Lab Activity 1

Project Ideas and Team Formation

Objective

In this lab you will choose your team members for your semester-long project and choose what you will develop for your project

Team Size

Form teams with 3 members. If the lab size is not divisible by 3, a minimum number of teams of 4 can be formed

Project Overview

You and your teammates will develop a real-time MMO (Massively Multiplayer Online) application. You may choose whatever you would like to build for this project so long as it meets the conditions given in this document. Many of the lab activities will be designed to help you with aspects of the project and the lecture material will cover the concepts needed to develop your project. It is up to you and your team to apply those concepts to accomplish your goals and realize your project idea.

Though you are choosing an initial idea today, you are free to change this idea later. You have no realistic way of knowing how much work will be involved in building this app so don't be afraid to choose an ambitious project now then switch to a simpler idea later in the semester.

Project Conditions

As you brainstorm your project idea please ensure that it will contain the following required features:

1. Your project will be a massively multiplayer web application (MMO)
    - For this project we will define MMO as a multiplayer experience that has no inherent upper limit to the number of simultaneous users (Ex. Chess, Football, and Poker all have upper limits on simultaneous players and web versions of these games would not qualify as MMOs)
    - For full credit it should be possible for everyone in class to use your app during lecture (This will be a reality for select projects during the last week of lectures)
2. Your users interact with each other in real-time

- We will consider your idea to be real-time if your users would notice a difference in the user experience if there were a delay of 1 second in the app
- The simplest way to meet this criteria may be to make a real-time game where all players share a game world and can see/interact with each other (Ex. Minecraft)
- An idea that would not meet this criteria is a standard social media site or messaging app. The user experience would only negligible be affected by a 1 second delay and most users would not notice the delay at all

Project Architecture

By the end of the semester you will build an application comprising of:

1. A server that will maintain the state of your application and allows users to interact with each other [Any language*]
2. A web front end [JavaScript/HTML/CSS]
3. A networked desktop front end that can run on a different machine than your server [Any language*]

*You may use languages not covered in this course

With this architecture you will have a single server that will maintain the state of your application and allow users to interact (The model in the MVC pattern). The web and desktop front ends will both connect to the same server. As an example, if you are making a game and a player using the desktop front end moves their character your code will send that movement to the server, the server will update the state of the game to reflect this movement, then send that game state (via websockets) to all web and desktop users where the GUIs will be updated and all players see the movement.

While all of the concepts needed will be covered in class, they might not be covered in your chosen languages and your team should expect to have to find libraries and study documentation on your own to complete your project. For example, we will not cover web servers in Scala so if you choose Scala for your server you will have to choose and setup a Scala web framework.

Deliverables for Today's Activity

Deadline: Before the end of your lab time this week. Do not believe the deadline listed on AutoLab. The submission on AutoLab is for all lab sections so that deadline is at the end of the week. You will not receive credit if you submit after your lab period.

Submit a pdf file on AutoLab under Lab Activity 1 with:

1. The names of all your team members
2. A brief description of your project idea

3. An initial plan for how you will divide the work among team members
4. What each team member will complete for project demo 1 in 4 weeks (See below)

All team members must submit on AutoLab. It is fine if each team member submits the same exact document, but all team members need a submission in AutoLab for grading.

Ask a TA to review and approve your document before submitting to AutoLab to avoid losing points for missing details. There is no reason to miss any points today if you follow the instructions carefully and verify with a TA before submitting.

Project Demo 1

The week of March 4 in lab.

For today's submission each team member will choose one of the following to complete. Project Demo 1 will be individually graded based on the completion of your chosen task. Multiple team members can choose the same option out of these three, but they must work on different parts of the project (Ex. Two members can work on back-end functionality, but must implement different functions).

Each team member chooses only 1 of these 3 options. Describe exactly what each team member will build for their chosen option.

Front-End Mockup: Using front-end web technologies (HTML/CSS/JavaScript) create a mock front-end for your project as a prototype. This prototype does not have to have any functionality or connect ot a server, but it must give someone unfamiliar with your project an idea of what your team plans to build just by looking at your prototype. For example, you might build the GUI for your app but none of the buttons do anything when clicked.

You must use web technologies for the mockup. You will not receive credit if the entire mockup is a static image or drawing. The idea is that this mockup will become your actual front-end once the buttons connect to JavaScript code and a server.

This is not an art class. Nothing on the mockup has to look good, but it should be understandable. For example, a stick figure drawn in MS Paint instead of full character model is fine (even for your final submission, in fact) as long as it's understandable that that's a character in your game.

Back-End Functionality: In your chosen server language write at least 3 functions that will be used in your completed software, write unit tests for each function, and pass those tests. You may choose which functions you implement so long as they are clearly related to your project and are not trivial (ex. A function that returns the name of your project is not allowed).

Example: If your project is to build a Fortnite clone (probably 2d, overhead view, and graphics like Asteroids to keep it feasible) you might want to implement functions for:

1. Hit detection. Given the orientation and location of the firing player detect who/if anyone was hit by the projectile
2. Elimination. A function that takes a username and removes them from a list of active players
3. Winner. A function to check if there is exactly 1 player remaining and return that player as the winner

Server Networking: In your chosen server language write a web server for at least one piece of functionality for your app allowing users to interact with each other. This will include:

1. At least 1 API endpoint to which a user - or rather, their browser - will connect to while using this feature. Realistically you might need 2 or more endpoints though
2. A database or file to store the interactions
3. A way to test this functionality

This does not have to be connected to a front end and testing can be done with a Python program that sends requests to the server directly (or any other way you would like to show your testing during the demo).

An example would be a way for players to join the server with endpoints:

- POST “/join” with the body containing a username which adds the player to the game
- POST “/leave” with the body containing a username which removes the player from the game
- GET “/players” returning a list of all the players currently in the game

You could then write a script that adds 5 players, removes 2 of them, then go to your browser to request “/players” and verify the player list

---

For all choices you must list in detail what each team member will implement and demo, then have a TA approve these choices before submitting. Any submissions that just list each member as “Front-End Mockup” or “Server Networking” with no details will not earn full credit for this activity.

Non-Participants

Any team member who does not attempt to complete their individual task for project demo 1 may be removed from their team at the discretion of the course instructor. For project demos 2 and 3 you will be graded as a team and the team should not suffer due to a non-participant.

Looking Ahead

For project demo 2 your team will develop and demo a working desktop application with no networking and no multiplayer. This will be written entirely in your server language(s) and can run as a single application on a single machine.

For project demo 3 your team will add, and demo, the web front-end and the networked desktop front-end and show that they are connecting to the same server. For the demo it is acceptable for all three parts of the app to run on a single machine, but you must prove that it would still work if all three parts were on different machines. This way you don't have find a server to host your app and deploy it for the demo, though it's highly recommended that you do so you can show off and use your app.

While developing your local app for demo 2 it is highly recommended that you follow the MVC design pattern by keeping your GUI separate from your game logic and then communicating between them using JSON strings. This may create more work during part 2, but will make part 3 much easier when the GUI will run on a separate machine than the server. If these parts are already separated logically you can move the GUI to another project and send the JSON strings over the Internet with minimal other changes to get your desktop app working. This will give your team more time to work on the web front-end and all the complex networking that must to take place.