

Win, Lose, or Draw

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

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

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/14/2021 | Shane Griffith | Added 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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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** | A web-based platform for this game is a good idea. This will let all users regardless of what physical platform they are using to be able to play together for. So, whether the user is playing IOS, Android, Windows, Mac, or Linux they will all be able to join the same game at the same time. Which allows for less development time and implementation. | | | |
| **Client Side** | The web has an infinite reach. Games that are web based typically work well for all platforms. PC, Mac, Linux and mobile. The only requirement for the client regardless of operating system (in this scenario) would be an internet connection and access to a working web browser. | | | |
| **Development Tools** | Java when paired with JavaScript can be a very powerful setup for web-based gaming. They can also be paired with HTML5 to enhance the interactivity of the game. Based on this information, I might recommend the JetBrains integrated Development Environments both for JAVA and JavaScript. Both IDEs are free of charge for these specific languages. The limitation here however would be the skill of the developers on the team. They would need to be well verse in the tools and languages that are required for this project. Someone who has experience developing PID controllers in C++ might not be the best fit to work on a project in JavaScript/HTML5. So, this project would require a team that has the required experience to make the project as successful as possible for the customer, and the company itself. | | | |

## 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**: For a server software platform for the game to be ran on, I would still recommend the Amazon Web Services (AWS) platform. Overall, especially for a company just starting our who might not have a budget to build a server system on premise, AWS will offer the flexibility and performance at a good price for the customer. Not having to worry about the maintenance or upgrades would be a huge advantage for a company like The Gaming Room.
2. **Operating Systems Architectures**: The game should be running on a browser. This will allow development and scalability much easier. If the game is developed for a specific operating system, then different development teams will need to work on the different “versions” of the games. Whereas, if the game was developed for a universal browser, any device with internet access could theoretically have access to the game, thus extending the “reach” of the game tremendously.
3. **Storage Management**:

Again, here my recommendation would be to work through a Storage Area Network through AWS. To determine how much space would be required, an “inventory” would have to be taken to establish the right amount. Starting off with not enough storage here could be detrimental to the growth of the game. Especially upon launch. Knowing how much space is needed for X number of users, the game itself, and the database that ties everything together is to key to a successful launch.

1. Memory Management:

Taking the AWS system into account yet again here, the memory would all be managed remotely and not by the staff at the gaming room. However, as with the storage, it would be up to The Gaming Room to decide how much memory they would need access to for their system. This would be determined by how much memory one instance of the game takes up multiplied by how many instances The Gaming Room expects to have at launch plus some kind of buffer to hopefully help out in the event that traffic is higher than expected.

1. **Distributed Systems and Networks**:

A SQL database would be what I would recommend to-be shared amongst the player base. Currently, network game developers typically implement shared databases and the player-to-player communication from scratch. Artery, which is a system that is designed to support network game applications provides a high-level interface that takes advantage of semantics that helps optimize the performance of the network itself. This is a difficult process to implement. However, if correctly done the benefits it gives to online games would be worth it in the long run. Given the fact that the game will be a web application makes some of this much easier in the sense that as long as the device has internet access (even through mobile) it should be able to connect to a game. This tied with the fact that the game will be hosted and managed on an AWS system, makes downtime for the game or its users extremely unlikely.

**Security**:

Security should be the number 1 priority for companies who require to gather and store user information. Perhaps one day The Gaming Room wants to implement some sort of “Pay to Play” feature here with Win, Lose, or Draw. If that’s the case storing customer credit card information would be required. If something were to go wrong and someone gathered this information maliciously, it could mean the end of the company. Therefore, implementing some simple things can help prevent this from happening. First, patches and updates can help keep security protocols up to date. Another way to confirm everything is being done to prevent unwanted data leaks would be to practice “Least Privilege” within the company so that only the people who absolutely need access to this information has that access in the first place. This would also be another advantage of creating the web application so that it would be possible to only have one database stored whereas multiple would be required if the game was to be developed on multiple systems.