

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 | 10/20/2024 | Brian Gregory | Added information on software design |

**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 project aims to create a web-based game that works across multiple platforms, expanding on the existing "Draw It or Lose It" game, which is currently available only on Android. The game involves multiple teams with several players, competing in four rounds, each lasting one minute. During each round, one team draws an image from a library, while their teammates try to guess it before time runs out. If they don’t guess correctly, members of the opposing teams have 15 seconds to make their guesses.

## [Design Constraints](#_2et92p0)

* The game must be compatible with multiple platforms.
* Each team must consist of multiple players.
* Only one instance of the game can run at any given time.
* Both game and team names must be unique, allowing users to check name availability when selecting a team name

## [Domain Model](#_8h2ehzxfam4o)

The Entity class links Game, Team, and Player, meaning they all inherit or get information from Entity as the superclass, shown in UML with inheritance. Team and Player have a "has a" relationship, where Game includes a Team, and GameService contains Games. In UML, this is called aggregation (HAS-A). Basically, one class holds a reference to another. In this case, GameService refers to Games, Games refer to Teams, and Teams refer to 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** | Mac servers come bundled with Apple hardware at no additional cost. They can efficiently host web-based applications and scale to support large user bases. | Linux is free and offers robust server deployment methods. It's highly reliable for hosting web applications and supports scalable solutions. | Windows servers require licensing, but they are capable of hosting web applications and handling scalability needs. | Mobile devices typically connect to external servers instead of hosting web applications. MDM solutions and their costs vary depending on the service provider. |
| **Client Side** | Requires responsive design to ensure compatibility across browsers and devices, with a focus on optimization and testing for seamless performance. | Also requires a responsive design for compatibility across different browsers and devices, ensuring proper functionality on various screen sizes. | Focus on responsive design to ensure compatibility across devices and browsers. This might require additional development time to handle all supported environments. | A mobile-first design is essential for functionality across iOS and Android. Testing on various devices and screen sizes is key to delivering a consistent user experience. |
| **Development Tools** | Commonly used languages include Java, Python, and JavaScript. Eclipse and IntelliJ IDEA are popular IDEs. Depending on the project scope, separate teams might be required. | Java, Python, and JavaScript are frequently used, with Eclipse and IntelliJ IDEA as the primary development tools. Development may involve multiple teams for larger projects. | Java, Python, and JavaScript are standard languages, with Eclipse and Visual Studio as preferred IDEs. Multiple teams may be needed based on the project scale. | For Android, Java with Android Studio is common, while iOS uses Swift with Xcode. Depending on the complexity, development may require multiple teams. |

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**: For expanding "Draw It or Lose It" to different environments, I recommend using Linux as the operating platform. This choice allows for easy customization without the burden of high licensing fees. Linux provides solid security and reliability, which are crucial for running an online game. Its flexibility makes it easier to adopt new technologies and scale as the player base grows.
2. **Operating Systems Architectures**: Linux runs on a monolithic kernel architecture, which efficiently manages hardware and software resources. This setup supports smooth data processing, which is important for gaming performance. The modular design of Linux allows you to add new features without needing to completely overhaul the system, keeping the game responsive to changes.
3. **Storage Management**: For storage management, MySQL is a suitable choice. It integrates well with Linux and effectively handles the player data generated as the game expands. MySQL can scale alongside the user base, ensuring that performance remains stable even as data loads increase. Its transactional capabilities help maintain data consistency during gameplay.
4. **Memory Management**: Linux uses techniques like paging and segmentation for memory management, allowing it to allocate memory efficiently across multiple processes. This is essential for maintaining a smooth gaming experience. By employing a virtual memory system, Linux optimizes RAM usage, ensuring that "Draw It or Lose It" runs well, even under heavy loads.
5. **Distributed Systems and Networks**: To facilitate communication between platforms, consider using a **microservices architecture**. This approach lets different components of "Draw It or Lose It" operate independently, which simplifies scaling and maintenance. Utilizing RESTful APIs will enable smooth interactions across platforms. A well-structured network with load balancers will help manage traffic effectively, even during connectivity issues.
6. **Security**: Security is crucial for The Gaming Room. Implementing TLS will help protect data exchanged between users and servers. Additionally, using role-based access control will restrict access to important system functions based on user roles. Regular security assessments are also necessary to identify and address vulnerabilities, ensuring that user information remains secure across all platforms.