

Draw It or Lose It Web Game

# **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 | 9/19/2024 | Jean Martinez | Made comparisons and explanations to the entire template, including an executive summary, requirements, design constraints, domain model, a detailed Evaluation table, and 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 wants to make their game, Draw It or Lose It, available as a web-based game that can work on lots of different devices, not just Android. The game lets players guess drawings to win points.

Some important coding techniques were used to make sure everything runs smoothly. For example, only one version of the game can run at a time (using something called the Singleton Pattern). Also, game and team names are always unique (using the Iterator Pattern). These methods help the game work better and make it easier to manage.

## Requirements

The Gaming Room has asked for the game to meet the following needs:

* The game must let players form teams and play against each other.
* Team and game names must be different from each other.
* Only one game service should be running at a time.
* The game should work on different devices, like computers and phones.

## [Design Constraints](#_2et92p0)

A few challenges came up during the design of the game. First, it was important to make sure that only one game service could run at a time. The Singleton Pattern was used to handle this, which stops multiple versions of the game service from running.

Another challenge was ensuring that each team and game had a unique name. The Iterator Pattern was used to check existing names and avoid duplicates. This prevents confusion when creating new games or teams.

Finally, since this is a web-based game, the design had to work on different devices and handle many players playing at once.

## [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 main parts of the game are GameService, Game, Team, and Player. All of them share common attributes, like having an ID and a name. These pieces work together to make the game run. GameService is responsible for creating and managing the games, teams, and players.

The Singleton Pattern ensures that only one game service runs, while the Iterator Pattern checks that all game and team names are unique. The structure shows how these different parts work together.

**"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** | Macs can handle server-side applications, but they are not commonly used for large-scale web hosting due to higher costs and less support in enterprise environments. | Linux is great for servers, and most versions of Linux servers are free. It has strong community support and works well with cloud services. | Windows servers are reliable and offer good integration with other Microsoft services, but they are more expensive and resource-heavy compared to Linux. | Mobile devices do not serve as web servers, but they connect to web-based services hosted on servers. Server-side services need to be optimized for mobile users. |
| **Client Side** | |  | | --- | |  |   Mac users typically use Safari or Chrome. Development for Macs requires testing to ensure compatibility with macOS-specific features, which can increase development time. | Linux users typically access the game through web browsers like Firefox or Chrome. The game should be optimized for open-source tools and common Linux environments. | Windows clients use a variety of browsers (Edge, Chrome, Firefox), and compatibility across these must be ensured. Windows is widely used, so it’s important to provide broad support. | Mobile devices need responsive design and touch-friendly interfaces. Special care must be taken to ensure fast load times and smooth gameplay on mobile browsers. |
| **Development Tools** | Mac development often uses Xcode or IntelliJ IDEA. Java-based tools work well, but additional time may be needed for macOS-specific optimizations. | Development on Linux is flexible, with tools like Eclipse, IntelliJ IDEA, and command-line utilities commonly used. Linux supports a wide range of programming languages and libraries for web applications. | Development for Windows clients can use Visual Studio, IntelliJ IDEA, or Eclipse. Windows also has good support for .NET-based applications, which might be beneficial if expanding beyond Java. | Development for mobile requires tools like Android Studio for Android or Xcode for iOS. Ensuring the game works well on both platforms is critical for mobile users. |

## 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 is recommended to host the game because it’s affordable, reliable, and works well with many players.
2. **Operating Systems Architectures**: Linux will handle the game’s server, and players can connect using their devices. It’s a good choice for managing a lot of users.
3. **Storage Management**: A cloud service like Amazon S3 is a good option for storing game data, as it allows for growth and quick access.
4. **Memory Management**: Linux handles memory management, ensuring the game runs smoothly without using too much space.
5. **Distributed Systems and Networks**: The game will connect players through the web, with a server to handle all interactions. This setup will allow the game to work across computers, phones, and tablets.
6. **Security**: To keep player data secure, the game will use HTTPS for web connections and encryption for any information being sent or stored.