

<Draw It or Lose 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 | 12/15/2024 | Gigi Cruz | Summary of client goals, update on design constrains, development breakdown and 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)

The game development company, The Gaming Room, wants to develop a web-based game that is similar to their current game, Draw It or lose it, which is currently only available on an Android app. Development for the game must be undergone by a new team as the current team is unclear on how to approach the development.

## 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)

One design constraint is that development stacks for Android operating platforms are not the same for web stacks. The lack of knowledge from the current staff could prove to be an obstacle.

Another design constraint is that Draw it Or Lose is a multiplayer games that allows teams. A game can only go through if the team and player names are unique. Multiplayer functionality adds complexity to the development of the game, and requires large data storage for player and team names. There is also a similar constraint on the large stock image library required for the games clues.

## [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)

ProgramDriver contains the main method and is associated with SingleTester to test if an instance of GameService already exist. This makes is so GameService can only have one instance of a game running at a time.

Entity is the parent class to the classes Game, Team, and Player. These three classes inherit from Entity, adapting Entity’s required attributes. This inheritance reduces repeated code, and allows more control over variables instances. Each Game can only have unique team names and each team can only have unique, individual players.

**"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 would make a decent server, but licensing maybe expensive. Also specific hardware that is required could be an added cost. | Linux developed well for webhosting. Popular option, as it free to use. | Windows maintains a secure server. Fairly simple to use. Can, however, be expensive in terms of licensing. | While possible, not advisable as this is not the mobile devices highest function. Issues will most likely occur with the rending of images. |
| **Client Side** | Mac falls on more of the expensive side as it not only requires ownership of a Mac computer, but requires a specific expertise in it. However, Mac can be user friendly. | Linux requires knowledge of python and may have an extend development time. | Windows may require a bit more expertise from all other options. | A focus on UI recommended for displaying web-based games on a mobile. |
| **Development Tools** | Coding is done in Swift | Linux has Python installed. | VSCode is a popular IDE allowing ability to code in any language. Most Windows programs are written in C++ and c#. | Android and Ios development can be down through Android Studio or Swift. |

## 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**: At first glance, I would recommend The Game Room use Linux, as it is free to use and is the most supportive of web-based development. However, cost aside, I would also recommend Windows operation system because of its wide user-base, its capabilities with gaming integration( windows has a built-in support for gaming-specific API’s, and facilitates efficient rendering and performance optimization) and its compatibility with many different development tools and frameworks. Cross-platform compatibility has also proven to be a bit more seamless when porting to other platforms like mobile devices and gaming systems.
2. **Operating Systems Architectures**: While Linux is stable and secure, and there is flexibility on what hardware you can use, Windows provides an efficient modular design through their utilization of microkernel architecture. This allows for effective memory management, resource allocation, and proper isolation between different processes. Windows server also supports large multi-threading capabilities which will be important for handling multiple player connections that is essential for accommodating a large number of users. Windows server also has comprehensive security features that include authentication, authorization, and encryption, which are essential in protecting player data and preventing unauthorized access. Lastly, windows servers have a well-developed networking stack that supports various network protocols, providing reliable communication between the client and server on different networks.
3. **Storage Management**: For storing game data, user-information, and other data types I would recommend using something like SQL Server or MySQL, which are relational database management systems. I would recommend this because they are great at storing and managing structured data, which in this case would be ideal for game-related information like player profiles, team names or tracking game scores. This management system also is great for scalability, enabling a solid handling of large volumes of data and remaining consistent when multiple user interact with the game at the same time.
4. **Memory Management**: Window Server uses paging and virtual memory to manage the games memory requirements. It ensure smooth gameplay by dynamically loading and unloading data a as needed, ultimately preventing excessive RAM usage and game crashes. Paging is a technique that separates the game’s memory into smaller “pages” that can be interchanged between RAM and hardware as needed, Virtual memory creates a virtual address space that appears much larger than the physical RAM, which allows the game to allocate memory without worrying about exceeding limits.
5. **Distributed Systems and Networks**: With Draw It or Lose it being required to communicate between various platform, a system could be designed with a central server and client applications that interact over a network, data exchange, API’s to handle game logic and user interface updates. The central server hosts the game logic and state, manage communication between clients, implement game rules, and handle player authentication. The client application for each platform would display the game interface and User Interface elements, send and receive updates from the central server, and provide user input like drawing actions, to the server. Considering connectivity, a reliable network connection is essential for real-time gameplay. UDP and TCP sockets can be used as they would be reliable for turn-based gaming with fast real-time updates. UDP is useful for fast data transmission and TCP is reliable in data delivery. Firewalls and network settings would also need to be configured to allow communication between server and client.
6. **Security**: To protect user information, I would recommend utilizing data encryption, authentication, and authorization, which are integrated in the Window operating system. Data Encryption protects sensitive user information like login and game data. Authentication and authorization verifies user identities and restricts access to sensitive resources.