

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_30j0zll)1

[**Table of Contents**](#_1fob9te)2

[**Document Revision History**](#_2et92p0)2

[**Executive Summary**](#_tyjcwt)3

[**Design Constraints**](#_3dy6vkm)3

[**System Architecture View**](#_1t3h5sf)3

[**Domain Model**](#_4d34og8)3

[**Evaluation**](#_2s8eyo1)3

[**Recommendations**](#_3rdcrjn)5

## [Document Revision History](#_2et92p0)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/18/2021 | Devin Shattuck | General Summary of Game Requirements |
| 1.1 | 08/01/2021 | Devin Shattuck | Addition of Operating System Requirements |
| 1.2 | 08/15/2021 | Devin Shattuck | Final Edition |

## [Executive Summary](#_tyjcwt)

The problem given by the client is that they want to make a web-based application. They requested that a game will be able to have one or more teams involved, and that each time will have multiple players on it. The game and team names must be unique so they can be easily found and so people can check if the name is in use. Only one instance of a game can exist at one time via UUIDs.

## [Design Constraints](#_3dy6vkm)

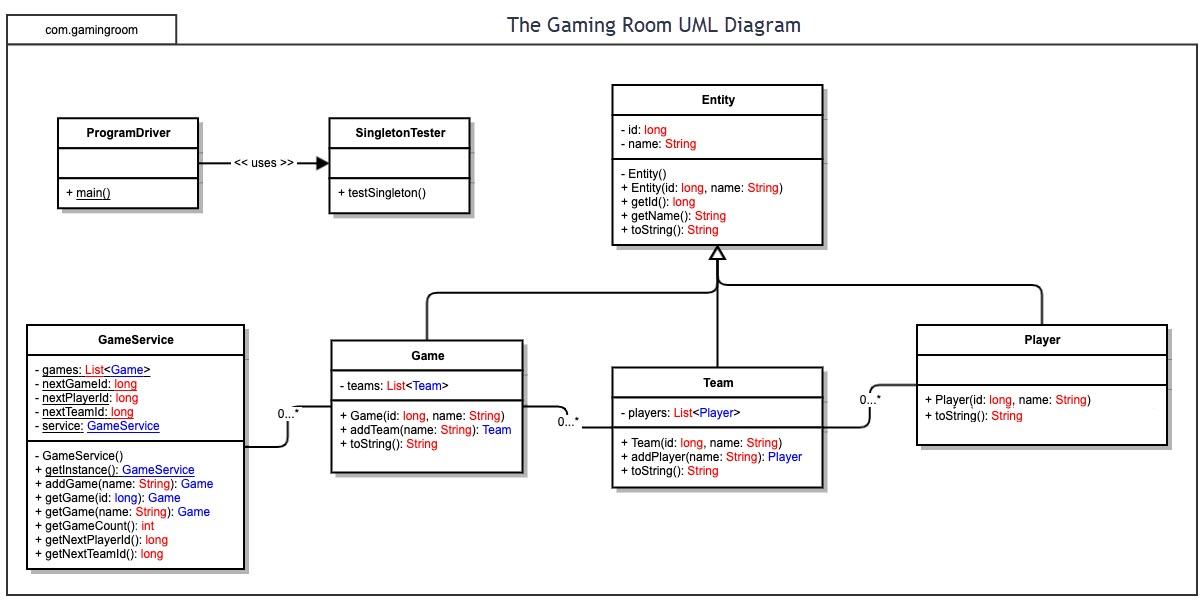
Making a Web Application will require you to send HTML, CSS, JavaScript, or PHP back to the client. Making a game with it will make using PHP very difficult, and JavaScript more helpful. This limits the options to writing the game in straight HTML, which will prove very difficult, or using node.js, which will make things considerably easier, albeit not easy. Having one or more teams with multiple players assigned to it will be entirely implementation based, and can be done numerous ways, but there must be an admin endpoint that can only be called internally that modifies the teams in some way so that users can’t create teams whenever they feel like it. Game and Team names being unique will require some way to check that an existing game/team doesn’t exist with a certain name, and only one instance of a game being there at once will require a singleton game manager to overlook and make sure only one pool of games exists and that each game exists once at a time. Everything else is up to implementation.

## [System Architecture View](#_1t3h5sf)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_4d34og8)

A ProgramDriver instance will use the SingletonTester class to start the program and test the GameService singleton. GameService is a singleton class that has private members, those being a static List of 0 to infinite Game objects to store the games, a static long nextGameId to store the next available game ID, a static long nextPlayerId to store the next available player ID, a static long nextTeamId to store the next available Team ID, and a static instance of the GameService itself for the singleton. It has a private constructor so people can’t create a new GameService instance, a static getInstance function that returns the GameService instance, an addGame function that takes in a String, the game name, to add to the pool, a getGame function that takes in a long ID to get a game from the pool, a getGame function that takes in a String name to get a Game from the pool, a getGameCount that returns an int count of the games, a getNextPlayerId function that returns a long next available player ID, and a getNextTeamId function that returns a long next available team ID. As stated before, the GameService stores 0 to infinite Game objects, which inherit from the Entity class. The Entity class stores a private long id, and a private String name. It has a private constructor with no arguments, but a public constructor with a long id argument and a String name argument. It has a getId function that returns a long id, a getName function that returns a String name, and a toString function that returns a String representation of the object. The Game object inherits this Entity object, gaining these members and all the functions, but on top of that it has a private List of 0 to infinite Team objects to store the teams. It has a public Game constructor with a long id and a String name to override the Entity constructor, an addTeam function that takes a String name and returns the Team with that name, and finally it overrides the toString function to return a String representing that Game. As stated, the Game stores 0 to infinite Team objects, which also inherit the Entity class, taking its members and functions and adding a private List of 0 to infinite Players objects to store the players. It also has a public Team constructor that takes a long id and a String name, an addPlayer function that takes a String name and returns the Player with that name, and finally it overrides the toString function and returns a String with that team’s info. As stated, the Team class contains a list with 0 to infinite Player objects. Player also inherits Entity, and has its members and functions, but has its own public constructor that takes a long id and a String name, and its own toString function that returns a String representation of the player.

****

## [Evaluation](#_2s8eyo1)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is a very expensive option and doesn’t give much in terms of advantages or disadvantages. It has good security, but takes a long time to set a web server up on. | Linux is a free OS and therefore the only expense is the space and power the machine takes up. It has good security if used correctly and is very easy to use for purposes like this. | Windows is also really easy to set up, but getting the license will cost some more money out of your pocket, just like Mac. That being said, it’s much more capable than Mac, but not as much as Linux. | Mobile Devices have very little power and as such are not capable of hosting a stable and large web server. They are not meant to do this, and while most do run on unix, they lock down the phone so you can’t run stuff like that. |
| **Client Side** | Because this is a web based application, any desktop OS’s will work the exact same, as it’s sent back in HTML which SHOULD BE parsed the same way for each. Make a working example for desktop and it should work for all desktop OS’s. | Because this is a web based application, any desktop OS’s will work the exact same, as it’s sent back in HTML which SHOULD BE parsed the same way for each. Make a working example for desktop and it should work for all desktop OS’s. | Because this is a web based application, any desktop OS’s will work the exact same, as it’s sent back in HTML which SHOULD BE parsed the same way for each. Make a working example for desktop and it should work for all desktop OS’s. | Mobile has different requirements as it has a different control schema and can also be a wildly different resolution, which has to be accounted for. HTML is parsed the same way but it needs to respond to the fact that it’s on mobile accordingly. |
| **Development Tools** | Mac has things like Visual Studio Code for programming node.js or making the HTML, but it takes a while to set up apache2 on Mac to host a server. | Linux has LAMP to host web servers and has many many many applications to code with (Visual Studio Code, Sublime Text, etc.) | Windows has the widest selection of applications to code with and also has WAMP to set up a server with, making it easiest to get tools for. | Unless you want to figure out how to use vim there are no premade applications to host a webserver on android and compilers will all be terminal based. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: Either Windows or a Linux-based OS would be appropriate for this task. MacOS would make it very difficult, as it doesn’t have the same cross-system capabilities as Windows and Linux do, and the features it does have aren’t easily accessible in the slightest. Windows and Linux make it easy to access other computers on the system and interact with them in whatever ways are needed.
2. **Operating Systems Architectures**: Windows is an operating system made more for office and home use, and as such isn’t inherently designed for purposes like this, but Microsoft makes it easy to install software made by them for these purposes. Linux on the other hand makes it very easy to run everything in the terminal and set up a system like this on the fly, without taking up any excess memory for things like user files and other files that would be necessary for other purposes. This frees up more memory and space to run and store other things. As such, Linux would probably be the best bet for this application.
3. **Storage Management**: For the server, a database that stores the currently running games, teams, and players so the program can easily grab and use the data in them would be easiest. It doesn’t take much space at all, is very quick to search (so long as there isn’t too much data), and as long as you take cautionary measures, such as deleting entries after they’re done being used, and only adding entries that you absolutely need, then there shouldn’t be too much to deal with there. For the client-side application, since it’s a web-based program, just have one computer host the necessary files (HTML, JS, Images, etc.) and send only the required ones to the user as they need it so it doesn’t take up too much memory or space on their end.
4. **Memory Management**: As stated before, for the server, Linux and a PostgreSQL/MySQL database should minimize memory use so long as, again, cautionary measures are taken not to use it too often or put in too many entries, and as such it’s easy to lighten the Memory load on the server side of things. On the client side, only sending the client needed material and coding efficiently will deplete their memory usage as much as possible. As long as your code isn’t a memory hog, and they’re not using anything they don’t need then there should be no reason for there to be any memory issues.
5. **Distributed Systems and Networks**: Writing a simple server on each device is very easy, and especially easy to do on Linux, as you can just write up a simple program that listens for packets, and can send packets to other machines accordingly. In case of a power outage, have constant backups of data and have the servers start on computer startup, which every OS has a feature to do. As long as nothing physical happens to the computers, then there’s not much to worry about in terms of data as long as you maintain it correctly.
6. **Security**: The two main concerns with security are making sure users can’t access permissions to anything they shouldn’t, and making sure users don’t have a way to bypass the permission check. A simple authentication system (implemented properly) would make sure users can’t access anything with admin permissions. Always make sure that any time a user can run or access any part of a program, that they cannot exploit it in some way. This comes to the second concern as well. SQL injections, if going with the database, are one of the main concerns when it comes to safety, and as such take basic precautions like sanitizing user input.