

“Draw it or Leave 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 | 05/26/2024 | Jimar Mims | Updated design template |

**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)

Our clients have a game that is only available on Android Operating Systems and are looking to expand to cross compatibility between Mac operating systems. For an expansion like this we, as a team need to develop their game using a Mac operating system to create the app for the apple store.

## Requirements

We are required to expand the app for use on multiple platforms. The technical requirements are related to the game itself. Requirements such as the game having the ability to host multiple teams, each team having multiple players, usernames and game names all must be unique, and only one instance of the game can exist at a time.

## [Design Constraints](#_2et92p0)

The game will cause for our company to utilize the MacOS to create this game for the apple store. The game will also rely on a stable internet connection, meaning we must strive to minimize network failure issues. Also, the game requires teams to communicate in real time. These constraints implicate the need for regular maintenance of the games servers and monitoring data being transmitted.

## [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)

gg

The UML diagram shows 7 classes simultaneously working together, utilizing object-oriented principles. The player, team, and game classes all inherit from the entity class which demonstrates inheritance. Encapsulation is utilized by the classes having private and public methods which keeps the internal moving parts behind the scenes while allowing access to the data. Also, we see the singleton pattern being used in the game service class which allows for only one instance of the game to exist. The principles, along with the singleton pattern, prove efficiency within the code because lines of code can be reused, and all the requirements are met.

**"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 is known for its Unix-based foundation which is stable and compatible with various server technologies. The advantage of MacOS is that it’s very user-friendly, secure, and reliable. But the cost and hardware restrictions do present a problem. | Linux is a great open-source operating system and is great for customization. The advantages of Linux are the cost effectiveness and an extensive software support. But the hardware compatibility and the learning curve Linux poses does raise concern. | Windows is also known for its user-friendly interface and wide software compatibility. The advantages to windows are the robust support and documentation on its products and ease of use. But windows do stand out amongst the most vulnerable when security is considered. | Mobile Devices are great for portability and touch interface. The advantages for using a mobile device to host web-based applications are the accessibility and having built in features. But mobile devices have limited performance and network limitations, which puts it last on my list. |
| **Client Side** | Supporting multiple clients on mac requires a great investment in terms of time and cost. It also takes an expert to deliver quality products when using MacOS. | Linux development benefits from free and open-source tools and the ongoing costs include hardware and maintenance. And the development cycle can be extended due to the need for comprehensive testing. | The developing for Windows can be resource-intensive and provides various avenues of support available, making it a worthwhile for projects. | Mobile development presents unique challenges due to the overwhelming options of devices. Developing mobile applications is also more time consuming. |
| **Development Tools** | The relevant programming languages available for mac are Python, JavaScript, and Ruby. | The relevant programming languages available for Linux are Python, Ruby, and Java. | The relevant programming languages available for Windows are C#, .NET, JavaScript, and Python. | The relevant programming languages available for mobile devices are Swift and React Native. |

## 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 that the company develop on a Mac Operating System. I choose this OS because the stability and the strong security features. It also incorporates an integrated development environment.
2. **Operating Systems Architectures**: the macOS architecture is designed to provide a secure and user-friendly environment for users. It also offers a great platform for developers to create various applications. It also handles great with the core functionalities of multitasking and memory management.
3. **Storage Management**: The FileVault is the leading storage management system that encrypts files and is a built-in feature with mac. This protects data from unauthorized use. Also, the APFS (Apple Filing System) is the default file system for macOS, optimized for SSD storage with features such as space sharing, snapshots, and fast directory sizing. Both works great and are optimized for speed and efficiency.
4. **Memory Management**: Mac uses virtual memory which would allow the game to run clean regardless of if the data becomes larger than the available ram. This is done by using disk space as extra “virtual” memory. MacOS efficiently allocates memory to applications and services based on demand, ensuring optimal performance.
5. **Distributed Systems and Networks**: This may be accomplished by introducing a client-server architecture which helps the game server act as a coordinator. This would allow for the server to manage games, players, and data. This would connect all parties over the network and allow for cross-platform functionality. Also we can utilize middleware technologies like Apache Kafka or RabbitMQ to manage communication between different platforms, ensuring smooth interaction and data streaming.
6. **Security**: The MacOS is the leading operating platform in terms of security. There is an immense amount of evidence proving this and MacOS doesn’t fall victim to as many cyberattacks compared to their competitors. MacOS applications are also sandboxed, restricting their access to system resources and user data, minimizing the risk of malicious behavior.