

Jamming Gaming

# **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 | 01/25/25 | Trevin Wadlingotn | Gaming software that meets the requirements of the consumer. |

**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 needs to set up an environment for their game. They are trying to develop a web-based version of the gaming app. The client has requested specific software requirements such as having one or more teams involved, Multiple players for each team, Unique identifiers and only one instance of the game at a time. We are going to develop the proper things behind this.

## Requirements

One or more teams

Multiple players for each team

Unique identifiers

Only one instance of the game at a time

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

## [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)

It depicts the relationships between different classes in this system. Entity is a class that has attributes and methods. Game, team and player extends entity. Gameservice is a singleton pattern with several attributes.

**"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** | Apple is a well-developed company. Server based application would be limited. Probably most expensive hardware. | Capabilities for server deployment. Open source, free and widely used for web based but not a common platform. | Fully capable. Most widely used tech ecosystem. Expensive license fees. | Scalability is high. Variations in pricing depending on which rout you take. A lot of open source options so costs are low. |
| **Client Side** | High costs for mac specific frameworks. Timeframe would be estimated at least 6 months. Must have macOS hardware compatibility. | Low costs, effective compatibility and good security. Open source with have less issues with web apps. Estimated shortest time to develop. | Moderate cost estimations. The user base is very wide but will have to use windows for development. Normal estimated time constraints. | Must be optimized for mobile screens and touch. Low cost with certain development paths. Between 3-6 months on average. |
| **Development Tools** | Primarily swift is apples preferred language. Also, can use React, electron and free opensource softwares. | Common languages include C++, python, java and bash. The most common IDEs are visual studio, eclipse and pycharm. | Windows most commonly uses C++, .NET and python. Java is also used. IDE is mainly visual studio. Free and open source | Most common languages include React Native, Flutter and Swift. Free Open-source options available. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

The goal of our software is to expand the gaming room application to multiple platforms. With a good number of well-know platforms to choose from, we look at them and decide upon the best one. We analyze system architecture, storage, memory, distribution across networks and security. Based upon our research we found these suggestions.

1. **Operating Platform**: Linux-based server would be most beneficial for us. We hit all our marks for cross platform use. Stability, security and compatibility with multiple platforms. Open source to support affordability too. Supports cloud based implementations, which is a huge bonus. Notable games like Minecraft and Counter Strike use this platform. Windows has a good structure but has licensing cost. Even though it may be best under different circumstances. MacOS is apple products dependent. It wont perform well for what we are striving for.
2. **Operating Systems Architectures**: Linux uses monolithic and kernel architecture. It provides for high performance. The kernel handles management for operations, memory, devices and more.
3. **Storage Management**: The most common file system used for Linux. Good performance and reliability.
4. **Memory Management**: The Linus system within the kernel has several features. Virtual Memory allows for larger apparent memory than physically exists. Paging divides memory into pages that can be swapped between RAM and swap space. The kernel itself manages memory space.
5. **Distributed Systems and Networks**: To ensure functionality over multiple platforms we can try a few things. Load Balancing to distribute traffic across multiple servers. Cloud is an option to help the availability. Restful APIs will allow the server and clients to communicate. Latency will be challenge to overcome if we have players globally. If it goes out we can use Failover mechanisms.
6. **Security**: Encrypted communication will protect data transfers. We will have our team regularly conduct pen tests. A multiple factor authentication will be put in place. RBAC will be in place to manage user permissions.