

The Gaming Room

# **CS 230 Project Software Design**

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 | 01/25/2025 | Sharif Ayesh | Added initial content, including software requirements, design constraints, evaluation, and recommendations for platform and architecture. |

**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 aims to expand its current Android-only game, "Draw It or Lose It," into a web-based, multi-platform application. This transition seeks to enhance accessibility and broaden the user base while preserving the engaging gameplay. The proposed solution involves developing a distributed, web-based system that supports multi-team gameplay, ensures unique identifiers for teams and players, and allows only one instance of the game to run in memory at a time. This design will address critical requirements such as scalability, synchronization, and efficient resource management, ensuring that the game delivers a seamless user experience across platforms.

## Requirements

The client’s primary business requirement is to develop a cross-platform game application accessible through web browsers while retaining the core gameplay of "Draw It or Lose It." The technical requirements include ensuring that team and player names are unique, allowing only one instance of the game in memory, and supporting multi-team functionality with multiple players assigned to each team. Additionally, the game must run efficiently in a distributed environment to ensure a responsive and enjoyable user experience.

## [Design Constraints](#_2et92p0)

* The application must be designed for a distributed web environment to support multiple platforms effectively.
* Real-time synchronization is required to accommodate team-based gameplay and ensure smooth user experiences.
* Low latency is critical for gameplay updates and interactions, particularly during timed rounds.
* A solid backend could use Node.js with WebSockets for real-time interactions, PostgreSQL for structured data, and Redis for caching, with AWS Lambda or Firebase ensuring scalability and reliability.
* The architecture must support scalability to accommodate an increasing number of players and teams without performance degradation.
* Reliable network connectivity is essential to minimize disruptions and ensure consistent gameplay for all users.

## [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 development of the game in a distributed web environment presents several constraints. The application must ensure real-time synchronization to accommodate team-based gameplay while maintaining low latency. Additionally, a robust backend is necessary to manage player data, team configurations, and unique ID generation for teams and players. These constraints imply a need for scalable architecture and reliable network connectivity to minimize delays or disruptions in gameplay.

The chosen design must also account for potential scalability challenges as the user base grows and more teams and players engage with the application.

**"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 provides a reliable environment for web hosting but may incur higher costs due to proprietary hardware and software. It supports web-based applications well but is less common for large-scale hosting compared to Linux. | Linux is ideal for hosting due to its open-source nature, cost-efficiency, and scalability. It offers robust support for web-based distributed environments. | Windows offers good compatibility and ease of use but may require higher licensing fees. It’s suitable for smaller-scale web hosting. | Mobile devices cannot directly serve as servers but can act as clients accessing the application hosted on a server. |
| **Client Side** | Development for Mac clients requires knowledge of macOS-specific development tools like Xcode. It can be time-intensive but ensures compatibility with Apple devices. | Linux clients are less common among end users. Development may require additional effort to ensure compatibility with Linux desktop environments. | Windows is widely used, making it essential to prioritize compatibility with this platform. Development is straightforward due to the availability of tools like Visual Studio. | Mobile client development should focus on responsive design and compatibility with Android and iOS devices. Frameworks like React Native can streamline this process. |
| **Development Tools** | Tools such as Xcode and Swift are used for macOS development. For web applications, frameworks like React and Node.js are also applicable. | Tools like Node.js, Python, and Apache are commonly used for Linux development. IDEs such as PyCharm or Eclipse are helpful. | Development tools such as Visual Studio and .NET are effective for Windows. JavaScript frameworks like React are also compatible. | For mobile devices, React Native or Flutter can be used for cross-platform development. Android Studio is suitable for Android-specific development. |

**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**: The recommended operating platform for expanding *Draw It or Lose It* is Windows. Windows Server provides a robust and user-friendly environment for hosting web-based applications. Its compatibility with .NET technologies ensures seamless development and deployment, making it an ideal choice for creating scalable, distributed applications. Windows is widely used, which simplifies integration with existing infrastructure and tools.
2. **Operating Systems Architectures**: A Windows-based architecture using a monolithic or microservices approach is recommended. The microservices model, supported by Windows Server and Microsoft IIS, allows the application to be divided into smaller, independently deployable services, such as user authentication, game logic, and data management. This structure enhances scalability and simplifies maintenance
3. **Storage Management**: Windows Server integrates seamlessly with Microsoft SQL Server for efficient database management. SQL Server provides robust storage solutions, including high availability and fault tolerance for critical game data such as player profiles, team configurations, and game state. Additionally, Azure File Storage can be utilized for external storage needs, offering reliable access to assets and backups.
4. **Memory Management**: Windows provides effective memory management tools through the .NET runtime and Windows Task Scheduler. Additionally, caching solutions like Azure Cache for Redis can be integrated to enhance performance by storing frequently accessed game data in memory, reducing load times and improving responsiveness.
5. **Distributed Systems and Networks**: Windows Server supports distributed systems with tools like Microsoft Load Balancer to handle traffic across servers, ensuring high availability and performance. For real-time updates and communication between platforms, SignalR (built into .NET) can be implemented to provide a seamless gaming experience. Failover mechanisms and redundancy can be configured to maintain connectivity and system stability during outages.
6. **Security**: Windows offers strong security features, including built-in firewalls and encryption protocols. Role-based access control (RBAC) through Active Directory ensures secure user authentication and permissions management. HTTPS and encrypted WebSocket protocols can be used to protect data transmission. Additionally, regular updates and security patches provided by Microsoft enhance the platform’s resilience against cyber threats.