

<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 | 03/22/24 | Timothy Milne | Updated Executive Summary, Requirements, Design Constraints, System Architecture View, Domain Model, Evaluation, and Recommendations needed for this project |
| 1.1 | 4/6/2024 | Timothy Milne | Evaluated the characteristics, advantages, and weaknesses of various operating platforms |
| 1.2 | 4/18/2024 | Timothy Milne | Analyzed the characteristics of and techniques specific to various systems architectures |

**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 seeking to have their game “Draw It or Lose It”, which is currently an Android only game, created into a multiple platform web-based game. They are asking for help to streamline their development of the cross-platform design for this game. For the game, the ability to have one or more teams involved, each team will have multiple players, game and team names must be unique, and only one instance of the game can exist in memory.

## Requirements

The requirements for this game, based on the client’s intel, are:

* The game must have the ability to have one or more teams involved.
* Each team will have multiple players.
* Game and team names must be unique.
* Only one instance of the game can exist in memory.

## [Design Constraints](#_2et92p0)

The main constraint for this project would be the development of the game on multiple platforms. Having members on a team that are not familiar with the concept of cross platform design will require multiple teams to work on this project. In the end, this will require a larger budget.

## [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)

In the UML model, The Entity Class is shown to be the Parent Class of the Game, Team, and Player (child) classes. This shows that the child classes will inherit the attributes of Entity’s class while also being assigned their own personal attributes. The “GameSevice” Class is used to make sure that only one instance of the game is in use at a time with a unique name. This is to make sure that the client’s requirements are being met. “ProgramDriver” contains the “main()” statement and uses the “SingletonTester” 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 has the ability to run the MacOS, Windows, and Linuz apps. Mac is typically easy to operate. The weakness that appears for Mac would be the cost. They are very pricey.  For Project Two  Mac is less common compared to Linux or windows servers when it comes to hosting web applications, but can still be used.  Not as scalable or used as much as Linux  Typically high cost to use and might have costs associated with hardware | Linux is free to use and allows for customization of certain aspects of the OS. Linux is optimized for performance and is adaptable for “high traffic” applications. A weakness would be the difference in the easiness to use.  For Project Two  Linux uses different server-based deployment options such as Apache, Nginx, and/or Lighttpd. These servers can be made to host web applications and host the thousands of people desired.  Linux also has little to no licensing cost since it is open source | Windows is easy to use and is super compatible with Microsoft apps. A weakness would be that it does not have the same level of compatibility as other OS systems in terms of web-based applications.  For Project Two  Windows is sort of costly when it comes to licenses.  More commonly used for hosting web applications.  Can support thousands of players and run various web applications built on other frameworks. | Mobile Devices have advantages such as mobility, hardware specific for certain devices, and the ability to have internet anywhere with the use of mobile data, and offline use as well. Some weaknesses would be the screen size, battery life, and different security issues.  For Project Two  Mobile Devices require specific expertise in different coding languages.  Some costs required  More resources need for design  Different specs from other OS could make it more difficult to use for certain things |
| **Client Side** | The necessary software developments considerations for supporting multiple types of clients are:  -easiness of use  -testing and quality assurance  -compatibility and cross-platform development  -the design of user interface  For Project Two  Mac has a high cost to use.  Requires some knowledge of the OS to use.  Typically easy to use. | The necessary software developments considerations for supporting multiple types of clients are:  -Compatibility across various Linux distributions  -Utilizing different Linux development tools  -  For Project Two  Linux requires more knowledge in order to navigate.  Linux allows for compatibility across various Linux distributions. | The necessary software developments considerations for supporting multiple types of clients are:  -Having specific development tools for Windows  -compatibility with different Windows versions  -  For Project Two  Windows has unique tools that can be utilized.  Easy to use and has support to help. | The necessary software developments considerations for supporting multiple types of clients are:  -specific development tools for Mobile Devices  -The use of additional design resources  -optimized performance  -  For Project Two  Mobile devices are more accessible.  Sometimes can be harder to use for certain things.  Requires platform specific programming |
| **Development Tools** | MacOS uses the programming language Swift among others as well. The main tools would be Xcode, AppCode, Visual Studio Code, etc…  For Project Two  Programing languages consists of Swift, Objective-C  Development teams would need to be familiar with these languages and must have access to Mac hardware  Uses Xcode for macOS | Linux uses the programming language C/C++, Python, Java and more. The tools consist of Visual Sudio, Atom, and Eclipse. There are many tools in Linux, these are just some of the main ones.  For Project Two  Programming Languages consists of HTML, CSS, JavaScript and more.  Depending on the project, development teams would need experience with certain languages.  Many different tool, depends on what a team wants to use  Tools consists of Visual Studio Code, Sublime Text, Emacs, and more | Windows uses the programming language C/C++, Visual Basic .NET, Java and more. Some tools consist of Visual Studio, JetBrain, Windows Sdk, .NET Framwork, and more.  For Project Two  Programming languages consists of HTML, CSS, JavaScrip, and more  Development teams would need expertise in certain languages  Different tools to use such as Visual Studio, Sublime Text, Atom, and more | Mobile Devices use the programming language Swift, Objectice-C, Java.  Some tools consist of Xcode, Android Studio, IntelliJ IDEA, and others like Android and Flutter SDK.  For Project Two  Programming languages consists of Java, Swift, Kotlin, and more  Tools consists of Android Studio, Eclipse, Xcode, and more  Teams would need to be well versed in platform specific languages |

## 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 Windows as the Operating System because it is the most widely used. The Windows operating system is also a well-known operating system that many people have knowledge of.
2. **Operating Systems Architectures**: The windows operating platform architecture handles different types of applications, (Win32, POSIX, WSL, etc…). It supports 32-bit and 64-bit computing environments. These components provide a versatile environment that supports design and hardware.
3. **Storage Management**: Windows provides tools that help with storage management. These tools, like Disk Management, Disk Cleanup, and Storage Sense, help users manage storage efficiently. They help with formatting, resizing, and monitoring the disk volumes and space.
4. **Memory Management**: Windows has many different memory management tools. One being OneDrive, which allows the user to save/store versions of the game in case there were issues. Others being Azure Storage, Visual Studio and more.
5. **Distributed Systems and Networks**: Using cross-platform development will help limit the need to have multiple teams for different operating systems. I would suggest implementing fault tolerance mechanisms in case of outages and other issues. There would also need to be a strong server network that could support the client’s game and the users that are planning to play the game.
6. **Security**: Having different security protocols would be beneficial for this. I would recommend purchasing a specific security monitor for a secondary source so that the team can focus specifically on the product. Hiring an outside source would also allow the client to keep having security even after the product is finished and distributed. Although Windows has some built in security, more security wouldn’t hurt.