

The Gaming Room

# **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 | <04/21/2024 | Zack Good | Adding information related to software design |

**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 has decided to develop a web-based game. At the moment the client’s game is only available on an android application. The application they are wanted to make is currently called: Draw it, or Lose It. The client is hoping to serve all platforms and not just android. The idea for the game was based off of a 1980s television game: Win, Lose, or Draw.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

For the Gaming Room: the design constraints are as follows:

The client requires the app to be web-based so that more people are able to access it. The game itself would consist of four rounds, one minute each. The game concept wants to revolve around the team members to guess part of the puzzle such as a phrase, title, etc. Drawings must be rendered at a steady rate and complete at the 30-second mark. If the team member doesn’t guess the answer before the time ends, then the remaining teams have 1 guess each to solve the puzzle with a 15 second time limit. A game will have the ability to have one or more teams involved. 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.

## [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 Entity class is essentially our handler for the following: Game, Team, and Player class. From the three they inherit characteristics of the Entity class. Of the four classes created: Game, Team, Player, and Game Service they all reference each other. The Program Driver class is how we create or build the project to meet the requirements of our clients. From the Program Driver class we can access all the classes we created, and execute them. With that in mind, there's also the Singleton Tester class. For each class that is organized, it allows the project to run according to the design constraints: it will allow for there to be more than one teams and players, while having one game session at the same time.

**"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 does have Mac OS X server available for use. Per Apple’s website, Mac OS X Server is only $20, so it would be inexpensive to implement. Mac, however, is not as popular as say Linux or Windows for performing these tasks. | Linux is interesting here as it has many distributions that have server capabilities. Linux Server would be low-cost, and open source (which provides a lot of resources). Not many users are savvy with Linux, so you would need someone who is familiar with Linux running the server. | Windows offers Windows Server. Looking at Microsoft’s website, it might be costly to implement, but it is fully functioning. Windows is likely the most used operating system, so finding users to operate Windows server would be a lot easier. | Given that mobile devices do not necessarily have the power that computers do, hosting a fully f ledged server on one may not be the best option compared to computers. Running servers on mobile devices is the most advantageous in terms of cost, as there is little to none to get one started. |
| **Client Side** | Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Mac would need less time and someone who does not have as much experience with Mac would need more time. | Cost would be low (if there even is a cost) with Linux, as it is open source. Maximum t ime and experience would be necessary, as Linux is not commonly used and you would need someone who is apt with Linux and allow them time to work, as Linux can be difficult even for someone with experience. | Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Windows would need less time and someone who does not have as much experience with Windows would need more t ime. | Cost would not be too much of an issue with Mobile devices. Experience may not be too much of an issue, as mobile devices can be easier to work with. More time would be needed, as there are multiple operating systems and multiple mobile devices that would need to be worked on. |
| **Development Tools** | Development tools for Mac may include Xcode and various programming languages like Swift and Objective-C. | Eclipse and Atom are commonly used IDEs on Linux. Eclipse is primarily used for Java, although it can support other languages like C+. | Eclipse and Visual Studio are popular IDEs for Windows. Visual Studio can be used for developing in HTML, C# and JavaScript among others. | Development for mobile devices typically involves platform-specific IDEs such as Android Studio and Xcode.  For iPhones, the development tools are like those for Mac, and iOS apps are typically written in Swift, though iOS and macOS are different. |

## 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**:  It is recommended that The Gaming Room initiates the project on the Windows OS due to its cost-effectiveness and the availability of numerous useful IDEs that can facilitate a quicker and smoother development process.
2. **Operating Systems Architectures**: Windows offers essential services for all Windows-based applications, enabling the creation of a Graphical User Interface (GUI) and accessing system resources, graphics, multimedia, messaging, and web services. These services can be accessed through user accounts or dedicated servers.
3. **Storage Management**: Windows allows for the easy addition of additional storage to the system, enhancing performance and data protection by merging extra disks into a storage pool using Storage Spaces. Utilizing cloud storage and optimizing existing storage capacity can further improve efficiency.
4. **Memory Management**:  Utilizing game engines for file generation can streamline the development process by leveraging pre-built libraries and resources, saving time and effort compared to manual coding. Organizing a database of libraries on the local system can aid in managing project files efficiently.
5. **Distributed Systems and Networks**: While starting with Windows-based development, it is essential to plan for cross-platform compatibility. Utilizing cross-platform game engines like Unity can support various operating systems and devices, ensuring a broader reach for the game. Selecting a robust server infrastructure is crucial to handle high user volumes effectively.
6. **Security**:  Windows offers built-in security features, but additional security software like McAfee or Norton Antivirus can provide enhanced protection against malware and viruses. Investing in reputable security programs is crucial when handling sensitive user data to safeguard against potential threats and breaches.

By adopting these recommendations, The Gaming Room can effectively manage and analyze the large volumes of consumer data acquired from the recent acquisition while ensuring scalability, security, and cost-efficiency.