Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-project-milestone-1/grade/mbh3

IT114-004-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/26/2024 10:44:12 PM

Instructions

COLLAPSE

1. Create a new branch called Milestone1
2. At the root of your repository create a folder called Project if one doesn't exist yet
    1. You will be updating this folder with new code as you do milestones
    2. You won't be creating separate folders for milestones; milestones are just branches
3. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
4. Copy in the latest Socket sample code from the most recent Socket Part example of the lessons
    1. Recommended Part 5 (clients should be having names at this point and not ids)
    2. https://github.com/MattToegel/IT114/tree/Module5/Module5
5. Fix the package references at the top of each file (these are the only edits you should do at this point)
6. Git add/commit the baseline and push it to github
7. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
8. Ensure the sample is working and fill in the below deliverables
    1. Note: The client commands likely are different in part 5 with the /name and /connect options instead of just "connect"
9. Generate the worksheet output file once done and add it to your local repository
10. Git add/commit/push all changes
11. Complete the pull request merge from step 7
12. Locally checkout main
13. git pull origin main

Branch name: Milestone1

Tasks: 9 Points: 10.00

Start Up (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Server and Client Initialization

Checklist

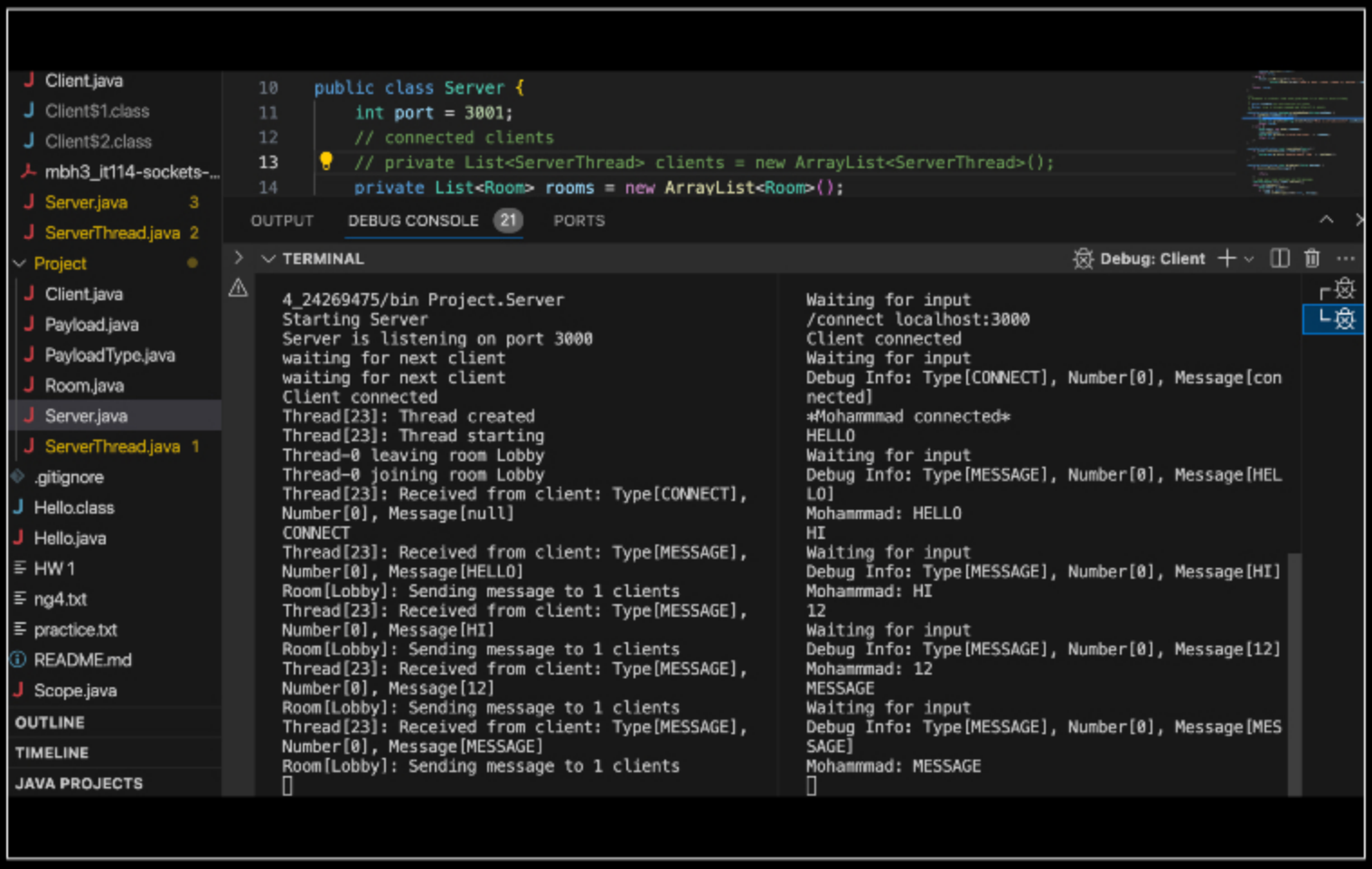
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Server should properly be listening to its port from the command line (note the related message)
#2	1	Clients should be successfully waiting for input
#3	1	Clients should have a name and successfully connected to the server (note related messages)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Successfully connected to the server and gave it my name which is Mohammad, as you can see from there.

Checklist Items (3)

#1 Server should properly be listening to its port from the command line (note the related message)

#2 Clients should be successfully waiting for input

#3 Clients should have a name and successfully connected to the server (note related messages)

COLLAPSE

Task #2 - Points: 1

Text: Explain the connection process

Details:

Note the various steps from the beginning to when the client is fully connected and able to communicate in the room.

Emphasize the code flow and the sockets usage.

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how the server-side of the connection works
#2	1	Mention how the client-side of the connection works
#3	1	Describe the socket steps until the server is waiting for messages from the client

Response:

When a client wants to connect to a server using WebSockets, the process starts with the client sending a connection request to the server. The server, already listening for such requests, accepts the connection if everything checks out (like security protocols). Once accepted, both the server and the client keep this connection open, allowing them to send messages back and forth in real-time. The server can handle multiple client connections by assigning each a unique socket. On the client side, it uses this socket to communicate with the server, sending messages that the server can then broadcast to other clients or respond to directly. During this whole process, the server is always waiting and ready to process any messages that come from the client, effectively making the communication between the two seamless and continuous.

Communication (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add screenshot(s) showing evidence related to the checklist

Checklist

*The checkboxes are for your own tracking

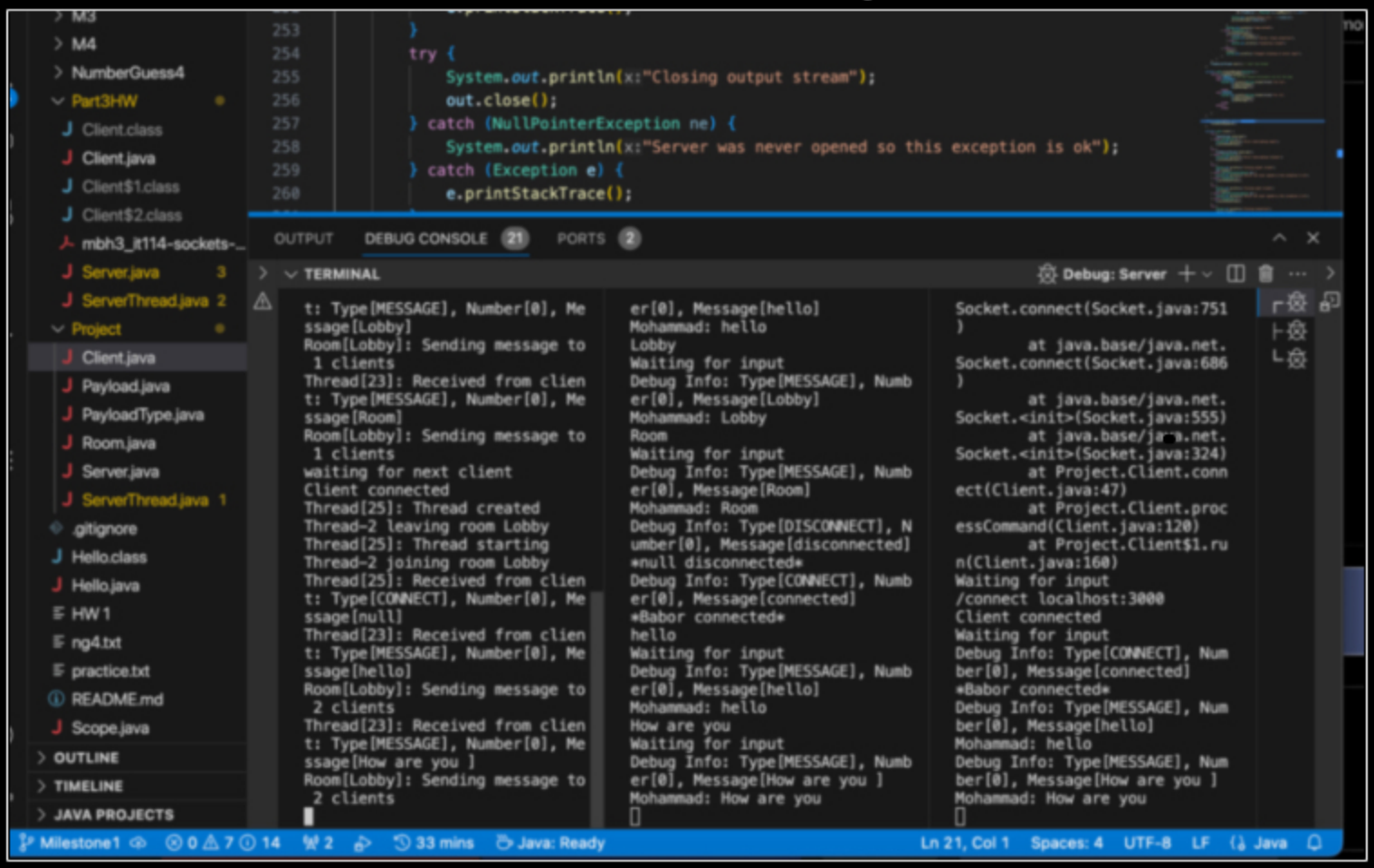
#	Points	Details
#1	1	At least two clients connected to the server
#2	1	Client can send messages to the server
#3	1	Server sends the message to all clients in the same room
#4	1	Messages clearly show who the message is from (i.e., client name is clearly with the message)

#5	2	Demonstrate clients in two different rooms can't send/receive messages to each other (clearly show the clients are in different rooms via the commands demonstrated in the lessons
#6	1	Clearly caption each image regarding what is being shown

Task Screenshots:

Gallery Style: Large View

Small Medium Large



As you can see 2 clients are connected to the server and can receive and send messages.

Checklist Items (6)

#1 At least two clients connected to the server

#2 Client can send messages to the server

#3 Server sends the message to all clients in the same room

#4 Messages clearly show who the message is from (i.e., client name is clearly with the message)

#5 Demonstrate clients in two different rooms can't send/receive messages to each other (clearly show the clients are in different rooms via the commands demonstrated in the lessons

#6 Clearly caption each image regarding what is being shown

COLLAPSE

Task #2 - Points: 1

Text: Explain the communication process

Details:
How are messages entered from the client side and how do they propagate to other clients?

Note all the steps involved and use specific terminology from the code.
Don't just translate the code line-by-line to plain English, keep it concise.

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention the client-side (sending)
#2	1	Mention the ServerThread's involvement
#3	1	Mention the Room's perspective
#4	1	Mention the client-side (receiving)

Response:

Messages from the client side are entered through the client's user interface (e.g., a terminal or graphical interface), where the user types a message and sends it to the server. This process begins with the client constructing a Payload object, setting its type to MESSAGE, including the message content and the client's name, then serializing and sending this object to the server over the established Socket connection.

Upon receiving the message, the ServerThread associated with that client deserializes the Payload, checks the payload type, and if it's a message, forwards it to the Room object that represents the current chat room the client is in. The Room then iterates over its list of ServerThread objects, each representing a connected client within the same room, and calls their sendMessage method to distribute the message. This method serializes a new Payload object with the message and the original sender's name and sends it over the socket to each client.

On the client-side (receiving), each client's listening thread deserializes the incoming Payload objects. If the payload type indicates a message, the client displays the message content along with the sender's name in the user interface, allowing messages sent by one client to be seen by all other clients in the same room

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add screenshot(s) showing evidence related to the checklist

Checklist

*The checkboxes are for your own tracking

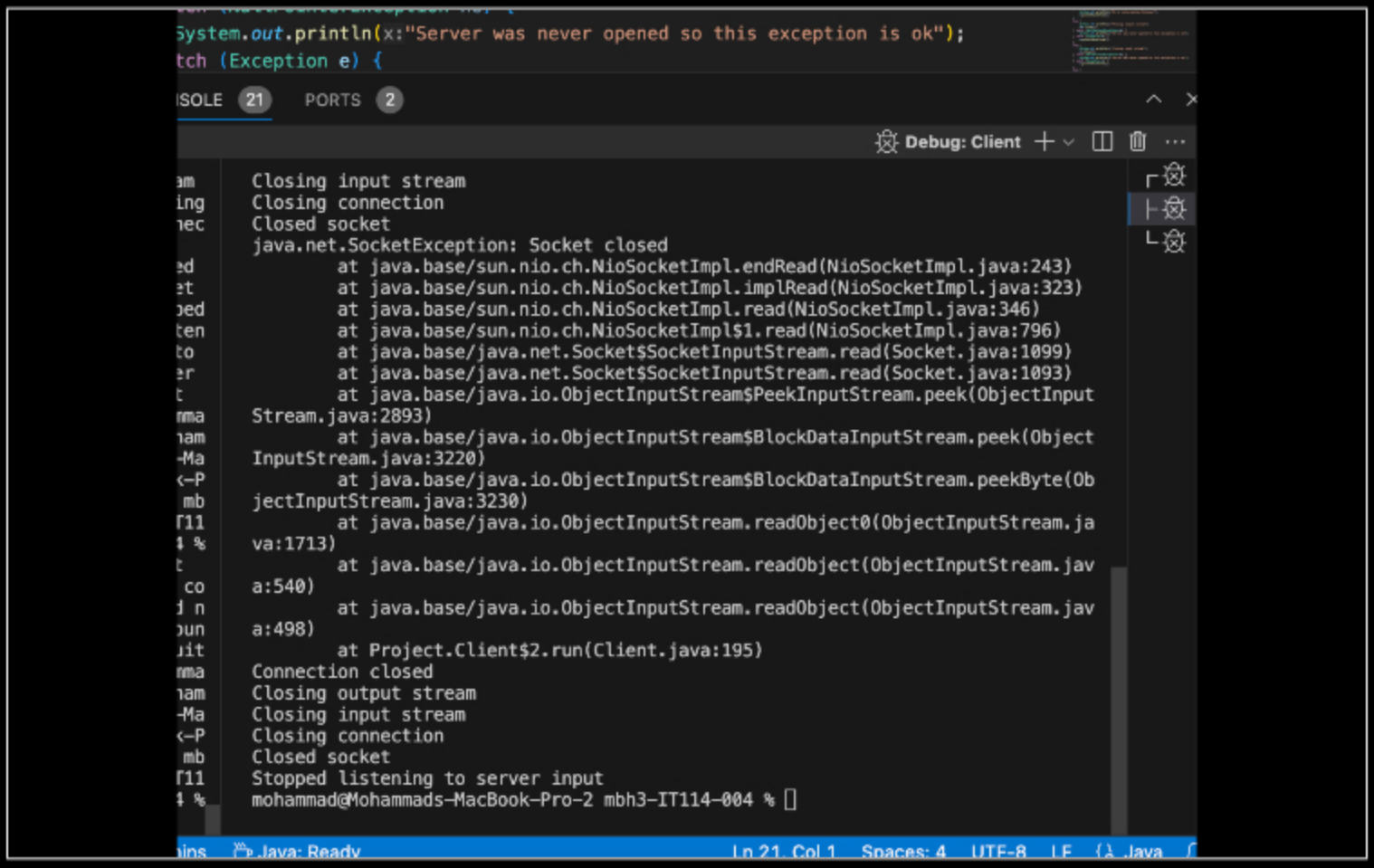
#	Points	Details
#1	1	Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)

#2	1	Show the server terminating; Clients should be disconnected but still running and able to reconnect when the server is back online (demonstrate this)
#3	1	For each scenario, disconnected messages should be shown to the clients (should show a different person disconnected and should show the specific client disconnected)
#4	1	Clearly caption each image regarding what is being shown

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot

Checklist Items (1)

#1 Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)

COLLAPSE

Task #2 - Points: 1

Text: Explain the various Disconnect/termination scenarios

Details:

Include the various scenarios of how a disconnect can occur. There should be around 3 or so.

Checklist *The checkboxes are for your own tracking

Checklist — The checklists are for your own tracking

#	Points	Details
#1	1	Mention how a client gets disconnected from a Socket perspective
#2	1	Mention how/why the client program doesn't crash when the server disconnects/terminates.
#3	1	Mention how the server doesn't crash from the client(s) disconnecting

Response:

Disconnections in a WebSocket connection can occur due to several scenarios. Firstly, a client may disconnect intentionally, such as when a user closes their browser window or navigates away from the page. Secondly, a disconnect could happen due to network issues, like poor connectivity causing a timeout or loss of connection. Lastly, the server itself might go down or restart, cutting off all active connections. When a disconnect happens, the client and server are designed to handle such events gracefully. From the client's perspective, the program doesn't crash because WebSocket clients are built to handle disconnects, often by trying to reconnect or simply ending the session without affecting the stability of the client application. Similarly, on the server side, when a client disconnects, it's a normal part of the server's operation. The server detects the disconnection, cleans up any resources used by that client, and continues running, ready to accept new connections. This resilience is built into the WebSocket protocol and the libraries that implement it, ensuring stability even in the face of disconnects.

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/Mohammadh222/mbh3-IT114-004/compare/main...Milestone1

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Details:

Few related sentences about the Project/sockets topics

Response:

One of the key learnings was handling real-time communication efficiently and robustly, particularly through WebSockets for instant messaging scenarios. Issues like managing connection lifecycles, including handling disconnects gracefully and ensuring the system remains stable and responsive even when network conditions are less than ideal, were critical. Another significant aspect was understanding how to design both the client and server sides to cope with the dynamic nature of connections. For instance, implementing reconnection strategies on the client side and managing resources on the server side when clients disconnect. This assignment highlighted the need for thorough testing across different scenarios to ensure a seamless and resilient user experience.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

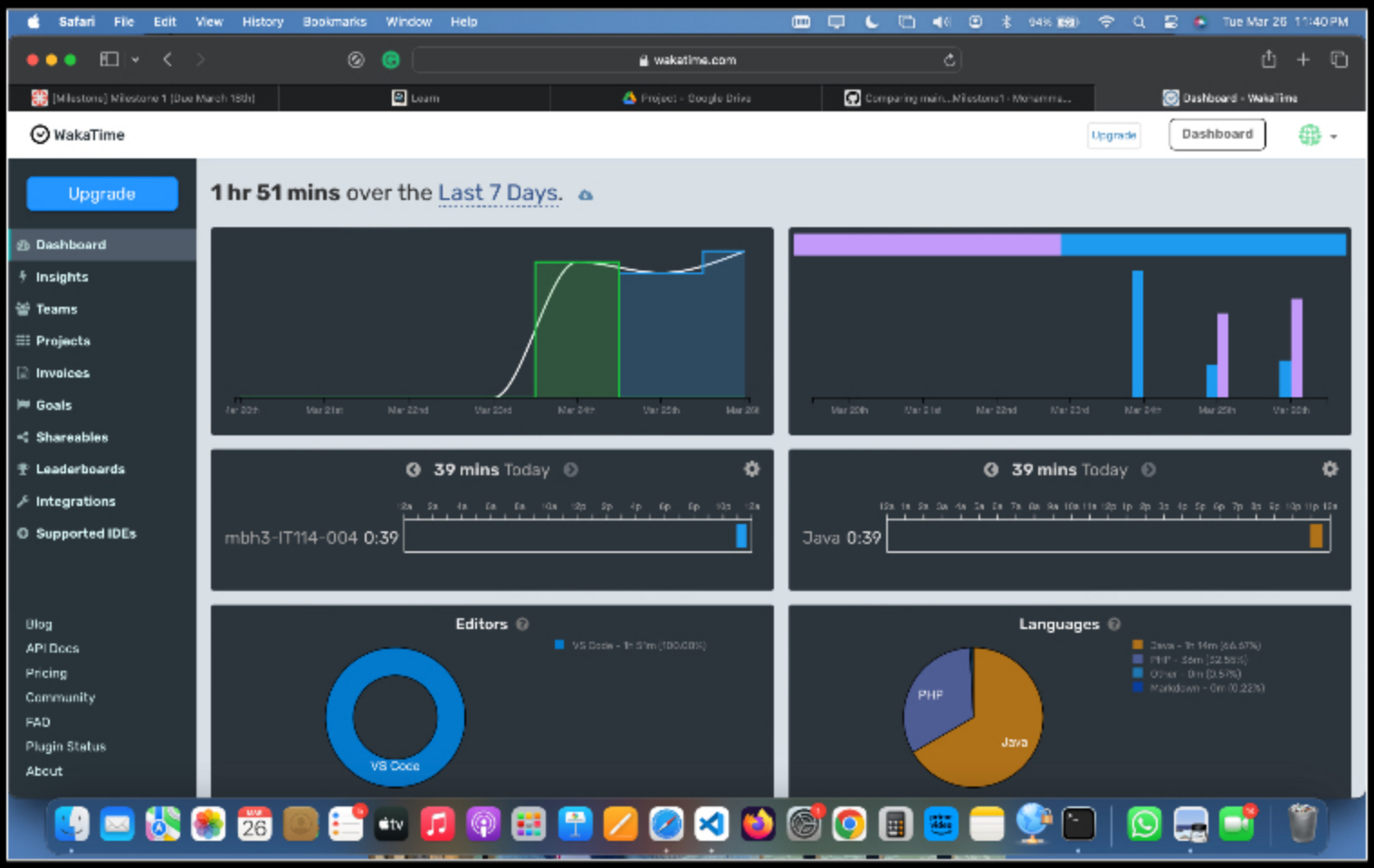
Grab a snippet showing the approximate time involved that clearly shows your repository.

The duration isn't considered for grading, but there should be some time involved.

Task Screenshots:

Gallery Style: Large View

Small Medium Large



waka time screenshot

End of Assignment