

# **The Gaming Room**

# **CS 230 Project Software Design Template**

Version 2.0

## 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 | 01/20/2025 | Johnathan Dumke | I made updates to the cover page. I also made updates to the document revision history. And finally changes to executive summary, design constraints, system architecture overview, domain model, and recommendations. |
| 2.0 | 2/4/2025 | Johnathan Dumke | Make changes to evaluation. |
| 3.0 | 2/17/2025 | Johnathan Dumke | Make changes to 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 Gaming Room project is a web based application that is based on the current game draw it or lose it. It is currently only available to Android users. The purpose of this game is it will render images from a big library and the game will have four rounds. Each round lasting one minute. Each drawing will render at the 30 second mark. And the team must answer within that time. If not answered the opposite team has 15 seconds to answer.

## Requirements

The Game Room Project requires business and technical requirements to ensure a seamless and scalable web-based application experience:

* Business requirements
  + Want to extend the application to IOS and Desktop.
  + Be user friendly and accessible.
  + Be able to support lots of users

Technical Requirements

* + Use development tools to ensure it works across all platforms
  + Make sure the application is secure and data is encrypted.
  + Use architectural patterns like the Singleton Pattern to enforce only one game instance in memory at a time.
  + Unique Team and Player Names

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

* **Needs to run on multiple platforms** (This requires choosing development tools and frameworks that support cross-platform compatibility, such as using web technologies (HTML, CSS, JavaScript.)
* **Each team will need to have multiple players** (Developers need to implement logic to check and enforce these limits when players are added to a team.)
* **Only one instance of the game should exist in memory at a time.** (This constraint necessitates the use of design patterns such as the Singleton Pattern to ensure only one instance of critical components, like the GameService, exists.)
* **Unique Naming for Teams and Players** (Requires efficient algorithms for checking name uniqueness, which might involve creating a centralized database or an in-memory data structure.)
* **Security and Privacy (**Developers must implement secure authentication tools, data encryption, and proper user permissions to prevent unauthorized access.)
* **Resource Constraints** (Developers need to optimize the application for low bandwidth and limited device resources.)

## [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 Entity class establishes a relationship between the Game, Team, and Player classes, serving as a superclass from which all these classes inherit common attributes and behaviors. In UML, this relationship is represented through inheritance, making Entity the base class.

When analyzing their relationships, we observe that Team and Player follow a "has-a" relationship. For example, a Game "has" a Team, and GameService "has" Games. In UML, such relationships are described as aggregation (HAS-A), where one class contains a reference to another class's instance. Specifically, GameService holds a reference to Games, each Game references its Teams, and each Team references its Players. This structure demonstrates how these classes are interconnected to fulfill the application's 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** | Mac server side is easy to use. It also has cross platform development. It also has flexible terminal commands. There are no additional server-side software costs if using the native tools. Macs developer program is costly. $99 annually fee. | Linux is very cost efficient. Very difficult to navigate. The command shell has simple server configuration and accessibility. It provides lightweight and efficient server configurations with broad support for tools like Apache, Nginx, and MySQL. | Server is expensive. Has a friendly UI. Also has a command prompt. Has an extensive combability. Has a wide application support. | Has high performance but does very from different devices. They have scalability. They offer a lot of security features.  Cloud services like AWS or Azure provide scalability for applications accessed via mobile. |
| **Client Side** | Macs developer program is costly. $99 annually fee.  Very little experience and expertise is needed. Will need to know languages like swift and objective C. Like Linux, modern web standards will ensure compatibility across macOS browsers like Safari, Chrome, and Firefox. Moderate time is needed. | It is cost friendly. It is free and open source. A lot of time and expertise is needed. Will need to become familiar with the Linux eco system. The application must adhere to modern web standards (HTML5, CSS3, JavaScript) to ensure compatibility across web browsers like Chrome, Firefox, and Edge.  It may need additional time to test compatibility with various Linux distributions. | Windows offers free and paid development tools. It is also easy to learn and navigate. Don’t need to be an expert to learn. Familiarize yourself with visual studio code. Testing should include various screen resolutions and configurations specific to Windows hardware. Short, as Windows browsers align closely with web standards and are less likely to introduce unique issues. | Got tools like Android studio which is free and Apples $99 annual fee to use their software. Very time intensive to develop for both platforms. Developers need to know Swift/Objective-C for iOS development and Kotlin/Java for Android. Optimization for touch inputs, smaller screens, and gestures. For expertise you must have knowledge of mobile-first responsive design. |
| **Development Tools** | HTML, CSS, and Java script are needed for front end development.  Visual studio code primary IDE. Tools like Electron, Flutter, and Python extend the capability to build cross-platform.  A team must have experience with Mac-specific tools (e.g., Xcode) and be able to troubleshoot Safari-specific issues.  Xcode is free, but macOS hardware costs can increase overall development expenses. | HTML, CSS, and Java script to support front end. Linus systems include JavaScript, Ruby, Php, and Python. Supported by IDEs like Visual Studio Code, Eclipse, and PyCharm.  Developers must be familiar with terminal-based tools and testing in different Linux environments.  Most tools like VS Code, Eclipse are free. JetBrains tools have licensing fees $99/year per developer. | HTML, CSS, and Java script to support front end. Tools that are needed are IDE Like Eclipse, Pycharm, ect.  Strong integration with Microsoft tools like Azure and SQL Server simplifies workflow.  Visual Studio Professional $45/month per developer or Community Edition free for small teams. Licensing for tools like JetBrains is an additional cost. | HTML, CSS, and Java script to support front end. IOS JavaScript (for web apps), Swift, Objective-C.  And Android JavaScript for web apps.  IDE that are Android Studio, Visual Studio Code, Xcode IOS.  iOS requires a team with Swift and Objective-C expertise for native features.  Licensing costs is Xcode is free, but Apple Developer Program costs $99/year per developer for app store deployment and advanced tools. |

## 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**: An operating platform to support the game The Gaming Room to expand Draw It or Lose It is windows server. Windows servers ensure broad accessibility, seamless integration with powerful tools, and a familiar user experience. It is the most cost efficient to reach a wide audience.
2. **Operating Systems Architectures**: Microsoft Windows server is a system developed by Microsoft. Supports a wide range of applications, including gaming, productivity, and enterprise software. Its hybrid kernel ensures stability and performance. It also has an advanced subsystems and APIs provide extensive compatibility. The networking, graphics capabilities, and security features, Windows offers the ideal architecture to support scalable, engaging, and secure gaming experiences. It operates in 64 bit architecture and provides great performance and security. It also supports virtualization through hyper V.
3. **Storage Management**: In Windows you can use Windows’ NTFS file system. It provides reliable local storage for temporary files, caching, or smaller data needs during gameplay. You can also use an SQL server for handling user profiles and keeping track of games scores.
4. **Memory Management**: Using NTFS for memory management will ensure secure, efficient, and reliable storage for Draw it or Lose it. It caters to the game's temporary storage and caching needs during gameplay. NTFS supports large file sizes and provides efficient storage for game assets, such as images, logs, and temporary files.
5. **Distributed Systems and Networks**: Draw It or Lose It can seamlessly communicate across various platforms by leveraging distributed systems and a robust network infrastructure. Having cloud based hosting, fault-tolerant architecture ensures consistent and engaging gameplay while addressing challenges like connectivity issues and outages. Need to implement a shared database.
6. **Security**: Windows comes with built in security features. Windows detailed security features, such as BitLocker, Defender, Secure Boot, and Encryption provide a solid protection for game data. Windows' comprehensive security features, such as BitLocker, Defender, and Secure Boot, provide a solid foundation for protecting game data.