

# 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 7**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <12/09/23> | <MINNU JOHN> | <Making changes to include more teams and players> |

**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 Draw it or lose it projects goal is to develop a web-based gaming application for the gaming room. The objective is to create an engaging and interactive game where multiple teams compete to guess phrases, titles, or things based on rendered images. This game has four rounds. Each round last for one minute, with teams attempting to guess the puzzle before time expires. The application will render images from a vast library of stock drawings as clues.

## Requirements

The client’s software requirement for this project are:

Ability to support one or more teams each with multiple assigned players.

Game and team names must be unique to prevent naming conflicts.

To achieve a single instance of the game in memory at any time, unique identifiers for each game, team and player will be implemented.

[**Design Constraints**](#_2et92p0)

The design constraints to develop this project in a web-based distributed environment involve considerations for hosting on different operating platforms like Mac, Linux, Windows and mobile devices. This influence decisions regarding server-side and clint-side development, as well as the selection of development tools.

## [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 relationships among the Game, Team, and Player classes are established through entities, with each class receiving or inheriting information from the Entity class. In UML, these relationships are demonstrated using inheritance, allowing common attributes such as "name" and "id" to be shared among all classes, with Entity serving as the superclass.

Upon closer examination, it becomes evident that there exists a "has a" relationship between Team and Player. This implies that a Team contains instances of Player, illustrating an aggregation (HAS-A) relationship in UML. When we say that a class "has a" relationship with another class, it typically means that it possesses both an instance of one class and a reference to an instance of another class.

Analyzing the diagram, it is apparent that GameService holds references to Games, Games contains references to Teams, and Teams, in turn, has references to Players. This intricate network of relationships showcases how the GameService orchestrates Games, which in turn contain Teams, and Teams consist of Players.

**"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** | Known for its stability and reliability, making them suitable for hosting web-based software applications.  Can be relatively expensive, potentially impacting deployment costs. | Recognized for its performance and stability, making it a popular choice for server-side hosting. Offers vast array of open-source tools and technologies, making it cost effective. Its diversity distribution may pose challenges in ensuring consistent performance across different environments. | Robust platform with a user-friendly interface, making it accessible for developers. Supports various web technologies and integrate well with Microsoft development tools. Its licensing cost might be expensive.  this server will be a good choice if the clients are familiar with Microsoft technologies | They offer portability and broad user reach.  They might have issues with processing power and memory may impact the server-side hosting power. Mobile devices are not suitable for hosting the server-side. The server-side logic, which involves handling data storage, business logic, and processing requests, is typically hosted on remote servers. These servers can be managed by the application developer or deployed on cloud platforms like AWS, Google Cloud, or Azure. |
| **Client Side** | Developing tools and hardware will be expensive.  Time depends on the team’s familiarity with the tools.  Investing in Mac development may lead to higher upfront costs but can enhance the user experience for Mac users. | The cost is lower due to the availability of open-source development tools. Time depends on the team’s familiarity with the tools.  This can contribute to efficient and cost-effective development. | Development cost on windows may include licensing fees for certain development tools. Time depends on the team’s familiarity with the tools. | Costs may be impacted by the need for platform specific development tools and potential licensing fees. Time depends on the team’s familiarity with the tools.  Expertise in swift or objective-c for iOS and java for android, is vital. |
| **Development Tools** | For developing software to be deployed on Mac, relevant programming languages include Swift and Objective-C. Xcode is the primary integrated development environment (IDE) for Mac applications, offering a comprehensive suite of tools for designing, coding, testing, and debugging. Leveraging the macOS Human Interface Guidelines is essential to ensure a seamless and intuitive user experience. | This involves using programming languages such as C++ and Python. Popular IDEs for Linux development include Eclipse, Code::Blocks, and Visual Studio Code. These tools provide features for coding, debugging, and version control. | For deploying software on Windows, relevant programming languages include C# and, to some extent, C++. Visual Studio is the primary IDE for Windows application development, offering a range of tools for coding, debugging, and designing user interfaces. | Developing software for deployment on mobile devices involves platform-specific programming languages and tools. For iOS development, Swift or Objective-C is used, and Xcode serves as the preferred IDE. Android development typically utilizes Java and Android Studio is the primary IDE. Cross-platform development tools like Flutter (Dart language), React Native (JavaScript), or Xamarin (C#) can also be considered for building applications that run on both iOS and Android. |

## 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 as the operating platform for Draw It or Lose It. Linux's open-source nature, stability, and scalability make it well-suited for hosting web-based applications. It provides a robust foundation for server-side hosting and supports a wide range of development tools.
2. **Operating Systems Architectures**: Linux operates on a monolithic kernel architecture, where the entire operating system resides in the kernel space. This architecture facilitates efficient communication between components and allows for better performance. Draw It or Lose It can benefit from this architecture by ensuring seamless interaction between the server-side components and the underlying system.
3. **Storage Management**: For storage management, a relational database management system (RDBMS) such as MySQL or PostgreSQL is recommended. Both are well-supported on Linux and offer features essential for managing game-related data, such as teams, players, and game instances.
4. **Memory Management**: Linux employs virtual memory management techniques, including demand paging and process memory isolation. These techniques optimize the usage of physical memory, ensuring that Draw It or Lose It can efficiently handle multiple game instances and maintain responsiveness.
5. **Distributed Systems and Networks**: Achieving communication between various platforms can be accomplished through the implementation of a microservices architecture. Each platform-specific component can be developed as a microservice, interacting through well-defined APIs. A message broker, such as RabbitMQ or Apache Kafka, can facilitate communication between these microservices.
6. **Security**: Linux, being inherently secure, provides a solid foundation. Additionally, employing HTTPS for secure communication over the network is crucial. User authentication and authorization mechanisms must be implemented, and sensitive data should be encrypted both in transit and at rest. Regular security audits and updates are essential.