

The Gaming Room Web-Based App Project: Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0  1.0 | 9/15/2023  10/15/2023 | Branden Boehnke  Branden Boehnke | Add required material for Project 1 Submission.  Complete final recommendations. |

**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 is looking to have software developed for their game application. The software must have the option for one or more teams with multiple individuals on each team. They also ask that the names for the game and teams be unique in order for users to check if a name is already in use. Finally, only one game instance can be present at a time.

## Requirements

The client needs the software to be web based as well as have options for multiple players on multiple teams. The client also requires only one game instance to be present at one time as well as each game having a unique name.

## [Design Constraints](#_2et92p0)

With the game being web-based, some constraints to think about would be the network latency, compatibility of different browsers, optimization, and the overall cost. The game could have network latency issues based on the network conditions of the players. With the multiple browsers that are available, not everyone uses the same one, so making sure that the game works on all browsers is needed. Focus on optimization helps to make sure that the game runs smoothly while not being too demanding. Lastly, overall cost would be a constraint as The Gaming Room could have a strict budget that causes some factors of the game to be prioritized over the other.

## [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)

The ProgramDriver Class is our focal point including our main() method. The SingletonTester has a testSingleton() method. The ProgramDriver and SingletonTester are connected with an arrow including “<<uses>>”, which refers to ProgramDriver using the Singleton Pattern and being tested by the SingletonTester. Next we have our Entity class containing the ID and name as private attributes as well as getters. This also includes a toString method. Our GameService class holds the responsibility of our games, players and teams. It uses getInstance() method which will make sure that there is only one instance of GameService at a time. Game class holds the information for each game such as the teams, game ID and name and the ability to add a team. Lastly the Player class holds the information for each player such has their ID and name.

**"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** | Macs typically are a bit more expensive which make them less likely to be used for hosting. This being said, since they are used less often, they are also more challenging to manage due to provider and support constraints. Macs also have constraints for scalability. | Linux is heavily used for server hosting based on its reliability, security and cost. Linux is open source which means that the OS is free to use. | Windows is used fairly often for server hosting due to its ease of use and compatibility. Though it is used often, Windows is fairly secure but not to the standard of Macs. | Mobile Devices have many advantages like mobility, but also have a lot of limitations such as its security concerns, life of its battery, and being dependent on a network connection. They are capable of hosting web-based games, but are designed more for smaller applications. |
| **Client Side** | Macs have high-quality hardware as well as a user-friendly UI, making them easier to use and maintain. Mac machines are typically more expensive than others, making them less budget friendly for clients. | Linux is free to use which makes it extremely cost effective for the client but can have a bit of a steep learning curve. | Hosting a server for Windows can come with licensing costs which could be a make or break for a client. Windows Security weaknesses could also steer client’s away from using the platform for development. | In this day in age, almost all of us have some type of mobile device, being either a smartphone, tablet, laptop, etc. This makes the cost and accessibility easy for the client. |
| **Development Tools** | Since Mac is a Unix-based system, it includes a Unix terminal, and languages such as Ruby and Python. Mac also has the ability to run containers which can make it easier for development and testing. | Linux works well with languages such as C++, Python, JavaScript, Java and a few others. It also works well with IDEs such as VS Code, Eclipse, PyCharm and others. | Windows supports well known languages such as C#, C++, JavaScript, Python, Java and others. It also has popular IDEs like VS, VS Code, Eclipse, PyCharm and others. | Mobile devices have a certain set of languages. For iOS there is Swift and Objective-C. Android has Java and Kotlin. As far as IDEs iOS has Xcode, Android has Android Studio, while they both have Visual 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**: Linux
2. **Operating Systems Architectures**: Linux offers plenty of support and attention from host providers to be a good candidate for Draw It or Lose It. Linux is used for server hosting often and is a reliable and friendly platform for both security and cost. With the platform being open source, it cuts down on client costs and supports both the language and IDE being used for the project.
3. **Storage Management**: MySQL would be a good storage management option as it is open source and works well with both Java and Linux. It allows for easy scalability changes for fluctuating popularity, stores players data (account info, achievements, progress, etc.), and can retrieve data quickly. MySQL also has systems in store for data recovery in case of issues such as lost data or corrupted info.
4. **Memory Management**: Virtual Memory is an option for Linux to access extra memory. Virtual Memory allows for developers to multitask and run more than one process at once while also securing memory information of one process to another.
5. **Distributed Systems and Networks**: Introducing APIs to the cross-platform construction would help with making sure there is smooth communication between the server and player while also using load balancing which allows connections to come in evenly. This would ensure that each player has the same experience and promotes fairness in the competitive game-ops.
6. **Security**: Implementing MySQL’s database encryption to guarantee that player data would not be compromised. Ensuring that secure APIs are in place is also important for player security such as implementing verification measures in order to access specific account data.