

Kamirah Pritchard

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_pxbk4st2o46e)

[**Table of Contents 2**](#_srfwl1zgevle)

[**Document Revision History 2**](#_k1xva9imq2c)

[**Executive Summary 3**](#_3e21tmpsxk64)

[**Requirements 3**](#_ceedai9s476f)

[**Design Constraints 3**](#_922761ef27fb)

[**System Architecture View 3**](#_pqli2yma2vo)

[**Domain Model 3**](#_v6f3m6cur45g)

[**Evaluation 4**](#_o3g801o4tp91)

[**Recommendations 5**](#_zdrkjvo2s1uc)

## [Document Revision History](#_gl7958jhn3ce)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <09/20/25>,  <10/06/2025>,<10/18/2025> | <Kamirah Pritchard> | Assessing the client’s requirements and creating a solution.  Evaluating the software requirements before development.  Final recommendations for the game’s development. |

**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](#_1n9fcg7ckwuo)

Client, the Game Room aims to create a game called “Draw it or Lose it”. This is a web-based game that involves multiple teams being able to guess the image that is being drawn. If a team does not guess correctly, the remaining teams have an option to choose.

## Requirements

*Requirements for this project consist of the game being able to have teams with multiple players within it.*

*Another, the game must consist of four rounds.*

*There needs to be auto auto-generated image that is displayed*

*Game and team names must be unique*

*Only one instance of the game at a time.*

## [Design Constraints](#_jelp55687rpg)

Some design constraints to consider while building a web-based environment for the game is the scalability, being able to support multiple users on the game at the same time. For this, the company must have a client-server architecture that can balance user growth so the website won’t basically crash. Another, the software will be built with the Java language. Therefore, the library and frameworks will be limited to this language. Next, regarding the user interface, the system must be user-friendly and easy to navigate for players.

## [System Architecture View](#_579ztmx10lu1)

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](#_oe3ncd673emm)

Classes Game, Team, and Player inherit from Entity. Inheritance helps the classes not have repeated code and ensures there is only one instance of an object.

**"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](#_u3dt0djl8nug)

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 consider the situation holistically, as everything must 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** | macOS discontinued its commercial server. It is still possible for smaller-scale developments. | Offers a server deployment that is dominant within operating systems. Has a large open-source system for web development. Many built-in images are in the cloud. Excellent security tools. | Offers a server deployment, has great vendor support, but requires subscriptions. | A robust operating system, such as the other OS, would be needed. |
| **Client Side** | The user will be able to access the web browser game via Safari, Chrome, and Firefox. No installation needed, provides seamless performance in web browsers. | No installation needed. Users are able to access the game via Firefox and Chrome. Efficient browser support. | No installation needed. | No installation needed. Depending on the mobile device used, the game can be accessed by Safari (iPhone), Google Chrome (Android), etc. |
| **Development Tools** | With macOS, you can use programming languages such as Python, C, etc. However, the IDE Eclipse is not available for this operating system. So for the Java language, you may have to use Visual Studio. | Has development tools such as VS Code, Eclipse, etc. | Has development tools such as .NET/.NET Framework, Windows Terminal, VS Code, PyCharm, etc. WSL (to use Linux on Windows). | N/A |

## Recommendations

Analyze the characteristics of and techniques specific to various system architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: The operating platform that is best for “Draw it or Lose it!” is Linux. It is known for its best scalability and flexibility. Since the game will involve many people playing together as teams, it will require a platform that can handle the enormous traffic with minimal freezes or crashes. Linux can run for extended periods of time without the need for reboots. In addition, as the game traffic and content continue to grow, Linux will be able to support growth with its flexibility.
2. **Operating Systems Architectures**: Linux has a monolithic kernel architecture. Meaning, everything runs as a single program, but components can be loaded dynamically. With a monolithic kernel architecture, there is a tight integration of system services and high performance, providing CPU Scheduling, Memory management, file management, and other operating system functions through system calls.
3. **Storage Management**: Using a cloud-based storage for scalability and accessibility across all platforms that can be used for the game. This reduces local storage and allows images to be downloaded when it is needed, which saves storage for things such as run time or even updates when they come about. Techniques that will also save storage are data compression, which reduces the size of data files to save space. In addition, data deduplication eliminates redundant copies of data.
4. **Memory Management**: In order for the program to be able to manage those high definition images at an efficient speed is necessary to invest in a dynamic memory allocation system. This helps by only requesting memory when it’s needed, which allows for memory to be freed when it is not required for use. In addition, when memory is allocated, the memory size can be changed, which allows for the memory to be reused. Also, the memory can be deallocated when more memory is needed.
5. **Distributed Systems and Networks**: For a distributed system, such as the client-server architecture, that allows the client and server to communicate over a network, it is crucial to have this communication seamless. Therefore, I suggest implementing a cloud-based web server (AWS, Ubuntu) that is compatible with the Linux operating system. This can ensure scalability and high performance, as a web server such as AWS would automatically scale the game's resources depending on the website's traffic. Furthermore, web-servers tend to have multiple zones in different regions, which reduces the latency of the client request reaching a server. Furthermore, web servers such as AWS have a CDN (Content Delivery Network) that speeds up the delivery of images to users by caching them closer to their location, which reduces latency as well. Lastly, using techniques such as patching (using intentional chaos engineering to expose any vulnerabilities in the network), to improve any bugs/errors.
6. **Security**: The Linux operating system has fewer known vulnerabilities than other operating systems. However, there are still security methods that can be put in place to protect gamers’ information and also protect the company from loss. First, implementing TLS, a cryptographic protocol that ensures privacy in communication in web browsing. As mentioned above, using OS patching helps with security vulnerabilities. On a network level, invest in patching, and control what a user can do by controlling who enters or leaves the website.