

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/25/2025 | Emilio Crocco | Completed Executive summary, Domain Model, and Design Constraints section |
| 1.1 | 2/8/2025 | Emilio Crocco | Completed Evaluation section |
| 1.2 | 2/18/2025 | Emilio Crocco | Added Recommendations section |

## [Executive Summary](#_heading=h.35nkun2)

The Gaming Room project is to develop a web-based game that can be used on multiple platforms based on the current game Draw It or Lose It. Draw It or Lose It is currently only available on Android. The game's purpose is to have multiple teams consisting of several people going four rounds at a minute each. When a picture is pulled from a library of images, one team guesses till the time runs out. If not answered, the opposing team has 15 seconds to guess

## Requirements

There are many requirements, including cross-platform functionality, having multiple teams and players that cannot be duplicated, teams must allow for multiple players, and only one game instance can exist at a time. (using unique IDs for each game, team and/or player)

## [Design Constraints](#_heading=h.1ksv4uv)

* Cross-platform functionality and cross-platform servers.
* Having only one instance of a game at a time.
* Storing and tracking unique IDs to prevent duplicates.
* The ability to have multiple teams, players, and games at once.
* The user interface should be almost identical across all platforms

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The Entity class created a relationship between the Game, Team, and Player classes. By doing this, they can inherit or get information from Entity. With the UML, we can show this through inheritance. This makes Entity a superclass. When we look at Team and Player, we can see that their relationship is “has a” type.

**"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](#_heading=h.z337ya)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Known for stability and reliability. Specifically built for web applications and services developed on the Apple platform. | Open-source-based and most popular for hosting applications featuring easy scalability and lower costs. Low system requirements and support are available through community and paid options. Widely available on most platform hosting options and easy to scale. | Requires a license to run services. Easy to navigate and use due to its easy-to-use GUI. High resource and initial load times. Fewer supported hosting applications are available. | Specifications can be different from device to device. |
| **Client Side** | While the MacOS offers a user-friendly environment, it is more expensive and has a much lower market share. Developing and maintaining multiple clients can elevate development costs and time investment and may require diverse expertise. | Using the operating system requires significant expertise and time investment, particularly in Linux, where familiarity with its data is essential. | License costs may be higher. Windows systems are easy to learn and support, and minimal expertise is required to set up. | Provide flexibility to clients or even developers to see updates at any time.  Platform-specific teams may need to develop the same application twice, one for IOS and one for Android. Swift and Kotlin knowledge will be necessary. |
| **Development Tools** | Any IDE that handles HTML, Javascript, and CSS.  XCode is also required for Mac systems. Any libraries or additional plugins need to be used for ease of development. Popular tools include Visual Studio Code, Eclipse, etc. | Any IDE that handles HTML, Javascript, and CSS. Any libraries or additional plugins need to be used for ease of development. Building software for deployment on Linux typically involves using programming languages like C, C++, and Python. | Any IDE that handles HTML, Javascript, and CSS. Any libraries or additional plugins need to be used for ease of development. Building software for deployment on Windows typically involves using programming languages like C# and .NET. Additional checks may be needed due to File System permissions on cross-platform development. Popular tools include Visual Studio Code, Eclipse, etc. | Android: Uses Java or Kotlin  iOS: Uses Swift or Objective-C  Most IDEs can handle development in all languages however, specialized IDEs can be more beneficial. |

## 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**: Linux would be best to host the game because it is cost-efficient, stable, and scalable. It is widely used in web hosting and can handle high traffic with fewer resources than other platforms.
2. **Operating Systems Architectures**: Linux is lightweight and efficient for hosting web applications.
3. **Storage Management**: Use cloud storage to handle game data so it’s easy to scale as the game grows. Cloud storage makes it easier to scale as the game grows and ensures data backups for security and disaster recovery.
4. **Memory Management**: Linux manages memory well, efficiently handling a single game instance preventing memory overload.
5. **Distributed Systems and Networks**: Use a cloud-based hosting provider to enable global access and ensure low latency for all users. Using dedicated servers or cloud services will help keep things fast and reliable.
6. **Security**: Implement HTTPS and secure authentication to protect user information. Regular updates and monitoring will help ensure ongoing security.