Team Project Requirements

This document will provide more details about the project than what is listed on the course website. You should first read the section on the website before reading this document as not everything mentioned there will be repeated here.

GitHub Repository

Create a GitHub repository for your project and add all your team members as collaborators. This repository will contain all the code for your project.

This repository can be public or private depending on what you decide as a team. If you make a private repository you must add Jesse as a collaborator so your project can be assessed. Jesse's GitHub username:

- hartloff

Note: It's recommended that you use GitHub issues (Or a task tracking app) to document what each team member is working on at any given point in time. You can submit links to these issues/tasks on the meeting form to make it very clear what each team member was responsible for completing. If the tasks are not clear, individual grading decisions may be made at the discretion of the course staff.

Readme

The readme in your repository should include a description of your project. This document should clearly state the end goal of your app from a user perspective (ie. What a user can do on your app when it's finished)

Open-Source Reports

Create a directory in your repository named "reports" that contains all the open-source reports for your project. Review the project section of the course website for details on what each report must contain. You need a separate report for each technology you use in your project that would not be allowed on a homework assignment.

Docker-Compose

Your app must be setup to deploy with docker-compose. In the root directory of your repository include a docker-compose.yml file with everything needed to run your app using docker-compose. At a minimum this must include creating a container for your database and another container for your app that will communicate with the database container.

- You may choose the local port for your app unless otherwise directed

When testing or deploying your app the procedure will be:

1. Clone your repository
2. cd into to the directory of the repository
3. Build the docker image (docker-compose build)
4. Create and run a docker containers (docker-compose up --detach)
5. Open a browser and navigate to http://localhost:<local_port> (We'll read your docker-compose.yml to find your local port)

Deployment

Details TBD.

Requirements

Below is more description of each of the 5 criteria listed on the course website.

- User Accounts with Secure Authentication
    - Users must be able to register accounts and log into their account. Once logged in, each user must see at least some content and functionality that is specific to them and cannot be accessed by other users (If you completed DMs then this is covered)
    - Each user must have at least 1 setting that they can adjust. This setting must persist across sessions. For example, if a user logs in and changes this setting it should still be changed when they log off and log back in from a different device
    - Requires use of a database in Docker to store uploads (Services such as FireBase are not allowed)
    - OAuth is allowed (Report required!), though you still have to store a profile for the user with their setting(s) on your server
- Users to see all users who are currently logged in
    - It is up to you what feature(s) you want to implement for this requirement, though there should be a way to see all other logged in users.
- Users can send direct messages (DM) to other users with notifications when a DM is received
    - From the previous requirement, each user is able to "see" other users. There must be an option to be able to DM users where you can see them.
    - DM's can be text-only
    - When a DM is received, the recipient must be notified in real-time and given an option to reply
    - You may assume the recipient is online at the time the DM is sent

- Users can share some form of multimedia content which is stored and hosted on your server
    - Requires use of a database in Docker to store uploads (Services such as FireBase are not allowed)
    - The multimedia content itself can be stored as files on your server with the file paths stored in your database
    - The details of the content are up to your team to design as long as multimedia is shared in some way (Ex. Uploading profile pictures which other users see is acceptable)
- Live interactions between users via WebSockets (Cannot be text)
    - Users interact in real-time via WebSockets. This cannot be raw text (ie. you can't build a chat app like we saw in lecture)
    - The WebSocket messages can be formatted as JSON Strings, so long as the feature from the user perspective is not sharing plain text with other users.
    - Example: You build a game where user locations are shared via WebSockets allowing all players to see other players move in the game world.
    - Example: A shared canvas on which users can draw and see each other's drawings

Security

Your site must be secure! If a vulnerability is found, the consequences will be decided based on the severity of the vulnerability. Severe vulnerabilities may result in a 0 for this phase of the project.

At a minimum, we will explicitly check for the following during grading:

- HTML/JS Injection (This is considered a severe vulnerability!)
- SQL Injection (Use prepared statements)
- Securing user accounts (If passwords are stored they must be salted and hashed properly. There must not exist a practical way for a user to authenticate as another user)
- Private content must be private (Do not assume users are using your site as intended. Your server must check that the user is authenticated before sending them private content even if your site doesn't offer a way to request that content without being authenticated)

Submission

Fill out this form (You only need 1 submission per team) each time you want to use a project checkpoint.