

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 | 03/17/2025 | Jael Ortiz | Added an Executive Summary and Design Constraints to explain the project needs, game requirements, and important technical considerations. |

**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 turn their Android game, Draw It or Lose It, into a web-based game that works on multiple devices. Since their team doesn’t have the technical skills to do this, we will create a system that makes the game easy to use and run smoothly online.

The web version will allow multiple teams to play, with each team having several players. To keep things organized, game and team names must be unique, and only one game can run at a time. This will be managed by assigning unique IDs to each game, team, and player. The game will follow the same four-round format, with drawings appearing in real-time and teams getting a chance to guess.

We will plan the technical details and development steps to make sure the game works well as a web-based version while keeping the fun experience of the original game.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

When making Draw It or Lose It a web-based game, there are a few important things to consider. First, the game needs to work on different devices and browsers, so we have to make sure it looks good and works well everywhere.

Each game and team name must be unique, so players can’t choose the same name. We also need to make sure only one game is running at a time by giving each game, team, and player a unique ID.

Since the game is played in real-time with many players, we need to make sure everyone sees the same thing at the same time. This means the system must quickly update and send information to all players, which can be tricky with lots of people playing at once.

We also need to keep player data safe with encryption, especially login details. If a player disconnects, the game needs to save their progress and update it when they come back.

By keeping these things in mind, we can make sure the game works smoothly, is secure, and gives a good experience for everyone.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

The UML class diagram for The Gaming Room system organizes games, teams, and players efficiently. The Entity class serves as a base, providing shared properties like id and name, which Game, Team, and Player inherit. This reduces duplicate code and keeps the structure clean.

Each Game contains multiple Team objects, and each Team contains multiple Player objects. This setup models real-world relationships, making it easy to manage game participants. The system also includes a GameService class, which follows the Singleton pattern to ensure only one instance handles all game-related operations. It tracks games, assigns unique IDs, and manages game data efficiently.

The program starts with ProgramDriver, which contains the main() method, while SingletonTester ensures the Singleton pattern works correctly. The system applies key object-oriented programming (OOP) principles like encapsulation (protecting data), inheritance (reusing code), and polymorphism (customizing methods like toString() for different classes).

**"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 are not commonly used because they are expensive and less supported. | Linux is the best choice for servers since it is free, secure, and reliable. | Windows servers work well for businesses, especially for Microsoft-based software. | Mobile devices cannot host servers but can access web applications. |
| **Client Side** | Making Mac apps requires learning Swift or using tools like Electron for cross-platform support. | Linux mostly runs web apps, but many big-name programs don’t work on it. | Windows is the most popular for software development and supports many programming tools. | Mobile apps need to be built separately for iOS and Android or use cross-platform tools like Flutter. |
| **Development Tools** | Developers use Xcode for Mac apps, along with VS Code and JetBrains tools. | Linux works well with VS Code, Eclipse, and many free coding tools. | Windows developers use Visual Studio, .NET, and PowerShell. | Mobile apps are made with Android Studio for Android, Xcode for iOS, or tools like Flutter for both. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

To make *Draw It or Lose It* available on more devices, I recommend using cloud platforms like Amazon Web Services (AWS) or Microsoft Azure. These platforms will help the game run smoothly on different devices like computers, phones, and tablets. They also allow the game to grow as more people start playing, without needing to worry about running out of space or power.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

The game will run on a cloud system that lets the game’s website and app work on different devices. The game will have two parts, the part you see (the game on your screen) and the part that runs the game behind the scenes (like tracking scores and showing pictures). Cloud platforms like AWS and Azure make sure these parts work together and can handle more players as needed.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

For storing game progress and user information, the game will use online storage. This will allow players to access their data from any device, whether it's a phone, tablet, or computer. The storage will automatically expand to keep up with the growing amount of data.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

The game needs to use memory to keep track of everything happening in the game. In the cloud, this memory is managed automatically. If lots of people are playing at once, the cloud can give the game more memory to keep it running smoothly. When fewer people are playing, it uses less memory.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

The game needs to work across different devices, like phones, tablets, and computers. The cloud platform will help send data back and forth between these devices in real-time. This way, everyone sees the same thing, even if they are using different devices. If the internet connection or a server goes down, the system will try to automatically fix it to keep the game running without interruption.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

Security is very important to protect player data. Both AWS and Azure have strong security features to keep information safe. For example, the game will use encryption to protect data as it travels between devices. It will also use secure logins to make sure only the right people can access the game. Extra security features like multi-factor authentication can be added to keep accounts safe.