

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

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

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

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

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

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/12/23 | Calvin Tucker | Created documentation for the Draw it or Lose it game for the Gaming room company. |
| 1.1 | 11/26/23 | Calvin Tucker | Added evaluation of comparison between different platforms for server, client and development tools. |
| 1.2 | 12/10/23 | Ca;vin Tucker | Added recommendation for the server architecture to be built in. This recommendation is Linux. |

**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](#_35nkun2)

The Gaming Room wants a web-based game developed that serves multiple platforms based on their current game, Draw It or Lose It. The problem that the staff has is to set up the environment in which they need our help streamlining the development.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

Currently there are no technical requirements.

Software requirements:

* Game has the ability to have one or more teams involved.
* Each team has multiple players assigned to it.
* Game and team names have to be unique to allow users to check if the name is already in use.
* One instance of the game is allowed on the memory at any given time. Which can be done by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_1ksv4uv)

The design constraints for developing web based game applications is firstly compatibility, multiple platforms should be able to play on the game, that includes mobile, windows, mac and linux. Next is efficiency, because it is a web based game it has to be efficient because of possible networking capabilities and possible server costs. The game has to also be able to store users to avoid issues with turns and to avoid stuff like spam. Lastly the game has to be scalable since there are multiple players and teams and requires servers.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

GameService has a 0 to many connection to Game, Game has a 0 to many connection with Team, and Team has a 0 to many connection with Player. Game, Team and Player all inherit the Entity class. The ProgramDriver uses the SingletonTester class, possibly for its testSingleton function. They are used to fulfill the software requirements efficiently by cleaning up the code by sharing necessary functions without repeatedly creating the same functions over and over again and possibly setting up infrastructure for future additions to the game.

**"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](#_z337ya)

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’s operating system is based on Unix and supports all major scripting languages, it has good server tools and native support for import server, it is secure and cost effective compared to windows, its weaknesses are scalability, lack of flexibility, and requires specialized skills to deploy apps. | Linux is open source and uses Apache and Nginx as web servers, it runs on Perl, PHP, Python and Ruby, Its strengths are that it requires much less system requirements, it is fairly secure since less likely to receive cyber attacks, distros are all free to use and there community support. The weaknesses are Steep Learning curves, low corporate support and compatibility issues with windows. | Windows operating system uses Microsoft IIS which scripting languages are VBScript, and ASP.NET. The Strength of Windows is LTS, drivers, good technical solving, beginner friendly interface and habitual updates. The weaknesses are costs, Resource heavy, security and dependency on Microsoft stack. | Mobile Devices are small cellular devices or laptops that have a variety of platforms that they use. There isn’t much advantage to using a mobile device as a server unless it is required to move around a lot. The Weaknesses are Battery life, Bandwidth issues if connected via wifi, applications locked behind store, prone to overheating and parts not being interchangeable requiring a full purchase of a new device to upgrade. |
| **Client Side** | Mac requires a developer account that costs 100 dollars a year to publish on IOS Apps, Cost to also port apps from Windows / Linux to run natively on Mac. Requires Additional time because of having to use Swift/Objective-C although there are some programs that convert code. Time required Redesign for Mac’s UI and UX, also older versions can have compatibility issues. Expertises have to practically want to use Mac exclusively. | Linux is free of charge and easy to install. Linux is fairly bareboned when it comes to dependencies so it requires time to set up. Also time to understand the different distros. For Expertise, developers must understand the Linux terminal commands, behaviors between distros and possibly GUI. | Windows requires a windows dev account, and cost of certain developer tools and OS licenses for the OS. The time added is minimal because of how well made the documentation and dev environment is. Expertise needed is .Net framework XAML, UWP and Win32. Developers need to be used to Windows UI. | Cost for mobile platforms usually requires a developer account fee, to publish on app stores, cost for devices in order to simulate using them especially with different screens. The time it takes is significant to develop on these platforms because of the multiple compatibility and the vast visual screen resolution differences. App Store also has different approval rates and possibly could get declined. Expertise requires knowledge of swift, objective-c, java and Kotlin. Expertise must also be well versed in the mobile guidelines in order to avoid legal issues and mobile guideline failures leading to being disapproved in the app stores. Mobile games on a mobile browser is just not feasible as mobile often doesn't support games on a web browser, which makes it a better choice to release an application. |
| **Development Tools** | Mac uses mainly Swift and Objective-C but can use other programming languages such as Python and C. Mac IDEs are Xcode, Cocoa and Cocoa Touch. Mac doesn’t require a licencing cost for their IDEs | Linux uses Java, Python, C/C++ JavaScript/Node.js and PHP, The IDEs and Tools for Linux are GNU, Bash, Make, Eclipse, but also can use Visual Studio Code. Linux also doesn’t require a licensing cost. | Microsoft uses Programming languages such as C#, C/C++, Visual Basic, Javascript and Python and more. The IDE are popular IDE tools such as Windows SDK, .Net Framework, Visual Studio and Code, XAML, UMP and NuGet. Most Microsoft IDE’s do not require licensing costs, although there are paid features for advanced purposes. | There are plenty of languages that are used in mobile devices but for cross platform, there is React native, Flutter, and Xamarin. There are also IOS and Android simulators, allowing for testing on the computer, some requiring the phone to use. Also most mobile IDE’s do not require a licensing cost. |

## 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**: The operating platform that I recommend is Linux Operating Platform, the reason why is because it is very good for newer companies and much more customizable without requiring much investment.
2. **Operating Systems Architectures**: Linux has multiple distros all free of cost for different use cases, these distro all are different but each have better compatibility with certain devices. Having multiple compatibility is very important if releasing on multiple platforms.
3. **Storage Management**: Linux can use storage management such as SAN or NAS which can consolidate storage efficiently and securely across multiple Linux Servers.
4. **Memory Management**: Linux uses memory techniques which are useful for The Gaming Room’s Draw it or Lose it, with its capabilities of demand paging and swapping mapped files.
5. **Distributed Systems and Networks**: Linux has plenty of ways to synchronize through cross platforms, one possible way is through using web sockets, the networking could be solved by using cloud hosting because of its virtualization and resource management capabilities. These provide flexible and scalable but also cost-efficient cloud hosting environments.
6. **Security**: Linux is a platform that has advanced security with features such as firewall, Encryptions such as SSH, TLS, and SSL to prevent sensitive data leaking, Access controls which is useful for limiting what clients can access, Authentication, constant updates and hardening techniques which makes the servers harder to attack.