

# 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 | 09/15/23 | Devin Judson | Filled in pages 3-5 |
| 1.2 | 09/28/23 | Devin Judson | Evaluation |
| 1.3 | 10/11/23 | Devin Judson | Architecture Recommendation |

**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 objective of the gaming room project is to create a cross-platform web-based game inspired by the existing "Draw It or Lose It" game, originally exclusive to Android. In this game, the goal is to form multiple teams, each composed of several players, and engage in four one-minute rounds. During each round, a random image is selected from a library of pictures. One team takes turns guessing the image, with a time limit for their guesses. If they fail to guess correctly within the allotted time, members of opposing teams can take a shot at answering within 15 seconds.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

## The game must be compatible with various platforms, accommodating multiple players within each team. Furthermore, the game system should ensure that only a single instance of the game can be active at any given time. Additionally, it is imperative that both game and team names possess uniqueness, allowing users to verify name availability when selecting a team name.

## [Domain Model](#_8h2ehzxfam4o)

**The entity class establishes a connection between the game, team, and player classes, allowing them to inherit or access information from the entity class. In UML notation, we can illustrate this relationship through inheritance, designating entity as the superclass.**

**When we examine the relationship more closely, we observe that teams and players are of a "has-a" type within the game entity. In the context of UML, this relationship is referred to as aggregation, signifying that a user instance belongs to one class and holds a reference to an instance of another class.**

**In this diagram, you can see that the game service class possesses references to games, indicating that it manages games, and games, in turn, hold references to teams. Teams, similarly, have references to players, showcasing the hierarchical structure of these classes within the 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** | Mac offers stability and security along with a user-friendly interface, compatibility with popular development tools, and hardware quality. However, weaknesses may include limited hardware customization options and potentially higher upfront costs compared to some other server options.  **Edit:**  For the backend, if rolling your own server, the project can start at minimal costs. | Linux is a strong choice for hosting web-based software applications due to its stability, security, and open-source nature, which allows for extensive customization and optimization. Its advantages include a wide range of hosting solutions and low cost of ownership, but potential weaknesses can include a steeper learning curve for administrators unfamiliar with Linux and a potentially less user-friendly interface compared to some other server operating systems.  **Edit**:  The best Linux practices include strong and unique passwords, multi-factor authentication, updating server software regularly, closing hidden ports, and utilizing backups. | Windows can effectively host web-based software applications, particularly those built using Microsoft technologies such as ASP.NET and C#. Advantages include strong support for Microsoft-centric development, extensive third-party software compatibility, and user-friendly management tools, but potential weaknesses may include higher licensing costs, susceptibility to security vulnerabilities, and a smaller market share in web hosting compared to Linux.  **Edit**:  A windows server solution can also start at minimal costs off and scale up if more computing power is needed. | Mobile devices are not typically used for hosting web-based software applications; instead, they are typically clients that access web-based software hosted on servers. Their characteristics include portability and diverse operating systems (iOS and Android). Advantages include the potential for a large user base, but weaknesses include limited processing power and resources compared to dedicated servers or cloud hosting.  **Edit**:  Mobile devices would require a ground-up approach of coding and hosting the server in addition to cross-app development. |
| **Client Side** | Mac for software development may require considerations such as additional development time to ensure compatibility with various operating systems (e.g., macOS, Windows, Linux) and browsers, potentially leading to increased development costs.  **Edit:**  Consider an independent React-native web developer. A hybrid app can be created rather than a native app allowing for portability, faster speed to market, and cheaper origination costs. | Linux in software development may require considerations such as additional development time to ensure compatibility with different distributions and configurations, potentially leading to increased development costs.  **Edit**:  Authentication can connect to our Flask or Firebase database. | Windows in software development may require considerations such as additional development time to ensure compatibility with various Windows versions and configurations, potentially leading to increased development costs. Moreover, expertise in Windows-specific technologies and cross-platform development tools is crucial to efficiently target diverse client environments, which can impact the skill set and expertise needed within the development team.  **Edit**: React-native for web codebase can also be worked on in a Windows environment using Visual Studio Code. | Mobile devices in software development necessitates considerations such as additional development time and cost to ensure compatibility with different operating systems (iOS and Android) and device screen sizes, which may require expertise in cross-platform development tools or separate development teams for each platform. Additionally, expertise in mobile user interface (UI) and user experience (UX) design is vital to create optimal experiences across various devices and form factors.  **Edit**: React will allow responsive media adjustments to conform to IOS and Android phone form factors and scale up the interface for web. |
| **Development Tools** | Developing software for deployment on Mac typically involves using programming languages like Swift or Objective-C for native macOS applications. Popular integrated development environments (IDEs) include Xcode for native Mac app development, while web-based applications can be built using HTML, CSS, and JavaScript with IDEs like Visual Studio Code or JetBrains WebStorm.  **Edit:**  **Flask, node.js, and Websockets** | Linux involves using programming languages like Python, C/C++, Java, or Ruby, while popular integrated development environments (IDEs) include Visual Studio Code, IntelliJ IDEA, and PyCharm. Additionally, web-based applications can be created using standard web development tools like HTML, CSS, and JavaScript, along with text editors or IDEs tailored for web development.  **Update:**  **Firebase and Amazon AWS** | Developing software for deployment on Windows commonly involves programming languages like C#, .NET, and JavaScript. Popular integrated development environments (IDEs) include Visual Studio, Visual Studio Code, and JetBrains Rider for Windows application development, while web-based applications can be built using standard web development tools like Visual Studio Code, Visual Studio, or Sublime Text.  **Edit:**  **MySQL** | To build software for deployment on mobile devices, commonly used programming languages include Swift and Objective-C for iOS development and Java or Kotlin for Android development. Popular integrated development environments (IDEs) include Xcode for iOS and Android Studio for Android, along with cross-platform development frameworks like React Native, Flutter, or Xamarin for building apps that work on both iOS and Android.  **Edit:**  Google Play Store and Apple App Store. |

## 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**: The most appropriate operating platform that will allow The Gaming room to expand Draw It or Lose It to other computing environments is Windows because you won’t run into a shortage of IDEs to work with.
2. **Operating Systems Architectures**: Microsoft Windows is a graphical operating system developed by Microsoft. It provides a way to store files, run software, watch media, and connect to the internet.
3. **Storage Management**: Windows comes with a feature called storage sense. This allows you to scrutinize and manage files on your hard drive.
4. **Memory Management**: Windows storage sense would allow for storage and management of Draw It or Lose It photos and game players. It also allows you to keep them together in one secure space in memory.
5. **Distributed Systems and Networks**: In network-based, multi-user interaction systems like network games, there is usually a shared database that connects players who are geographically dispersed and enables them to interact over the network. Presently, developers of network games are required to build shared databases and inter-player communication systems from the ground up.
6. **Security**: Windows comes with built-in security protection software.