Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-project-milestone-1/grade/pd438

IT114-002-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/18/2024 9:42:03 AM

Instructions

COLLAPSE

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

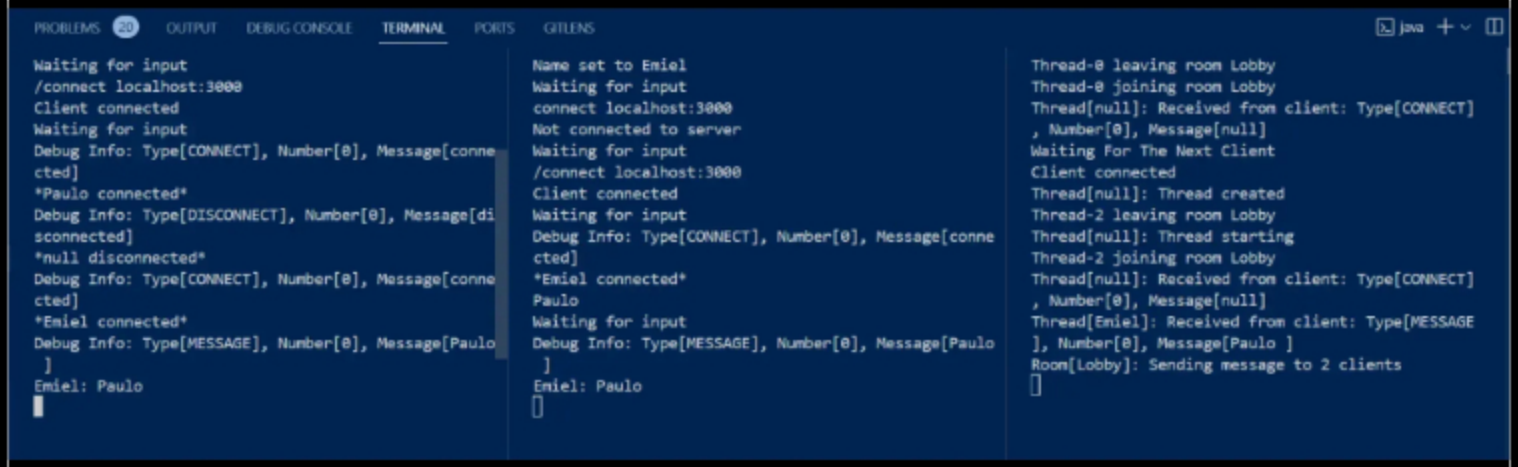
*The checkboxes are for your own tracking

#	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



This displays multiple clients being successfully working. Displays two users emiel and Paulo being able to send and receive messages.

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:

Server connection works by listening in on current port, and creating a socket out of port by setting the socket equal to the port, in which the case of the code is 3000 for the part 5 sockets. On client side of connection when client sends its connection to the port through output sream, will send the name through payload to the server and becomes cooperative with the server.

Communication (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

Checklist

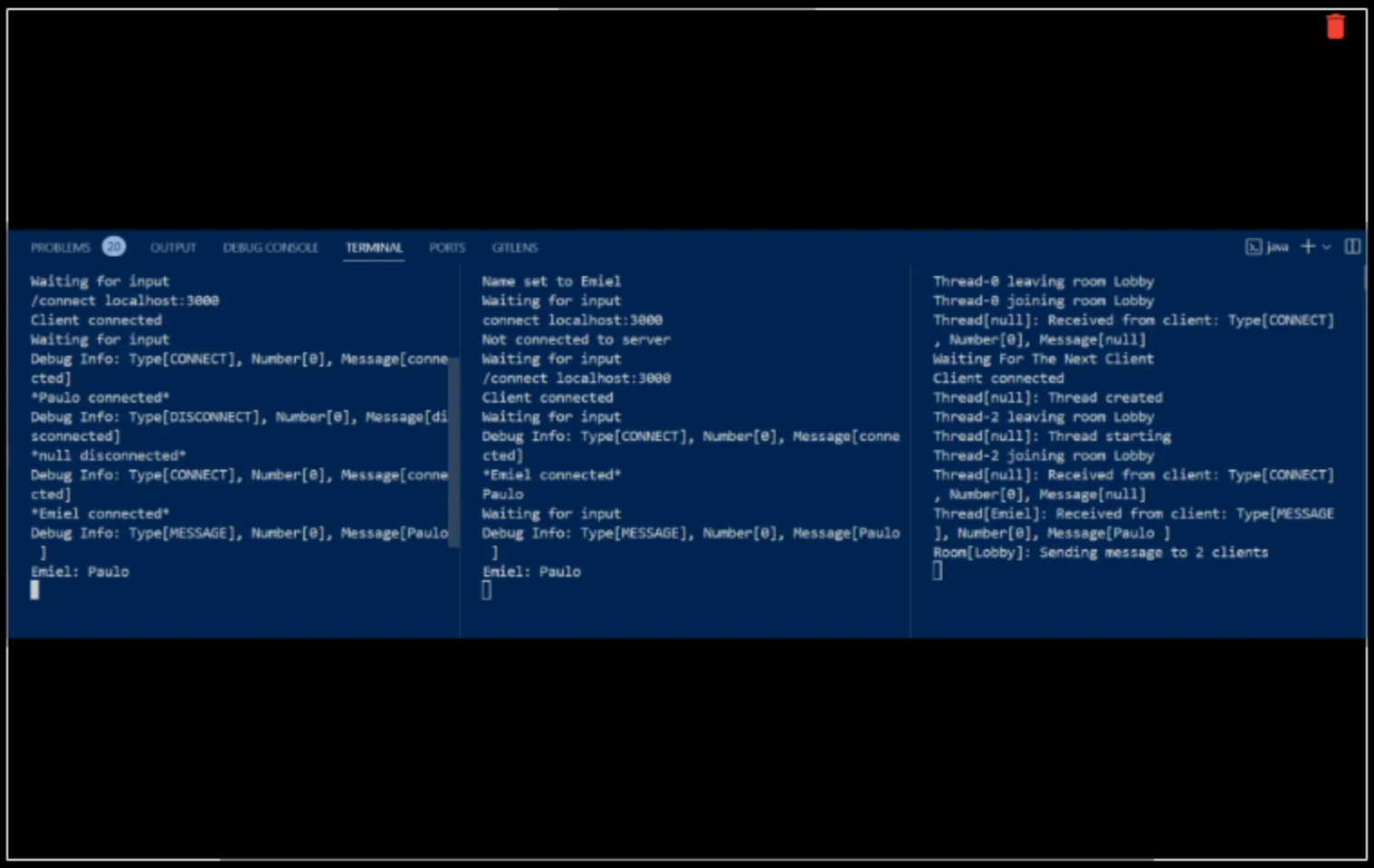
*The checkboxes are for your own tracking

#	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



Images being displayed that user is giving messages to the server and is displayed between the two clients.

Checklist Items (5)

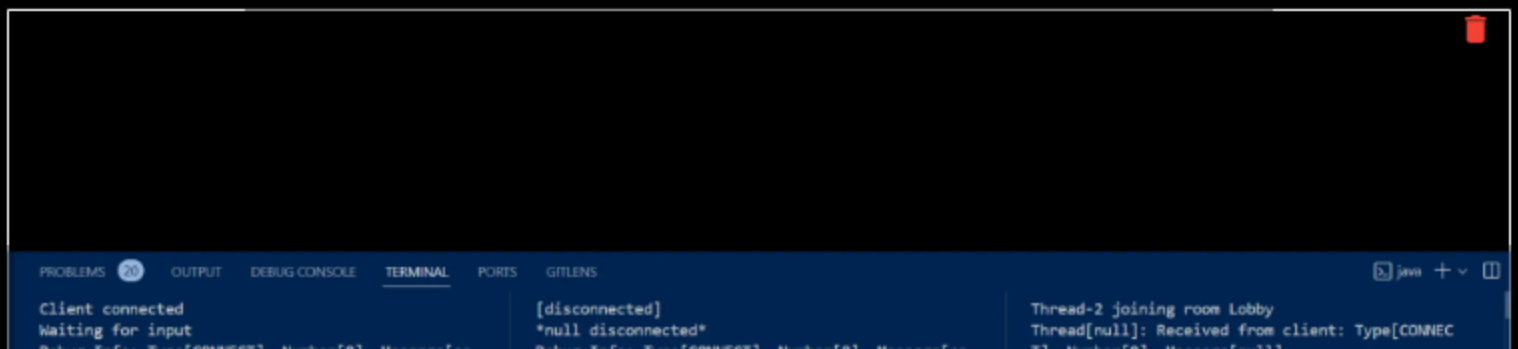
#1 At least two clients connected to the server

#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)

#6 Clearly caption each image regarding what is being shown



```
Debug Info: Type[CONNECT], Number[0], Message[co
nnected]
*Paulo connected*
emiel
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[em
iel]
Paulo: emiel
Debug Info: Type[DISCONNECT], Number[0], Message
[disconnected]
*Emiel disconnected*
Paulo
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Pa
ulo]
Paulo: Paulo
```

```
Debug Info: Type[CONNECT], Number[0], Message[co
nnected]
*Paulo connected*
Debug Info: Type[MESSAGE], Number[0], Message[em
iel]
Paulo: emiel
/createroom 1
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[co
nnected]
*Emiel connected*
Emiel
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Em
iel]
Emiel: Emiel
```

```
1], Number[0], Message[null]
Thread[Paulo]: Received from client: Type[MESSA
GE], Number[0], Message[emiel]
Room[Lobby]: Sending message to 2 clients
Thread[Emiel]: Received from client: Type[MESSA
GE], Number[0], Message[/createroom 1]
Room[Lobby]: Sending message to 2 clients
Created new room: 1
Thread-0 leaving room Lobby
Thread-0 joining room 1
Thread[Emiel]: Received from client: Type[MESSA
GE], Number[0], Message[Emiel]
Room[1]: Sending message to 1 clients
Thread[Paulo]: Received from client: Type[MESSA
GE], Number[0], Message[Paulo]
Room[Lobby]: Sending message to 1 clients
```

This displays room being created and having users message each other and have messages not being displayed

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:

On client sending side, client send message through PAYLoad Variable and see it as a message. On server thread side, strictly handling communication between the clients. On room side, the room gets the name of the client in the room, checking at least 1 client in the room and if there is no one , then closes becuase it be redundant to keep it open if there is less than 1 client. on receiving end, processed the message that was delivered to them through process Message

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

Checklist

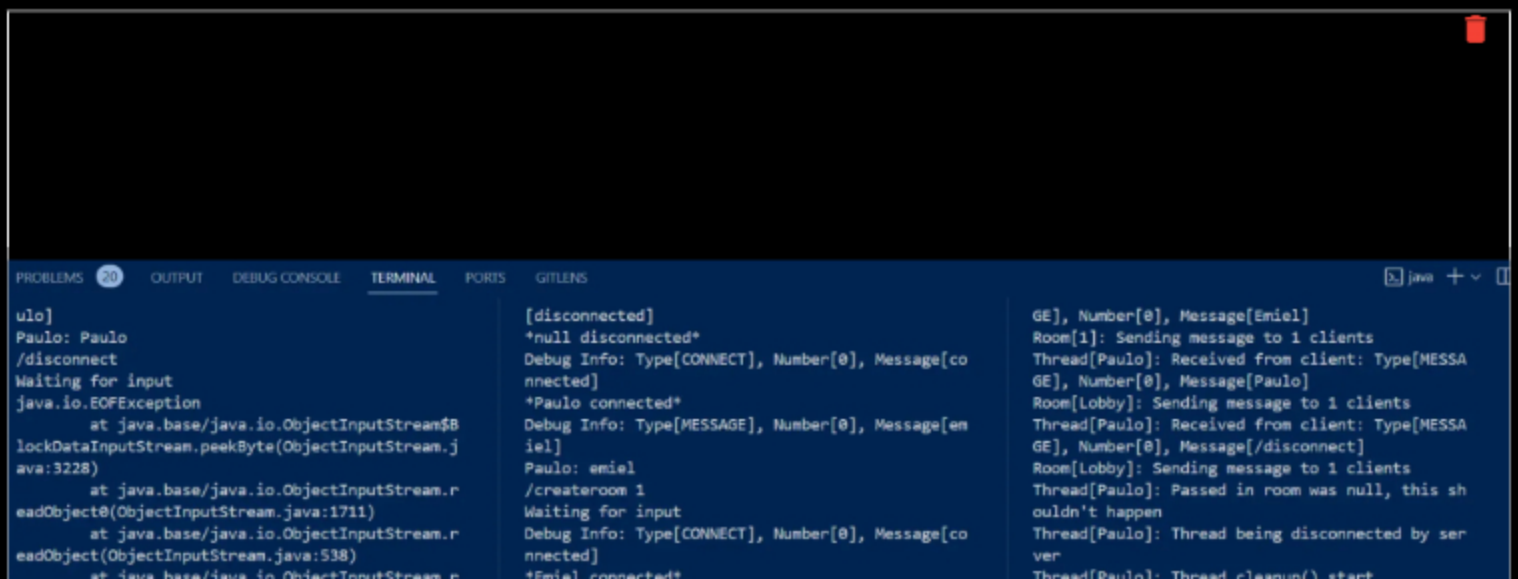
*The checkboxes are for your own tracking

#	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



```
eadObject(ObjectInputStream.java:496)
        at Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
```

```
Emiel
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Em
iel]
Emiel: Emiel
```

```
Thread[Paulo]: Thread cleanup() complete
Thread[Paulo]: Exited thread loop. Cleaning up
connection
Thread[Paulo]: Thread cleanup() start
Thread[Paulo]: Thread cleanup() complete
```

This displays user working perfectly fine when Client Disconnect from server

Checklist Items (2)

#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)

#4 Clearly caption each image regarding what is being shown

PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
Debug Info: Type[MESSAGE], Number[0], Message[Emiel]
Emiel: Emiel
java.net.SocketException: Connection reset
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:318)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:345)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:794)
lockDataInputStream.peek(ObjectInputStream.java:3216)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStr
eam.java:3226)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1711)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:538)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:496)
        at Project.Client$2.run(Client.java:195)Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
```

```
Room[Lobby]: Sending message to 1 clients
Thread[Paulo]: Received from client: Type[MESSAGE], Number[0], M
essage[/disconnect]
Room[Lobby]: Sending message to 1 clients
Thread[Paulo]: Passed in room was null, this shouldn't happen
Thread[Paulo]: Thread being disconnected by server
Thread[Paulo]: Thread cleanup() start
Thread[Paulo]: Thread cleanup() complete
Thread[Paulo]: Exited thread loop. Cleaning up connection
Thread[Paulo]: Thread cleanup() start
Thread[Paulo]: Thread cleanup() complete

emiel@LAPTOP-B9RDLNS1 MINGW64 ~/OneDrive/Pictures/Documents/IT11
4/PD438_IT114_$
```

This displays server being disconnected and

Checklist Items (0)

PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
eam.java:3226)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1711)
```

```
Thread[Paulo]: Exited thread loop. Cleaning up connection
Thread[Paulo]: Thread cleanup() start
```

```
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:538)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:496)
        at Project.Client$2.run(Client.java:195)Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
name
Not connected to server
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Emiel connected*
```

```
Thread[Paulo]: Thread cleanup() complete


emiel@LAPTOP-B9RDLNS1 MINGW64 ~/OneDrive/Pictures/Documents/IT11
$ java Project/Server
Starting Server
Server is listening on port 3000
Waiting For The Next Client
Waiting For The Next Client
Client connected
Thread[null]: Thread created
Thread[null]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[null]: Received from client: Type[CONNECT], Number[0], Me
ssage[null]
```

This displays when it is reconnected.

Checklist Items (3)

#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)

#3 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 Clearly caption each image regarding what is being shown

^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

#	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:

the client gets disconnected with /disconnect, /logoff or /logout. This completely disconnects the user. The client program does not crash when server terminates because they are in different serverthreads. Meaning that they do not depend on each other. Same process goes for the Server.

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/PD438/PD438_IT114_002/pull/11

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:

There were quite a few issues that i had, when copying pasting the code. The code did not want to cooperate with me and caused me great amounts of stress during the break.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

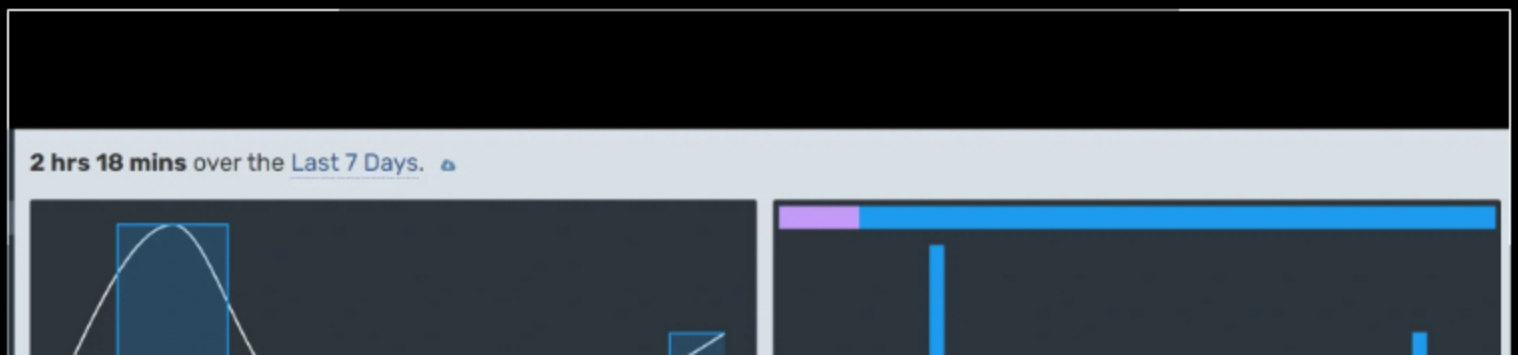
Grab a snippet showing the approximate time involved that clearly shows your repository.

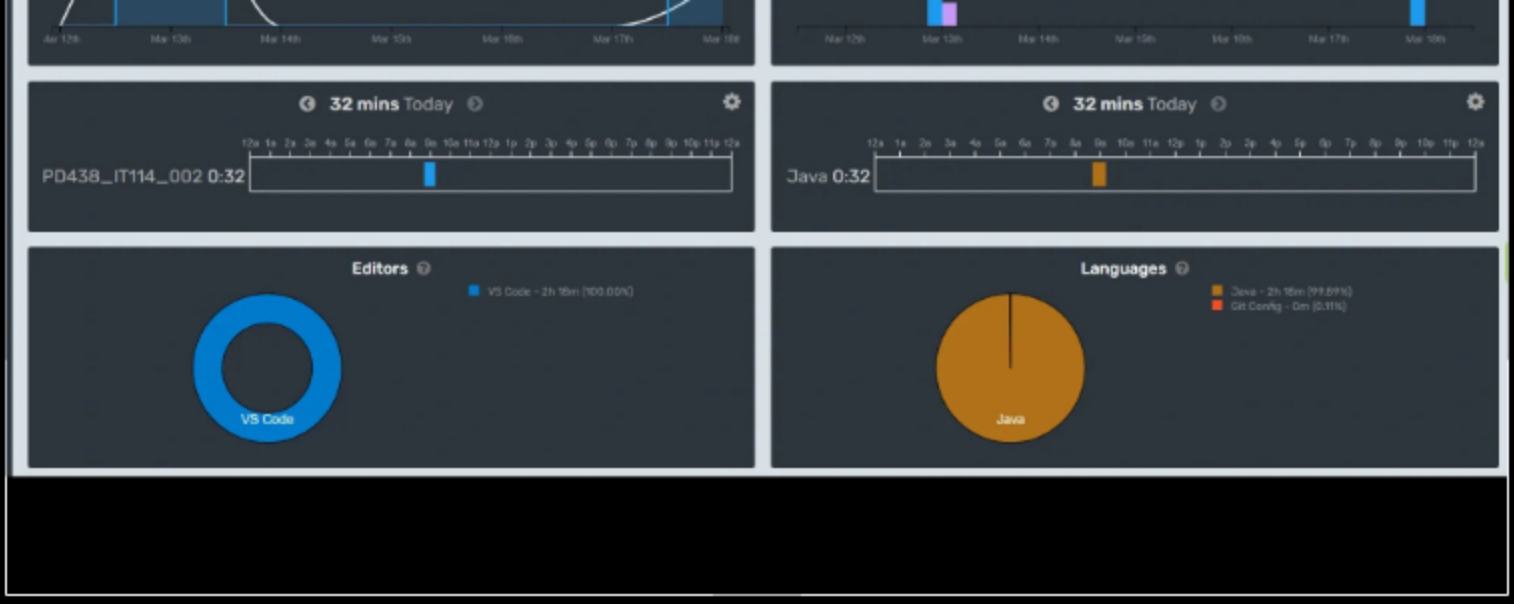
The duration isn't considered for grading, but there should be some time involved.

Task Screenshots:

Gallery Style: Large View

Small Medium Large





Wakatime being displayed

End of Assignment