

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/14/2022 | Farooq Ahmed | Added Recommendation to help understand better about the system. |

**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 problem we have here is to implement the game as web-based instead of relying on different architectures. Currently, The Gaming Room only has it on Android and they want to make the game accessible for everyone regardless of which platform the user is playing the game. Since we already have game developed in Android. So, it would not be a much hassle to implement the same logic for web and have to make few modifications. The company will need buy the server to host the game on the web and to make it accessible and reliable access, I would recommend using AWS Cloud to deploy and launch the game. It would be cheaper, reliable, secure, and fast.

## [Design Constraints](#_2et92p0)

There will be no design constraints as far as I am concerned. It is much simpler than building the game for different Operating System because you must choose different languages and Platforms, where we need different team to build a game for each individual architecture. Web-based is not dependent on any platform, it can run on any device.

## [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)

## 

Let’s began with ProgramDriver class, it uses the SingletonTester class to verify that GameService has singleton pattern applied and there is only a single instance of GameService exist in the system. GameService class can create multiple classes Game instances. Game instance can have multiple instances of Team. Team instance can have multiple instances of Player. Game, Team, and Player class inherits attributes and methods from Entity class. The reason we create Entity class is because the attributes and methods were all used in those classes. So, we don’t have to repeat each time.

**"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** | Hosting web-based software application is going to cost more because you are going to need Mac Hardware, which cost a lot of money. Current, new Mac machine minimum cost around $1,000. But it is the more secure option out of three Operating System. Security and Stability comes with a huge price, but it’s worth it. You don’t have to worry about someone hacking your machine. Operating System Is not open source that’s why It’s a lot harder to run some third-party software on your machine. Terminal commands are configured the same way as Linux servers. | Linux servers has low-cost maintenance. Linux web hosting is probably the best and cheapest option. It’s open source and continue to get updates and patches without having to pay anything out of your pocket. Since it’s open source, you do not often get updates on your machine. You will have to put your security measure to keep web server secure. Terminal commands are configured to access server and make any changes. There are less resources need to run Linux server compared to Windows or Mac. | Windows servers are ideal for hosting web-based application. Windows servers cost is between Mac and Linux server (can run on any low-cost machines). For Windows, you need high-end hardware for hosting web application. Otherwise, it will crash and go down in no time due to not enough RAM or lack of CPU speed. Instead of terminal like Mac or Linux, you can use PowerShell or Command Prompt to configure to access server and make any changes. You can third-party software easily. It is not restricted like Mac OS. There needs to be frequent updates and OS patches to keep server secure and stable. There will be downtime for web application during updates and patches. | Mobile Devices are not able to host web-based application. Mobile application can use third-party server. Best option would be to get from Cloud Service providers. |
| **Client Side** | You are going to need someone Mid-Level expertise for software development. Development time and cost will be moderate, it’s not going to be too high. Preferably someone, who has done software development in Mac OS and understand Mac architecture. Mac OS comes with built-in Apache servers to allows basic web code like cPanel, vBulletin, and WordPress. | Need someone with High-Level of expertise and it is going to require more time than Windows and Mac. There aren’t many people out there that used Linux in their day-to-day life. Development cost will be very low. Need expert guidance through the software development process. Just like Mac, it comes with built-in Apache server to host web-based application. | You are going to need someone Mid-Level expertise for software development. Development time and cost will be moderate, it’s not going to be too high. Preferably someone, who has done software development in Windows and understand Windows architecture. Windows application can run application that use Microsoft SQL Server Database, AST, and .NET core apps. | Mobile application software development cost is not going to be high since we already have Android mobile application. This will help us building mobile application for iOS. |
| **Development Tools** | Relevant programming languages: HTML, CSS, JavaScript, Java, Python, and PHP  Databases: MySQL, MongoDB, or PostgreSQL  Tools: Eclipse, Visual Studio Code, IntelliJ, and GitHub repository. | Relevant programming languages: HTML, CSS, JavaScript, Java, Python, PHP  Databases: MySQL, MongoDB, or PostgreSQL  Tools: Eclipse, Visual Studio, IntelliJ, and GitHub repository | Relevant programming languages:  HTML, CSS, JavaScript, C# .NET  Database: Microsoft SQL Server  Tools: Eclipse, Visual Studio, and GitHub repository | Relevant programming languages:  HTML, CSS, JavaScript, Java, Dart, Kotlin, Swift Databases: MySQL, MongoDB, or PostgreSQL  Tools: XCode, Visual Studio, Android Studio. |

## 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**: I would recommend Linux Server platform to expand Draw It or Lose It to other computing platforms. Linus servers are low cost and don’t need high end hardware to host this game. For instance, Raspberry Pi would be sufficient, and it is affordable, low power, has built0in wireless networking and Ethernet, which most of the older hardware lacks. For optimal performance, I would recommend getting dedicated Linux server with newer

Hardware installed or you can lease it from Cloud Service Provider like AWS, Azure, or Google Cloud.

1. **Operating Systems Architectures**: As for OS architecture. I would recommend x86 architecture because it is easy to develop small games and it provide hardware compatibility and support 32-bits, approximately 4 GB memory, which is enough for this game. Nowadays, most personal computer comes with x64 architecture instead of x86, but x64 can run games that built on x86.
2. **Storage Management**: We have two options for storage management, we can either use Hardware Disk Drive (HDD) or Solid State Drive, which is most popular nowadays. SSD is fast and reliable compared to HDD, and new computers comes with SSD. It can boot up your machine very fast and quickly retrieve data. SSD is slightly expensive than HDD.
3. **Memory Management**: We have two options for memory management as well just like storage management. We can either use Static Random Access Memory (SRAM) or Dynamic Random

Access Memory (DRAM). DRAM must continuously refresh, and SRAM does not, which results in better performance and lower power usage, but it most more than DRAM.

1. **Distributed Systems and Networks**: I would consider using Client-Server design pattern to implement distributed systems and networks for this system. It is simple and efficient to handle multiple clients at the same time. This system uses common connections such as IP/TCP.
2. **Security**: Make sure to encapsulate sensitive data and choose secure database to keep user’s data safe. Abide by Security compliance and have penetration testing by third party to avoid getting hacked, which can result in loss of data and financial damage. Apply security patches regularly as viruses keep evolving.