Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-3-trivia-2024-m24/grade/ma2633

Course: IT114-003-F2024
Assigment: [IT114] Milestone 3 Trivia 2024 M24
Student: Mohamad A. (ma2633)

Submissions:

Submission Selection
1 Submission [submitted] 12/1/2024 9:24:37 PM

Instructions

COLLAPSE

Implement the Milestone 3 features from the project's proposal document: https://docs.google.com/document/d/1h2aEWUoZ-etpz1CRI-StaWbZTjkd9BDMq0b6TXK4utI/view Make sure you add your ucid/date as code comments where code changes are done All code changes should reach the Milestone3 branch Create a pull request from Milestone3 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: Milestone3

Group

Group: Basic UI
Tasks: 1
Points: 2

COLLAPSE

Task

Group: Basic UI
Task #1: UI Panels
Weight: ~100%
Points: ~2.00

COLLAPSE

Details:
All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task
100%
Group: Basic UI
Task #1: UI Panels
Sub Task #1: Show the ConnectionPanel by running the app (should have host/port)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

ConnectionPanel

Caption(s) (required) ✓

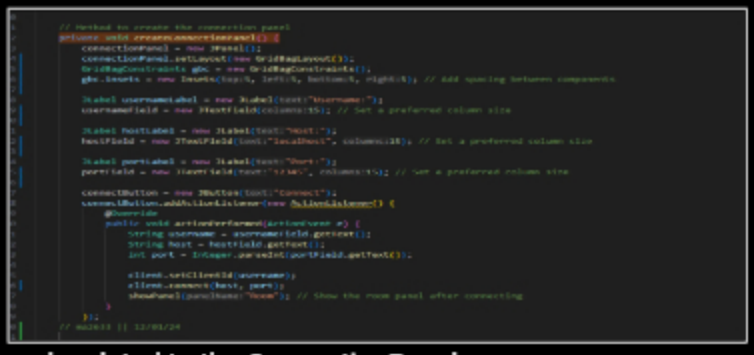
Caption Hint: *Describe/highlight what's being shown*

Sub-Task
100%
Group: Basic UI
Task #1: UI Panels
Sub Task #2: Show the code related to the ConnectionPanel

Task Screenshots

Gallery Style: 2 Columns

4 2 1



code related to the ConnectionPanel

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain how it works and how it's used

Response:

This method creates a simple panel where users can input their username, host, and port to connect to a server. When they click "Connect," it uses the entered details to establish a connection through the client object and switches to the "Room" view if successful. It's basically the login screen for connecting to the server

Sub-Task

100%

Group: Basic UI

Task #1: UI Panels

Sub Task #3: Show the UserDetailsPanel by running the app (should have username)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

UserDetailsPanel

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Basic UI

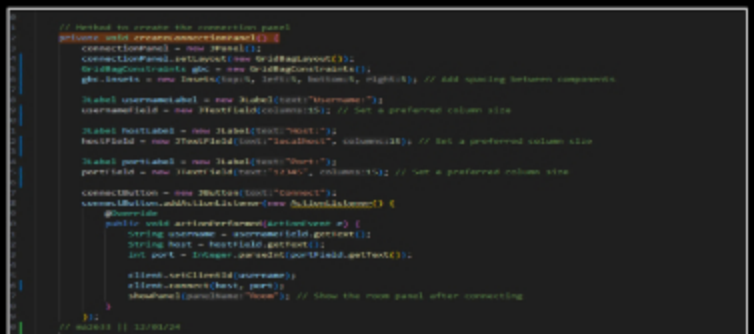
Task #1: UI Panels

Sub Task #4: Show the code related to the UserDetailsPanel

Task Screenshots

Gallery Style: 2 Columns

4 2 1



code related to the UserDetailsPanel

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain how it works and how it's used

Response:

This method creates a simple panel where users can input their username, host, and port to connect to a server. When they click "Connect," it uses the entered details to establish a connection through the client object and switches to the "Room" view if successful. It's basically the login screen for connecting to the server

End of Task 1

End of Group: Basic UI
Task Status: 1/1

Group

91%

Group: Game Area
Tasks: 6
Points: 7

COLLAPSE

Task

50%

Group: Game Area
Task #1: ReadyCheck UI Panel
Weight: ~17%
Points: ~1.17

COLLAPSE

Details:

All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task

100%

Group: Game Area
Task #1: ReadyCheck UI Panel
Sub Task #1: Show the screen with the ready panel open in a fresh session

Task Screenshots

Gallery Style: 2 Columns

4 2 1

ready panel open in a fresh session

Caption(s) (required) ✓

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

Sub-Task

0%

Group: Game Area
Task #1: ReadyCheck UI Panel
Sub Task #2: Show the screen with the ready panel open after a session ends (there should be output in other parts of the UI showing this)

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)

End of Task 1

Task

100%

Group: Game Area
Task #2: User List
Weight: ~17%
Points: ~1.17

COLLAPSE

Details:

All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson says

App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task

100%

Group: Game Area
Task #2: User List
Sub Task #1: Show the UI indicating that a user locked in an answer for the round

Task Screenshots

Gallery Style: 2 Columns

4 2 1

user locked in an answer for the round

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Game Area
Task #2: User List
Sub Task #2: Show the related code (from server-side to UI) that marks the user list item properly

Task Screenshots

Gallery Style: 2 Columns

4 2 1

code (from server-side to UI) that marks the user list item properly

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

When a player clicks an answer, the client sends the locked answer to the server using the lockInAnswer method. The server processes this and notifies all players that someone has locked in their answer using notifyPlayersAnswerLocked. On the UI, this update reflects by showing the player's status, keeping everyone informed in real-time

End of Task 2

Task

100%

Group: Game Area
Task #3: GameEventPanel
Weight: ~17%
Points: ~1.17

COLLAPSE

Details:
All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task

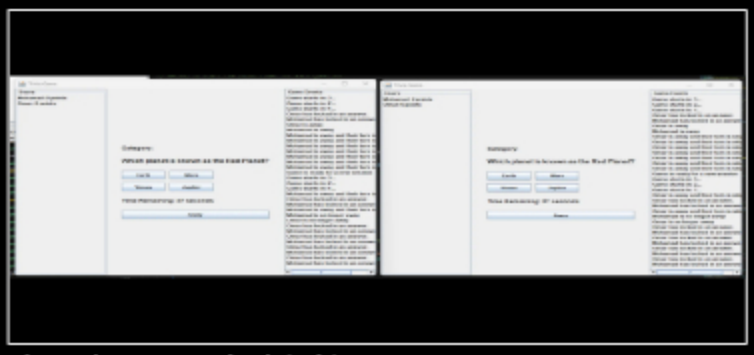
100%

Group: Game Area
Task #3: GameEventPanel
Sub Task #1: Show the answer lock-in history

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the answer lock-in history

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Game Area
Task #3: GameEventPanel
Sub Task #2: Show earned points (should show the name and the points acquired)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Show earned points

Caption(s) (required) ✓

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

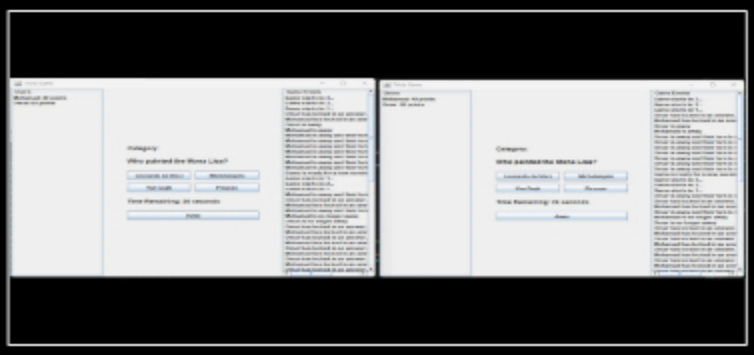
Sub-Task 100%

Group: Game Area
Task #3: GameEventPanel
Sub Task #3: Show the scoreboard updates from Milestone 2 (should display on the UI)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



scoreboard updates from Milestone 2

Caption(s) (required) ✓

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

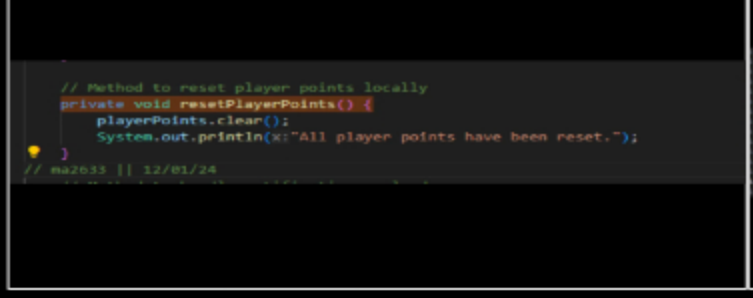
Sub-Task 100%

Group: Game Area
Task #3: GameEventPanel
Sub Task #4: Show the code for the UI flow (Client receiving to UI) for each of the 3 event examples above

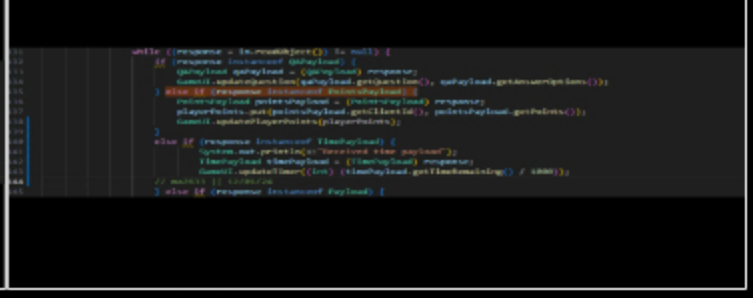
Task Screenshots

Gallery Style: 2 Columns

4 2 1



Method to reset player points locally



Receiving PointsPayload

Calculate points based on response time

sync points to all clients

Updating Points on UI

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

The server calculates points, updates the player's score, and broadcasts the changes to all clients using a PointsPayload. The client processes the PointsPayload, updates the local points map, and calls GameUI.updatePlayerPoints(). The updatePlayerPoints method updates the scoreboard in the UI, showing each player's points in real time.

End of Task 3

Task

100%

Group: Game Area

Task #4: Question and Category

Weight: ~17%

Points: ~1.17

COLLAPSE

Details:
All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task
100%
Group: Game Area
Task #4: Question and Category
Sub Task #1: Show the question category

Task Screenshots

Gallery Style: 2 Columns

4 2 1

question category

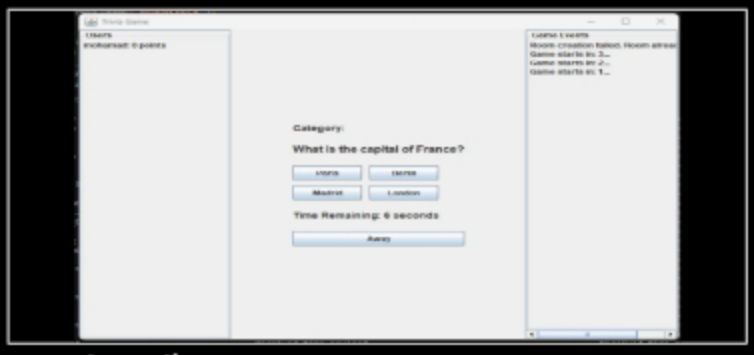
Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Sub-Task
100%
Group: Game Area
Task #4: Question and Category
Sub Task #2: Show the current question

Task Screenshots

Gallery Style: 2 Columns

4 2 1



current question

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Game Area
Task #4: Question and Category
Sub Task #3: Show the UI code related this data (from Client receiving to UI)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

broadcast the current question to all clients

load questions from a file

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

The server selects a question and its category, creates a QAPayload with the question text and answer options, and sends it to all clients. The client processes the QAPayload and calls GameUI.updateQuestion with the received question text and options. The updateQuestion method updates the question label and, optionally, the answer options on the game screen for the player.

End of Task 4

Task

100%

Group: Game Area
Task #5: Answers
Weight: ~17%
Points: ~1.17

COLLAPSE

Details:

All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task

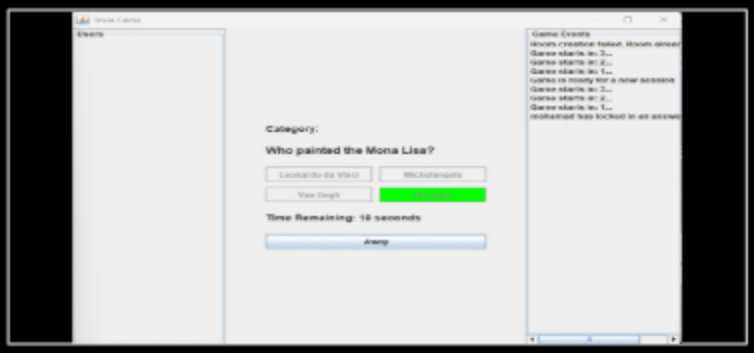
100%

Group: Game Area
Task #5: Answers
Sub Task #1: Show the current answers each with a button to lock in that choice (Locking in changes the color of the button and disables all answer choices)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



current answers each with a button to lock in that choice

Caption(s) (required) ✓

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

Sub-Task

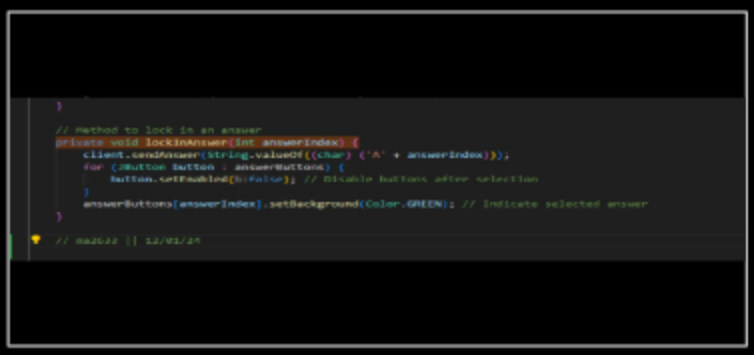
100%

Group: Game Area
Task #5: Answers
Sub Task #2: Show the code related to managing and interacting with these components (UI to Client sending)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
    }

    // method to lock in an answer
    private void lockInAnswer(int answerIndex) {
        client.sendAnswer(String.valueOf((char) ('A' + answerIndex)));
        for (JButton button : answerButtons) {
            button.setEnabled(false); // Disable buttons after selection
        }
        answerButtons[answerIndex].setBackground(Color.GREEN); // Indicate selected answer
    }

    // main() || 12/01/24
```

code related to managing and interacting with these components

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Sends the chosen answer to the server using client.sendAnswer. Disables all answer buttons to prevent further selection using button.setEnabled(false). Changes the background color of the selected button to green using

setBackground(Color.GREEN).

End of Task 5

Task

100%

Group: Game Area
Task #6: Countdown Timer UI
Weight: ~17%
Points: ~1.17

COLLAPSE

Details:
All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

Columns: 1

Sub-Task

100%

Group: Game Area
Task #6: Countdown Timer UI
Sub Task #1: Show the UI of the countdown (few examples to show it changes)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

UI of the countdown 1

UI of the countdown 2

UI of the countdown 3

Caption(s) (required)

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

Sub-Task

100%

Group: Game Area
Task #6: Countdown Timer UI
Sub Task #2: Show the code related to managing the timer

Task Screenshots

Gallery Style: 2 Columns

4 2 1

start the round timer for each question

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works, also note if you're doing two separate timers or just syncing the ticks (or something else)

Response:

It initializes the timer with a 30-second duration (roundDuration) and updates every second (updateInterval). Each second, the server creates a TimePayload with the remaining time and sends it to all connected clients. The timer decrements timeRemaining by updateInterval. When the timer reaches zero, it prints "Round timer expired," calls endRound(), and cancels the timer.

End of Task 6

End of Group: Game Area
Task Status: 5/6

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

Group: Misc

100%

Weight: ~33%

Points: ~0.33

COLLAPSE

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

100%

Group: Misc

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

Weight: ~33%

Points: ~0.33

COLLAPSE

Task Response Prompt

Response:

One of the challenges during this assignment was managing the round timer synchronization between the game room logic and the UI. The timer needed to be accurately reflected on both the server and client sides, and making sure that all players see consistent countdowns was tricky. It required careful attention to timing updates and broadcasting notifications without delays or mismatches

End of Task 2

Task

100%

Group: Misc

Task #3: WakaTime Screenshot

Weight: ~33%

Points: ~0.33

COLLAPSE

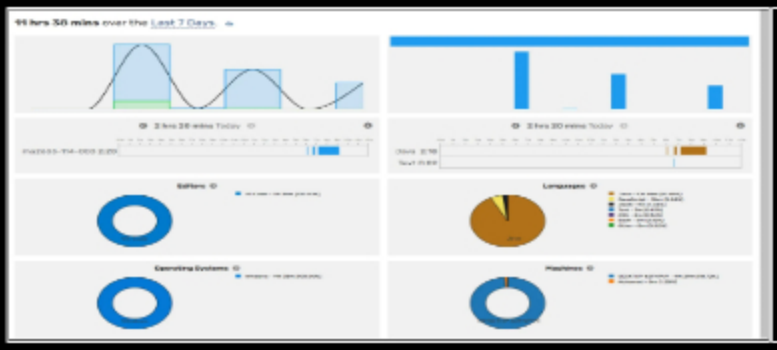
Details:

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

WakaTime Screenshot

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment