letter_J: Button
Letter_K: Button
Letter_L: Button
Letter_M: Button
Letter_N: Button
Letter_O: Button
Letter_P: Button
Letter_Q: Button
Letter_R: Button
Letter_S: Button
Letter_T: Button
Letter_U: Button
Letter_V: Button
Letter_W: Button
Letter_X: Button
Letter_Y: Button
Letter_Z: Button
WordGuessClient()
<<Java Class>>
GameInfo
(default package)
gameMode: int
clientNum: int
numLetter: int
numRemainGuess: int
numRemainAttempt: int
charLocation: int
category: char
guess: char
win: boolean
lose: boolean
letterInWord: boolean
GameInfo(int,int,int,int,int,char,char,int,boolean,boolean,boolean)
<<Java Class>>
WordRepository
(default package)
Disease: ArrayList<String>
MythicalCreature: ArrayList<String>
ProgrammingLang: ArrayList<String>
Default: ArrayList<String>
holdString: String
WordRepository()
getRandWord(char):String
<<Java Class>>
ClientInfo
(default package)
clientNum: int
categoryChoice: char
givenWord: String
categories: char[]
numGuessLeft: int
numAttemptLeft: int
ClientInfo(int)
giveNewCategory(char):void
finishCategory():void
~serverConnection 0..1
<<Java Class>>
WordGuessServer
(default package)
listItems: ListView<String>
WordGuessServer()
main(String[]):void
start(Stage):void
playerHasWon(ClientInfo):boolean
playerHasLost(ClientInfo):boolean
categoryName(char):String
<<Java Class>>
Server
(default package)
~server
count: int
0..1
~clientInfos
callback: Consumer<GameInfo>
0..*
port: int
Server(Consumer<GameInfo>,int)
<<Java Class>>
TheServer
(default package)
TheServer()
run():void
~clients 0..*
<<Java Class>>
ClientThread
(default package)
connection: Socket
count: int
in: ObjectInputStream
out: ObjectOutputStream
ClientThread(Socket,int)
updateClients(GameInfo):void
run():void
playerHasWon(ClientInfo):boolean
playerHasLost(ClientInfo):boolean


Connect
Client Activity Diagram
Choose Category
3rd Attempt
Choose Letter
Process
Wrong
Server places letter in correct spot
Right