

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 |
| --- | --- | --- | --- |
| 1.4 | 02/23/2025 | Richard Pinkham | Recommendations for operating platforms, operating systems architectures, storage management, memory management, distributed systems and network, and security. |

**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)

To help the client in their effort to revamp "Draw It or Lose It" into a web-based game that operates on multiple platforms, we can delineate a neat, structured plan. This plan encompasses (1) setting up the work environment; (2) adopting development strategies; and (3) implementing vital game features. Select a Cross-Platform Framework: Use frameworks such as Unity, Phaser, or React Native that enable web deployment. Unity is outstanding for game development and can export to WebGL, but all these frameworks have their strengths depending on your project's needs. Managing Game Instances: Use the singleton pattern to ensure that only one instance of the game exists in memory at any given time. This can be managed through a game server that handles sessions. Setting up teams: Let players create or join a team. Customize the name and logo for each team. Setting up teams: Let players create or join a team. Customize the name and logo for each team. Launch the game softly in select markets to gauge its performance and collect data on your users.

## Requirements

A structured approach is needed to translate the requirements for the game with teams and players into an implementation. We can define this structured approach with the following outline: Game Structure, Unique Identifiers, Name Uniqueness Check, Implementation Steps. This layout guarantees that every occurrence of the game, each team, and every player has a distinct identifier and name, ensuring that the game can be managed in an orderly fashion and that name conflicts are preemptively resolved.

## [Design Constraints](#_2et92p0)

The work of developing a game that runs across multiple platforms is inherently fraught with challenges that can affect not only the schedule for getting the game done but also the quality of the finished product. The main constraint is the need for cross-platform development. Moreover, guaranteeing that development fulfills all client demands across various platforms adds yet another layer of complexity. Each platform has its own individual specifications, performance criteria, and user interface mandates that must be followed. This makes for a lot of communication with the client to ensure that we understand their expectations and with a lot of testing to ensure that we adhere to these specifications. To summarize, the demands of cross-platform development and client insistence on certain features mean that we have to plan, allocate resources, and manage our team with great care to achieve a smooth, successful launch of our game on all platforms, intended and otherwise.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: Nothing is 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 Entity Class is the parent (super) class of the Game, Team, and Player classes. This means that the Game, Team, and Player classes, as Entity’s child classes, will inherit Entity’s attributes, while each being assigned characteristics of their own that are separate from the parent class. The Game Service Class is used to ensure the client’s requirements are met, providing a single game instance at a time, a unique team name (id), a unique game name (id), and a unique player name (id). Program Driver contains the main statement and uses the Singleton Tester class. The Game class contains a team list, and the Team class contains a Player list. The Player class does not contain a list, as it ensures that each player has a unique id, that can be assigned to a team. While a player can be on a team, and a team does have players, the player class does not contain or have a team or a game.

**"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** | One of MacOS's standout characteristics is its capacity to execute programs from multiple operating systems, namely MacOS, Windows, and Linux. This accommodation of varied software makes MacOS a compelling choice for developers and any kind of power user who needs to traverse different computational terrains. Users are almost guaranteed a predictable MacOS experience with its consistent user interface. | Linux provides a vast array of distributions (distros), including but not limited to Ubuntu, Fedora, Debian, and CentOS, from which users can choose an operating system that serves their specific needs and fits their particular environment, whether that be personal use, software development, or enterprise deployment. Linux provides an immense level of personalization options, allowing users not only to tweak the installed desktop environment but also to change the behavior and appearance of the operating system itself. This is mainly an advantage for developers and system administrators who need specific setups for servers or embedded systems. | For corporate users, the main advantage of Windows is its ability to integrate with corporate servers based on Active Directory. This allows users to be authenticated in a Windows way without additional add-ons or expenses. However, the real benefit of this feature is not felt at the level of the individual user. It is felt by corporate IT whenever user accounts need to be managed. There are, however, some disadvantages to using Windows. One big negative is that it offers limited tools for mobile development. That can be a problem for anyone who needs to create flexible and mobile solutions. | Hosting a web application on a mobile device has pros and cons. Compared with conventional web hosting services, mobile devices can be economical substitutes, mainly when the number of users is small, and the hosted application is relatively straightforward. Compared with conventional web hosting services, mobile devices can be economical substitutes, mainly when the number of users is small, and the hosted application is relatively straightforward. Disadvantages Security Risks, Susceptibility, and Restricted Resources Compared to dedicated servers or personal computers, mobile devices have less power in terms of processing and memory. |
| **Client Side** | Apple's proprietary operating system, MacOS, has several advantages and disadvantages that can profoundly influence the developer experience and user accessibility. Pros of MacOS: Interface That Is Friendly to Users: MacOS is recognized for its simple interface, which is easy to use. Users can easily navigate and operate the system, especially after they have gained a little skill and understanding of the operating system. Cons of MacOS: Diminished Customization: Other operating systems, like Linux, allow the user much more flexibility in customizing and controlling the system. This kind of flexibility is rare in MacOS and is not present in the closed system that Windows represents. | Linux offers several benefits as an operating system, especially regarding cost and user control. As a free and open-source operating system, Linux drastically cuts the price of software licenses. For that reason alone, Linux is a winner for individuals and organizations that want to save money. And then consider this: You can rework the Linux operating system. You can change the source code as you see fit. You can do all that and still be legal. Again, that’s a rare combination in the world of computing, and it demonstrates that Linux is a flexible OS that can serve many different kinds of people in many different ways. | Pros of Windows: Accessibility: The Windows operating system is accessible to almost all software and hardware vendors, which allows for nearly universal compatibility and a diverse range of available programs for the user. Customizable Price Ranges: Various editions (Home, Pro, Enterprise) of Windows cater to different needs and budgets. Users can choose an option that fits their project requirements. Cons of Windows: Demand for Knowledge: The Windows OS expert may be necessary to competently manage, troubleshoot, and optimize the system— tasks that novices cannot effectively perform. This necessitates the allocation of either financial or human resources. | Mobile applications and tools versus their PC counterparts are one of accessibility and equality of function. These days, when people need to accomplish a task, they are just as likely to use a mobile device as a desktop or laptop computer. Pros of Mobile Applications: Accessibility: Mobile applications are available on various platforms, making them accessible to a large audience. Users can effortlessly obtain the apps from app stores at a range of price points, including many free options. Cons of Mobile Applications: Restricted Functionality: Numerous mobile apps lack many features on their desktop counterparts, which can lessen functionality and lead to a poorer user experience. Performance Constraints: Mobile devices typically have much less processing power and memory than personal computers. |
| **Development Tools** | The integrated development environment (IDE) for macOS, known as Xcode, furnishes developers with all the necessary tools to engineer applications for macOS and iOS, watchOS, and tvOS. Its many features contain a powerful code editor, an array of debugging tools, and a user interface designer. Xcode Cloud, in contrast, is a direct service that continuously integrates and delivers (CI/CD) applications built in Xcode. It allows developers to automate the building, testing, and deploying their app projects in a virtual cloud environment. This makes it just that much easier for teams of developers to apply the best practices for project integration and collaboration consistently. | The platform known as Docker is compelling. Yet, it's simple because it has a relatable user interface. Docker really offers a way to deal with a container, a very sturdy vessel into which you package not just an application but also what you need to run that application, from operating systems to libraries to whatever else you might need—directly inside the container. "Containerization" is what we call the use of a container. And you're familiar with the term "container" from other contexts—key Features of Docker Development Environment Consistency, Cross-Platform Applications, and Docker Hub. | The operating system most people use—the one that runs on most personal computers—was primarily written in the C programming language, although some components were implemented in assembly language. Windows was first released to major retailers in November 1985. Visual Studio is the most renowned and widely used among the various integrated development environments (IDEs) available for Windows. Visual Studio has multiple functions: a solid IDE, a powerful code editor, and, in some instances, a source/version control system. | Mobile app development has a premier programming language: Java. With its robust object-oriented features, Java has developed a massive following among developers. Yet, despite its dominant position, some of the other languages we’ve been discussing—like Python and C++ and their variants—are also gaining followers, especially in sectors where performance is of the essence, like gaming. The most common choices for Integrated Development Environments (IDEs) used in mobile app development are Visual Studio Code, IntelliJ IDEA, and Eclipse. Many alternatives are available, but these three remain at the top. |

## 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**: In addition, the development environment on Windows is very solid, featuring tools like Visual Studio that support a multitude of programming languages and frameworks. This allows for an almost effortless user experience across Windows and Android devices. Moreover, a broad spectrum of libraries and APIs are at developers' disposal to enrich the functionality of their applications and to help them meet user expectations for modern software. In addition, Windows' ubiquity means that the app will touch a lot more people, giving it an even bigger platform for engagement. This is crucial for apps that thrive on community, such as "Draw It or Lose It."
2. **Operating Systems Architectures**: The Universal Windows Platform (UWP) was introduced with Windows 10 as a way to create applications that can run across a wide range of devices, from PCs to tablets to smartphones and Xbox consoles. This evolution of the Windows Runtime model allows developers to build applications that leverage the capabilities of different Windows devices while maintaining a consistent user experience.
3. **Storage Management**: The modern IT infrastructure relies heavily on storage that is server based, and for good reason. It's not just one thing that makes server-based storage desirable; it is a combination of several factors—and these factors relate to both how well server-based storage does its "job" and how dependable it is. Here are some of the main advantages: Centralized Access to Files, Enhanced Functionality, Failover Clustering and Faster, Optimized Performance. All these aspects of server-based storage make it an invaluable asset for any organization.
4. **Memory Management**: The Windows OS offers a well-rounded suite of storage and memory management features that address a range of user needs. One of its most prominent features is Azure Storage. A cloud-based service, Azure Storage allows users to securely and efficiently store and manage copious amounts of data. Unlike many cloud services, which offer a single type of storage solution, Azure Storage provides several—each optimized for different kinds of data and usage scenarios.
5. **Distributed Systems and Networks**: An environment for cross-platform development can significantly simplify the application development process and can considerably reduce the need for specialized skills in several programming languages or frameworks. One prominent example is "Develop 4," which provides tools and resources to make this approach work smoothly. The potential for connectivity or service outage issues can be minimized if the server infrastructure has been built up to—and not just to, but over—the expected user load.
6. **Security**: In today’s digital environment, protecting user data is more important than ever, given the constant danger that cybercriminals pose. One good solution to this problem is Aura, a security service that goes well beyond the most basic, inadequate building blocks of system security. Aura protects users on a multitude of platforms, including PC, Mac, and all kinds of mobile devices. To employ Aura is to ensure not just a baseline security presence but a robust defense that operates around the clock and even the calendar.