Course: IT114-002-S2025
Assignment: IT114 Milestone 1
Student: Hitarth P. (hp627)
Status: Submitted | Worksheet Progress: 96.43%
Potential Grade: 9.90/10.00 (99.00%)
Received Grade: 0.00/10.00 (0.00%)
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-002-S2025/it114-milestone-1/grading/hp627

Instructions

1. Refer to Milestone1 of any of these docs:
    2. Rock Paper Scissors
    3. Basic Battleship
    4. Hangman / Word guess
    5. Trivia
    6. Go Fish
    7. Pictionary / Drawing
2. Ensure you read all instructions and objectives before starting.
3. Ensure you've gone through each lesson related to this Milestone
4. Switch to the Milestone1 branch
    1. `git checkout Milestone1` (ensure proper starting branch)
    2. `git pull origin Milestone1` (ensure history is up to date)
5. Copy `Part5` and rename the copy as `Project`
6. Organize the files into their respective packages `Client`, `Common`, `Server`, `Exceptions`
    1. Hint: If it's open, you can refer to the Milestone 2 Prep lesson
7. Fill out the below worksheet
    1. Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
8. Once finished, click "Submit and Export"
9. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"`
    3. `git push origin Milestone1`
    4. On Github merge the pull request from `Milestone1` to `main`
10. Upload the same PDF to Canvas
11. Sync Local
    1. `git checkout main`
    2. `git pull origin main`

Section #1: (1 pt.) Feature: Server Can Be Started Via Command Line And Listen To

Connections

Task #1 (1 pt.) - Evidence

Combo Task:

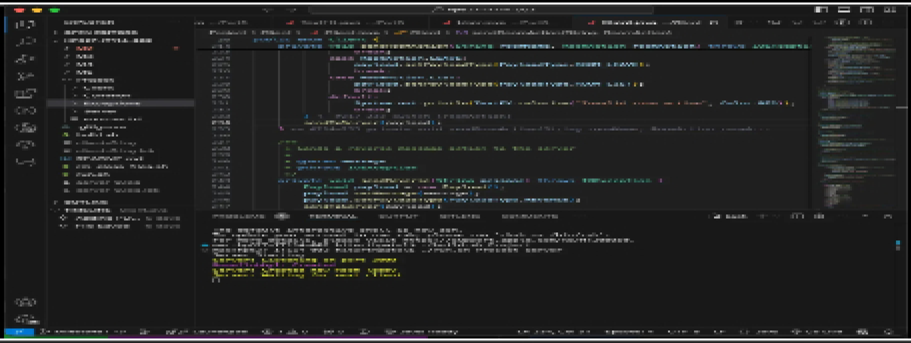
Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of the server started and listening
- Show the relevant snippet of the code that waits for incoming connections



Evidence

Saved: 4/7/2025 9:14:13 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the server-side waits for and accepts/handles connections

Your Response:

The server adds a ServerSocket runs and listens to a port 3000. The lobby has been created. The server enters a loop where it waits for client to connect.

Saved: 4/7/2025 9:14:13 AM

Section #2: (1 pt.) Feature: Server Should Be Able To Allow More Than One Client To Be Connected At Once

Task #1 (1 pt.) - Evidence

Combo Task:

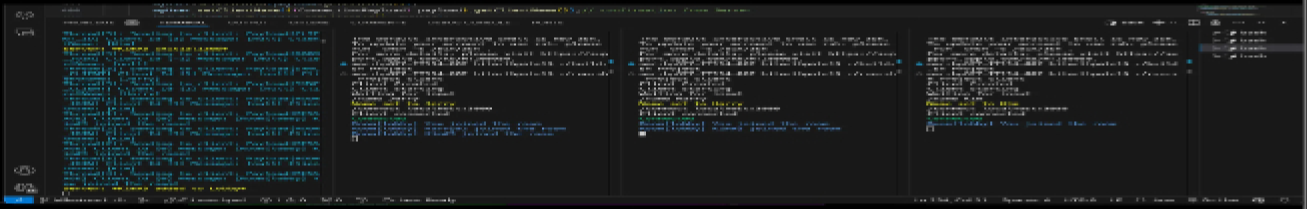
Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of the server receiving multiple connections
- Show at least 3 Clients connected (best to use the split terminal feature)
- Show the relevant snippets of code that handle logic for multiple connections



[illegible]

Evidence

Saved: 4/7/2025 9:35:00 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the server-side handles multiple connected clients

Your Response:

The server reference various different clients by listening to the Server Socket and is in a loop where it wait for the clients to connect. It makes a new ServerThread for the new clients that joins and where each client is independelty communicating with the sever. SeverThread tends to executes on a the independent thread, having the server to manage all the clients at the same time.

Saved: 4/7/2025 9:35:00 AM

Section #3: (2 pts.) Feature: Server Will Implement The Concept Of Rooms (With The Default Being "I obby")

Task #1 (2 pts.) - Evidence

Combo Task:

Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of rooms being created, joined, and removed (server-side)
- Show the relevant snippets of code that handle room management (create, join, leave, remove) (server-side)

Evidence

Saved: 4/7/2025 9:46:51 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the server-side handles room creation, joining/leaving, and removal

Your Response:

The server utilizes the HashMap in order to store the ongoing rooms. When the client runs the command /createroom, it makes sure that the room exits and creates a room if not. When the client joins the room they are added to the room that all the client are already in and removes you form the last room.

Saved: 4/7/2025 9:46:51 AM

Section #4: (1 pt.) Feature: Client Can Be Started Via The Command Line

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of the /name and /connect commands for each of 3 clients (best to use the split terminal feature)
- Output should show evidence of a successful connection
- Show the relevant snippets of code that handle the processes for /name, /connect, and the confirmation of being fully setup/connected

Evidence

Saved: 4/7/2025 12:11:54 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the /name and /connect commands work and the code flow that leads to a successful connection for the client

Your Response:

The /name command lets the client name in the local repository by having the myUser object. The /connect command makes a socket on the server which sends the name to the register and awaits a confirmation response.

Saved: 4/7/2025 12:11:54 PM

Section #5: (2 pts.) Feature: Client Can Create/j oin Rooms

Task #1 (2 pts.) - Evidence

Combo Task:

Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of the /createroom and /joinroom
- Output should show evidence of a successful creation/join in both scenarios
- Show the relevant snippets of code that handle the client-side processes for room creation and joining

Evidence

Saved: 4/7/2025 12:33:27 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the /createroom and /join room commands work and the related code flow for each

Your Response:

The /createroom sends a create_room to the payload for the server which makes the room and if the room doesn't exist it just ends up creating it. The /joinroom sends a join_room payload and the server make sures that the room is valid and moves the client to the room. If its sucessufull the client will print out You have joined the room and also update the room.

Saved: 4/7/2025 12:33:27 PM

Section #6: (1 pt.) Feature: Client Can Send Messages

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show the terminal output of a few messages from each of 3 clients
- Include examples of clients grouped into other rooms
- Show the relevant snippets of code that handle the message process from client to server-side

and back

Evidence

Saved: 4/7/2025 1:13:19 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the message code flow works

Your Response:

When a client send a message and it gets wrapped into the payload with a message which is sent to the server. The server receives it in Server thread and gets passed into the room that the client is in. Then the room relays the message to all clients to each recipient.

Saved: 4/7/2025 1:13:19 PM

Section #7: (1 pt.) Feature: Disconnection

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: *100%*

Objective: *Evidence*

Image Prompt

Weight: *50%*

Details:

- Show examples of clients disconnecting (server should still be active)
- Show examples of server disconnecting (clients should be active but disconnected)
- Show examples of clients reconnecting when a server is brought back online
- Examples should include relevant messages of the actions occuring
- Show the relevant snippets of code that handle the client-side disconnection process
- Show the relevant snippets of code that handle the server-side termination process

Quit

Rejoin

Saved: 4/7/2025 1:37:26 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how both client and server gracefully handle their disconnect/termination logic

Your Response:

The close() is used for clients handles for disconnect which stops the listening and prints a confirmation message or a error message that the server was lost. The server handle client disconnect by removing the client from the room and terminate everything.

Saved: 4/7/2025 1:37:26 PM

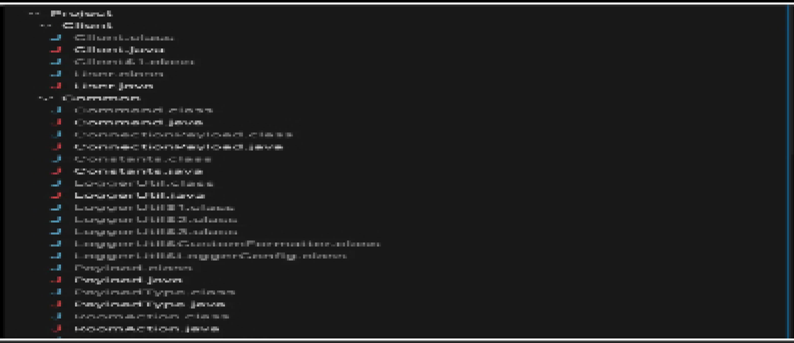
Section #8: (1 pt.) Misc

Task #1 (0.25 pts.) - Show the proper workspace structure with the

Image Prompt

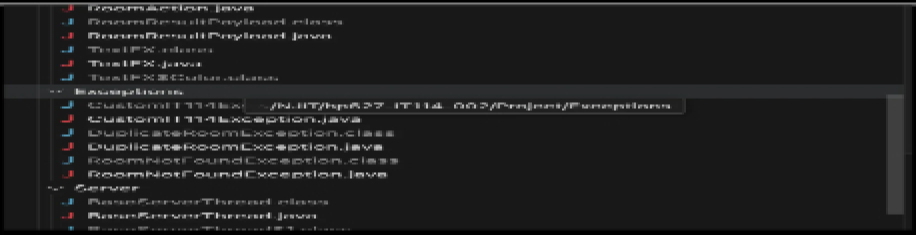
Weight: *25%*

Objective: *Show the proper workspace structure with the new Client, Common, Server, and Exceptions packages*



```
⌄ Project
  ⌄ Client
      Client.class
      Client.java
      Client$1.class
      User.class
      User.java
  ⌄ Common
      Command.class
      Command.java
      ConnectionPayload.class
      ConnectionPayload.java
      Constants.class
      Constants.java
      LoggerUtil.class
      LoggerUtil.java
      LoggerUtil$1.class
      LoggerUtil$2.class
      LoggerUtil$3.class
      LoggerUtil$CustomFormatter.class
      LoggerUtil$LoggerConfig.class
      Payload.class
      Payload.java
      PayloadType.class
      PayloadType.java
      RoomAction.class
      RoomAction.java
```

structure



```
      RoomAction.java
      RoomResultPayload.class
      RoomResultPayload.java
      TextFX.class
      TextFX.java
      TextFX$Color.class
  ⌄ Exceptions
      CustomIT114Ex  ~/NJIT/hp627-IT114-002/Project/Exceptions
      CustomIT114Exception.java
      DuplicateRoomException.class
      DuplicateRoomException.java
      RoomNotFoundException.class
      RoomNotFoundException.java
  ⌄ Server
      BaseServerThread.class
      BaseServerThread.java
```

Task #2 (0.25 pts.) - Github Details

Combo Task:

Weight: *25%*
Objective: *Github Details*

Image Prompt

Weight: *60%*

Details:
From the Commits tab of the Pull Request screenshot the commit history

Missing Caption

Saved: 4/7/2025 1:39:57 PM

Url Prompt

Weight: *40%*

Details:
Include the link to the Pull Request (should end in `/pull/#`)

URL #1
https://github.com/hitarthpat/hp627-IT114-002/

URL
https://github.com/hitarthpat/hp62

Saved: 4/7/2025 1:39:57 PM

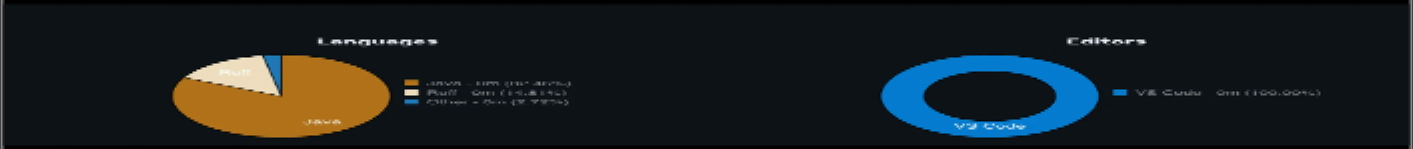
Task #3 (0.25 pts.) - WakaTime - Activity

Image Prompt

Weight: *25%*
Objective: *WakaTime - Activity*

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



WakaTime

Saved: 4/7/2025 1:40:34 PM

Task #4 (0.25 pts.) - Reflection

Weight: *25%*
Objective: *Reflection*

Sub-Tasks:

Task #1 (0.33 pts.) - What did you learn?

Text Prompt

Weight: *33.33%*
Objective: *What did you learn?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

Using Java Sockets, I have learned the client-server communication, where clients connect to a server, send messages, and join/create chat rooms. I have learned how to handle the entire procedure of disconnection from the client end and server end and a way of reconnecting clients after server restart. I can see how payload process and the message that it relays with from one client to the other one in a shared room. I also learned understand structuring multi-client server applications and handling real-time communications.

Saved: 4/7/2025 1:45:24 PM

Task #2 (0.33 pts.) - What was the easiest part of the assignr

Text Prompt

Weight: *33.33%*
Objective: *What was the easiest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of the assignment was using the /name and /connect commands to connect to the server. All I had to do was run the server and the commands would just work normally.

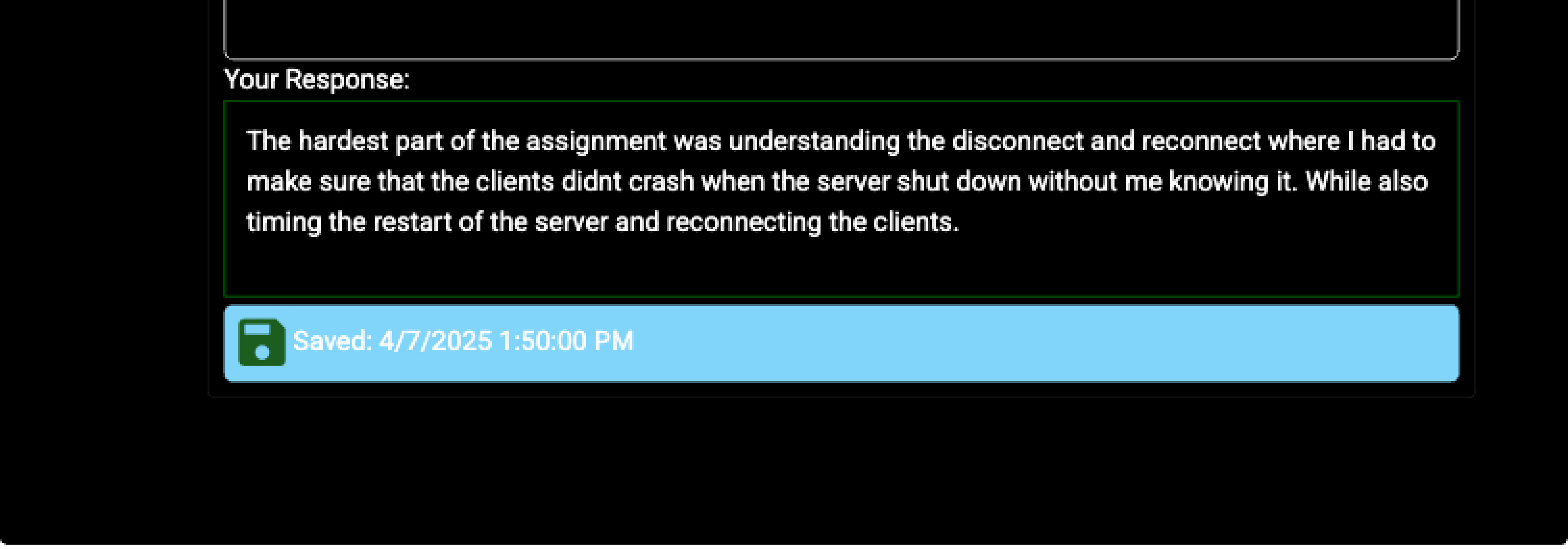
Saved: 4/7/2025 1:47:14 PM

Task #3 (0.33 pts.) - What was the hardest part of the assign

Text Prompt

Weight: *33.33%*
Objective: *What was the hardest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)



Your Response:

The hardest part of the assignment was understanding the disconnect and reconnect where I had to make sure that the clients didnt crash when the server shut down without me knowing it. While also timing the restart of the server and reconnecting the clients.

Saved: 4/7/2025 1:50:00 PM