

# **CS 230 Project One**

# **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 | 09/21/2024 | Errol Foley | Inserting pictures with descriptions |

**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 purpose of this application is designed as a “Draw it or lose it” where it will be distributed across the web services. Current solution was to design a web-based game that can be accessed through any web browser that would provide portability to the user wherever they travel. UML diagram will be used to set forth for the design of the game.

## Requirements

Requirements will include devices that has internet easily accessible for the user where the user can make their character and play the game. A game will have the ability to have one or more teams involved. Some requirements as listed from the company…

* “Each team will have multiple players assigned to it.”
* “Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.”
* “Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.”

## [Design Constraints](#_2et92p0)

Some of the design constraints includes security to everyone’s information safe and sound. Compatibility allowing multiple platforms to play. Performance which allows the user to play smoothly. Scalability to make the player to feel more in depth. Finally, accessibility which will allow people to play the game in more locations. These constraints are important because these are the main issues that needs to be fixed to make an efficient game based on the requirements of the client.

## [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)

The UML model is designed to illustrate the domain model where the class diagram can be seen. Guess, Word, Drawing, Game, Entity and Player will be the six classes for the class diagram where each class will play a role in how the product will be. Guess will be what the drawings will be. Word represents the player skills to create which is typing into the drawing class where it has a one-to-many relationship. Game will be representing the game where the player is tied into this since the player is the one playing 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](#_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** | Secure, strong encryption, VPNs, storage space, and backups  Flexibility, adaptability, ease of use to make adjustments.  Advantage: room for upgrades options, can be used on different platforms  Disadvantages: web services are lacking and could be improved | Security, stability, community support, cost effectiveness, and ease of use  Friendly user, flexibility, cost appropriate, popular vote by users.  Advantage: Security is good, most used for webservices  Disadvantages: ease of use is harder, not very user friendly, hosting webservices | Windows admin center, better support, scalability functionality, data deduplication  More options for available software, competitor to OS, most dominate option, closed platform  Advantage: requires a lot of requirements, most comfortable platforms to use, loading times are low  Disadvantages: security is low and viruses are prone, customer/ tech support is lacking | Data control, quick load, remote management, application management, centralized updates  Mobility which is not ideal but still is an option, trackability, user friendly  Advantage: wide range of use, compatibility uses are above average, cost worth is valuable than others  Disadvantage: security is low and is easy to breach the software |
| **Client Side** | Built in apps, less support required, build quality, user friendly, and better integrations  Somewhat easy to learn and most people learn pretty fast, competitor to windows | Stability, easy to update, customization, compatibility, easy to install, license costs  Difficult to learn and requires more attention to teach people the software, pretty cheap in cost | Reduce server load, easier separation of concerns, eliminates driver mismatch | Cost saving, faster approval workflows, increased efficiency, client-side caching  Flexibility, portability, a bit more difficult to implement all devices |
| **Development Tools** | Compatibility, debugging, terminals, task management, and profiling tools  Able to run all programming languages, will pick the reliable option to run the program | Flexibility, software updates, small footprint, security  Can use multiple software, can use most programming languages | Programming languages, development tools, windows 11, easy of use, software compatibility  Somewhat better than Linux but is on par to the competitor | Improved customer loyalty, performance, cost savings, offline access, customer engagement  Limited amount of programming 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**: For this project I would recommend Windows since the platform is easy to learn and not too costly.
2. **Operating Systems Architectures**: Windows would be the operating system because it has a graphical user interface for resources.
3. **Storage Management**: Windows 11 has a good amount of storage which will help manage the files, save mapping of apps, location management. Not to mention the functionality of the cloud storage to make sure there’s a backup of files for more important usage.
4. **Memory Management**: There should be enough storage that will allow users to save information or photos for personal use. Creating a database for these requirements would help the project by allowing the user to have a more secure feature for their personal information.
5. **Distributed Systems and Networks**: The game should be allowed cross play which will give access to more users that are on different platforms. Testing should be done to test out the servers to see what it can handle and make necessary adjustments to allow the users to have a seamless game time.
6. **Security**: Built In protection and would allow protection for the users against viruses and malware.