

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 | 05/25/2025 | Dylan Harmon | This revision includes additions to the executive summary, design constraints, and domain model portions of the document. |
| 1.0 | 6/8/2025 | Dylan Harmon | Evaluating potential new platforms. |
| 1.0 | 6/17/2025 | Dylan Harmon | Final 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)

“Draw it or Lose it”, designed for The Gaming Room, is an exciting game that merges competitive drawing with real-time teamwork and guessing. The primary objective of this game is to create an engaging multiplayer experience where teams compete to decipher images as they slowly render. Each game consists of 4 rounds, with team racing against the clock and each other to guess the image.

## Requirements

The game will be implemented as a web application, compatible with major web browsers allowing for quick response times for real-time interactions. All user data must be securely handled, adhering to regulations. The user interface must be intuitive and user-friendly, ensuring that players can easily navigate and interact with “Draw It or Lose It”. Player and team names require unique identifiers. The game must only have one active instance in memory at any given time. The design needs to be scalable to ensure the anility to handle the number of users as the player base grows.

## [Design Constraints](#_2et92p0)

* Technology
  + Web-Based platform
  + Cross-Browser compatibility
* Performance
  + Real-time interaction
  + Able to handle concurrent users
* Security
  + Data protection
* Scalability
  + Limit to Single Instance
  + Resource Allocation for multiple players and teams
* Usability
  + Unique Identifiers for usernames and team names

## [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 class diagram displays inheritance by having the Game, Team, and Player classes inherit from the Entity class, allowing them to share common attributes and methods. Composition is displayed in the class diagram where the Game class contains a list of Team objects, the Team class contains a list of Player class objects, and the GameServices class contains a list of Game class objects. In order to achieve a single instance, the Singleton Pattern is used within the GameService class ensuring only one instance exists.

**"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 offers a versatile development environment that supports cross-platform development. Macs offer limited hardware (in terms of customization (hardware)), the initial cost of Macs are generally more expensive as is licensing for developing for Mac (iPhone and iPad apps/ app store licensing). As for developing on Mac, there can be a steep learning curve for those not accustomed to developing for MacOS. | Linex provides a robust and flexible environment for server-side development. There is extensive support for various languages and frameworks allowing for scalability. Linux provides great performance, security and compatibility with server environments. There is also a learning curve which may require additional time and effort. | Windows offers advantages for integrating into the Microsoft ecosystem such as integration with Azure and more. Windows offers robust development tools and offers support for .NET and other languages to enhance development efficiency. However, similar to Mac, windows requires more system resources and can increase cost with licensing fees, and is not as flexible as Linux may be. | Complex platform fragmentation and app store requirements offer challenges. However developing for mobile devices offers upside in reach and offers the ability to offer dynamic, scalable and personalized experiences to a vast number of users. |
| **Client Side** | Xcode offers a free development environment, however may raise in cost due to the integration for third-party tools. Sufficient time must be allocated for our developers to possess the technical skill required to create an application that aligns with macOS standards. | Offers cost advantages due to its open source nature, but requires careful consideration in time and expertise of developing with Linux. However developers should be able to use the languages they are comfortable in. | Licensing and infrastructure costs may be higher development time may be shortened with our current development team being accustomed to working with Windows. | Can be costly and time consuming with fees for using the app store, and google play store. Requires technical expertise in platform specific skills and UI/UX design (we may have this due to already developing for android.) Has potential to reach more users. |
| **Development Tools** | Languages: Swift, Objective-C, C/C++,Python, Java.  IDE’s: Xcode, AppCode(Jetbrains)  Potential Tools: Homebrew, Interface Builder (part of Xcode) | Languages: C/C++,Python, Java, Ruby, JavaScript.  IDEs: Eclipse, PyCharm, Visual Studio Code.  Potential Tools: GCC, Docker | Languages: C#,C/C++, Visual Basic, Java, Python, JavaScript, Shell  IDEs: Visual Studio, VS Code, JetBrains, Eclipse, PyCharm.  Potential Tools:.NET, MSBuild, Docker | Languages: Swift, Objective-C, Java,JavaScript  IDEs:Xcode, Android Studio,Eclipse  Other Tools: React,Flutter |

## 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 Gaming Room take advantage of a cloud based serverless architecture. I would also recommend the serverless architecture with Linux as the underlying operating system. The serverless architecture fits the project requirements allowing for auto scaling, where “Serverless functions automatically scale in response to demand” (Chouhan, 2024). Serverless promotes scalable projects with less maintenance and overhead which can lead to “significant cost savings, especially for applications with unpredictable or infrequent workloads” (Chouhan, 2024).

The long-term benefits of using serverless is the reduction in monthly operational costs, increased developer productivity by allowing developers to focus on building features and functionality rather than server architecture, and increased reliability by scaling server resources “dynamically in response to changes in demand” Elliott (2022).

AWS Lambda, API Web Sockets, and DynamoDB offers a serverless solution which can build on the current Draw It Or Lose It environment. Android, similar to AWS Lambda, utilizes the Linux Kernal. This allows dev teams to continue developing in languages they are familiar with and allows The Gaming Room to integrate Draw It Or Lose It to serverless. In addition to the android application, a web browser for Draw It Or Lose It will act as a gateway and handle requests to the game endpoints using HTTPS (WebSockets).

1. **Operating Systems Architectures**:

Draw It Or Lose It requires “player to player” and “service to player” communication (Building a Serverless Multi-player Game That Scales | Amazon Web Services, 2021). API Gateway WebSockets provides communication and enables AWS Lambda functions, and when needed the Lambda function will send messages to players. These Lamba Functions will be written to accommodate the needs of Draw It Or Lose it. Lambda Functions will also be created to work with DynamoDB for game connections, player profiles etc. This leads us to storage management.

1. **Storage Management**:

DynamoDB is a fully managed, secure and scalable storage management option. Auto-scaling can be scheduled and offers a cost-effective setup for Draw It Or Lose It. User growth across multiple regions can be managed effectively with *Global Tables* that will “ensure data is close to users located in various locations, leading to a reduction in latency” (Eisenberg, 2024). In addition to cost efficiency, scalability, and reduction in overhead features that a serverless storage management like DynamoDB offers is its “constant stream of new features, and support from the AWS ecosystem” (Eisenberg, 2024).

1. **Memory Management**:

AWS Lambda applies settings to memory configuration (i.e. between 128 MB and 10,240 MB) to allocate CPU Power. Higher settings → more allocated cores → higher cost. In many cases this can lead to memory over-allocation which “increases node utilization and waste, and can even cause cold-start and latency issues” (Optimizing Memory Allocation in a Serverless Architecture Through Function Scheduling, 2023). Serverless offers many benefits, in order to properly manage memory and reduce latency issues within this framework to benefit players, I suggest incorporating in-memory caching within the function. Faa$T presents a caching method to increase performance and further reduce Lambda costs. “Each application gets its own Faa$T cache. After a function executes and the application becomes inactive, the cache is unloaded from memory with the application. Upon reloading for the next invocation, Faa$T pre-warms the cache with objects likely to be accessed.” -for example, and image for the next round- ”In addition to traditional compute-based scaling, Faa$T scales based on working set and object sizes to manage cache space and I/O bandwidth.” (Romero et al., 2021).

In addition to auto-caching methods AWS offers two approaches to lower cost while increasing execution times in order to fine tune memory allocation. AWS Lambda Power Tuning is an open-source tool than can help find optimal memory configurations for Draw It Or Lose It (Understanding Techniques to Reduce AWS Lambda Costs in Serverless Applications | Amazon Web Services, 2023).

1. **Distributed Systems and Networks**:

As the application scales to support growth there will be a need for a distributed system (i.e. running the application across multiple servers). As mentioned in **Storage Management**, Global Tables or *Global Load Balancing* is a “technique used to distribute traffic across servers or data centers located in multiple geographic regions”. By routing users to less congested data centers we can reduce latency and load times, and scale across multiple regions GeeksforGeeks (2024). The API Gateway, which provides support in player-player and service-player communication, provides built-in load balancing support.

1. **Security**:

The user authentication system should provide a good user experience, easy to register, sign-in and reset passwords, while providing security and protection to player information. Social integrations (i.e. google, Facebook, twitter/X) can help in making the process easier for players. OAuth 2.0 “*is an authorization framework that allows users to access resources from an API without giving the API access to their credentials, such as username and password … The client requests authorization from the resource owner to access resources on the resource server, which is authenticated by the authorization server. If authorization is granted, the client receives an access token that it can use to access resources … It offers security advantages … including the ability to revoke access to a specific user, and the ability to delegate authentication to a trusted third party*” (API Authentication Vs. Authorization: Methods & Best Practices, 2025). In addition to OAuth 2.0, OpenID Connect (OICD) “*is an authentication protocol that extends the OAuth 2.0 framework by providing an identity layer on top of it. OICD enables users to authenticate with a web application using an identity provider”* *(API Authentication Vs. Authorization: Methods & Best Practices, 2025)* or “social integration” such as Facebook, google, etc.

Although Draw It Or Lose It does not currently propose any in-game purchases, if this changes in the future, I suggest using a 2FA (two-factor authentication) to further protect users.

**Citations:**

API Authentication vs. Authorization: Methods & Best Practices. (2025, March 25). Frontegg. <https://frontegg.com/guides/api-authentication-api-authorization>

Building a serverless multi-player game that scales | Amazon Web Services. (2021, March 2). Amazon Web Services. <https://aws.amazon.com/blogs/compute/building-a-serverless-multiplayer-game-that-scales/>

Eisenberg, J. (2024, August 25). AWS DynamoDB: Everything you need to know. N2W Software. <https://n2ws.com/blog/aws-cloud/amazon-dynamodb-latest>

Elliott, E. (2022, November 22). What’s up with Serverless? - JavaScript scene - Medium. Medium. <https://medium.com/javascript-scene/whats-up-with-serverless-1b9bfa80f21f>

GeeksforGeeks. (2024, September 26). Global Load Balancing vs. Local Load Balancing. GeeksforGeeks. <https://www.geeksforgeeks.org/system-design/global-load-balancing-vs-local-load-balancing/>

Optimizing Memory Allocation in a Serverless Architecture through Function Scheduling. (2023, May 1). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/10181224/figures#figures>

Romero, F., Chaudhry, G. I., Goiri, Í., Gopa, P., Batum, P., Yadwadkar, N. J., Fonseca, R., Kozyrakis, C., & Bianchini, R. (2021). Faa$T. Faa$T: A Transparent Auto-Scaling Cache for Serverless Applications. <https://doi.org/10.1145/3472883.3486974>

Understanding techniques to reduce AWS Lambda costs in serverless applications | Amazon Web Services. (2023, May 8). Amazon Web Services. <https://aws.amazon.com/blogs/compute/understanding-techniques-to-reduce-aws-lambda-costs-in-serverless-applications/>