Project Part 3

Objective 1: A Sense of Time

Add a feature to your app that takes timing into consideration. You can design your own feature for this objective as long as it meets the following criteria:

- Timing must be controlled and verified by your server (No front end timing control)
- The current timing is sent from the server and displayed to all relevant users in real-time
- Timing must have <= 1 second accuracy
- There is a fixed point in time where the functionality of the app changes
- Timing must be initiated by a user[s]
- The feature must involve interactions between users

Any feature that meets these criteria will complete this objective. If you are having trouble thinking of an idea, here are some sample features that would meet all criteria:

- A game/auction/sale/quiz/etc with a time limit. The server tracks the time and sends the time remaining to all users every second which is displayed on the front end. When time expires, the game ends / auction ends and is awarded to the highest bidder / the sale ends and the price returns to the original pre-sale price / the quiz ends and submissions are no longer accepted / etc
    - Each game/auction/sale/quiz/etc is created/started by a user
- DMs/video chats/meetings started by the users with a set end time. The server sends the current time every second to all users which is displayed along with the end time. When the event ends, users cannot send messages/all video feeds end
- Users can schedule posts. A user can create a post and select a timestamp or delay after which the server will make the post public to all users. The server sends the user who created the post the time remaining before the post goes public every second which is displayed on their app
- A turn-based game that tracks the total time taken by each player during their turns. The time is tracked by the server and sent to all players every second
- A user list that displays the total time each user has been active on the app. All times are tracked by the server and sent to all users every second
    - Same idea, but display how long a user has been inactive. Check if the tab is focussed or react to mouse movement to tell the server when to reset the timer

Testing Procedure

1. Start your server using docker compose up
2. Open two browsers and navigate to http://localhost:8080/
3. Create an account and login in both browsers

4. Find a feature that implements a sense of timing. When you find a way to do this, move on to step 5
    a. If there is no clear way to do this, create and interact with posts and check if the feature is made apparent
    b. Look for any links to other pages on the app that may contain the feature
    c. Refresh the home page
    d. If there's still no clear feature that uses timing after following all these steps, the feature is assumed to not be implemented.
5. Interact with the timing feature and verify that it meets all the required criteria
    a. This step is open-ended as it will depend on the specific feature implemented by each team. Do whatever is needed to test the feature and verify that it meets the required criteria
    b. If necessary to test interactions effectively, use a 3rd/4th/etc. browser/account
6. Check for relevant **security** vulnerabilities

Objective 2: DoS Protection (IP rate limiting)

Add very basic DoS protection to your app based on rate limiting IP addresses. Your IP protection should:

- Block requests from an IP address if it has made more than 50 requests within a 10 second period
    - Every single request from an IP should count towards this limit (eg. Just loading your homepage requires ~5 requests that all count toward the limit)
- While an IP is blocked, respond to all requests from the IP with a 429 "Too Many Requests" response with a message explaining the issue to the user
- When an IP becomes blocked, it should remain blocked for 30 seconds
- After 30 seconds pass, requests from the IP should be handled as usual unless they hit your rate limit and become blocked again

Testing Procedure

1. Navigate to the public deployment
2. Count the number of requests made to load the page
3. Refresh slowly enough times to reach 70 requests, but not fast enough to trigger the rate limiter (<50 requests per 10 seconds)
    a. Ensure all requests are handled as expected
4. Wait ~10 seconds
5. Refresh quickly such that >50 requests are sent in at 10 second window
    a. Ensure that after the 50th request, you start seeing 429 responses and the page no longer loads
6. Wait ~20 seconds
7. Refresh the page and ensure that you are still blocked

8. Wait another ~10 seconds
9. Refresh the page and ensure that it loads
10. [Hit the rate limit again on the same device] Refresh quickly such that >50 requests are sent in at 10 second window
    a. Ensure that after the 50th request, you start seeing 429 responses and the page no longer loads
11. Within 30 seconds, on a second device with a different public IP address, navigate to the public deployment and ensure that the page loads
    a. Make sure the other device has a different **public** IP address. If you are at a residence, all your devices probably have the same public IP address. To ensure you have a different IP, you can use your phone on mobile data, or turn on a VPN

Objective 3: Creativity and Documentation

Add one more feature to your app of your own design and document this feature in your project's README. You have a significant amount of flexibility for this objective.

Requirements:

- Your feature cannot be a subset (⊆) of the requirements, of any other objective in this course across all project parts and homework assignments
    - It's ok to have overlap with an existing objective [or be a proper superset of the requirements] as long as you add something extra that was not required (eg. You can use WebSockets as long as you do something in addition to basic interactions [part2o2] or timing [part3o1])
- Add a description of your feature in the README of your repository on the default branch
- Write your own testing procedures for this feature in your README that the TAs will follow during grading

Note: This objective is extremely open-ended and there will likely be many very simple features that can be developed that technically meet the "no subset" requirement. These features will be accepted for credit no matter how simple as long as you can argue that what you did was not required in any other objective.

Testing Procedure

1. Navigate to the repository on GitHub and read the README on the default branch (eg. Click the repo link and scroll down to find the documentation. If any other steps are required, the objective is assumed to not be implemented)
2. Find and review the description of the extra feature in the README
    a. Verify that the description of the feature is clear and understandable
    b. Verify that the feature is not a subset of any other objective in this course

3. Find the testing procedures for this feature in the README
4. Follow the testing procedure and verify that the app passes the teams testing procedures

Submission

All of your project files must be in your GitHub repo at the time of the deadline. Please remember to include:

- A docker-compose file in the root directory that exposes your app on port 8080
- All of the static files you need to serve (HTML/CSS/JavaScript/images)

It is **strongly** recommended that you download and test your submission. To do this, clone your repo, enter the directory where the project was cloned, run docker compose up, then navigate to localhost:8080 in your browser. This simulates exactly what the TAs will do during grading.

If you have any Docker or docker compose issues during grading, your grade for each objective may be limited to a 1/3.

Team Scoring

Each objective will be scored on a 0-3 scale as follows:

3 (Complete)	Clearly correct. Following the testing procedure results in all expected behavior
2 (Complete)	Mostly correct, but with some minor issues. Following the testing procedure does not give the *exact* expected results, but all features are functional
1 (Incomplete)	Not all features outlined in this document are functional, but an honest attempt was made to complete the objective. Following the testing procedure gives an incorrect result, or no results at all, during any step. This includes issues running Docker or docker-compose even if the code for the objective is correct
0.3 (Incomplete)	The objective would earn a 3, but a **security** risk was found while testing
0.2 (Incomplete)	The objective would earn a 2, but a **security** risk was found while testing
0.1 (Incomplete)	The objective would earn a 1, but a **security** risk was found while testing
0 (Incomplete)	No attempt to complete the objective or violation of the assignment (Ex. Using an HTTP library)

Note that for your final grade there is no difference between a 2 and 3, or a 0 and a 1. The numeric score is meant to give you more feedback on your work.

3	Objective Complete
2	Objective Complete
1	Objective Not Complete
0	Objective Not Complete

Please note that there is only one chance to earn these application objectives. There will not be a second deadline for any part of the project.

Individual Grading

The grading above will be used to determine your team score which is based on the functionality of your project. Your actual grade may be adjusted based on your individual contributions to the project. These decisions will be made on a case-by-case basis at the discretion of the course staff. Factors used to determine these adjustments include:

- Your meeting form submissions: team meeting and eval form
    - You must fill out this form after every meeting. Failure to do so is an easy way to earn a negative individual grade adjustment. Since you have weekly meetings, you are expected to have as many form submissions as there were weeks for this part of the project
    - The quality of your submissions will be taken into account as well (eg. Saying "I'll do stuff" before the next meeting is a low quality submission)
    - You may submit more meeting forms than are required even if there was no meeting (eg. If you want to adjust your evals after a deadline without waiting for the next meeting)
- Your evaluations from the meeting form
    - If your teammates rated you poorly/excellently, your grade may be adjusted down/up respectively
- Your commits in the team repo
    - Your commits may be checked to see if you did in fact complete the work you mentioned in the meeting form, as well as compare the amount of work you completed to that of your teammates
    - You **MUST** commit your own code! It is not acceptable for your teammate to commit your code for you. You should have a clear separation between your tasks and commit the code for your task. If a commit is not in your name, you

effectively did not write that code. If a teammate is making this difficult, let the course staff know in the meeting form
    - If you don't have any commits on the default branch of the repo, you effectively did not work on the project and will earn a 0 after individual grade adjustments

Security Essay

For each objective for which your team earned a 0.3 or 0.2, you will still have an opportunity to earn credit for the objective by submitting an essay about the security issue you exposed. These essays must:

- Be at least 1000 words in length
- Explain the security issue from your submission with specific details about your code
- Describe how you fixed the issue in your submission with specific details about the code you changed
- Explain why this security issue is a concern and the damage that could be done if you exposed this issue in production code with live users

Any submission that does not meet all these criteria will be rejected and you will not earn credit for the objective.

Security essays are individual assignments. It is important that every member of the team understands the importance of preventing security vulnerabilities even if they did not write the offending code. Multiple members of a team submitting the same, or similar, essays is an academic integrity violation.

Due Date: Security essays are due 1-week after grades are released.

Any essay may be subject to an interview with the course staff to verify that you understand the importance of the security issue that you exposed. If an interview is required, you will be contacted by the course staff for scheduling. Decisions of whether or not an interview is required will be made at the discretion of the course staff.