

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 | 11/11/2024 | Nyzheir Warner | Finished Writing Executive Summary, Requirements, and Design Constraints |
| 1.1 | 11/12/2024 | Nyzheir Warner | Added description for UML Diagram under Domain Model section |
| 1.2 | 11/26/2024 | Nyzheir Warner | Completed Development Requirements Sheet |
| 1.3 | 12/12/2024 | Nyzheir Warner | Completed Recommendation Sheet |
|  |  |  |  |

## [Executive Summary](#_sbfa50wo7nsh)

The client The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw it or Lose it, which is currently only available on Android. By choosing to build a web-based version of Draw it or Lose it we can easily target multiple platforms including PC, Android and iPhone using a single build.

## Requirements

* 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.

## [Design Constraints](#_2et92p0)

* Shippable to multiple platforms.
* Network Latency and Bandwidth must be optimized to ensure seamless performance.
* Consistency and Synchronization are a must for an enjoyable multiplayer experience.
* Web based frameworks such as Node and/or Electron are required to target multiple platforms in one build
* Cloud database is required for Data Storage and Persistence of user progress across devices.
* Resource Management is crucial the app should be able to run on as many devices as possible.
* The UI should be a seamless and intuitive experience that translates to other platforms for consistence cross platform experiences.

## [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 diagram expresses multiple 0 to many relationships between GameService, Game, Team, and Player classes. This helps achieve the requirements of only one instance of the game existing in memory at any given time. A Player entity can only be assigned to one team at Time, a Team can only be assigned to one Game at a time, and a Game can only be assigned to one GameService. This is achieved using a 0 to many relationships. Additionally, it shows a Parent Child relationship between Entity Class and its three children Game, Team, and Player classes. This is done to avoid redundant code as well as ensure each Game, Team, and Player has a unique identify and name which is one of the program requirements.**

**"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** | Pro:  - History of Stability  - Easy to Use  Con:  - High Cost of Licenses  - Closed Source | Pro:  - Open Source  - Low Operating Cost  - History of Stability  - User Selected Updates  Con:  - Difficult to Use  - A more niche OS  - lower software compatibility  - User Selected Updates | Pro:  - Easy to Use  - Most used OS in the World  - Very Supported  Con:  - High Cost of Licenses  - Closed Source  - History of Low Stability  - Forced Updates | A mobile device should never be used to host a server. Not enough resources and is not intended use. Mobile OS would not allow such things most often. |
| **Client Side** | Higher Cost due to the proprietary IDE needed to develop a client for Mac Devices. Along with this we need a certified expert in the Mac OS Dev Suite. Across both IOS and OS X Apple accounts for 24.25% of OS market share worldwide. Targeting this platform seems costly but important. | Lower Development cost due to its open-source nature. Requires a higher level of expertise as Linux is one of the harder OS to work with having a very high learning curve. Linux accounts for 1.58% of global OS market share, making this less desirable OS to ship due to its low market impact. | Lower Development Cost because windows are the premier OS for most of the globe. Lower dev time as this is a tried-and-true OS meaning not as much research need to be done. No proprietary idea is needed to develop applications for this OS. Largest Desktop OS with 26.81 % market share making this a desirable platform to target. | Higher Cost of development because both apple and android mobile OS require proprietary IDEs to develop applications for mobile devices. With this higher level of expertise in these IDE’s and ecosystems is needed for Development.  Different approaches to user interaction are needed because most mobile devices are touch navigated. |
| **Development Tools** | - Xcode IDE  - Homebrew  - Swift Programing Language  - HTML  - CSS  - Node JS  - Electron  - JS | - JS  - HTML  - CSS  - VS Code  - MySQL  - Node JS  - Electron | - JS  - HTML  - CSS  - VS Code  - MySQL  - Node JS  - Electron | - Android Studio and Xcode IDE’s  - JS  - HTML  - CSS  - MySQL  - Node JS  - Electron |

## 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**: I recommend the use of Linux as the operating platform for the game room server. Due to its high level of stability, low operation cost as well as its open-source nature, I believe Linux meets all the required specs needed of the Game Room’s Application. Additionally, Linux offers much more system customizability compared to competing platform providers as well as being easily scalable using tools like Kubernetes and Apache Kafka.

1. **Operating Systems Architectures**: My recommended operating platform Linux is a UNIX-based operating system that follows a modular layered architecture. At the top is the Application Layer, which consists of user applications and programs that interact with the system. Below that is the Shell Layer, a command-line interface that allows users to interact with the system through commands. The Kernel Layer is the core of the operating system, responsible for managing hardware resources, system processes, memory. Beneath the kernel is the Utilities Layer which includes system libraries and utilities that provide essential functions to both the kernel and user applications. Finally, at the bottom is the Hardware Layer, which consists of the physical components that make up the system such as the CPU, GPU, and Memory. This layered structure allows Linux to be flexible, modular, and customizable, supporting a wide range of devices which makes this a great platform to build from.
2. **Storage Management**: In order to meet the performance requirements, I recommend using a cloud file storage system to store our files while using an SQL DB to hold tables of all of our valid file paths. By using this setup, we can keep the cost low while also assuring that we can operate at the appropriate speed needed for our game.
3. **Memory Management**: In order to effectively manage memory for our game I recommend we use techniques such as lazy loading to ensure a smooth transition between rounds while also freeing up memory from past images. Additionally, we can integrate caching into our application to save on data when trying to access frequently requested images. We should also make use of image compression techniques in order to deliver high quality images with small storage and memory footprints.
4. **Distributed Systems and Networks**: To achieve the goal of targeting multiple platforms I recommend the use of cloud services and API applications in order to communicate with clients on different platforms. Using a service like AWS will allow us to efficiently target multiple platforms while also being able to scale easily. Additionally, AWS boosts a 99.99% uptime making it one of the most reliable service providers for cloud computing software.
5. **Security**: In order to keep user data safe, I recommend we use services like Oath 2.0 in order to manage user sessions and access tokens. Additionally, it is important to tighten up API access by creating roles that have controlled access to only the necessary parts of the API needed to function, again this can be done by using OAuth and using encryption when saving data to our DB.