

<Name-of-Software-Application>

# **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 | 03/20/2025 | Anjel Cobbs | Completed the Executive Summary, Design Constraints, and Domain Model |
| 1.0 | 04/13/2025 | Anjel Cobbs | Completed Evaluation Table |
| 1.0 | 04/23/2025 | Anjel Cobbs | Completed the Recommendations |

**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 has a game, Draw It or Lose It, currently only available to Android users. They would like to extend their game to more users by developing a web-based game inspired by their app that will be available on all platforms. Draw It or Lose It is similar to the 1980s television game *Win, Lose, or Draw*.

## 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)

The web-based version of the Android application must be accessible on all platforms. This may require using a software development kit, such as Flutter or React Native, to develop a cross-platform application with a single source code.

Each game can have at least one team playing, and each team consists of multiple players. Each player and each team must have unique names. We’ll need to utilize an iterator pattern to traverse over every player’s name and team name in memory.

There can be only one instance of a game in memory at any time. To accomplish this, a Singleton design pattern should be implemented in the class that manages game creation and identification.

The structure of the game consists of four, one-minute rounds. The drawings for each round must be completely rendered within the first thirty seconds.

## [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 ProgramDriver class is dependent on the SingletonTester class.

The Entity class is the parent class of the Game, Team, and Player classes.

A flexible relationship exists between the GameService class and Game class. For every instance of the GameService class, there can be zero or more instances of the Game class.

A flexible relationship exists between the Game class and Team class. For every instance of the Game class, there can be zero or more instances of the Team class.

A flexible relationship exists between the Team class and Player class. For every instance of the Team class, there can be zero or more instances of the Player class.

**"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 has great security features that would help protect user data. Technical support for Mac software and hardware is readily available, with high success rates, which can be helpful if an issue arises. Some disadvantages to using Mac could be that its software and hardware are expensive. Mac isn’t always compatible  with outside tools and software. | Linux is free to use, making it a cost-effective choice. It’s also highly secure. Some of Linux’s weaknesses are that it isn’t beginner-friendly; it takes someone with a lot of technical experience to use it successfully. | Windows is a commonly used operating system, making its software compatible with a wide range of hardware and software. A major weakness of Windows is its weak security. It is vulnerable to viruses and cyberattacks. It also updates regularly, which can either be a good or bad thing, considering the updates are forced. | Mobile devices are convenient because they’re portable. Their touchscreen interface makes for easy and efficient navigation and workflow. There are some obvious weaknesses, like the smaller screens, limited processing power, battery life, and memory. It would be cumbersome to manage large, complex projects. |
| **Client Side** | Mac hardware and software tools can be expensive. Testing the client side of the codebase can be costly and time-consuming. Developers must also be familiar with Mac-specific software development tools and programming languages. | Linux is a free resource. The costs come from the software tools and hardware necessary for app development. However, Linux can run on minimal, low-cost workstations. If developers have little experience with Linux, the development process will take longer. To support multiple types of clients, developers should have experience with cross-platform frameworks. | Testing the client side of the codebase with Windows can be expensive because there are so many versions of Windows. Since Windows software is regularly updated, the codebase will require constant maintenance and bug fixes, which will be costly long-term. Developers should have experience \working with Windows to ensure compatibility. | Developing software using mobile devices could be cost effective because of the convenient access to inexpensive programming apps, but programming complex code on a mobile device would be time consuming. The advantage of building a cross-platform codebase with mobile devices is that developers can be either beginners or experts and find a way to work efficiently because mobile devices are customizable. |
| **Development Tools** | Popular programming languages for developing Mac software are Swift and Objective-C. Some compatible IDEs are Visual Studio Code and Xcode, Apple’s IDE for iOS and macOS app development. | Popular programming languages for developing software on Linux are Python and C/C++. Common IDEs include Eclipse, PyCharm, and Visual Studio Code. | Common programming languages for developing software on Windows are C#, C++, and JavaScript. Common IDEs are Visual Studio Code and SQL. | For developing software for Android mobile devices, Java is the primary programming language. For iOS devices, Swift is the primary programming language. |

## 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**: Windows is a suitable operating system for expanding the Draw It or Lose It software as a cross-platform application.
2. **Operating Systems Architectures**: Windows utilizes a layered architecture, consisting of two layers: user and kernel mode. Operating in user mode consists of running user applications. Different applications are processed independently of each other and the core operating system, minimizing dependencies and interference. Kernel mode involves interacting with the operating system by managing critical functions like process control, memory allocation and hardware interaction. This separation of concerns divides the workload of the operating system, allowing for multitasking between environments.
3. **Storage Management**: Windows offers a storage management tool called Dev Drive, a storage drive dedicated to developing and debugging software. Separating development files from the main system drive eliminates conflicts and helps developers stay organized. It also frees up space on the system driver, improving overall performance.
4. **Memory Management**: Windows uses virtual memory, storing less frequently used data on the hard drive, which allows the game to continue running with limited memory space. Windows also provides tools to manage memory allocation and deallocation, helping prevent memory leaks and fragmentation.
5. **Distributed Systems and Networks**: The Draw It or Lose It software should follow a client-server design where the multiple clients—game consoles, mobile devices, computers—communicate with a central server that manages the game and allows seamless collaboration between platforms. Distributed software distributes the workload across multiple nodes to process tasks faster and enhance the game’s performance. If a client can’t connect to the server, or vice versa, redundancy strategies and fault-tolerance patterns help prevent network failures from cascading.
6. **Security**: Some client-end practices that can protect user information are strong password validation and regular software updates. It is crucial to encrypt the data in storage and memory that is transmitted over a network. Leveraging Windows security features like Windows Firewall can protect data by monitoring network traffic. Windows-compatible IDEs, like Visual Studio, offer tools that allow developers to check the security of their code.