Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/owe

IT114-006-S2024 - [IT114] Milestone 2 Chatroom 2024

Submissions:

Submission Selection

1 Submission [active] 4/28/2024 5:44:26 PM

Instructions

COLLAPSE

Implement the Milestone 2 features from the project's proposal document:
https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145XizQG4/view
Make sure you add your ucid/date as code comments where code changes are done
All code changes should reach the Milestone2 branch
Create a pull request from Milestone2 to main and keep it open until you get the output PDF from this assignment.
Gather the evidence of feature completion based on the below tasks.
Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
Run the necessary git add, commit, and push steps to move it to GitHub
Complete the pull request that was opened earlier
Upload the same output PDF to Canvas

Branch name: Milestone2

Tasks: 12 Points: 10.00

Demonstrate Usage of Payloads (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of your Payload class and subclasses and PayloadType

Checklist

*The checkboxes are for your own tracking

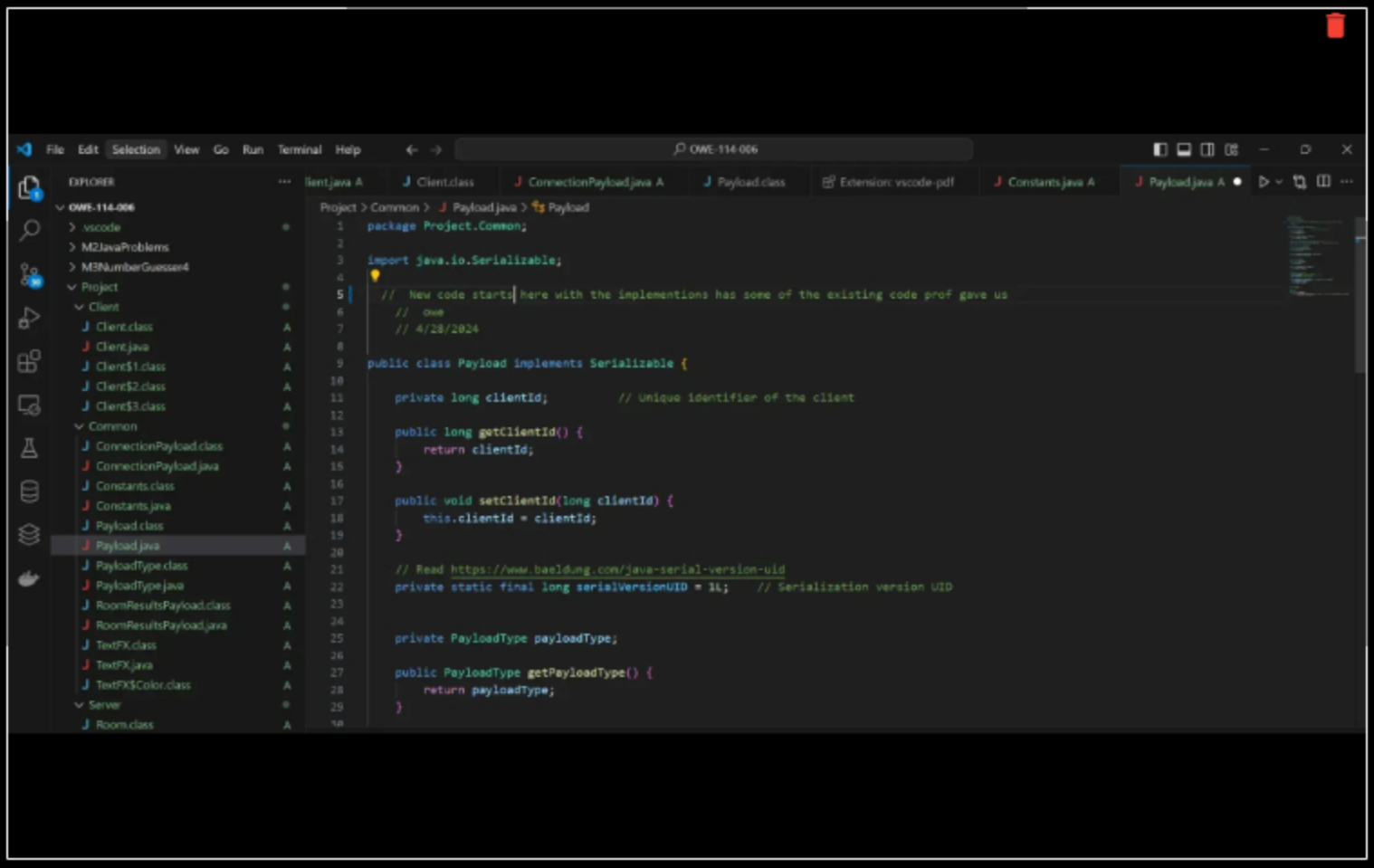
#	Points	Details

#	Points	Details
#1	1	Payload, equivalent of RollPayload, and any others
#2	1	Screenshots should include ucid and date comment
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

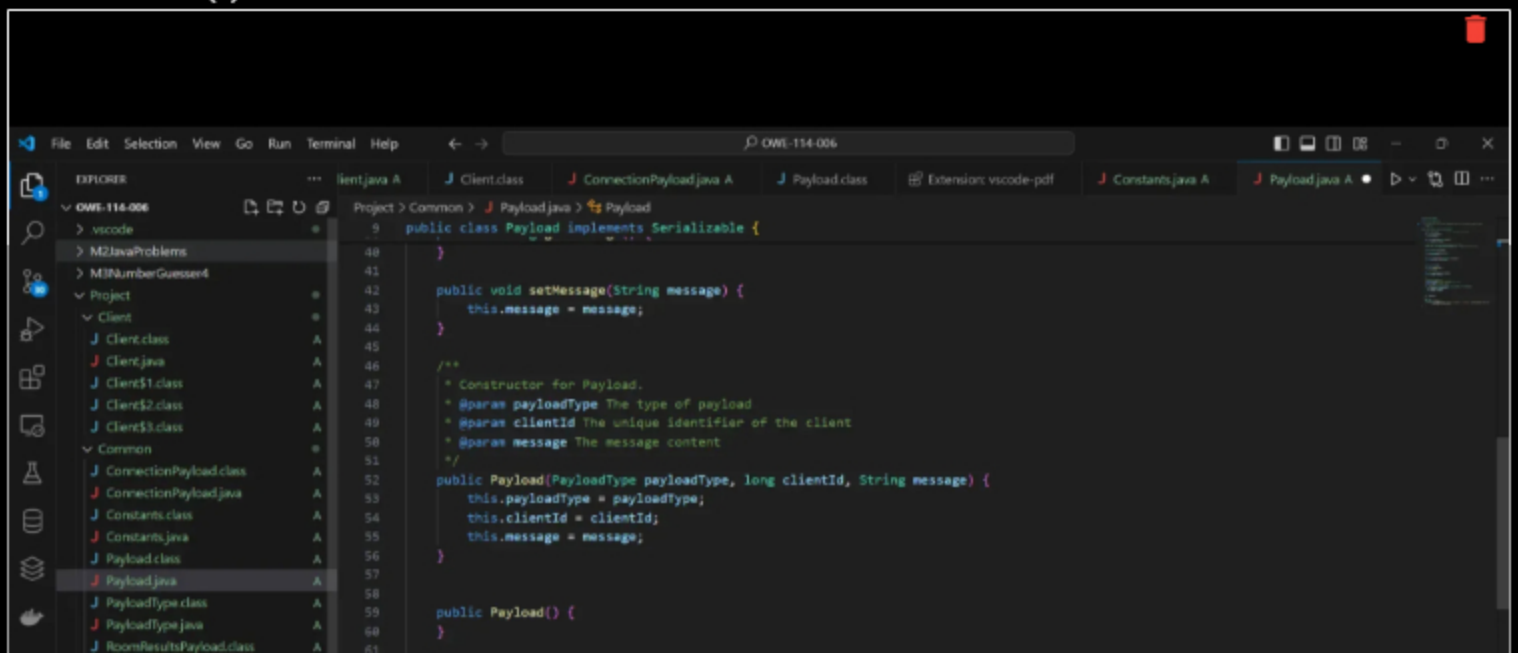
Gallery Style: Large View

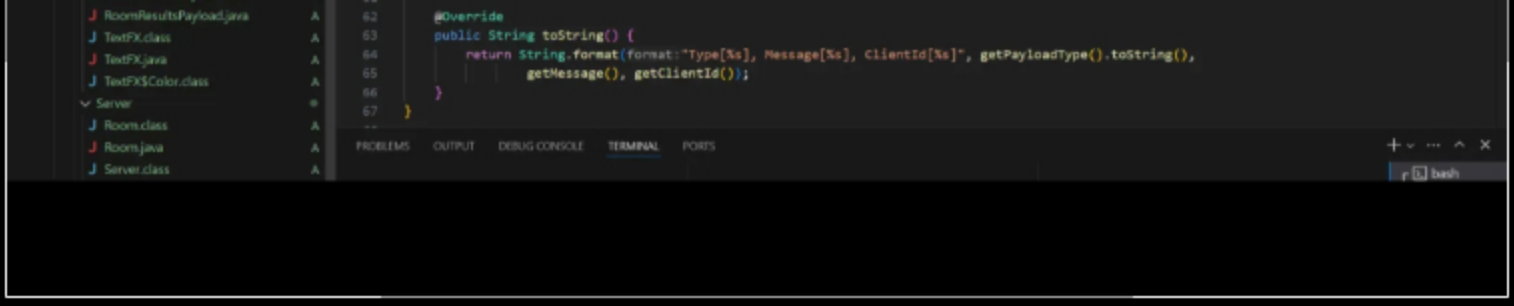
Small Medium Large



payload class with implications

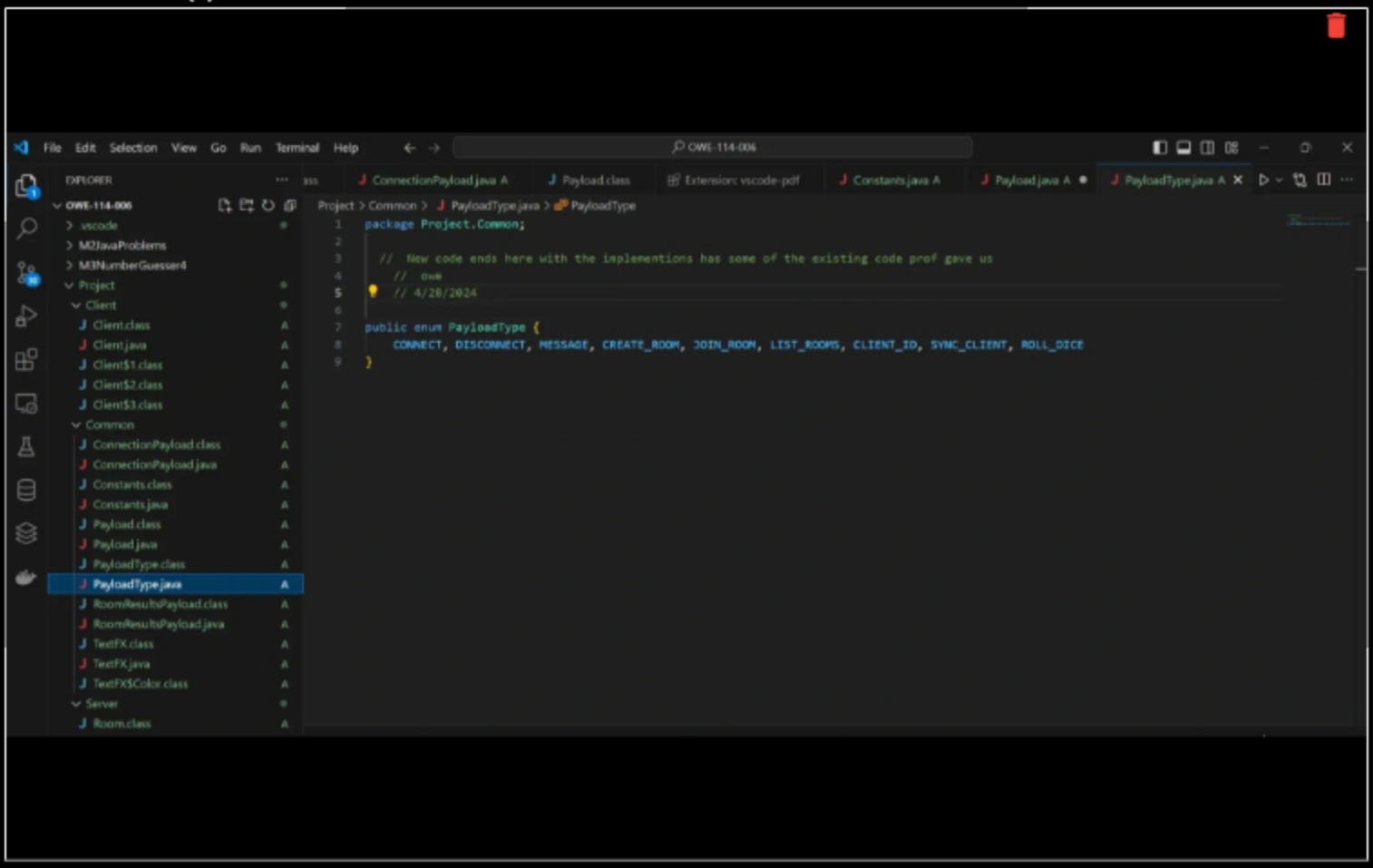
Checklist Items (0)





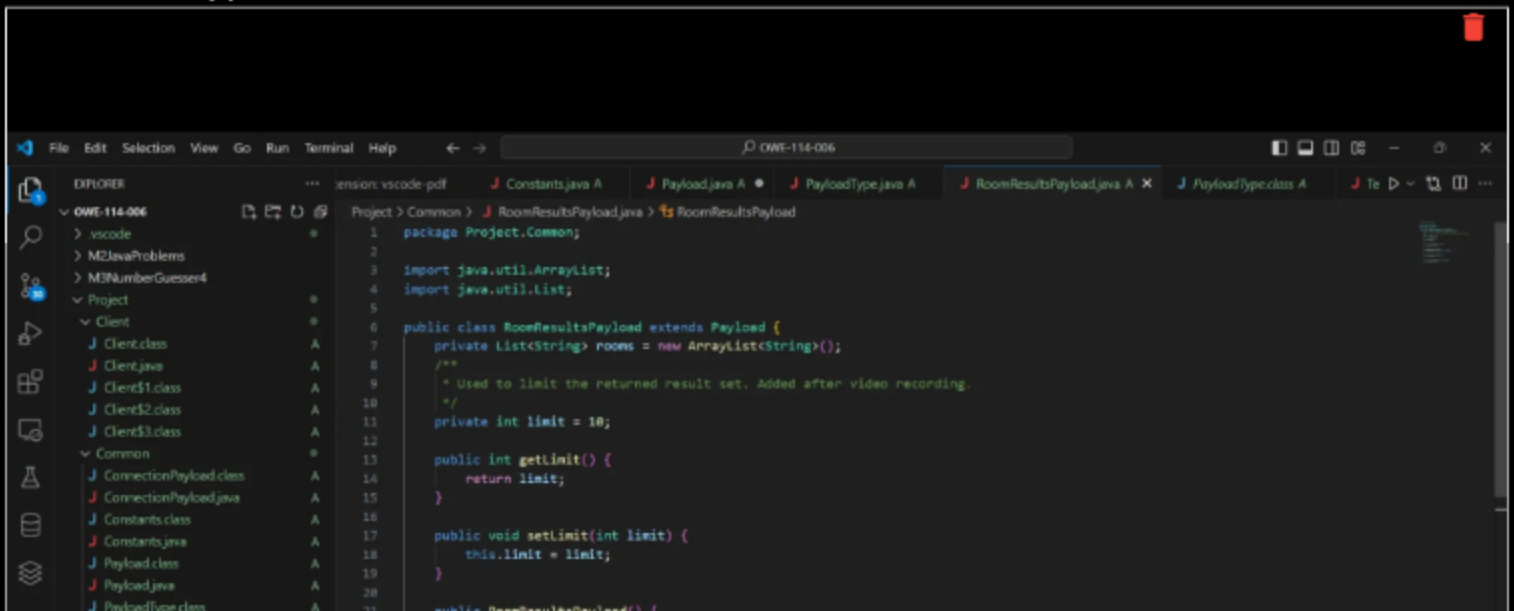
payload continued

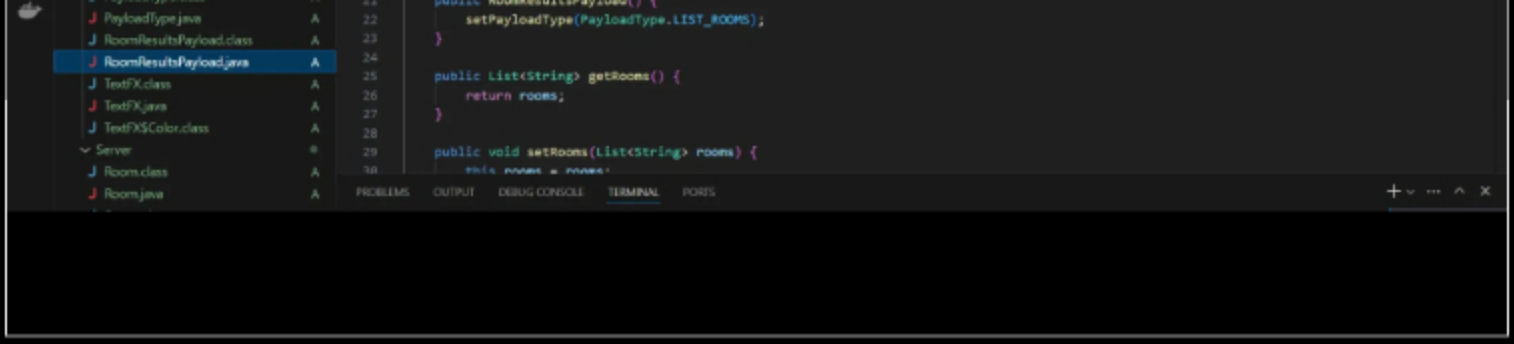
Checklist Items (0)



payload type class

Checklist Items (0)





subclasses for payload

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the payloads being debugged/output to the terminal

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Demonstrate flip
☐ #2	1	Demonstrate roll (both versions)
☐ #3	1	Demonstrate formatted message along with any others
☐ #4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Missing Caption

COLLAPSE

Task #3 - Points: 1

Text: Explain the purpose of payloads and how your flip/roll payloads were made

Response:

In our system, payloads are structured data packages that help clients and servers communicate with each other. They contain a variety of commands and messages, including CREATE_ROOM, JOIN_ROOM, MESSAGE, CONNECT, DISCONNECT, and LIST_ROOMS. We have added a new payload type called ROLL, which includes the data needed to simulate dice rolls. The system's functionality is improved by this new feature, which lets users participate in interactive games like dice.

Demonstrate Roll Command (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#
#2	1	ServerThread code receiving the payload and passing it to the Room
#3	1	Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients
#4	1	Code screenshots should include ucid and date comment
#5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

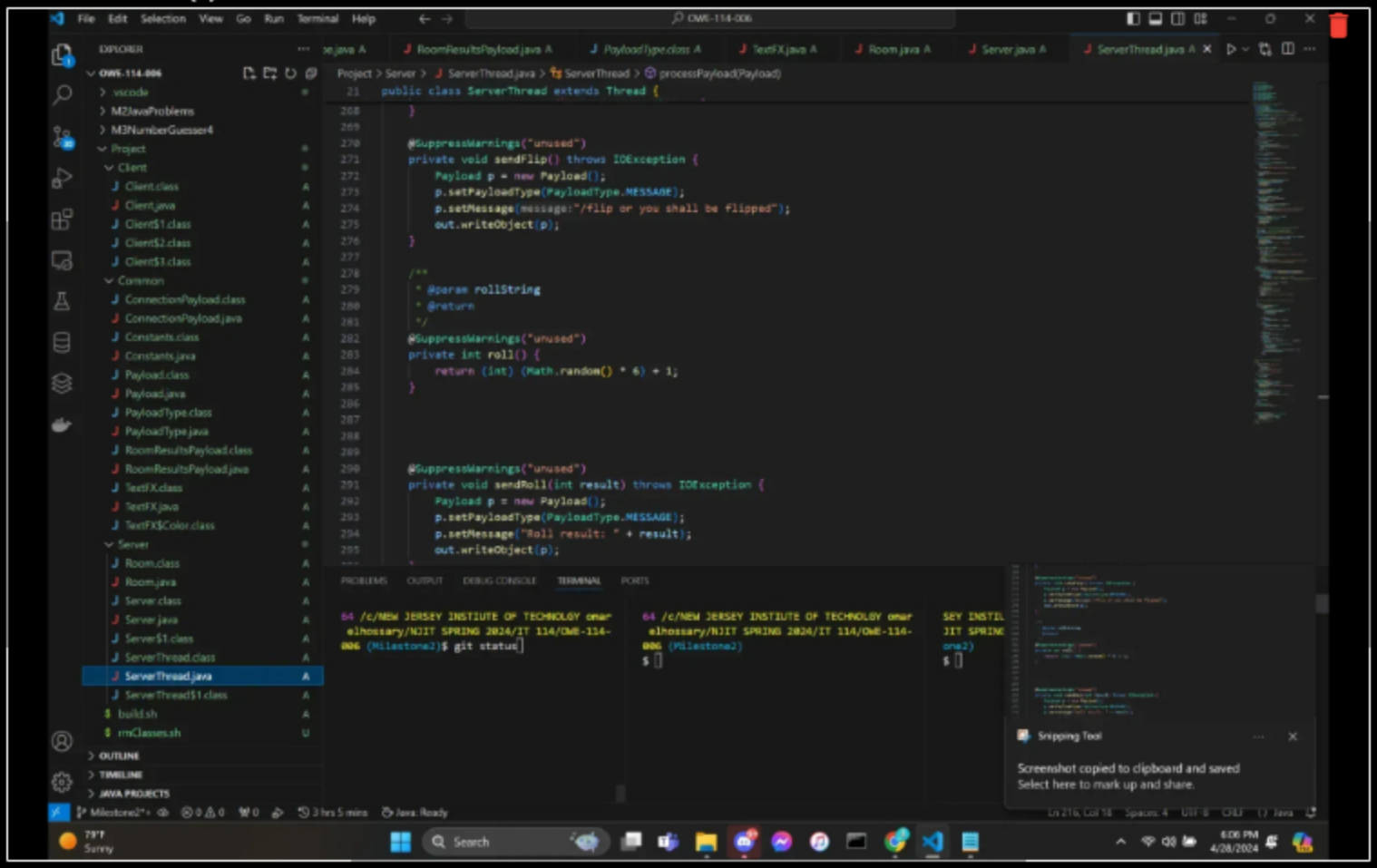
Small Medium Large

```
            }
            return true;
        } else if (text.equalsIgnoreCase(FLIP_COMMAND)) {
            try {
                sendFlip();
```

```java
            } catch (IOException e) {
                e.printStackTrace();
            }
            return true;
        } else if (text.startsWith(ROLL_COMMAND)) {
            String rollString = text.replace(target:"/roll", replacement:"").trim();
            try {
                int result = roll(rollString);
                sendRoll(result);
            } catch (IOException e) {
                sendMessage(message:"Wrong Format. Use : '/roll 123' or '/roll 456'.");
            }
            return true;
        }
        return false;
    }
```

the client code for the dice for roll

Checklist Items (0)



more server thread code for roll

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the two different roll formats are handled and how the message flows from the client, to the Room, and shared with all other users

Response:

In our system, the Room class receives a roll message sent from the client, which is then processed to extract the roll details. All other users in the room receive this roll message after it is broadcast by the Room class. Real-time communication is made possible during gameplay as each user receives the message and can see the outcome of the roll. This logic improves the user experience by keeping all users informed and synchronized about the current dice rolls.

Demonstrate Flip Command (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

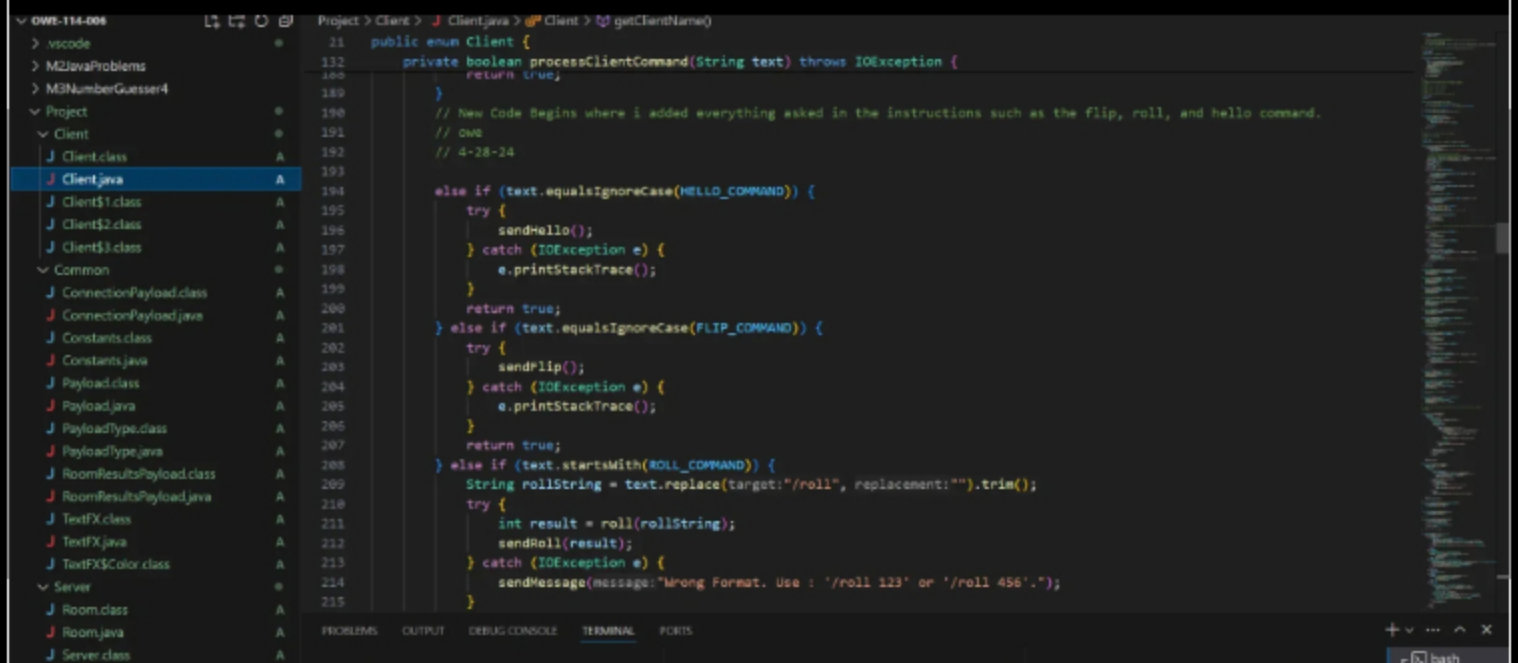
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Client code that captures the command and converts it to a payload
#2	1	ServerThread receiving the payload and passing it to the Room
#3	1	Room handling the flip action correctly
#4	1	Code screenshots should include ucid and date comment
#5	1	Each screenshot should be clearly captioned

Task Screenshots:

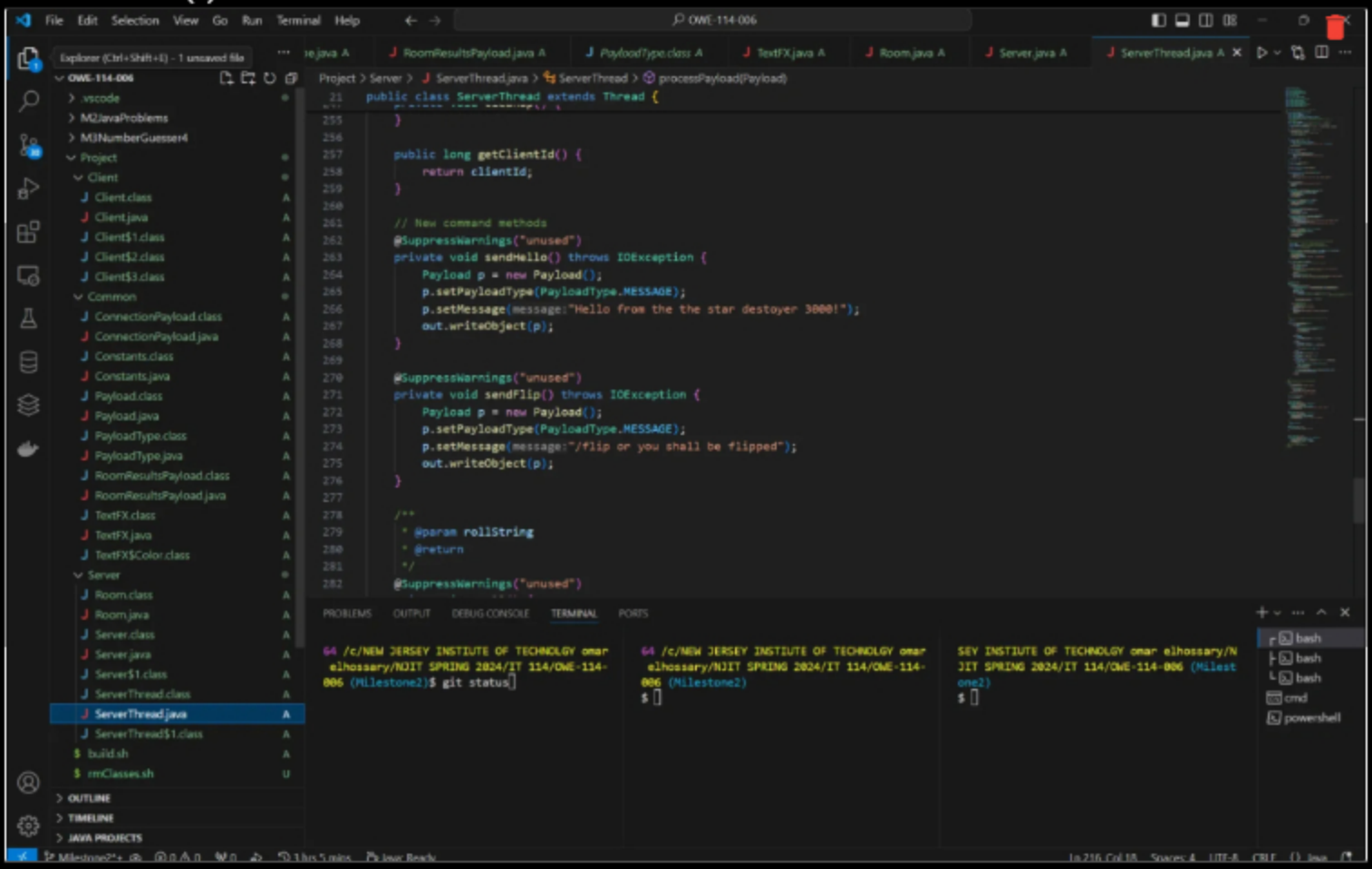
Gallery Style: Large View

Small Medium Large



client code for flip command

Checklist Items (0)



serverthread code for flip

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the flip command is handled and processed and how the message flows from the client, to the Room, and shared with all other users

Response:

In order to handle the flip command, it is sent from the client to the Room class, where it is processed to ascertain the flip's outcome. The message is then broadcast to all other users in the room with this outcome. Real-time communication and coordinated gameplay are made possible by the message that each user receives and the ability to view the flip's result. This logic makes sure that everyone is aware of the outcome of the flip, which makes it easier for people to interact and communicate in the shared virtual environment.

Demonstrate Formatted Messages (4 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of Room how the following formatting is processed from a message

Details:

Note: this processing is server-side

Slash commands are not valid solutions for this and will receive 0 credit

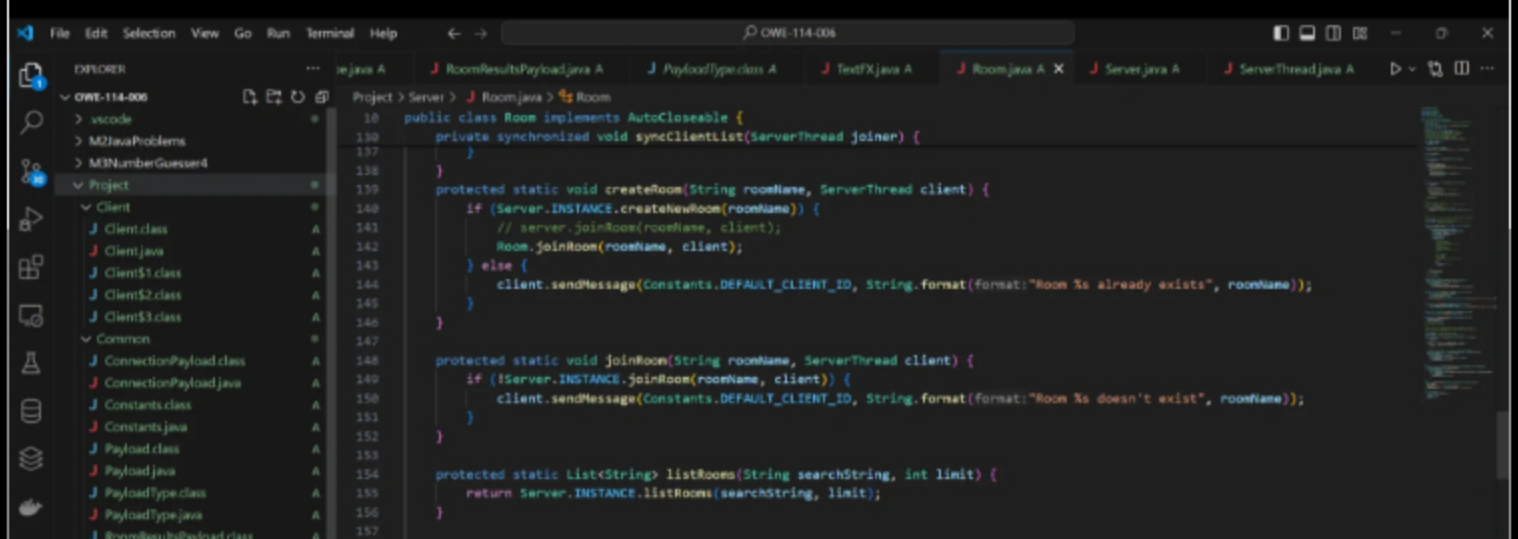
Checklist *The checkboxes are for your own tracking

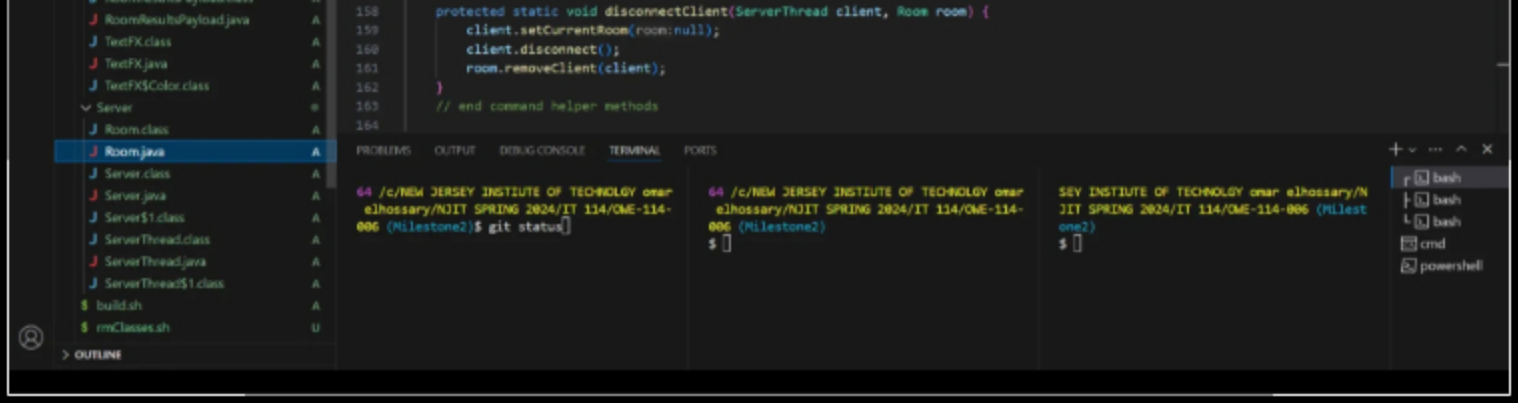
#	Points	Details
☐ #1	1	Room code processing for bold
☐ #2	1	Room code processing for italic
☐ #3	1	Room code processing for underline
☐ #4	1	Room code processing for color (at least R, G, B or support for hex codes)
☐ #5	1	Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal
☐ #6	1	Must not rely on the user typing html characters, but the output can be html characters
☐ #7	1	Code screenshots should include ucid and date comment
☐ #8	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large





code for room was not finished

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the following

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Which special characters translate to the desired effect
☐ #2	1	How the logic works that converts the message to its final format

Response:

i did not get to do this

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for the branch

Details:
Note: the link should end with /pull/#

URL #1
https://github.com/WayguBeef5/OWE-114-006/pull/9

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Response:

Missing Response

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

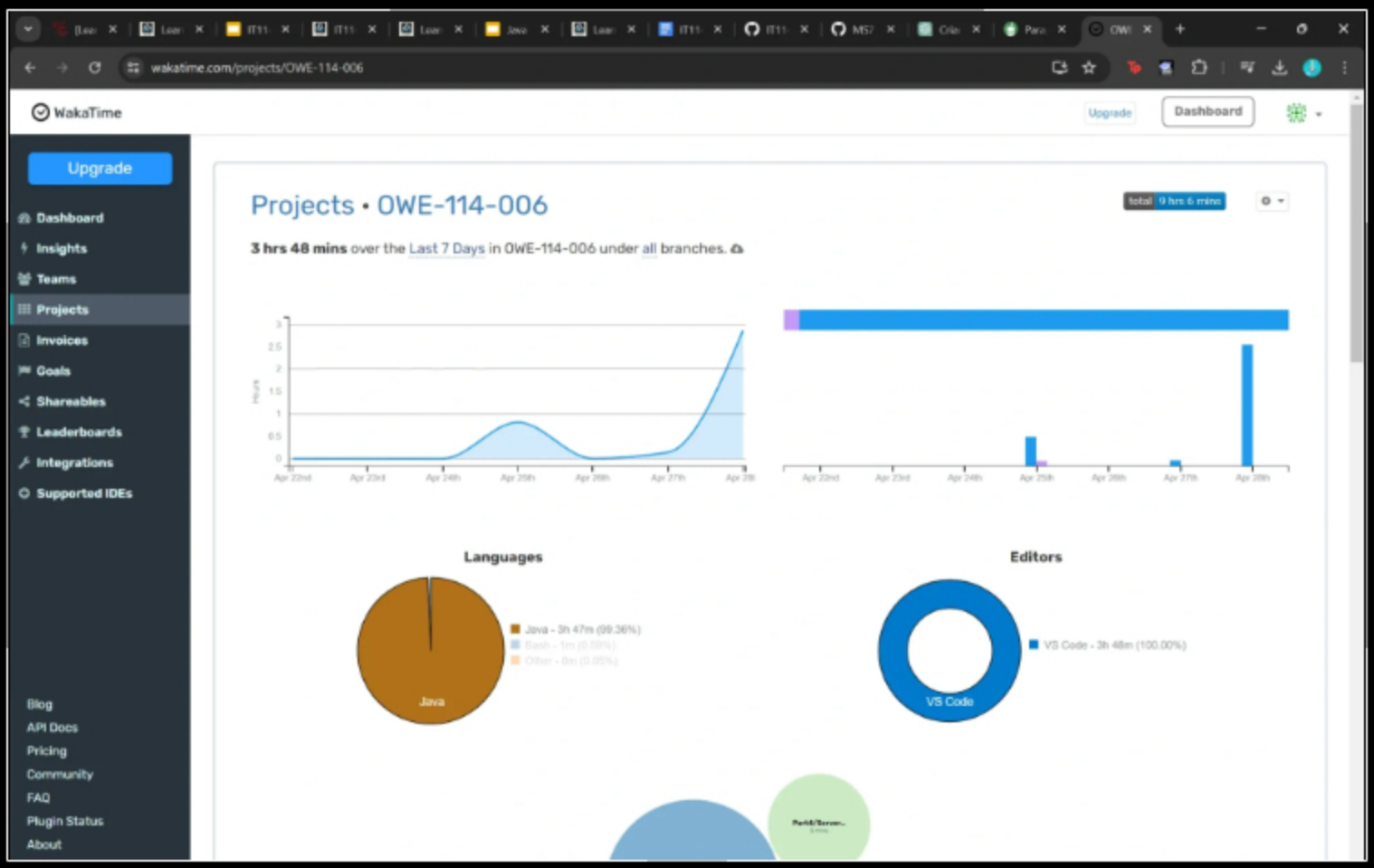
Details:

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



screen shot for the waka time

End of Assignment