Project !Failure

Documentation

Team 5 - Project !Failure

Rouqi Chen

Alex Ketavongsa

Ethan Marschean

Fernando Rodriguez

Rachael Simmonds

Date: 04-April-2019

unaccepted Revision: 0.5

Table of Contents

Revision History

Name	Date	Reason for Change	Version
Rachael Simmons	02-April-19	Team name change.	0.2
Fernando Rodriguez	02-April-19	Executive overview, To-do list, what we got done, Assumptions.	0.2
Rachael Simmonds	03-April-19	Audience	0.25
Alex Ketavongsa	04-April-19	Modified Gantt chart	0.4
Rouqi Chen	04-April-19	Chat interaction between Clients and Server.	0.4
Rachael Simmonds	04-April-19	Created first draft of UML Diagram.	0.5
Rouqi Chen	15-April-19	Client screenshots	0.6
Alex Ketavongsa	16-April-19	Documentation of Email class	0.7
Rouqi Chen	16-April-19	Client Screenshots	0.8
Rouqi Chen	18-April-19	Client Screenshots	0.9
Alex Ketavongsa	22-April-19	Documentation of “save log” function	1.0
Rouqi Chen	23-April-19	Documentation of “login”, “register”, and “game” functions.	1.1

Ethan Marschean	23-April-19	Renamed Project from Discord Jr. to !Failure for continuity	1.2
Ethan Marschean	23-April-19	Minor edits made to the Exec. Overview	1.3
Fernando Rodriguez	23-April-19	Updated the data files that were used in the project.	1.4
Rachael Simmonds	24-April-19	Added screenshots of themes and updated the UML diagram.	1.6
Alex Ketavongsa	29-April-19	Modifications: validate textbox if empty before send in Client class, added save log confirmation message, removed date from being printed to console. Created JAR Files for Client/Server files to submit.	2.0
Rachael Simmonds	29-April-19	Added the final version of the UML diagram. Checked final code to see if anything was missing. Fixed themes with Alex. Added the game and help menu items to final code. Added SetTheme operation to document and explained its function and purpose.	2.2
Alex Ketavongsa	6-May-19	Added JavaDoc Comments to the 'Client' class file.	2.3
Rouqi Chen	6-May-19	Added JavaDoc Comments to the 'Dots' class file.	2.4
Fernando Rodriguez	7-May-19	Added to unresolved issues, figure captions and additions to Gantt chart	2.5

Executive overview

The project is about having an open source way of chatting with multiple people. If you ever need to have a group chat with your co-workers then you'll need to find a way to get everyone together into one chat. With our project as long as the people have a client code and the Server IP then they can connect whenever and wherever they want. There will be no confusion as to who is who when connected because there is a nickname feature to use to identify each user who is connected to the chat room. Our project enables users to send an email and choose a theme for their chat room. They can play a stand-alone game if they want while waiting for others to reply. This project is important because of the ease of access it allows co-workers to communicate with each-other quickly, clearly and efficiently.

Audience

If you are unfamiliar with chat services this document will teach you about how the program operates and how you can interact with the service independently or with many other people.

Application intentions

The target audience for this application is more so for students and business people who need a way to chat with others on a project they're working on.

Assumptions made for this project

This project will need a server and client network code along with subsidiary codes for the extra functions. It will need a connection with just one server.

Gantt chart

Gantt task descriptions:

WBS	M	Tasks	Date (if known)
1	M	Assigned project ideas (Private Chat Program)	March 5th, 2019
2	M	Created the server-client program	March 10th, 2019
2.1		Drew a GUI mockup of the client	March 11th, 2019
2.2	M	Created the GUI for client	March 16th, 2019
3	M	Multithreaded Client created for chat	March 20th, 2019
4	M	Save log function - started brainstorming on how we wanted it to work.	April 1, 2019
5	M	Design document being worked on	April 4th, 2019
4.1		Alex started writing code for the save log function, referring to previous labs on how to use PrintWriter.	April 3rd, 2019
5.1		UML documentation	April 4th, 2019
5.2		Work on making themes	April 5th, 2019
4.2		Save log function mostly working. Saves from the text area to a log file but formatting is off.	April 7th, 2019
4.3		Fixed the formatting problem in the log files. Save log function fully working. Code given to Ethan to be implemented to the GUI.	April 8th, 2019
5.25		Themes were created	April 9th, 2019
6	M	Send Email function - researched how to get started and started writing code.	April 9th, 2019
6.1		Tested if emails were being sent to the Gmail server by hardcode. Problem with ports.	April 10th, 2019
6.2		Started implementation of a way to attach files, using JFileChooser.	April 12th, 2019
6.3		Able to send emails with attachments. Provided working email code to Ethan to be integrated into the GUI.	April 14th, 2019
7	M	Log in GUI was made	April 15th, 2019

7.1		Color Blindness Feature to Themes were added	April 16th, 2019
7.2		Theme colors were added	April 17th 2019
8	M	*Stand-alone game was made*	April 14th, 2019
9	M	SHA-256 password encryption	April 25th, 2019
9.1		Prepared Presentation	April 25th, 2019
10	M	Presented our project	April 26th, 2019
		Created JAR Files - for Client and Server. Submitting to dropbox.	April 29th, 2019
11	M	Final UML Diagram Created	April 29th, 2019
11.1		fixed the LoginGUI Array Error, the code that looked through the array needed to be moved up one element because the first element in the array would be blank everytime the code ran (by design),	May 5th, 2019
11.2		LoginGUI and SendEmailGUI codes Commented	May, 7th, 2019
	M	Trade show (System Deployment)	May 8th, 2019

M = Milestones

Class and method overview

Overview of the classes and functionality.

Project parameter Interface

- Contains Port number constants, used by Client and Server
- Contains Author and About messages

Client

- Main method

LoginGui()

The LoginGui class has register and login functions. Register button enables the users to register with their username and password. After the user successfully registered, user inputs saved to the "Registered_list.csv". For the login button, it allows users to enter their username and password that they previously registered with and then check if it matches the records saved in the "Registered_list.csv." If it matches, a success message show up, if not, try again. The users are able to enjoy other functions of our project once they are successfully login.

- Client constructor
    - Set Gui for chat
    - Buttons for nickname, connect and send.
    - text fields for nickname, connect and send.
    - Options menu contain function for themes, send to email, save log, game and help.

-
    - Thread allow multiple clients to chat.

SetTheme()

This operation allows a user to modify the background and foreground colors of the main GUI. It comes with a light and dark feature, a colorful feature, RIT theme, and accommodating themes for those who are color blind. Color blindness is very common in men and for our male users who are blind it would be useful for them to see the colors of the theme. If I had more time and possibly help I would have implemented a customizable feature for the users to create their own themes or even changing the background to an image of their choice.

Dots()

The Dots class allows our users to play a stand-alone game while they're waiting for others to reply. This game is called "connect the dots." Users need to enter their name before starting the game. 1. Each time you can only connect 2 dots to a line. If you connect the line to a box then you gain one point. If your name is red, then it's your time to connect the dots. 3. If you gained one point, then you got one extra term. 4. The save button save the game for the next time, when you click on the load you will access the archive

Server

- Main method
    - Run Server class
-
    - Thread started for many clients

SendEmail()

The SendEmail class uses the JavaMail library to connect to send emails over Gmail. It works by creating the Session object, which handles the server configuration and authentication. Once the username and password is verified, the MimeMessage subclass handles the actual email content. The session object is passed into the MimeMessage constructor to create the message. The email is written by using several 'setter' methods in the MimeMessage class such as setFrom(), setSubject(), and setText(). Next, The MimeBodyPart and DataFileSource class handles the file attachments. The file is passed to the DataFileSource parameters which then gets put into the BodyPart using the setDataHandler() method. Next, the InternetAddress subclass is used to set the To/From email address. Finally, the Transport class uses the SMTP protocol to send the message.

SaveLog()

This function is incorporated in the GUI and its purpose is to write the chat log to a text file and save it on the computer. This method is invoked by the ActionListener when the user clicks the JButton created in the GUI. The method calls the SimpleDateFormat class to get the current timestamp. The time stamp prints the date and time in the log file. The FileWriter class is used to write the contents from the JTextArea to a log file. The purpose of this function is if the user wants to send a copy of their chat log over email. It is possible to attach the log and send it over the program.

!Failure. Chat Server UML Rough Draft

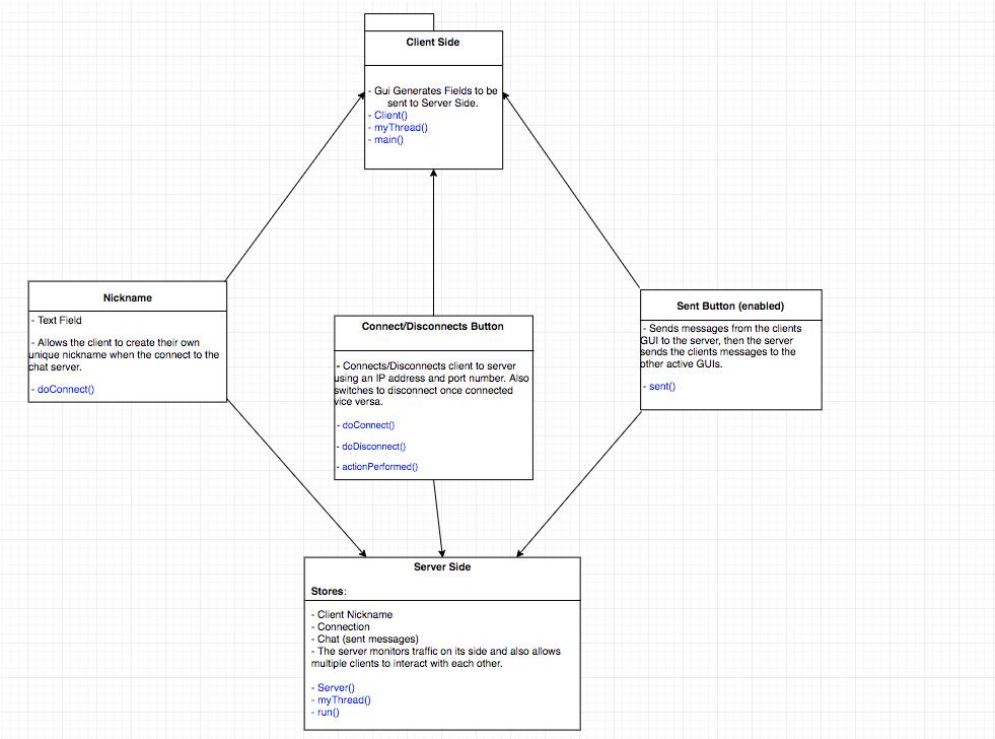

Figure 1. UML diagram rough draft #1

!Failure Chat Server UML Draft #2

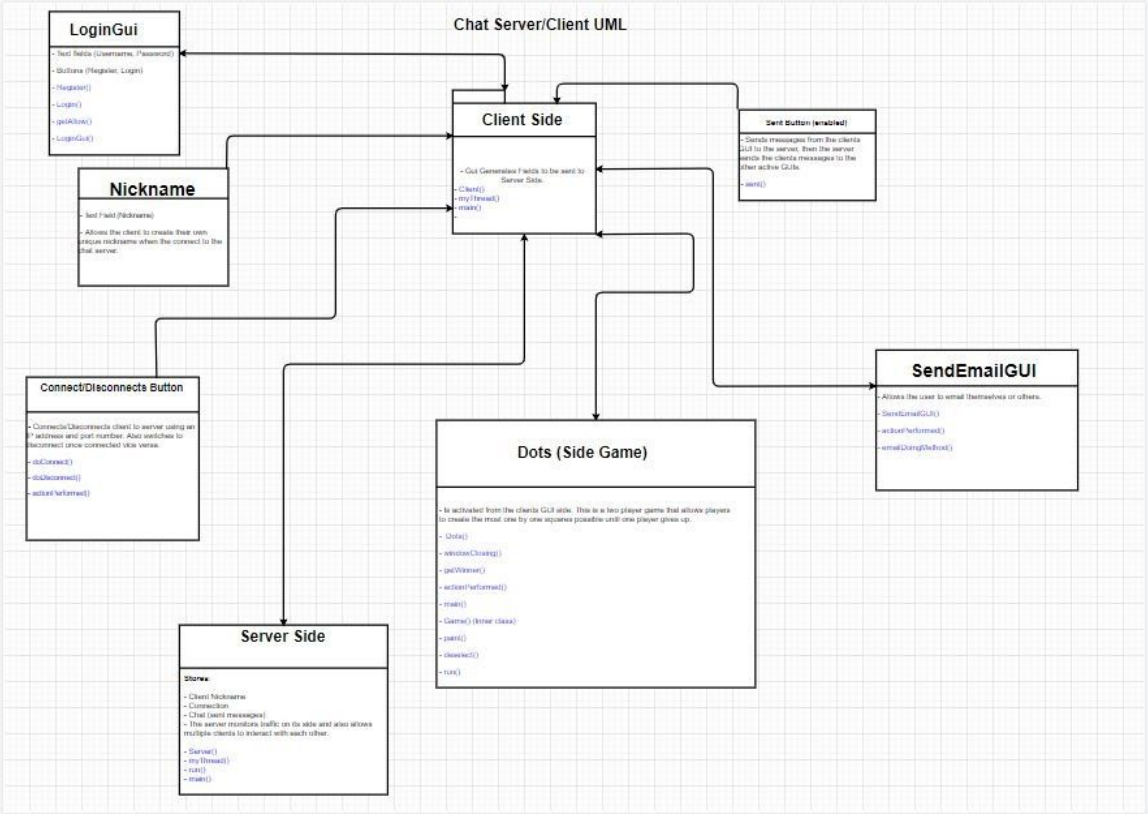

Figure 2. UML diagram rough draft #2

!Failure Chat Server UML Final Version

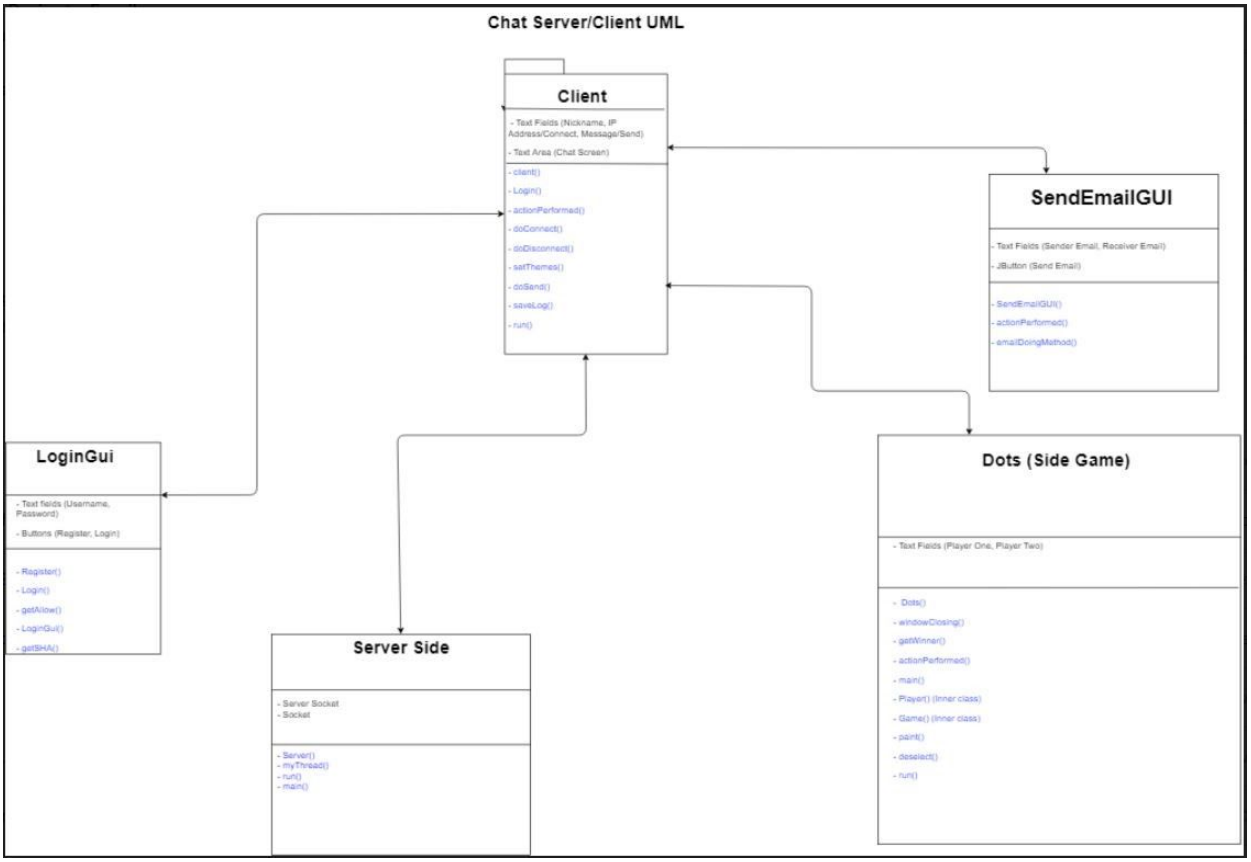

Figure 3. UML diagram final version

Client GUI

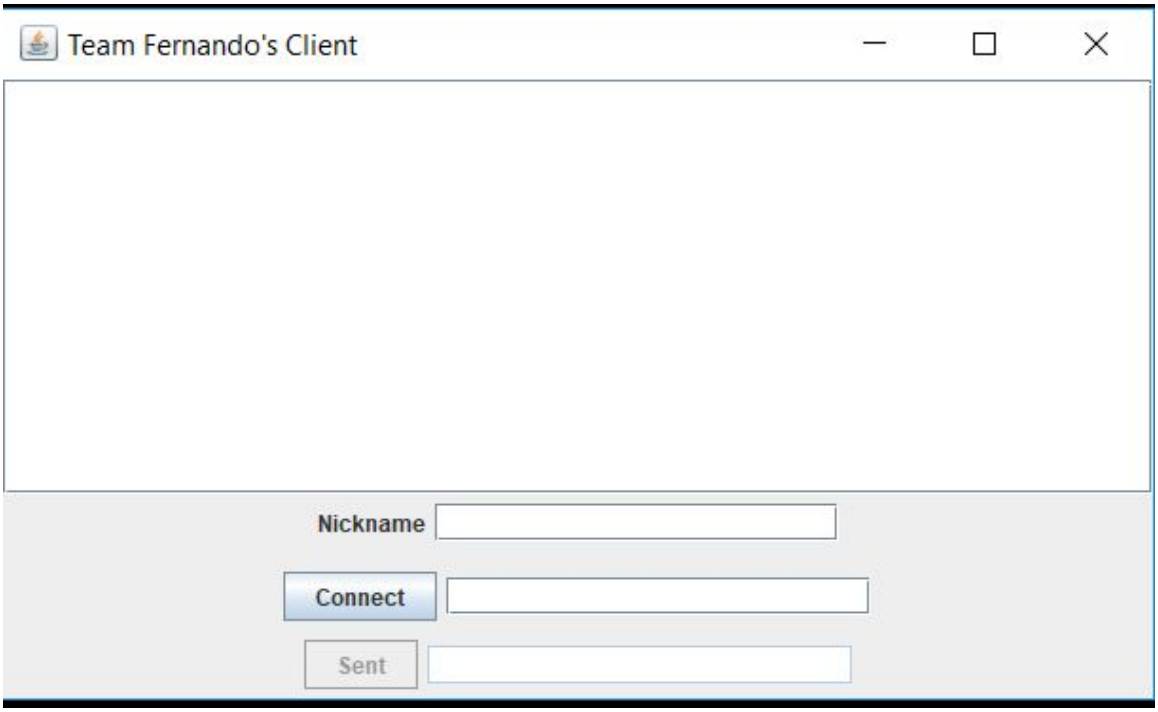

Figure 4. Screenshot of Client GUI in the beginning development phase

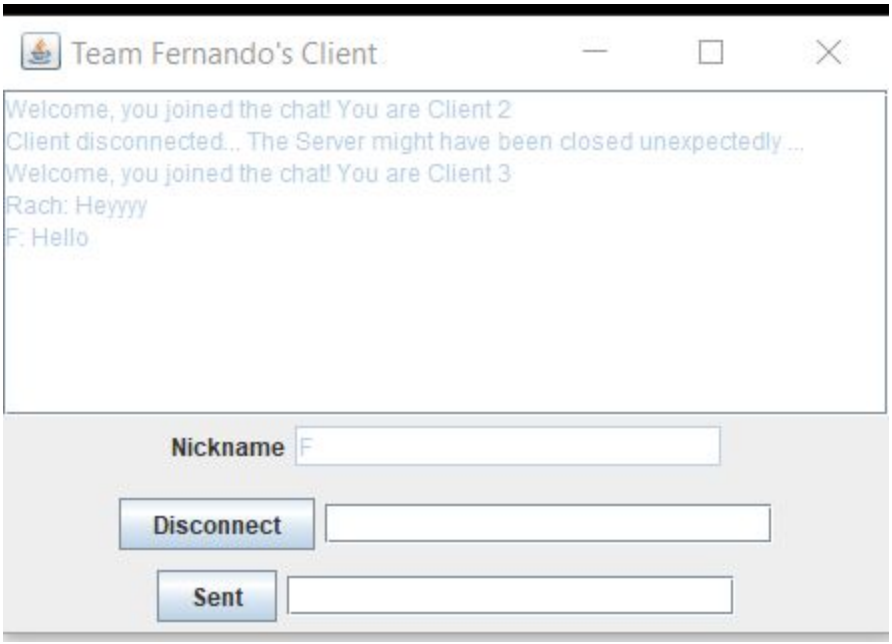

Figure 5. Screenshot of Client GUI - sample chat

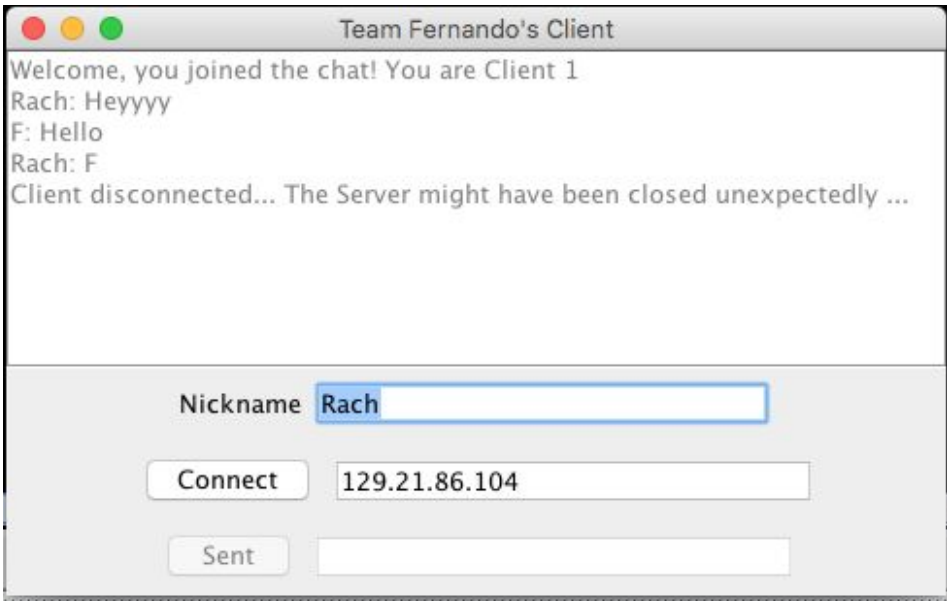

Figure 5a. Screenshot of Client GUI - sample chat

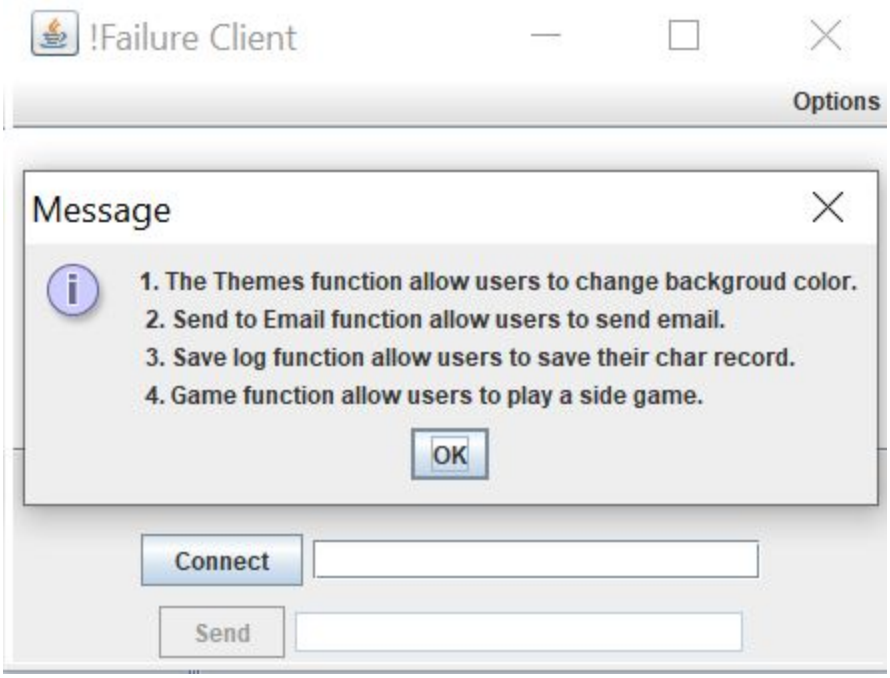

Figure 6. Screenshot of Client GUI - Table of Contents dialog

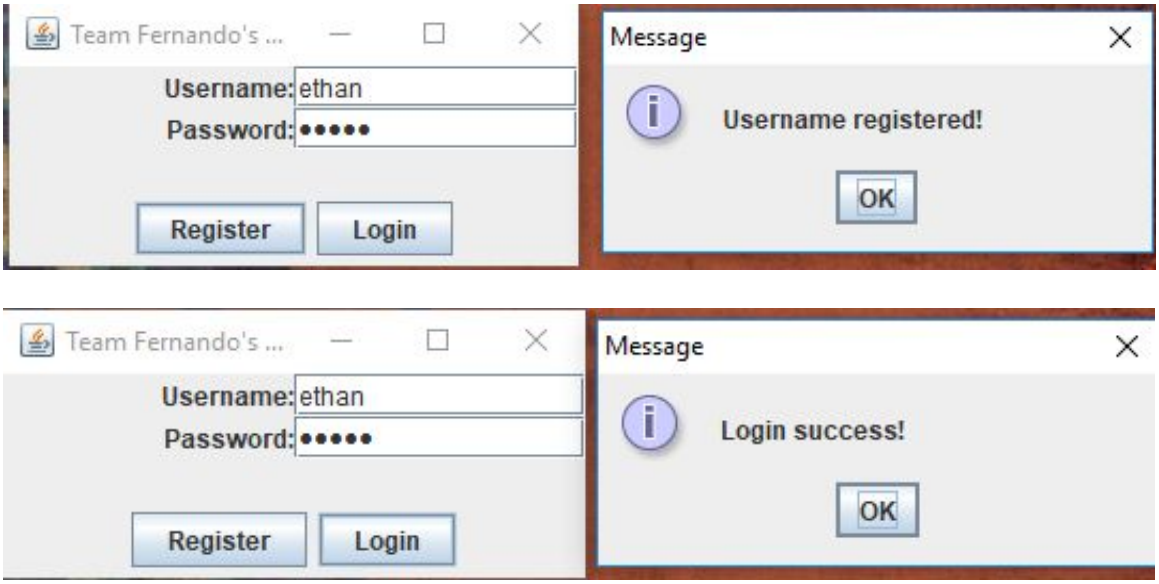

Figure 7. Screenshot of Login GUI - username registration/login

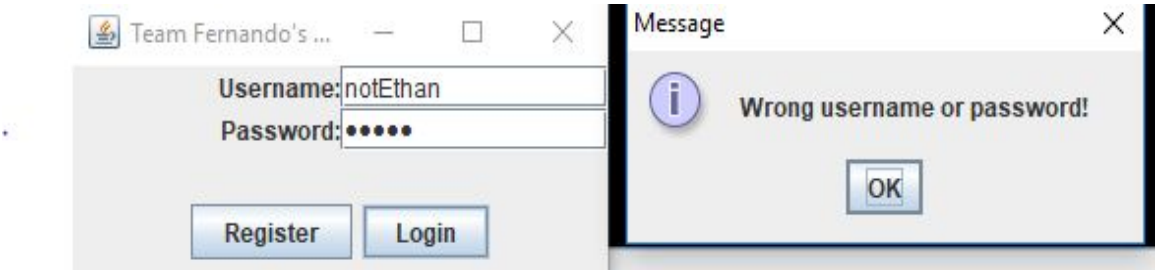

Figure 7a. Screenshot of Login GUI - error: incorrect login credentials

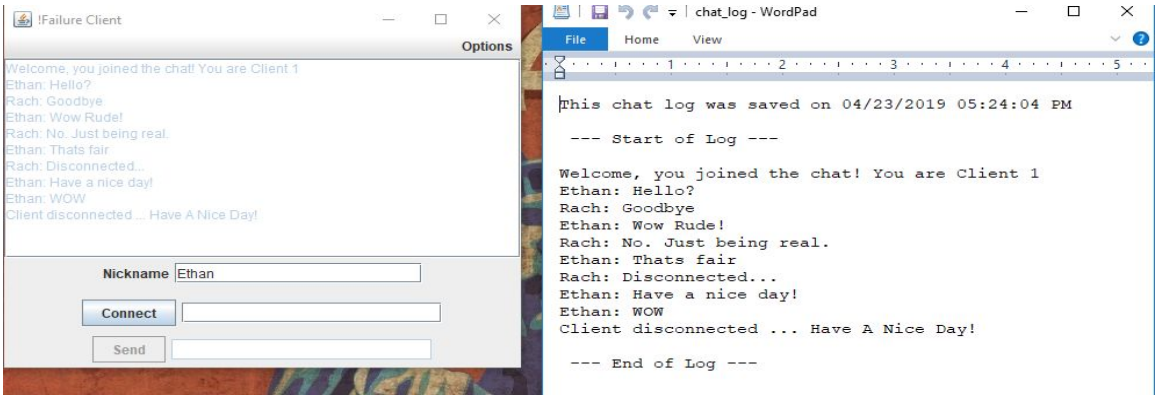

Figure 8. Screenshot of Client GUI - chat log saved to .txt file

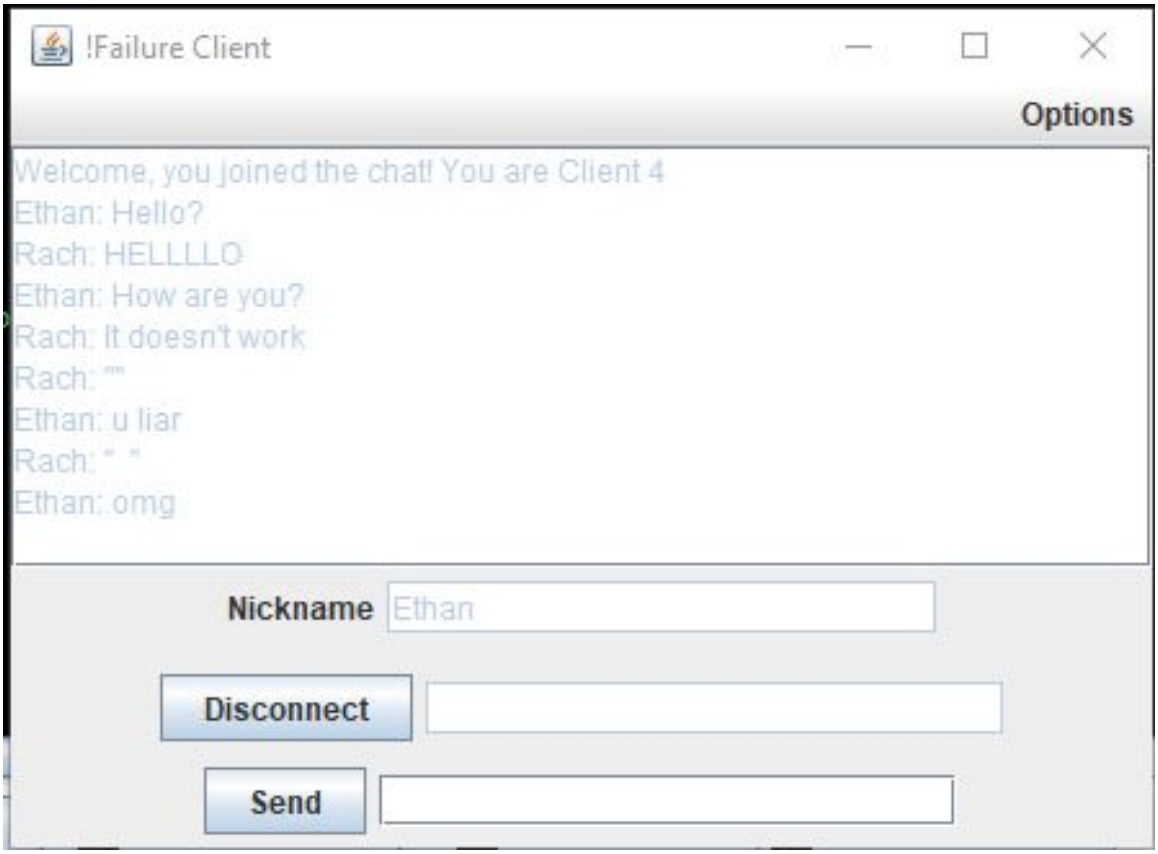

figure 9. Screenshot of two users being able to communicate.

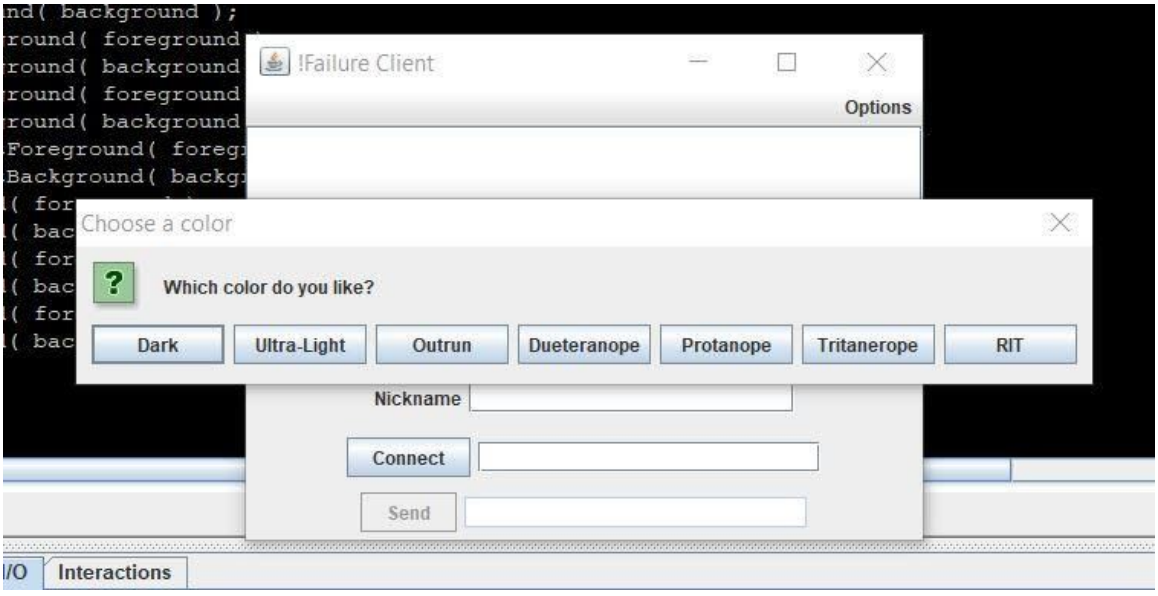

figure 10. Screenshot of being able to pick a theme for the GUI.

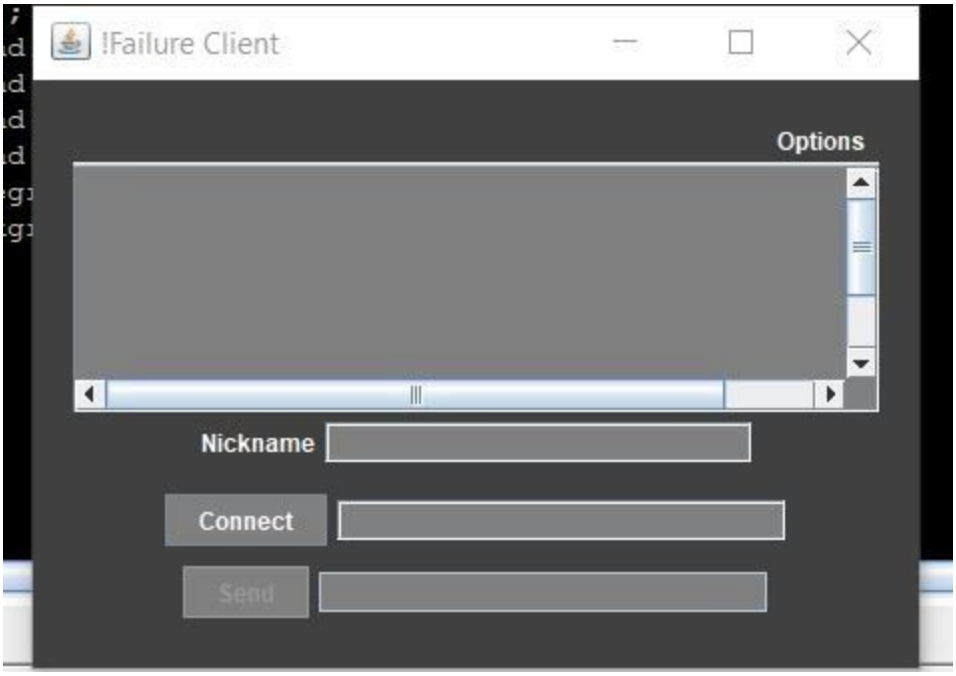

figure 10a. Screenshot of the Dark theme applied to the GUI.

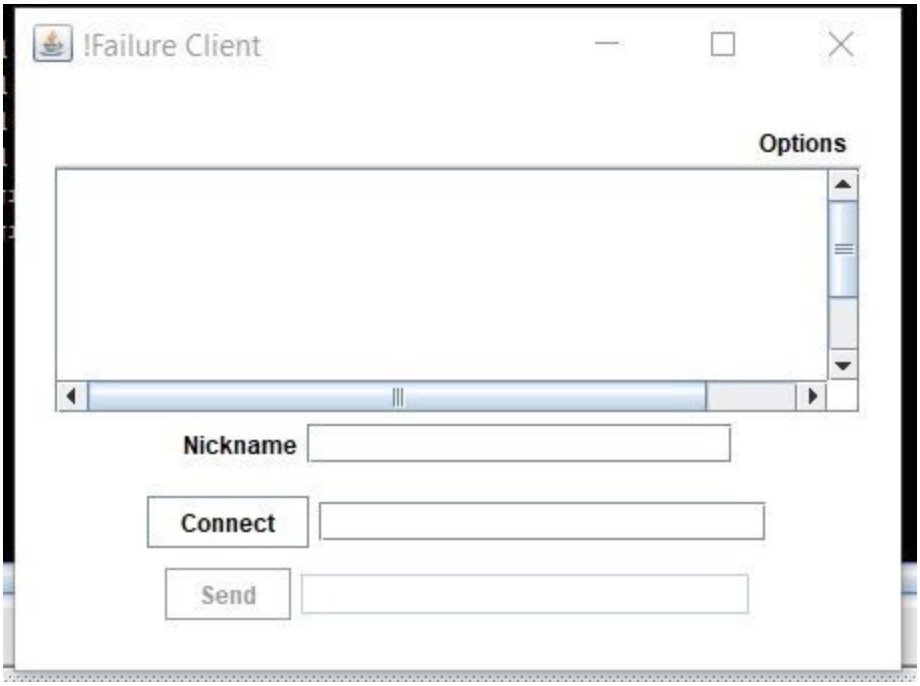

figure 10b. Screenshot of the Ultra-light theme applied to the GUI

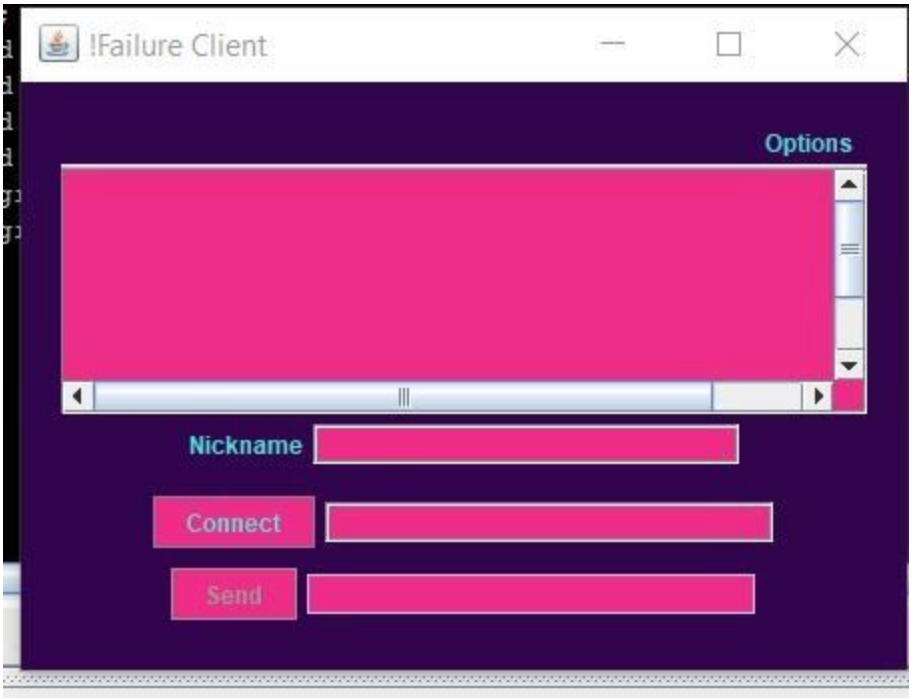

figure 10c. Screenshot of the Outrun theme applied to the GUI

figure 10d. Screenshot of the Dueteranope theme applied to the GUI

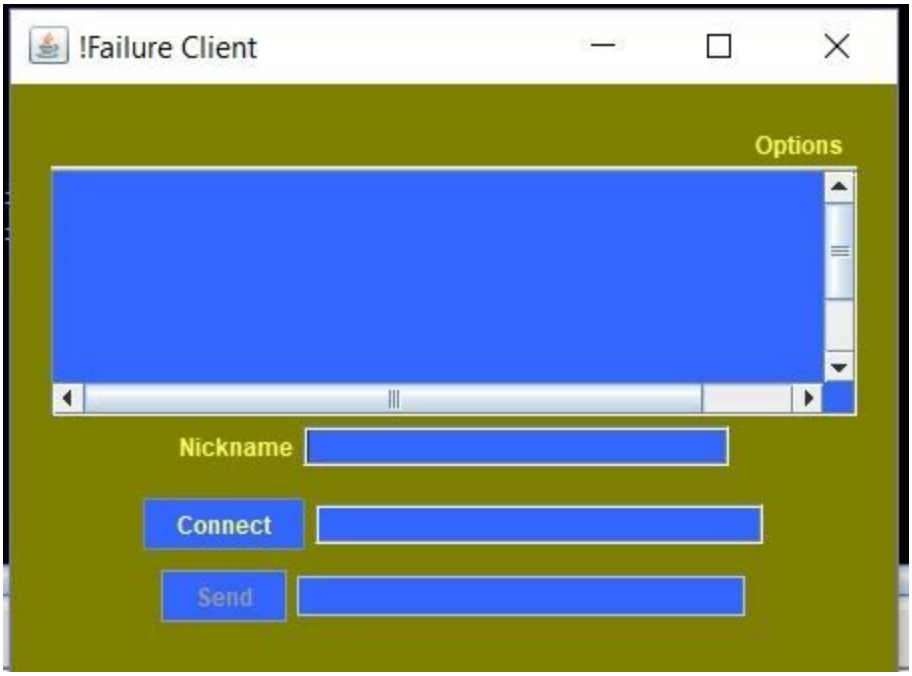

figure 10e. Screenshot of the Protanope theme applied to the GUI

figure 10f. Screenshot of the Tritanerope theme applied to the GUI

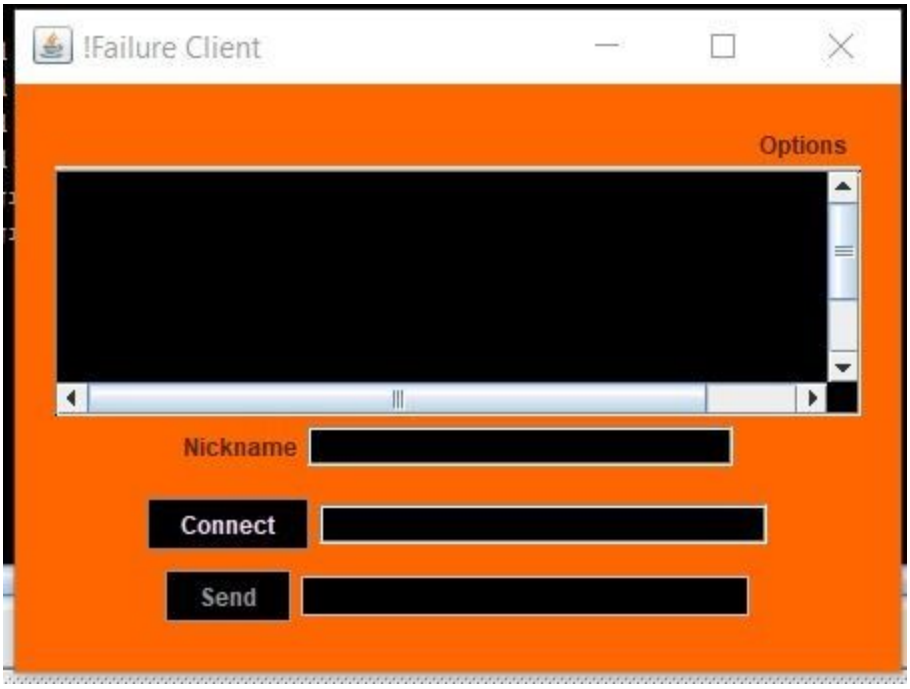

figure 10h. Screenshot of the RIT theme applied to the GUI

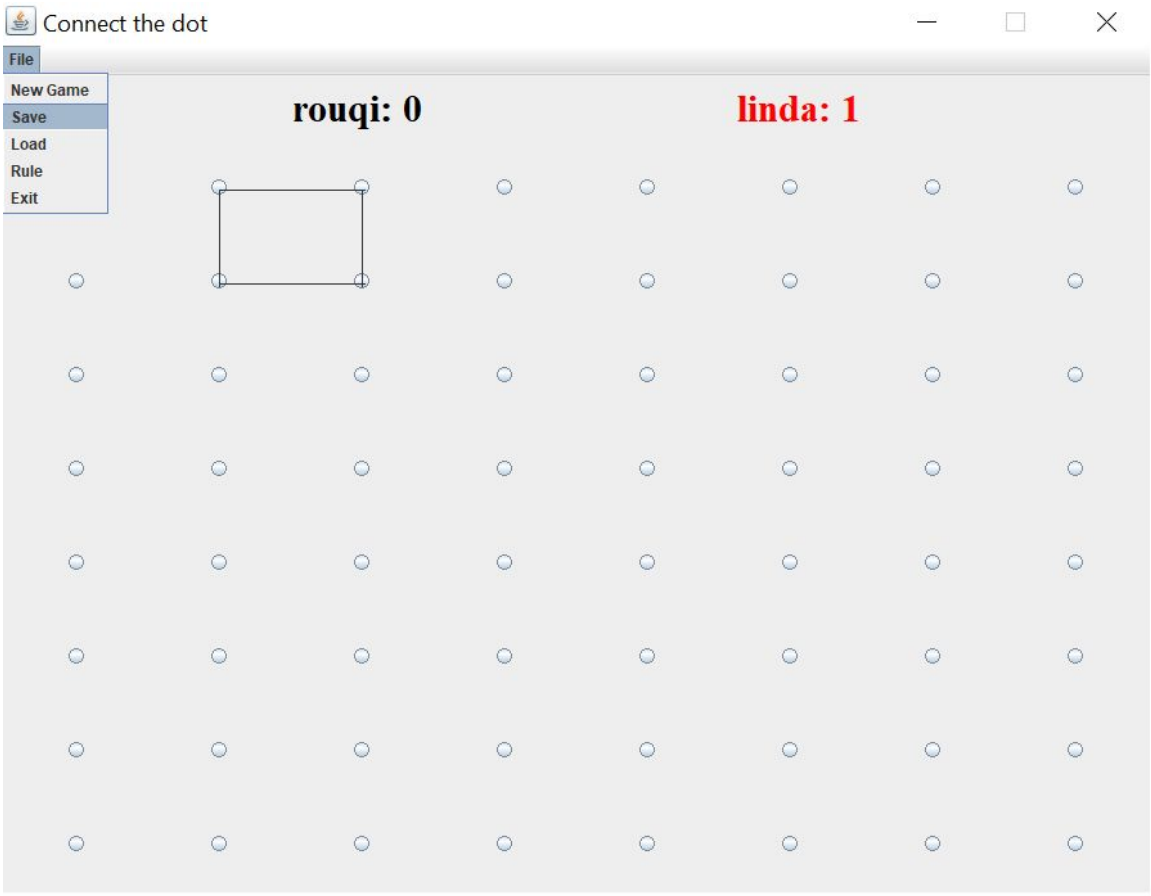

figure 11. Screenshot of Connect the Dots

Networking connections & Protocols

Client connection to server: Word of mouth from the person who is hosting the Server, for Security.

Port number(s): 16789 is the port number that we used. The purpose of the port number is so that there was a way to connect the server and the client.

Protocol interface code for both client and server:

Regular port numbers to connect to. The server accepts the socket with .accept().

Communication class

Chat interaction between Clients and Server.

Client	**Communication**	**Server**
		Startup
		Waits for client to connect
Client connection	Connection, no data **->**	Accept connection
Client create account	registration, username and password **->**	Server reads info
Client receives successful registration message	**<-** send message	Save username and password
Client login	Login, username and password	Server read info
Client receives successful login message	**<-** send message	Server check username and password if data match, Client successfully login, else, try again.
Client send message	message **->**	Server read message
		Server arrange message
Clients receive message.		

Data used

Javax.mail.jar was used for the email communication from the GUI to the user's email of choice.

Data files

Javax.mail.jar is used to set a classpath so that our code has access to the internet. It goes to the iste121project@gmail.com. This is where the code sends the emails from to any other existing email that the user typed in.

Punch List used

To do:

- Need to make a nicer GUI for the client chat room
- Create a way to make a private chat for one on one communication.
- Have an arraylist to see who is on at what times.
- Save the chat logs.
- Maybe create a way to register a username in a way so they don't have to set their own names each time they connect.
- Make a comparator so that all nicknames are unique.
- A stand-alone game.
- Allow users to make their own themes.

Done:

- Setting up a way to create nicknames for each individual client connected.
- Client and a nice GUI for chat room.
- Server that is multithreaded.
- stand-alone game.
- Save the chat logs.
- Function allow users to register and login.

Unresolved Issues

- When selecting a theme, it will shrink the GUI.
- On Mac devices the themes would not work like it would on Windows. Jacob suggested tinkering with the 'look-and-feel' options. Unresolved due to time constraints.
- The GUI hangs for almost a minute before sending the email.

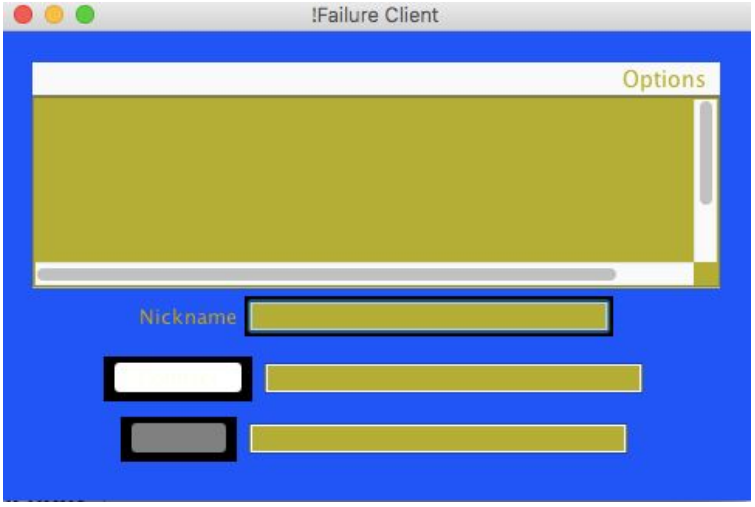

Figure 12. Black borders around JButton and the text is missing