

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 | 11/13/24 | Logan Boyd | Initial information added per client request. |
| 1.1 | 11/29/24 | Logan Boyd | Update Evaluation for better clarity and specific information. |
| 1.2 | 12/20/24 | Logan Boyd | Finalize recommendations for final submittal. |

**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 wants us to develop a web-based version of their game, Draw it or Lose it, that works across multiple platforms, including macOS, Linux, Windows, and more. The current game is only available on Android. The game is similar to Pictionary, where teams compete to guess images being drawn, with limited guesses and four rounds lasting about a minute each.

## Requirements

* Must be able to operate on multiple operating systems.
* Must function the same as the current Android version
  + Limited Guesses
  + 4 Rounds
  + 1 Minutes Each Round

## [Design Constraints](#_2et92p0)

* Multiple Operating Systems: Code will need to be written universally to be deployed across multiple systems.
* User Interface/Design: The game will need to have a similar UI as the current version of the game running on Android. The game will have to be accessible on many different models of phones, computer and more.
* Unique Game ID: Each instance of the game must have its’ own ID to limit the amount of games being ran to only one.
* Cross Platform: The game must allow teams from other platforms.

## [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 contains the main method and uses directed association with SingletonTester to check if an instance of GameService already exists for unique IDs. The Entity class serves as the parent class for the 3 classes: Game, Team, and Player, with each inheriting the required attributes from Entity. A Player can belong to a Team, and a Team can have Players. A Team can participate in a Game, and a Game can involve Teams. A Game can be managed by a GameService, and a GameService can oversee multiple Games. GameService must ensure that only one instance of each game is running at a time, and each Game can include multiple Teams. Each Team can have multiple Players at any given time.

**"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** | Stable hosting, less common. Advantages: Stability, strong ecosystem. Weaknesses: Fewer community resources. Licensing Costs: High, due to proprietary nature and Apple hardware. | Popular, highly stable. Advantages: Stability, security, low cost. Weaknesses: Requires technical expertise. Licensing Costs: Low to none, open-source. | User-friendly, strong support. Advantages: Ease of use, large user base. Weaknesses: Higher costs. Licensing Costs: High, due to Windows Server licenses. | Not typically used for hosting. Advantages: Useful for testing applications. Weaknesses: Not suitable for hosting. Licensing Costs: Not applicable. |
| **Client Side** | Requires design aesthetics and user experience focus. Cost: High, Apple hardware. Time: Moderate to high. Expertise: macOS development tools, design principles. | Uses open-source tools and frameworks. Cost: Low. Time: Moderate. Expertise: Linux development tools. | Benefits from a wide range of development tools and a large user base. Cost: Moderate to high. Time: Moderate to high. Expertise: Windows development tools. | Requires understanding of iOS and Android platforms. Cost: Moderate to high. Time: High. Expertise: iOS and Android development tools. |
| **Development Tools** | Xcode, Swift, Objective-C. Impact on Development Team: Requires macOS expertise. Licensing Costs: High, due to Apple hardware. | Eclipse, NetBeans, text editors. Impact on Development Team: Requires Linux expertise. Licensing Costs: Low, open-source. | Visual Studio, C#, .NET, Visual Basic. Impact on Development Team: Requires Windows expertise. Licensing Costs: High, Visual Studio licenses. | Xcode (iOS), Android Studio (Android). Impact on Development Team: May need multiple teams. Licensing Costs: Moderate to high. |

## 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**: Although it may be the most expensive option, I believe that Windows will allow us the most benefit as it has a large amount of IDEs and is compatible with many other operating systems.
2. **Operating Systems Architectures**: Windows supports several architectures, each offering unique benefits. The x86 architecture is known for its versatility and broad compatibility with various software and hardware, making it a reliable choice for many applications. The x64 architecture builds on this by allowing Windows to handle more RAM and run more demanding applications, which is crucial for tasks that require significant memory and processing power.
3. **Storage Management**: For the Windows operating platform, I recommend New Technology File System. NTFS offers advanced features such as file compression, encryption, disk quotas, and improved metadata support. It also provides better security with file and folder permissions, which is essential for protecting sensitive data.
4. **Memory Management**: Windows uses a few ways to manage memory. Virtual memory allows the system to use disk space to extend RAM, letting larger applications run smoothly. Paging efficiently swaps data between RAM and disk storage, while caching speeds up access to frequently used data. These methods work together to ensure the software runs efficiently and responsively.
5. **Distributed Systems and Networks**: To make "Draw It or Lose It" work across different devices, we need a good network and software setup. This can be done using a system where one device (the server) helps other devices (the clients) connect and share information. Two key things we will need are a stable internet connection to keep everything synced between the devices. We will also need good error handling to fix any issues if the connection were to drop.
6. **Security**: To keep user information secure across different platforms, several key practices are important. Windows has built-in security features like encryption to protect data when it's stored or being transferred. Multi-factor authentication helps verify user identities and prevent unauthorized access. Windows also uses advanced threat protection and regular security updates to defend against cyber threats.

Harwood, R., Gerend, J., & Barnett, J. (2024, November 01). *NTFS overview.* Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview

Stack, E. o. (2022, November 14). *The Best Operating System for Programming: Windows, Mac, Linux in 2023?* Retrieved from Medium: https://medium.com/@cannon\_circuit/the-best-operating-system-for-programming-windows-mac-linux-1d9091d85c48