

Draw It or Lose It

# **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 | 4/18/2024 | Elyssa Creed | Summary, constraints, and OS specifications |

**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)

We are wanting to design a web-based game application called “Draw It or Lose It”, inspired by the television show Win, Lose, or Draw, where teams compete to guess what is being drawn. In our version, teams will guess phrases or titles based on stock drawings made by the application. Each game is four rounds, and drawings are fully rendered by the 30-second mark. If a team fails to guess the answer in time, other teams have a chance to offer guesses within a 15-second time limit.

## Requirements

* The game must support one or more teams, each with multiple players.
* Game and team names must be unique to avoid conflicts.
* One instance of the game can exist in memory at any given time, so unique identifiers must be used for games, teams, and players.

## [Design Constraints](#_2et92p0)

* **Web-Based Environment:** the application must adhere to web development best practices and compatibility with various web browsers and platforms.
* **Unique Names:** Unique identifiers for games, teams, and players will add complexity to the app’s data management and validation processes.
* **Memory Management:** Memory leaks and performance issues must be handled to ensure that only one instance of the game exists at one time.
* **Real-Time Rendering:** The application must render stock drawings in real time. The app must have efficient image processing and delivery to maintain smooth gameplay.

## [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 represents a generic entity with properties like ID and name, along with methods for retrieval and string representation. Below Entity, the GameService class manages games and their creation, with methods for adding and retrieving games. The Game class represents individual games and their associated teams, with methods for adding teams and generating strings. The Team class handles teams within games, allowing for player management through methods for adding players and generating strings. Finally, the Player class represents individual players, providing methods for creating players and generating strings. The diagram also depicts associations between classes, such as GameService being connected to Game, Game to Team, and Team to Player, indicating the relationships between these entities in the gaming system.

**"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** | Offers a stable work environment with robust security features, but may have limited support for certain software libraries. | Flexible and scalable, although it may require more expertise to configure and maintain. | Seamless integration with Microsoft technologies, but may have additional licensing fees and targeted by malware. | Limited capabilities for hosting web-based apps and may face constraints like limited processing and network connectivity. |
| **Client Side** | Supporting multiple types of clients on Mac requires consideration of development costs, time, and expertise. | May require open-source tools and languages like C/C++, Python, and Java for development. | Development would include Microsoft Visual Studio and languages like C#, .NET, and JavaScript, with considerations for compatibility and documentation. | May require Xcode for iOS and Android Studio for Android, along with responsive web development for different screen sizes and device capabilities. |
| **Development Tools** | Involves using Xcode and Swift/Objective-C to enhance development workflow. | Involves using open-source tools like GC, Python, and Java, along with text editors like Vim and package managers like apt-get(APT). | Involves Microsoft Visual Studio, languages like C#, features such as IntelliSense and project templates for streamlined development. | Might involve using Xcode for iOS and Android Studio for Android, with access to cross-platform frameworks like React Native and Flutter. |

## 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 a suitable platform. Its open-source nature allows for greater flexibility and scalability, and can ensure seamless deployment across various operating systems.
2. **Operating Systems Architectures**: Linux supports multiple architectures, including x86, ARM, and PowerPC, which provides compatibility with a wide range of hardware. It also offers a modular kernel design, which allows for customization and optimization.
3. **Storage Management**: A reliable system such as sxt4 or XFS would be suitable for the Linux platform. They offer features like journaling, scalability, and efficient data handling, ensuring data integrity.
4. **Memory Management**: Linux has virtual memory management techniques, including demand paging, memory mapping, and process isolation. This protects and optimizes the memory.
5. **Distributed Systems and Networks**: Implementing APIs and standardized communication protocols would ensure a seamless interaction while considering connectivity, reliability, and fault tolerance.
6. **Security**: Linux offers security features such as file system encryption and network security protocols like SSH and IPsec. By implementing regular updates and security audits, Draw It or Lose It can protect user’s data across different platforms.