Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Module 4 Sockets Part3 Challenge
Student: Daniel C. (dvc2)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 6/24/2025 2:01:23 AM
Updated: 6/24/2025 3:22:02 AM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/grading/dvc2
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/view/dvc2

Instructions

- Overview Link: https://youtu.be/_029E_aBTFo

1. Ensure you read all instructions and objectives before starting.
2. Create a new branch from `main` called `M4-Homework`
   1. `git checkout main` (ensure proper starting branch)
   2. `git pull origin main` (ensure history is up to date)
   3. `git checkout -b M4-Homework` (create and switch to branch)
3. Copy the template code from here: GitHub Repository - M4 Homework
   - It includes Sockets `Part1`, `Part2`, and `Part3`. Put all into an `M4` folder or similar if you don't have them yet (adjust `package` reference at the top if you chose a different folder name).
   - Make a copy of `Part3` and call it `Part3HW`
     - Fix the `package` and `import` references at the top of each file in this new folder (Note: you'll only be editing files in `Part3HW`)
   - Immediately record to history
     - `git add .`
     - `git commit -m "adding M4 HW baseline files"`
     - `git push origin M4-Homework`
     - Create a Pull Request from `M4-Homework` to `main` and keep it open
4. Fill out the below worksheet
   - Each Problem requires the following as you work
     - Ensure there's a comment with your UCID, date, and brief summary of how the problem was solved
     - Code solution (add/commit periodically as needed)
     - Hint: Note how `/reverse` is handled
5. Once finished, click "Submit and Export"
6. 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 M4-Homework`
        4. On Github merge the pull request from `M4-Homework` to `main`
7. Upload the same PDF to Canvas
8. Sync Local
        1. `git checkout main`
        2. `git pull origin main`

Section #1: (3 pts.) Challenge 1 - Coin Flip

Progress: 100%

Task #1 (3 pts.) - Implement a Coin Flip Command

Progress: 100%

Details:

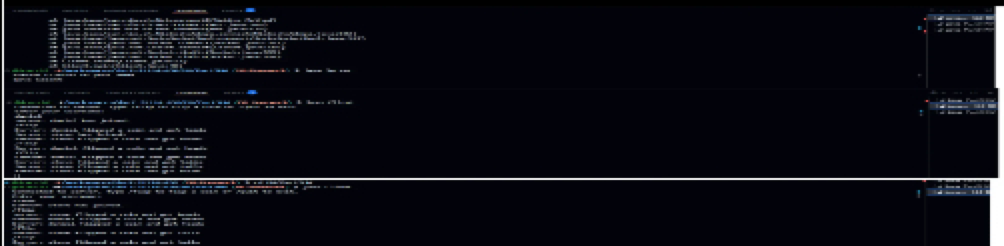
- `Client` must capture the user entry and generate a valid command per the lesson details
    - Command format must be `/flip`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
    - The message must be in the format of `<who> flipped a coin and got <result>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

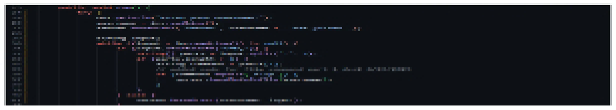
Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
    - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
    - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
    - Should only need to create a new method and pass the result message to `relay()`
4. Show 5 examples of the command being seen across all terminals (2+ Clients and 1 Server)
    1. This can be captured in one screenshot if you split the terminals side by side


[illegible]

code



results

Saved: 6/24/2025 2:22:01 AM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/dcarch2/dvc2-it114-450/main/M4/Part3HW/Client.java

URL
https://github.com/dcarch2/dvc2-

Saved: 6/24/2025 2:22:01 AM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenge (note: this isn't the same as `what` the code does)

Your Response:

My code solves the challenge by creating a client-server connection then handling the message broadcasting and displaying my output text through real-time communication.

Saved: 6/24/2025 2:22:01 AM

Section #2: (3 pts.) Challenge 2 - Private Message

Progress: 100%

Task #1 (3 pts.) - Implement a Private Message Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/pm <target id> <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic
  - The message must be in the format of `PM from <who>: <message>` and be from the Server
  - The result must only be sent to the original sender and to the receiver/target
- Add code to solve the problem (add/commit as needed)

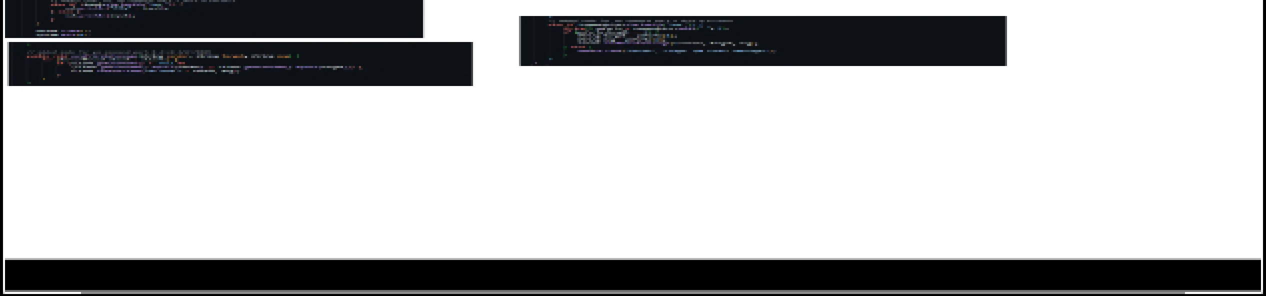
Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
   - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
   - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
   - Should only need to create a new method and send the result message to just the sender and receiver
4. Show 3 examples of the command being seen across all terminals (3+ Clients and 1 Server)
   1. This can be captured in one screenshot if you split the terminals side by side
   2. Note: Only the sender and the receiver should see the private message (show variations across different users)

results



code

Saved: 6/24/2025 2:59:35 AM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/dcarch2/dvc2-it114-450/M4-Homework/M4/Part3HW/Client.java

URL
https://github.com/dcarch2/dvc2-

Saved: 6/24/2025 2:59:35 AM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

My code solves the challenge by using threads and sockets to manage multiple clients and allow for private and or public messaging outputs.

Saved: 6/24/2025 2:59:35 AM

Section #3: (3 pts.) Challenge 3 - Shuffle Message

Progress: 100%

Task #1 (3 pts.) - Implement a Shuffle Message Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/shuffle <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
  - The message must be in the format of `Shuffled from <who>: <shuffled_message>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
   - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
   - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
   - Should only need to create a new method and do similar logic to `relay()`
4. Show 3 examples of the command being seen across all terminals (2+ Clients and 1 Server)
   1. This can be captured in one screenshot if you split the terminals side by side

code

results

Saved: 6/24/2025 3:14:51 AM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/dcarch2/dvc2-it114-450/M4-Homework/M4/Part3HW/Client.java

URL
https://github.com/dcarch2/dvc2-

Saved: 6/24/2025 3:14:51 AM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

My code solves the challenge by detecting the shuffle command in the client, sending it to the server and having it processed through serverthread, then finaly using the code I provided to shuffle the message and output the result to all the clientside users.

Saved: 6/24/2025 3:14:51 AM

Section #4: (1 pt.) Misc

Progress: 100%

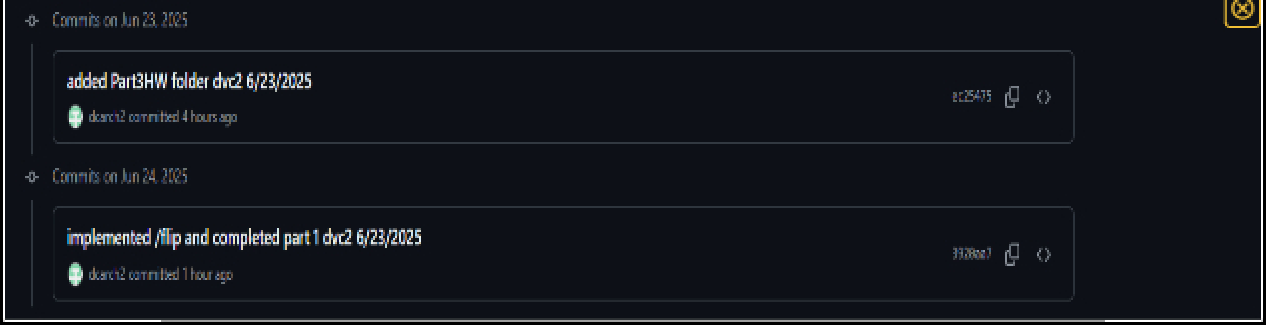
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

Details:
From the Commits tab of the Pull Request screenshot the commit history Following minimum should be present


Commits on Jun 23, 2025

added Part3HW folder dvc2 6/23/2025
dcarch2 committed 4 hours ago
ec25475

Commits on Jun 24, 2025

implemented /flip and completed part 1 dvc2 6/23/2025
dcarch2 committed 1 hour ago
3328ea7

commits, not sure why the others are not showing, should be visible in main commits

Saved: 6/24/2025 3:18:38 AM

Part 2:

Progress: 100%

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

URL #1
https://github.com/dcarch2/dvc2-it114-450/pull/4

URL
https://github.com/dcarch2/dvc2-

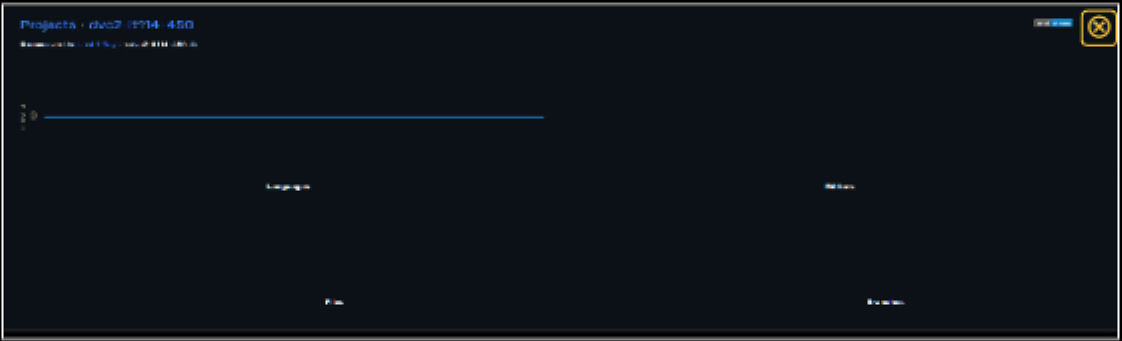
Saved: 6/24/2025 3:18:38 AM

Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

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


Projects · dvc2-it114-450

not showcasing my time spent, but it is visible on github, not sure what this issue is.

Saved: 6/24/2025 3:19:22 AM

Task #3 (0.33 pts.) - Reflection

Progress: 100%

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

Progress: 100%

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

Your Response:

I learned how to implement command based messaging between clients in a server within Java. I was able to practice user commands, broadcasting messages, and more logic use for my special commands which helped a lot.

Saved: 6/24/2025 3:20:33 AM

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

Progress: 100%

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

Your Response:

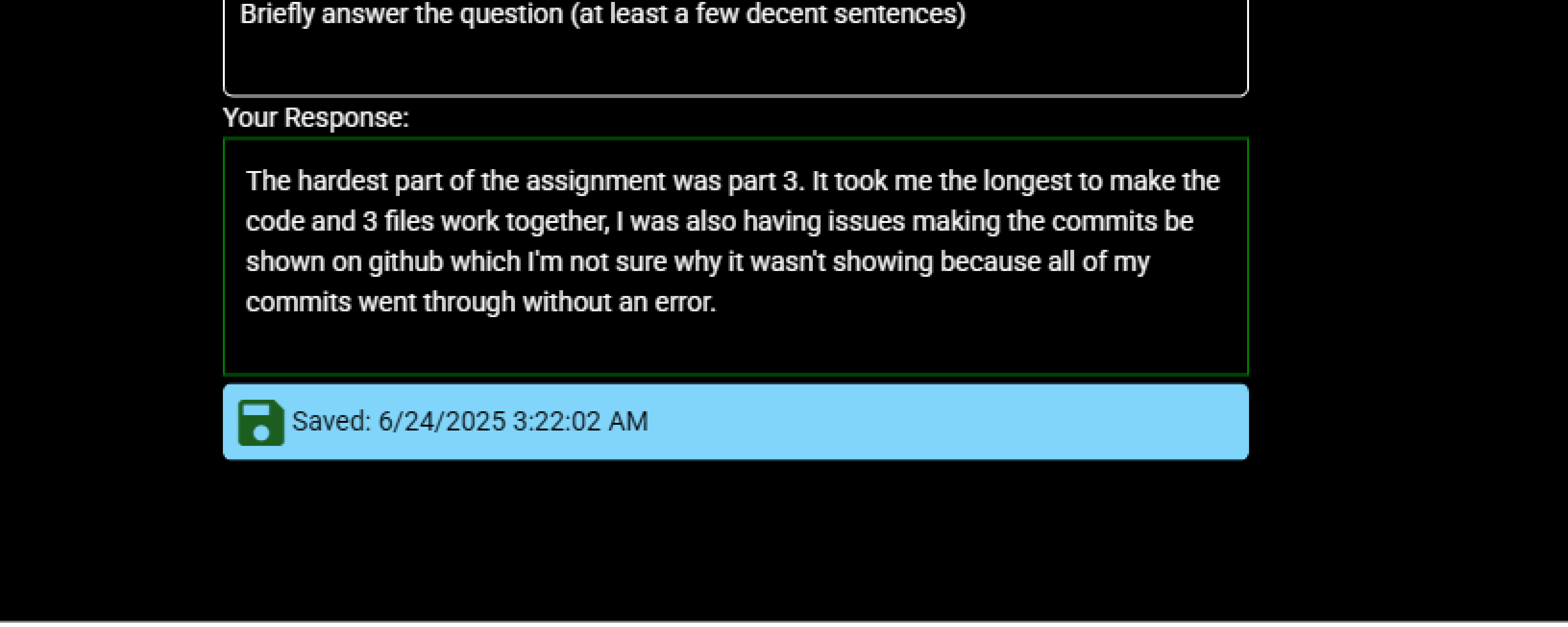
The easiest part of the assignment was part 1. I was able to complete the code for it with the least difficult out of all the other parts and it was commited smoothly.

Saved: 6/24/2025 3:21:04 AM

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

Progress: 100%

Details:



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

Your Response:

The hardest part of the assignment was part 3. It took me the longest to make the code and 3 files work together, I was also having issues making the commits be shown on github which I'm not sure why it wasn't showing because all of my commits went through without an error.

Saved: 6/24/2025 3:22:02 AM