

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc1669637303)

[Table of Contents 1](#_Toc1679078926)

[Document Revision History 2](#_Toc721444607)

[Executive Summary 2](#_Toc888158757)

[Requirements 3](#_Toc1526749849)

[Design Constraints 3](#_Toc592019327)

[System Architecture View 3](#_Toc793503323)

[Domain Model 3](#_Toc820886460)

[Evaluation 4](#_Toc718722879)

[Recommendations 5](#_Toc548348535)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/24/24 | David Hammons | Initial Version |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to have a web-based application running their game that allows it to be less platform dependent, which allows it to run on multiple different operating systems. This means that it will need to run as a web app to allow it the ability to run on any platform. This also means we need to pay attention to details as every device it will run on is different and we also need to ensure optimization is key as we create this program. We want this to run just as good on a lower end device as it does on a higher end device. All of this is within the constraints of the budget given for this project.

## Requirements

* The programming language used need to be platform agnostic – needs to work on iOS, Android, MacOS, Windows and Linux
* Needs a scalable UI for the different screen sizes that will playing the games
* Well optimized to run within a web browser or web app
* Games need to have the ability to have one or more teams involved
* Game and Team names need to be unique
* Unique identifiers need to be used for each instance of a game, team, or player

## [Design Constraints](#_2et92p0)

* Every platform has it’s own SDK to develop applications on.
* You won’t be able to hardcode alot of the application due to how different every device and OS will be
* Even web-based we need to ensure application is properly optimized for each browser
* We need to also account for the added load to the servers with the application being able to reach a wider audience
* Optimization of the code – we need to also ensure that the code base is optimized, clean and easily readable as well as modular so we can add features much more easily

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

**This model shows that the GameService class creates a single instance that is only created by the GameSevice class. This means only one instance of the object can be running at a single time. The GameService class calls the Game class which is an extension of the Entity class. This means that Game has access to the methods and variables stored in the Entity class. The Teams and Players classes also extends the Entity class allowing those classes to access the methods and variables stored in Entity. The Game class calls on methods within the Team class to get data as it is a separate object that the game class am the Team class calls on Player class as it is also a different class than the Player class.**

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

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 could be used to host a web based software application. The OS overall is relatively hands free so we shouldn’t encounter to many issues trying to host off of a Mac | Linux would be the best when it comes to Server Side. Linux is the most capable and with the different kernels, you can find a kernel that fits server side much more and use that | Windows is also more robust to use than Mac but less robust that Linux. We could use it for server side but we could encounter issues trying to keep it up and running using Windows | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | MacOS has it’s own language for the applications it uses means that we will have to develop an application that only works on Mac and won’t be cross platform with Windows but can work across iPhone and iPad. | With Linux being open source you can use really any programming language or tools that you want to use for client side. There is a learning curve to use the Operating System but if you are determined, you can do it. | Windows uses C for its programming language. It also is much more used in the general public so it’s much more worth creating for than other operating platforms except for phones. | Mobile devices use different languages for their applications. Android tends to use Java more while iOS uses the Swift programming language. This means we would need to write two different applications. |
| **Development Tools** | MacOS uses the Swift operating system which is Apple’s variant of Objective-C. The IDE that works best for programming Apple devices is XCode which is only available on Mac. | Linux uses many different kernals meaning that their are many different development tools you can use for the operating platform. | Windows you can code any language you’re looking for and the best IDE’s to use would be Visual Studio Code or Eclipse. | Every device has their own SDK that they use to program on. Android has their Android SDK that you use java to make apps and Apple uses XCode on Mac to create Swift applications on iPhone. |

## Recommendations

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

1. **Operating Platform: The best operating platform to use will be Linux, as it is the most robust operating system in this scenario.**
2. **Operating Systems Architectures**: There are a couple kernals that could be used with Linux but my recommendation would be using the Linux kernel due to how robust, user friendly, and popular it tends to be overall.
3. **Storage Management**: The best storage management would possibly be an SSD or an NVMe Drive as it would help in making sure any data stored lasts a long time but if cost becomes an issue, we can always use HDD storage as it is much more viable to use due to how much cheaper they are than SSD storage but they are slower in the long run.
4. **Memory Management**: For linux you can allocate memory dynamically to ensure that you don’t use up all of the memory that is being used by the application. This, in combination with using Virtual memory could help the program in case it ever runs into a situation where it runs out of memory and is unable to use anymore. Instead of slowdowns or crashing it can allocate and use virtual memory as needed.
5. **Distributed Systems and Networks**: We would need to ensure that a great infrastructure would be in place to allow a smooth networking experience. Redundancy would need to be built in just in-case of any issues that may arise such as outages or anything of that nature. This means we would need multiples of switches, routers, as well as backup networking solutions to keep the game going for as long as we can if anything were to happen.
6. **Security**: For security you could use something like SELinux. This is a way to secure access within Linux by limiting access to what every application can connect with. Like a web browser has no need to ever connect to SSH so therefore that permission wouldn’t be allowed by the SELinux application. This allows restrictions that make it so unauthorized applications cannot interact with apps that it has no reason being used by.