

# Submission Worksheet

**CLICK TO GRADE**

<https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-4-trivia-2024-m24/grade/ma2633>

Course: IT114-003-F2024

Assignment: [IT114] Milestone 4 Trivia 2024 M24

Student: Mohamad A. (ma2633)

## Submissions:

Submission Selection

1 Submission [submitted] 12/11/2024 11:48:53 PM

## Instructions

[^ COLLAPSE ^](#)

- Implement the Milestone 4 features from the project's proposal document:  
<https://docs.google.com/document/d/1h2aEWUoZ-etpz1CRI-StaWbZTjkd9BDMq0b6TXK4utl/view>
- Make sure you add your ucid/date as code comments where code changes are done
- All code changes should reach the Milestone4 branch
- Create a pull request from Milestone4 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: Milestone4

Group

Group: Away

Tasks: 1

Points: 2.5

100%

[^ COLLAPSE ^](#)

### Task

Group: Away



Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Weight: ~100%

Points: ~2.50

[▲ COLLAPSE ▲](#)

#### ⓘ Details:

Screenshots of editors must have the frame title visible with your ucid and the client name. Code screenshots must have ucid/data comments.

Columns: 1

#### Sub-Task

Group: Away



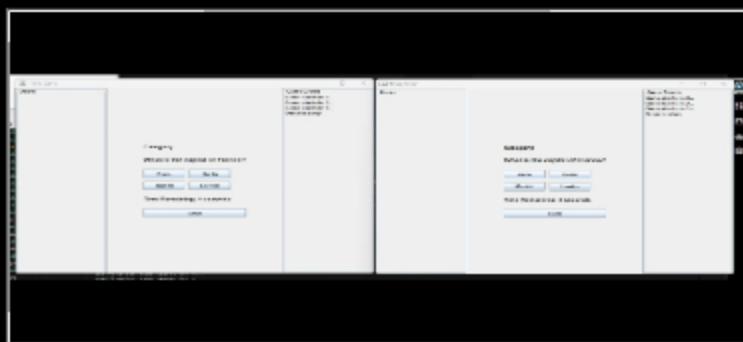
Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #1: Show the UI that allows the Client to toggle their away status

## ☒ Task Screenshots

Gallery Style: 2 Columns

4      2      1



UI that allows the Client to toggle their away status

#### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

#### Sub-Task

Group: Away



Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #2: Show a few examples of the Game Events Panel showing away/not away status in a clear message

## ☒ Task Screenshots

Gallery Style: 2 Columns

4      2      1





Game Events Panel showing away/not away status in a clear message

Game Events Panel showing away/not away status in a clear message



Game Events Panel showing away/not away status in a clear message

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

#### Sub-Task

100%

Group: Away

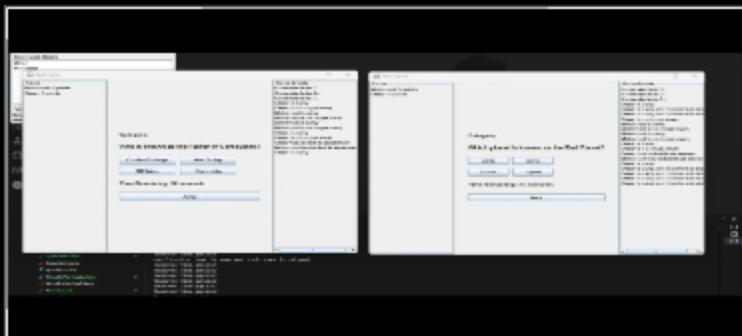
Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #3: Show a few examples of the User List panel showing a Client away (demonstrate that this changes when the status is toggled)

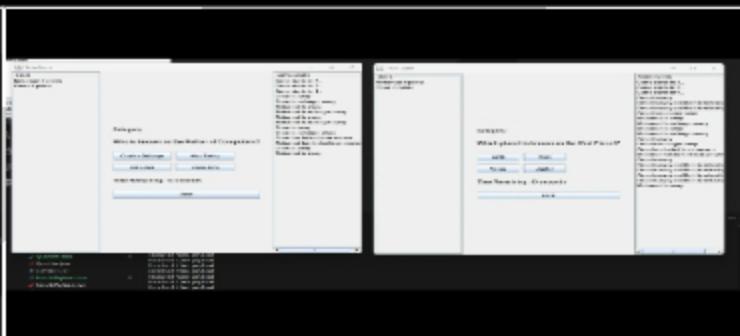
## Task Screenshots

Gallery Style: 2 Columns

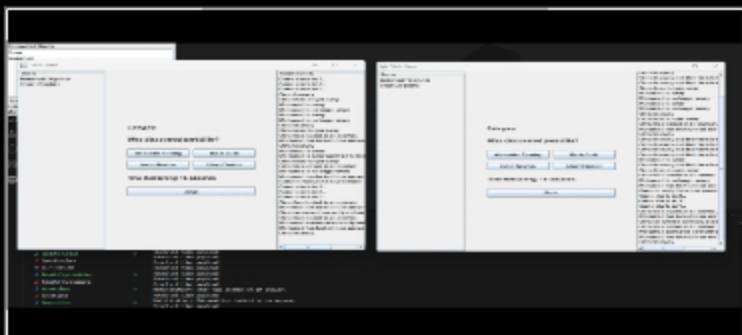
4      2      1



User List panel showing a Client away



User List panel showing a Client away



## User List panel showing a Client away

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

#### Sub-Task

Group: Away

100%

Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #4: Show the code flow that handles the away toggle (from Client interaction to updating server-side state)

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
// Method to send away status
public void sendAwayStatus(boolean isAway) {
    Payload payload = new Payload(isAway, String.valueOf(isAway), PayloadType.AWAY_STATUS);
    sendPayload(payload);
}
```

method to send the "away" status to the server

```
private void markClientAway(ClientData client, boolean isAway) {
    client.setAway(isAway);
    String message = client.getUsername() + " is " + (isAway ? "away" : "no longer away");
    Payload awayStatusPayload = new Payload(client.getService(), message, PayloadType.AWAY_STATUS);

    // Send notification to all clients
    for (ClientData c : clients) {
        ClientServerThread cThread = c.getClientServerThread();
        cThread.sendPayload(awayStatusPayload);
    }
}
```

Method to mark a client as "away"

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

**Client Action:** A player clicks a button or sends a command to toggle their "away" status. **Notify Server:** The client sends a message (payload) to the server with the updated "away" status. **Update State:** The server updates the player's status (isAway) in its records. **Create Notification:** The server generates a message like "Player X is away" or "Player X is back." **Broadcast to All:** The server sends this notification to all connected players to keep them updated. **Skip Turns:** When the game logic runs, players marked "away" are skipped during their turns.

#### Sub-Task

Group: Away

100%

Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #5: Show the code that handles the Game Events Panel message (from the server-side state changing to sending the payload)

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
// Button to mark away from the game
Button awayButton = new JButton("Away");
awayButton.addActionListener(new ActionListener() {
```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (awayButton.getText().equals("Away")) {
        awayButton.setText("Back");
        client.setAwayStatus(true);
    } else {
        awayButton.setText("Away");
        client.setAwayStatus(false);
    }
}
// mohamed Almagid 10/13/24

```

code that handles the Game Events Panel message

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

i don't understand the requirements for this task but i assumed is the what you need the ui handling

### Sub-Task

Group: Away



Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #6: Show the code that handles updating the User List Panel (from Client receiving -> the UI change)

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



User List Panel

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

The userListPanel is a JPanel that's used to show all the users in the game. It stacks the users vertically using a BoxLayout set to Y\_AXIS. A border is added around the panel with the title "Users" to make it look nice and clear. The panel is also wrapped in a JScrollPane, so if there are too many users, you can scroll to see them all. The scroll pane's size is set to 200x400 pixels to fit properly in the game's layout.

### Sub-Task

Group: Away

Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the

100%

game  
Sub Task #7: Show the project logic that skips/ignores away players (away players can't take a turn) and include a UI screenshot of an applicable message if someone tries the action while away

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```

// Method to broadcast the current question to all clients
private void broadcastQuestionToClients(Question question) {
    Payload payload = new Payload(clientId:"Server", message:"New Question", payloadType:NOTIFICATION, question:question);
    for (Client client : clients) {
        if (!client.isAway()) {
            String message = client.getUsername() + " is away and their turn is skipped.";
            Payload skipTurnPayload = new Payload(clientId:"Server", message, PayloadType.NOTIFICATION);
            client.getTurnHandler().sendPayload(skipTurnPayload);
        } else {
            client.getTurnHandler().sendPayload(payload);
        }
    }
    System.out.println("Broadcasting question to clients: " + question.getQuestionText());
}

```

project logic that skips/ignores away players



i don't have a UI message but the buttons are displayed and cant take a value until the player is back

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

i already have the Away button and also make the player Away but it has some issues also you can say it is half done

End of Task 1

End of Group: Away

Task Status: 1/1

Group



Group: Spectator

Tasks: 1

Points: 2.5

▲ COLLAPSE ▲

Task



Group: Spectator

Task #1: Spectator: Client can join as spectator

Weight: ~100%

Points: ~2.50

▲ COLLAPSE ▲

**Details:**

Screenshots of editors must have the frame title visible with your ucid and the client name.

Code screenshots must have ucid/data comments.

Spectator control/access logic must be handled in the GameRoom.

They can see all chat but are ignored from turns and can't send messages



Columns: 1

**Sub-Task**

0%

Group: Spectator

Task #1: Spectator: Client can join as spectator

Sub Task #1: Show the spectator UI and demonstrate how they're blocked/ignored from game actions (turns and messages)

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



Missing Caption

**Caption(s) (required)**

Caption Hint: *Describe/highlight what's being shown*

Missing caption(s)

**Sub-Task**

0%

Group: Spectator

Task #1: Spectator: Client can join as spectator

Sub Task #2: Show the User List Panel representation of Spectators (must differ from active Players)

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



Missing Caption

**Caption(s) (required)**

**Caption Hint:** *Describe/highlight what's being shown*

## Missing caption(s)

### Sub-Task

### Group: Spectator

## Task #1: Spectator: Client can join as spectator

Sub Task #3: Show the code logic that blocks/ignores spectators from game actions (turns and messages)

## ■ Task Screenshots

## Gallery Style: 2 Columns

4 2 1

```
// Method to process a player's answer
public void processAnswer(@NonNull String client, String answer) {
    if (client.isSpectator()) {
        System.out.println(client.getUsername() + " is a spectator and cannot answer.");
        return;
    }
    // Threaded answer
    if (isThreaded)
        return;
}
```

**ode logic that blocks/ignores spectators from game actions (turns and messages)**

**Caption(s) (required) ✓**

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

#### **Response:**

The code checks if the client is a spectator by calling `client.isSpectator()`. If they are a spectator, it prints a message saying they can't answer. Then it just stops the method with `return`, so nothing else happens for that client.

### Sub-Task

### Group: Spectator

Task #1: Spectator: Client can join as spectator

Sub Task #4: Show the code flow of the server-side sending the spectator status to Clients and having the User List Panel updated accordingly

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

Show the code flow of the source code according the

Show the code now or the server-side sending the spectator status to Clients and having the User List Panel updated accordingly

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

this code will make the user join as Spectator if the room exists and make the user Spectate the game

End of Task 1

End of Group: Spectator

Task Status: 0/1

Group

Group: Project Specific

66%  
Tasks: 3  
Points: 4

▲ COLLAPSE ▲

Task

Group: Project Specific  
Task #1: Spectator Extra  
Weight: ~33%  
Points: ~1.33

▲ COLLAPSE ▲

**i** Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.  
Code screenshots must have ucid/data comments.



Columns: 1

Sub-Task

Group: Project Specific  
Task #1: Spectator Extra  
Sub Task #1: Show that a spectator can see the correct answer

0%

## Task Screenshots

Gallery Style: 2 Columns



Missing Caption

**Caption(s) (required)**

Caption Hint: *Describe/highlight what's being shown*

Missing caption(s)

Sub-Task

Group: Project Specific



Task #1: Spectator Extra

Sub Task #2: Show the code related to sending/showing the correct answer to spectators

## Task Screenshots

Gallery Style: 2 Columns

4 2 1



Missing Caption

**Caption(s) (required)**

Caption Hint: *Describe/highlight what's being shown*

Missing caption(s)

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Missing Response

End of Task 1

Task



Group: Project Specific

Task #2: Add option to choose which categories will be used for the session

Weight: ~33%

Points: ~1.33

**i Details:**

Screenshots of editors must have the frame title visible with your ucid and the client name.  
Code screenshots must have ucid/data comments.



Columns: 1

**Sub-Task**

100%

Group: Project Specific

Task #2: Add option to choose which categories will be used for the session

Sub Task #1: Show the pre-game screen (can be ready panel) that provides the option for category selection

**Task Screenshots**

Gallery Style: 2 Columns

4 2 1



pre-game screen (can be ready panel) that provides the option for category selection

**Caption(s) (required) ✓**Caption Hint: *Describe/highlight what's being shown***Sub-Task**

100%

Group: Project Specific

Task #2: Add option to choose which categories will be used for the session

Sub Task #2: Show the code that sets and uses the chosen categories in the GameRoom

**Task Screenshots**

Gallery Style: 2 Columns

4 2 1

```

    // Method to start a round
    private void startRound() {
        ArrayList<Category> selectedCategories = getSelectedCategories();
        ArrayList<Category> allCategories = new ArrayList<Category>();
        for (ClientSession client : clients) {
            allCategories.addAll(client.getAvailableCategories());
        }
        return allCategories;
    }

    // m2055 12/13/20
    private void warning() {
        List<Category> chosenCategories = getSelectedCategories();
        Question selectedQuestion = questionList.get(0);
        questionList.remove(selectedQuestion);
        questionList.add(chosenCategories.get(chosenCategories.indexOf(selectedQuestion)));
        Random random = new Random();
        int index = random.nextInt(chosenCategories.size());
        if (index == 0) {
            System.out.println("No questions available for the selected categories.");
        }
    }
}

```

how the code that sets and uses the chosen categories in

the GameRoom

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Collect Player Categories: Grabs all categories players picked. Filter Questions: Finds a question that matches those categories. Ask Question: If it finds one, it shares it with everyone. Handle No Matches: If no question fits, it says no questions are available.

End of Task 2

**Task**

Group: Project Specific

100%

Task #3: Add ability to add new questions

Weight: ~33%

Points: ~1.33

▲ COLLAPSE ▲

**1 Details:**

Screenshots of editors must have the frame title visible with your ucid and the client name.  
Code screenshots must have ucid/data comments.



Columns: 1

**Sub-Task**

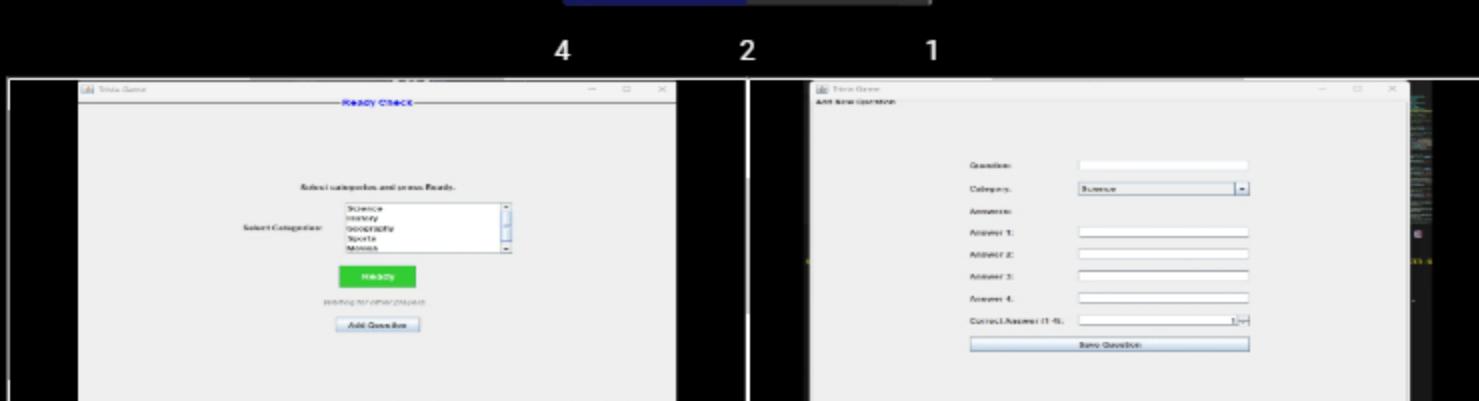
Group: Project Specific

100%

Task #3: Add ability to add new questions

Sub Task #1: Show the pre-game screen (can be ready panel) that allows adding new questions  
(see details above)**Task Screenshots**

Gallery Style: 2 Columns



Show the pre-game screen (can be ready panel) that allows adding new questions

**Caption(s) (required) ✓**

Caption Hint: *Describe/highlight what's being shown*

**Sub-Task**

Group: Project Specific

Task #3: Add ability to add new questions

Sub Task #2: Show the code that handles new questions and properly saves them to a file that can be used later

100%

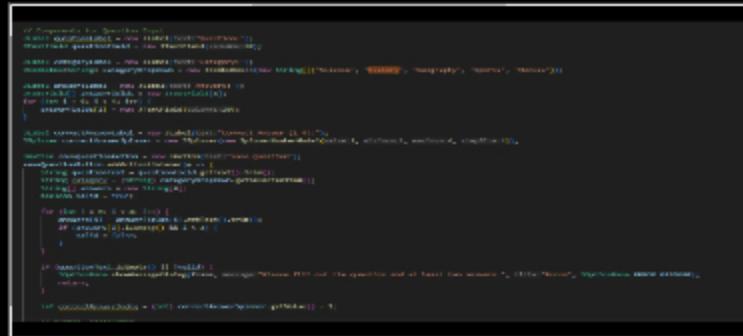
## Task Screenshots

Gallery Style: 2 Columns

4

2

1



```
public class QuestionHandler {
    public void handleNewQuestion(String question, String answerA, String answerB, String correctAnswer) {
        String formattedQuestion = String.format("Question: %s\nCategory: %s\nAnswer A: %s\nAnswer B: %s\nCorrect Answer: %s", question, category, answerA, answerB, correctAnswer);
        try {
            FileHandler.writeToFile("questions.txt", formattedQuestion);
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }
}
```

handles new questions and properly saves them to a file that can be used later

**Caption(s) (required) ✓**

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Check Input: Makes sure the question, two answers, and a correct answer are provided.

Save to File: Formats the question details and adds them to questions.txt.

Error Handling: Shows an error if saving fails.

Reset Form: Clears the fields for the next question.

**Sub-Task**

Group: Project Specific

Task #3: Add ability to add new questions

Sub Task #3: Show a before and after of the questions file and the related terminal output messages showing that the process works

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1



Show a before and after of the questions file and the related terminal output messages showing that the process works

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡ Task Response Prompt

*Explain in concise steps how this logically works*

Response:

here the user will be notified that the question been added

End of Task 3

End of Group: Project Specific

Task Status: 2/3

Group



Group: Misc

Tasks: 3

Points: 1

▲ COLLAPSE ▲

Task



Group: Misc

Task #1: Add the pull request link for the branch

Weight: ~33%

Points: ~0.33

▲ COLLAPSE ▲

**i** Details:

Note: the link should end with /pull/#



## ☞ Task URLs

URL #1

<https://github.com/Mohamad4322/>

ma2633-114-003/

URL

<https://github.com/Mohamad4322/ma2633-114->

End of Task 1

Task



Group: Misc

Task #2: Talk about any issues or learnings during this assignment

Weight: ~33%

Points: ~0.33

[▲ COLLAPSE ▲](#)

## Task Response Prompt

Response:

At first, there was no way to send or handle player-selected categories, so we had to add the logic from scratch, both on the client and server sides. Spectators weren't properly excluded from game actions, which caused confusion. We added checks to block their participation and notify them when they tried. There wasn't a clear way to save new questions for reuse. We implemented a system to validate input and store questions in a reusable file format.

End of Task 2

Task



Group: Misc

Task #3: WakaTime Screenshot

Weight: ~33%

Points: ~0.33

[▲ COLLAPSE ▲](#)

Details:

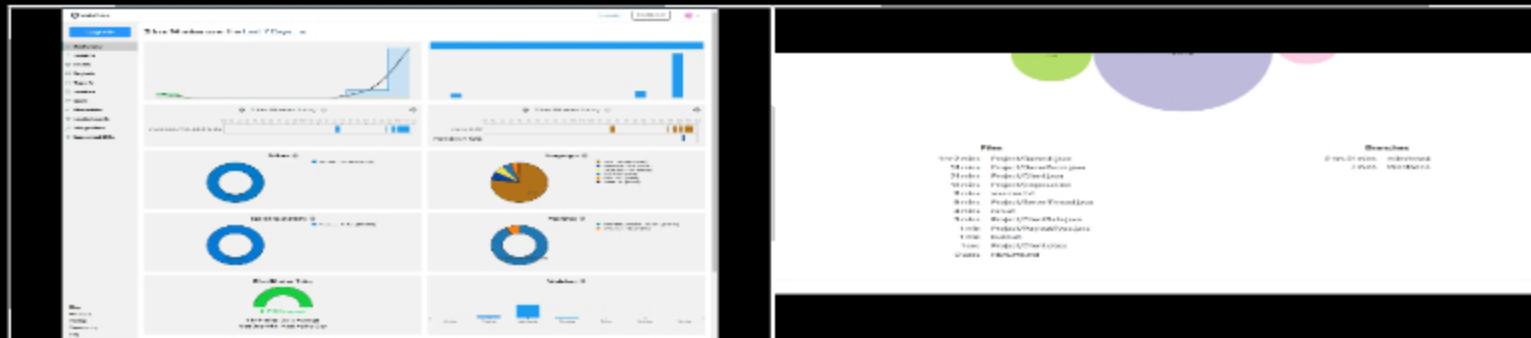
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved



## Task Screenshots

Gallery Style: 2 Columns

4      2      1



WakaTime Screenshot

WakaTime Screenshot

End of Task 3

End of Group: Misc

Task Status: 3/3

End of Assignment