

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 | 01/15/2025 | Selva Path | Updated project summary, requirements, constraints, and any other relevant information necessary for successful project execution. |

**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 client aims to transform their existing Android-only game, *"Draw It or Lose It,"* into a multi-platform, web-based game. The goal is to retain the core features of the current Android app while transitioning to a more scalable and cross-platform environment. The client requires assistance with setting up the development environment and streamlining the development process.

The game will need to be developed using a cross-platform approach to ensure compatibility across various platforms such as web, Android, and iOS. It is essential that each game and team name remains unique to avoid duplication, and the game should be designed so that only one instance of the game exists in memory at a time. Additionally, each team will consist of multiple players, and the game must have the capability to support multiple teams per game, whether there is a single team or several teams involved.

## Requirements

The client requires the game to support one or more teams, with each team being assigned multiple players. To ensure proper game management, only one instance of the game should exist in memory at any given time. This will be achieved by creating unique identifiers for each game instance, team, and player. Additionally, the game should allow users to check whether a chosen team name is already in use, ensuring that each game and team name remains unique.

## [Design Constraints](#_2et92p0)

The primary constraint for this project is developing the game across multiple platforms. Due to the lack of team members with cross-platform development expertise, this will require multiple teams working on different aspects of the product based on their specific environment knowledge. Additionally, another key constraint is ensuring that the development meets all client requirements for each platform, maintaining consistency and quality across the various versions.

## [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 tires, 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 serves as the parent (super) class for the **Game**, **Team**, and **Player** classes. As child classes of Entity, the Game, Team, and Player classes inherit the attributes of the Entity class while also having their own unique attributes, separate from those of the parent class.

The **Game Service** class ensures that the client's requirements are met by managing a single game instance at a time using a singleton design pattern. It also ensures that each game, team, and player have a unique identifier (ID), maintaining uniqueness for team names, game names, and player names.

The Program Driver contains the main statement and utilizes the Singleton Tester class. The Game class maintains a list of teams, while the Team class holds a list of players. The Player class does not contain any lists; instead, it ensures that each player is assigned a unique ID, which can be associated with a team. Although a player can belong to a team and a team has players, the Player class does not directly reference or contain information about the team or the game itself.

**"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 the major advantages of macOS is its ability to run macOS, Windows, and Linux apps side by side. Another often overlooked benefit is its consistency. macOS and its system have remained stable over time, making it easy to navigate for users familiar with the OS. However, the main drawbacks of macOS are its high cost and the limited hardware options compared to Windows or Linux. | The main advantages of Linux are its wide variety of distribution options and its open-source nature, offering several free or affordable choices. Linux is particularly renowned for its versatility, especially in server and embedded solutions, thanks to its high degree of customization. Additionally, Linux is known for having superior security protocols when compared to both Windows and macOS.  However, the drawbacks include a limited selection of pre-built machines and potential compatibility issues with certain file formats. | The main advantage of Windows is its seamless integration with Active Directory-based corporate servers, making it an ideal choice for corporate users. This integration comes right out of the box, with no need for add-ons or additional costs.  However, the disadvantages of Windows include limited support for mobility development, as well as weaker security protocols compared to other operating systems. This can make Windows more vulnerable to malware, spyware, and ransomware attacks. | Using a mobile device to host a web application can be beneficial when the user base is small and the application is simple or not resource-intensive. Several low-cost web server applications are available to Android users, which adds to the advantage of using mobile devices for hosting web applications.  However, the main disadvantage is that, in most cases, mobile-based hosting options rely on cloud services, which can expose companies to greater security risks and make them more vulnerable to hackers. Additionally, mobile devices are generally more susceptible to security threats compared to PCs, making them a less secure option for hosting applications in many scenarios. |
| **Client Side** | The advantages of macOS include its ease of use once users have developed the necessary skills and understanding. However, one of its main drawbacks is its limited availability outside of Apple's ecosystem. macOS is only accessible on Apple devices, which can restrict developers who have the knowledge and training to build macOS applications but do not have access to a Mac for development. | The advantages of Linux include its affordability and the control it offers developers over their environment. Being a free and open-source operating system, Linux is typically easier to maintain, requiring less time for upkeep. However, there are also drawbacks to these advantages. With Linux being open-source and user-controlled, security can become a concern, as users may need to address issues on their own, unlike Windows or macOS, which offer dedicated technical support services for their products. | The advantages of Windows include its widespread availability and the flexibility to choose from several price ranges, allowing customization based on project needs. Windows also provides robust technical support and advanced security options, which are often superior to those offered by Linux. However, the drawbacks include the need for expertise in Windows OS and the potential additional costs for specific features or functionalities you may want to include. | While there are many applications and tools available for mobile devices, they often lack the accessibility and full range of features that are typically offered on a PC. The advantages of mobile devices include their wide availability at various price points. However, the main drawback is that mobile devices run on many different operating systems, and each device is often tailored to a specific OS, making it incompatible with others. |
| **Development Tools** | macOS utilizes the Swift programming language. The primary tool available to macOS and iOS developers is Xcode, along with Xcode Cloud. Xcode Cloud is a service designed specifically for Apple developers, providing teams with a faster and more streamlined way to build, test, and deploy applications efficiently. | Linux offers a vast array of development tools to choose from. Due to the sheer number of options, I’ll highlight one: Docker. Docker is widely used to provide a consistent development environment, build cross-platform applications, and streamline deployment. Docker Hub further simplifies the process by allowing developers to skip the environment setup and dive straight into development. | Windows was primarily written in C, with some components written in assembly language. One of the most widely known and used IDEs for Windows is Visual Studio. It serves not only as an integrated development environment (IDE) but also as a code editor, and in some cases, a source/version control tool. Visual Studio offers a wide range of tools, and although I was initially slow to recognize their benefits, I now use it as my primary development platform for both school and work. | For most mobile applications, Java is the go-to language due to its object-oriented features, making it an optimal choice for many developers. However, Python, C++, and other C-based languages are also used, particularly in mobile game development. While there are countless IDEs available for mobile app development, the most popular ones include VSCode, IntelliJ IDEA, and Eclipse. Personally, I’m not a fan of Eclipse, as I find it a bit cluttered. Instead, I prefer using Visual Studio with Xamarin, which allows me to develop cross-platform native applications. |

## 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 the recommended operating system for this project, as it is the optimal choice for integrating with the current Android application, *"Draw It or Lose It."* With Windows commanding a large user base, comprising around 90% of global OS users, it provides access to a broader range of developers, skill sets, and tools, all of which will be instrumental in successfully completing the project as a cross-platform application.
2. **Operating Systems Architectures**: Windows 10 introduced the Universal Windows Platform (UWP), which builds upon the Windows Runtime model. UWP applications are capable of calling Win32 APIs, the Microsoft .NET Framework, and Windows RT APIs. This allows developers to create a single app that can run seamlessly across all Windows devices, leveraging a dual-stack approach. UWP is the recommended architecture for this project due to its versatility and cross-device compatibility.
3. **Storage Management**: Cloud-based storage enables centralized access to files, enhancing functionality and supporting failover clustering. It also offers redundancy, automated backups, and optimized performance for faster and more efficient operations.
4. **Memory Management**: Windows provides a range of storage and memory management options, including Azure Storage. The operating system offers both virtual and physical address space for memory allocation. Additionally, services like OneDrive, Visual Studio, and Azure Cloud can be utilized to store and manage different versions of files and projects.
5. **Distributed Systems and Networks**: Using a cross-platform development environment can minimize the need for multiple areas of expertise, leading to a more streamlined application development process. One such option is *Develop 4.* To address potential connectivity or outage issues, I recommend ensuring that the servers are built with sufficient capacity to meet the client’s needs. This should be based on forecasting game usage and user volume once the application is launched in the new environment.
6. **Security**: Given the constant and evolving threat of data theft, I highly recommend placing special emphasis on security protocols. One service that provides protection for PC, Mac, Android, and iOS devices is Aura. While it comes at a cost, relying solely on the standard security features included with an operating system is not advisable; an additional layer of protection is essential to keep systems secure and running efficiently. Additionally, Aura offers 24/7 U.S.-based customer support, which is a significant advantage of this service.