

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 | 09/22/24 | Alex Zumbado | Initial Design |
| 1.1 | 10/06/24 | Alex Zumbado | Executive Summary, design constraints, domain model |
| 1.2 | 10/12/24 | Alex Zumbado | Client side, server side, development tools, 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)

Draw It or Lose It will be a web-based game loosely based on the 1980s television game Win, Lose or Draw. Players will access a library of stock images to provide as clues in four one-minute rounds where they will guess the puzzle before time expires. The remaining teams have 15 seconds to offer one guess each to solve the puzzle. The Gaming Room needs help setting up the environment and has requested help in streamlining the development of the app. A software design document is required along with beginning development of the app. These will later inform hardware requirements.

## Requirements

* *A game will have the ability to have one or more teams involved.*
* *Each team will have multiple players assigned to it.*
* *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
* *Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.*

## [Design Constraints](#_2et92p0)

Web-based distribution – This brings network connectivity, security, and compatibility concerns into the constraints where the game needs to be able to be safely connected to on any network from a range of different browsers and devices.

## [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)

In the UML diagram, the Entity superclass along with various attributes and methods that the subclasses all have in common which extends to the three main subclasses, game, team, and player. Each game will have multiple teams and each team multiple players. The GameService class has a composition relationship to the Game class which manages instances of the Game class. The Game class likewise has a composition relationship to the Team class which itself has one to the Player class. The ProgramDriver class holds the main function. Through this class the GameService singleton is created so that only one instance of the game will exist at a given time. The ProgramDriver class is dependent on the SingletonTester 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** | MacOS is built on Unix. Unix is known for being very secure. Mac can work with many different programming environments. The GUI for Mac is known to be very good. Mac is not specifically designed for web hosting which can cause difficulties especially at scale. Mac hardware is typically more expensive. Businesses typically are not using Mac for web hosting which could cause compatibility issues. | Linux is a very popular web hosting OS. This means there is going to be a lot of support resources for it. Linux is free and open-source. Users can configure Linux to their particular needs which can boost efficiency. Linux is based on Unix which is known for its security. Linux is widely used in web hosting and has a wide variety of compatibility with hardware and technologies in the space. Linux has been known to be a bit on the complex end of the spectrum and the command line interface can be a challenge for some. The GUI for Linux is not fantastic. | Windows is the best choice when working in the Microsoft environment. The GUI is user-friendly. Windows is well integrated within the entire Microsoft ecosystem of tools, support, services, and the like. Windows is widely used in the business world and as a result enterprise-level support is available. Windows typically needs a more built out setup for web hosting requiring more components which cost money and take up space. There are many licenses required for Windows which all together can cost a decent chunk of money. Security issues have been an issue with Windows. Not a great option when running non-Microsoft technology. | Mobile devices realistically are not a good option for web hosting. In theory they could be used for testing especially if you are on the go, but the hardware capabilities ultimately bring a major roadblock. |
| **Client Side** | Software development costs associated with MacOS are high on the hardware side with some licensing costs depending on what is needed. Apple has guidelines for native MacOS applications which can add additional development time and costs in order to comply with them. Developers need expertise in Xcode, Swift, Objective-C and Apple’s APIs. | Linux is free and open source so costs are going to be very low. There may be some hardware costs but compared to other OSs the savings are still very high. Since the Linux ecosystem is fragmented there may be additional time concerns when making apps compatible across the ecosystem and deploying. Expertise in CLI, GTK+, Qt, Flutter, React, and POSIX APIs | Windows has high licensing costs particularly when developing applications at the professional or enterprise level. Development time is mostly dependent on the complexity of the application that is being developed. Expertise in C#, .NET, DirectX, and Win32 APIs are required. | Developing applications for mobile devices comes with some distribution costs. Extra time for testing and developing mobile compliant applications. Expertise is required in mobile app development and deployment. |
| **Development Tools** | Xcode, Swift, Objective-C, and Apple APIs are the main considerations for MacOS although others may be needed for specific use cases. | CLI, GTK+, Qt, Flutter, React, and POSIX APIs the main considerations for Linux although others may be needed for specific use cases. | C#, .NET, DirectX, and Win32 APIs are the main considerations for MacOS although others may be needed for specific use cases. | The development tools required for mobile depends on which mobile devices are being developed for. Generally, Xcode, Objective-C, Swift, Kotlin, Java, Android Studio, React, Flutter, and C# are required. |

## 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**: I would recommend using Windows since The Gaming Room seems to not have much expertise in software development and the setup for web-hosting. Windows is very common and user-friendly with a solid support system in place. Windows will integrate with the additional Windows tools and products that they likely already have. I think the main downside would be the licensing and hardware fees.
2. **Operating Systems Architectures**: The Windows architecture is set up with the operating system kernel as the base which then goes to system services, and ultimately apps.
3. **Storage Management**: I would recommend Windows Storage Spaces, Windows Storage Spaces is scalable, has support for configurations that improve redundancy, and can dynamically allocate disk storage space. For a business like The Game Room I think this provides all they would need while allowing the setup to scale if they need more.
4. **Memory Management**: Windows OS gives each 32-bit process its own virtual address space. This allows up to 4GB of memory to be viewed. For 64-bit processed this is upped to 8TB. All threads have access to the visible address space of the process, but the do not have access to the memory of other processes which protects them from being damaged.
5. **Distributed Systems and Networks**: APIs can allow different platforms to interact with each other, though this method we can ensure that all uses are able to connect regardless of their device or browser. Replication and redundancy of data can ensure that the data is stored even in the case of one platform or connection going down.
6. **Security**: Data encryption can provide security across distributed systems and networks. Additionally, authentication methods can ensure safety.