

Draw It or Lose It

# **CS 230 Project Software Design**

Version 3.0

## 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 | 11/24/24 | Michael Wood | updated requirements, constraints, summary, and any relevant data required for this project. |
| 2.0 | 11/25/24 | Michael Wood | system architecture perspective and updated recommendations. |
| 3.0 | 12/13/24 | Michael Wood | updated data that explains how to build the architectures according to the needs of the client and the architecture requirements for the various operating platforms. |

**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 wants to turn their current game, "Draw It or Lose It," which is only available as an Android application, into a web game that runs on several platforms. Due to the game’s success and popularity on the Android OS, they are now seeking to expand their user base to include Windows, Linux, and MacOS. They are looking for assistance with setting up the environment and expediting their development process, and they want the game to be based on the existing Android application. Cross-platform development will have to be used to create the game. Only one instance of the game should be able to exist in memory at a time, and each team and game name must be distinct. many players will be assigned to each team, and the game must support either single teams or many teams playing at once.

## Requirements

The client demands that the game allow for the participation of one or more teams, with several players assigned to each team. There should only be one instance of the game running at once. They ask that each player, team, or game instance be given a unique identification to achieve this. All games and teams should have unique names since the user needs to be able to check if a name is already being used when selecting a team name. The game must have expandable storage and memory allocation that factors run-time into consideration.

## [Design Constraints](#_2et92p0)

Creating the game on several platforms is the primary limitation of this project. Several teams will need to create distinct products based on environmental knowledge if team members are not experienced in cross-platform development. The other limitations include making sure the development satisfies every client's request on every platform.

## [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 Game, Team, and Player classes are all subclasses of the Entity Class. This implies that as child classes of an entity, the Game, Team, and Player classes will all inherit the attributes of the entity while also receiving unique attributes distinct from those of the parent class. By offering a single game instance at a time, a unique team name (id), a unique game name (id), and a unique player name (id), the Game Service Class helps to guarantee that the client's needs are satisfied. The main statement is in Program Driver, which makes use of the Singleton Tester class. The Game class holds a list of teams, while the Team class contains a list of players. The Player class does not include a list since it ensures that each player has a distinct id that can be given to a team. While a player might be a member of a team, and teams do have players, the player class does not contain or have a team or 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 must 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** | MacOS has a significant benefit in that it can run MacOS, Windows, and Linux software simultaneously. MacOS also has the virtue of being consistent, which is sometimes underestimated. MacOS and its system have stayed consistent, making it simple to use for people who are accustomed to the OS. The disadvantages of MacOS include its high cost and limited hardware possibilities as compared to Windows or Linux. | The main advantages of Linux are its diverse distribution options and its open-source nature, with several free or low-cost solutions. Linux is well-known for its adaptability in servers and embedded solutions, thanks to its ease of customization. In terms of security mechanisms, Linux outperforms Windows and MacOS. The downsides include a limited selection of pre-built workstations and file format compatibility difficulties. | The primary benefit of Windows for business users is that Windows-based authentication automatically connects with corporate servers that use Active Directory; no further costs or add-ons are required. Windows's lack of mobile development and its lax security measures, which can result in malware, spyware, and ransomware, are its drawbacks. | If the number of users is small and the application is not vast or complicated, using a mobile device to host it can be useful. Android users have access to several low-cost web server software, which is beneficial when hosting web applications on mobile devices. The negative is that most mobile-based hosting choices are cloud-based, which might make businesses more vulnerable to hackers. Mobile devices are also generally more vulnerable than computers. |
| **Client Side** | The advantages of MacOS include its simplicity of usage after knowledge and abilities are acquired. Its availability outside of Apple OS is a drawback. Because it can only be accessed on an Apple device, developers who may possess the skills and expertise necessary to create MacOS programs but do not use or work on a Mac device may be limited. | Pros include Linux's autonomy over your advancements and its price. Linux often requires less time for maintenance because it is an open-source, free operating system. Each of the advantages has drawbacks as well. Compared to Windows or MacOS and their technical support services for their products, Linux's open-source and user-controlled nature may make security a problem that only it faces. | Windows has the advantage of being widely available and available at a variety of pricing points that may be tailored to the needs and usage of a project. In contrast to Linux, they provide additional security choices and professional assistance. Cons include the need for a Windows OS specialist and the potential expense of adding any features you might want. | Although there are a lot of tools and programs for mobile devices, they typically do not have the same accessibility and comprehensive functionality as PCs. Advantages include their wide availability and range of prices. The drawbacks include that mobile devices come with a wide variety of operating systems, and most of them are designed to work with a single OS, making them incompatible with other systems. |
| **Development Tools** | Swift is the programming language used by MacOS. For MacOS and iOS developers, Xcode and Xcode Cloud are the primary tools that are easily accessible and used. A service called Xcode Cloud was created especially for Apple developers to give teams a quicker and easier way to create, test, and release apps more effectively. | A vast array of development tools is available for Linux. Since there are a lot of options, such as Docker. Docker provides a consistent development environment, facilitates the development of cross-platform programs, and facilitates deployment. Additionally, Docker Hub helps customers bypass the setup of the development environment so they may start working on projects immediately. | With a small amount of assembly language, Windows was mostly written in C. Visual Studio is arguably Windows' most well-known and popular IDE. An IDE, code editor, and occasionally source/version control are among its uses. Visual Studios offers a lot of capabilities, most of which I am just now starting to discover. It is the primary platform that I personally chose to work with. | Java is the language of choice for many mobile apps. Because of its object-oriented capabilities, it is the best option for many developers who are making mobile applications. However, alternative C languages, like Python and C++, are widely used for mobile development projects, especially those in the gaming industry. Although there are many IDEs that facilitate the creation of mobile apps, VSCode, IntelliJ IDEA, and Eclipse are the most widely used. Since it is a little too cluttered, Eclipse is not my thing. Instead, I prefer Visual Studio with Xamarin, which enables me to create native cross-platform apps. |

## 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 this project, Windows is the suggested operating system. Because Windows can interact with their present Android application, "Draw It or Lose It," it is the best option. The vast user base of Windows, which accounts for around 90% of all operating systems globally, will provide a greater variety of developers, skill sets, and tools to finish the project for a cross-platform application.
2. **Operating Systems Architectures**: Windows 10 improved the Windows Runtime architecture by introducing the Universal Windows Platform (UWP). Windows RT APIs, Microsoft.NET Framework, and Win32 APIs can all be called by applications made for UWP. This dual stack technique is the suggested design for this project since it allows developers to produce a single software that works on all devices.
3. **Storage Management**: Given that Microsoft integrates perfectly with Windows, Microsoft Azure is highly recommended for this project. This storage management system is the clear choice due to its low cost, competent and readily available customer service, and consistent upgrades.
4. **Memory Management**: Windows provides a multitude of storage and memory management options. For example, consider Azure Storage. The operating system comprises both virtual and physical address space for memory allocation. There is also the option of using OneDrive, Visual Studio, or Azure Cloud services to store and manage versions. Using industry-standard best practices when building the application will aid in runtime and memory allocation and management, particularly when comparing storage container types, such as those that do not allow memory expansion to those that can. Windows has demonstrated with recent releases that it intends to continue to improve its memory management capabilities.
5. **Distributed Systems and Networks**: As I previously stated, Azure's benefits and applications extend beyond storage management. The Azure Cloud service provides simple ways for accessing distributed systems and networks. Azure ensures optimum uptime using cloud-based email warnings, App Insight Logging, and Monitoring Service. Azure services, for example, will allow the client to more easily keep up with their users' demands by automating rather than manually doing operations. This will eventually reduce the time and effort spent on network loads, allowing them to devote more attention to the application and its features.
6. **Security**: From the beginning, one of the top goals has been security. Our staff has received and will continue to receive ongoing security awareness training. The safety factor will also be increased if static code analysis tools and well-maintained libraries and frameworks are used. We need to keep abreast of software vulnerabilities and steer clear of certain behaviors. Penetration testing must be done, and secure coding standards and rules must be adhered to. These are some methods for safeguarding the client's software. In addition, I advocate standardizing permission processes, such as limiting access to information only, when necessary, rather than granting everyone the same level of access or giving everyone access to everything. With the foregoing in mind, and given my previous Azure recommendations, I recommend that the customer use Azure's App Service via their App Service Plan, with Active Directory for login. Aura is a service that provides protection for PCs, Macs, Linux, Android, and iOS. While this may incur a fee, using the baseline security that comes with an operating system is not recommended, as an additional layer of protection is required to keep systems safe and running smoothly. Aura also provides 24/7 U.S.-based customer assistance, which is a significant advantage for this business. Another concern is ensuring that corporate personnel understand the significance of password security protocols and adopting an authentication process in conjunction with authorization.