

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 | <05/23/2025> | Jose Espinoza | Initial draft includes Executive Summary, Requirements, Design Constraints, System Architecture View, Domain Model Explanation, Evaluation, and Recommendations based on the evaluation. |
| 2.0 | <06/8/2025> | Jose Espinoza | evaluated the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac and Windows) and mobile devices (iOS and Android). Also evaluated the possible server-side cost for each option and determined each platforms programming languages, tools, and IDEs as well needed developer experience. Cost for tools was determined |
| 3.0 | <06/22/2025> | Jose Espinoza | Updated the recommendations to be more specific for Operating Platform, Operating Systems Architecture, Storage Management, Memory Management, Distrusted Systems and Networks, and Security recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is seeking to develop a web-based and cross-platform game for their popular game, Draw It or Lose it, on the Android App Store. The main challenge involves ensuring only a single instance of the game runs in the memory and this single instance can support a scalable player base with multiple teams. The architecture will include a server-client model and the back-end will be cloud-based. A distributed system approach ensures centralized management and allows for various single instance game sessions, with these single sessions having high player count scalability. Game Instances, Players, and Teams will have unique identifiers following an Object-oriented java design that ensures scalability and maintenance.

## Requirements

**Business**

* Develop a cross-platform web application from the existing Android Application.
* Ensure only a single instance of the game exists in the memory with the instance supporting multiple teams and players.
* Minimal latency and a user-friendly interface for a smooth user experience on web.
* Protect data and privacy of the player (IP ADRESSES and sensitive information).

**Technical**

* Support Android and Website platforms
* Ensure Game Instance, Teams, and Players have unique identifiers.
* Optimize for large user-base and interactions (Game Sessions, Teams, Players and their actions)

## [Design Constraints](#_2et92p0)

* *Web-based Architecture: Internet Connection plays a heavy role in ensuring performance and latency between the players and the server are reliable.*
* *Memory Management: The Singleton pattern is used to ensure only one instance per game session is allowed.*
* *Distributed Environment: Ensuring optimized and efficient code is used within a distributed cloud-based back-end to reduce resource utilization of the Memory and CPU.*

## [System Architecture View](#_ilbxbyevv6b6)

* Server-Client Based Model
* Cloud-Based Environment such as AWS or Google Cloud
* Database

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram below represents the model for The Gaming Room that will be used for Draw It or Lose It. The Entity Class is an Abstract Class in which the Game, Team, and Player classes will inherit the attributes and methods within the Entity Class. However each class has its own unique attributes and methods as well, The Game class has unique attributes such as a List of teams and unique methods such as adding teams to the game, The Team Class has a method to add players to the team, and The Player Class has a method to show the player id and name. The Game Service class uses a Singleton Pattern ensuring only a single instance manages the games through a list of the games, generating game, team, and player ids. Aggregation is represented as GameService has 0… Games has 0… Teams has 0… 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is typically not used to host web-apps due to the cost of hosting on a mac server. | Linux is the most popular for databases and server hosting due to the open-source, stable, secure, customizable, and low cost in the cloud and enterprise environment. | Windows is resource intensive and costly due to it being license based software. Normally used in Windows-based enterprise companies such as airlines due to it offering powerful tools and integration with .NET. | Not used for primary server platform due to limited hardware capabilities. |
| **Client Side** | Developing native clients for macOS requires Apple hardware and expensive licenses. Swift code is mainly used for this as well and would be restricted to the macOS environment tools which require monthly or yearly licenses. | Linux client side supports Java or web-based development. Linux client side is open-source and affordable. | Windows client side has a large user base, and the cost is low on the client side compared to server-side. Development is supported in Java and using .NET. | Requires React Native cross-platform development tools or native app develop tools. Xcode and swift code for iOS and Kotlin or Java for Android development. Can be expensive especially when choosing to build a native application. |
| **Development Tools** | IDES: Xcode (native macOS development), IntelliJ IDEA (JetBrains), Visual Studio  Code Languages:  Swift, Java, Kotlin, Python, JavaScript. | IDES: Eclipse, IntelliJ IDEA (JetBrains), Visual Studio  Code Languages:  Java, Python, JavaScript, Node.js | IDES: Eclipse, IntelliJ IDEA (JetBrains), Visual Studio  Code Languages:  Java, Python, JavaScript | Android Studio, Xcode, React Native  Mainly SDKs and emulators for testing on mobile devices. |

**Software Design Document**

**Server-side:**

**The need for the creation of a web-based application with a server-style configuration for hosting The Gaming Room’s website with the ability to scale up to thousands of players means that our choice of the operating system (Linux, Mac, Windows, or Mobile[iOS and Android] ) will be determined by various factor such as the advantages, weaknesses, cost, and server-based development.**

**Linux’s advantages include scalability, security, wide support, and the support for Docker/Kubernetes containers. The disadvantage of Linux is that there is a large learning curve. Linux offers server-based deployment method (such as AWS, Azure, and Google Cloud) where the website will be hosted. The potential licensing costs to the client The Gaming Room for the server operating system licensing is free keeping the cost low on Linux with the cost of hardware being around $1500.**

**Mac’s advantages are that this is an ideal environment for development and testing of macOS and iOS applications. The disadvantages are the lack of compatibility, scalability, and the high cost needed for server hosting. Mac server-based deployment method macOS is no longer sold, The potential licensing costs to the client The Gaming Room, for the server operating system was $500-$1000 the hardware could range from $4000-$8000.**

**Window’s advantages include the support for .NET and the support for enterprise level web hosting tools. the disadvantages include the cost, resource intensive nature, and low performance compared to Linux for cloud based. Windows offers server-based deployment methods such as IIS and .NET where the website will be hosted. The potential licensing costs to the client The Gaming Room, for the server operating system is roughly $400-$500 per month when the architecture and code is optimized but the cost could scale up to $1000+ when the architecture and code isn’t optimized.**

**Mobile devices (iOS or Android) aren’t used server-side due to limited hardware and only mainly host a single user**

**Client-Side:**

**To make sure that the application is compatible with all web browsers and mobile devices the development process requires the use of standards, tools, and practices such as the use of the HTML, CSS, and JavaScript frameworks React.js which allow developers to build applications that are responsive to the user’s screen size or type such as mobile or web. For the support of all browsers and outdated versions cross-browser compatibility tools can be used such as the autoprefixer css tool that reads the CSS file and changes the current prefixes to the prefixes needed in the user’s outdated web browser other tools for outdated browser support include the JavaScript Babel and polyfill tools which allow older JavaScript versions to support modern JavaScript features on users using browsers with outdated JavaScript versions. This saves developers from having to make interfaces for each platform and instead use the single codebases that include (HTML5, CSS, JavaScript (React.js)) for development on all web and mobile platforms bringing the time needed developing and the cost down drastically. React Native can be used to develop applications for iOS and Android that can interact with the back-end architecture aswell.**

**Development Tools**

**Linux:**

**programming languages include Java (Sprint Boot), JavaScript (Node.js), HTML5, CSS, and python. tools (IDEs and other tools) Include JetBrains’s IDEs (IntelliJ IDEA), Eclipse, VS Code, Docker which are used to build this type of software at low cost. Backend developer with Linux, API, and cloud are required to do these technical requirements. Front-end developers with Html, CSS and JavaScript experience are also needed.**

**Windows:**

**programming languages include Java, JavaScript, C#, Python, HTML5, CSS. Tools (IDEs and other tools) that are used to build this type of software include Visual Studio, Eclipse, Include JetBrains’s IDEs (IntelliJ IDE), and VS CODE. Developers experienced with Windows and .NET integration as well as Front-end developers with Html, CSS and JavaScript experience are also needed.**

**Mac:**

**programming languages include Swift (native development), Objective-C, Java, JavaScript, HTML5, CSS, and Python. Tools (IDEs and other tools) that are used to build this type of software are Xcode, JetBrains’s IDEs (IntelliJ IDE), and VS CODE, as well as Android Studio. Developers need to have experience with the iOS development platforms, Swift code, and Apple developer accounts/resources.**

**Mobile:**

**programming languages IOS (include Swift, Objective-C ) Android (Kotlin and Java). The tools (IDEs and other tools) that are used to build this type of software include React native for cross-platform Android and iOS development simultaneously. Android Studio, Xcode, and VS code can be used for cross-platform development between both mobile devices. Mobile developers need to be experienced in native Swift, Kotlin, and cross-platform React Native.**

* + **Are there licensing costs related to the development tools?**

The licensing cost for JetBrains IDE’s (IntelliJ) or the other IDEs can range from $250 per user for only IntelliJ to $700+ for access to all IntelliJ IDES per user. React Native is free and could be developed using VS Code. Developing with Xcode for Mac and iOS requires a yearly $99/year developer account but the IDE is free. Android Studio is free and the Docker Linux tools are free.

## Recommendations

1. **Operating Platform**: The recommended OS for The Gaming Room’s server-side development would be Linux due to the opportunity for scalability at low-cost. The Linux environment also offers reliable DevOps and production environments. It being open source has made it widely supported and can integrate with other needed development tools (Docker and Node.js).
2. **Operating Systems Architectures:** The OS Architecture that will be used is 64-bit because it is widely supported, the industry standard, and compatible while offering a wide variety of tools and libraries while maintaining a reliable performance client and server-side. This architecture is supports low resource consumption for Android and Web as well. This architecture handles high volume of concurrent users and memory consumption.
3. **Storage Management**: An appropriate storage management system that will be used with Linux will involve a combination of local storage in the server and cloud-based storage for scalability and performance while maintaining a low cost due to the expense of cloud-based environments such as AWS and google cloud which will handle backups, static content, and distributed access. Distributed storage solutions like LVM will be used to manage the Linux Servers by dynamically managing disk space improving flexibility. For the file system ext4 or XFS will be used because both are high performance and reliable on the Linux operating system which is the industry standard. Local storage will be the main storage used on the primary server for increased performance during live gameplay.

**Memory Management**: Memory management techniques to optimize performance on Linux include paging and swapping which manage memory when there are heavy loads. Processes are temporarily moved to the disk to stabilize the memory. Virtual Memory Abstraction ensures processes operate in their own memory space which enhance security and isolation. Linux’s kernel also has an Out-of-memory killer that will activate once it detects that a process uses too much memory and creates a threshold that triggers this. This prevents the server from overloading and crashing. This ensures server stability by terminating processes which aren’t critical when the threshold is exceeded which helps to increase performance and prevent a crash from occurring even during high concurrent user count and large amounts of user interaction.

1. **Distributed Systems and Networks**: For communication between various platforms in real-time Web Sockets will be needed enabling communication between the Linux servers and mobile/web clients which enables real-time multiplayer gameplay. Web Socket servers such as Node.js will be used along with the Docker tool which offer modularity and scalability in the back end which contribute to the multiplayer gameplay experience in real-time. Load balancing will be used to distribute traffic between the server and clients in case of outages. Load balancing allows traffic to not only target one server but evenly distribute the traffic through all the servers based off the relative location to someone then the server load. If a server is taking too much traffic the load will distribute to other servers. This reduces latency as well as helps in the case of an outage on one of the servers.
2. **Security**: To protect user information on and between various platforms data will be encrypted in rest and while in transit with the use of AES-256 at rest and HTTPS / SSL/TLS protocols when the data is in transit. User authentication tools supported on Linux such as OAuth 2.0 and JSON Web Tokens will be used to ensure the actual user is accessing our network and not bots or suspicious actors. Linux’s build-in security tools like SELinux and AppArmor will secure the servers aswell as Firewall rules and Role-Based Access Control will be used to restrict access to the backend for unwanted users. IDS will be used for security audits that help responding to any security threats.