

# 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 | 03/22/24 | Joseph Ruiz | Initial version of the software design template |

**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 software design template for the CS 230 Project "Draw It or Lose It" provides a structured approach to developing a web-based game application. The template outlines the requirements, design constraints, system architecture view, domain model, evaluation, and recommendations for the project.

## Requirements

The game application "Draw It or Lose It" must support multiple teams with multiple players. Game and team names must be unique to avoid duplicates. Only one instance of the game should exist in memory at a given time.

## [Design Constraints](#_2et92p0)

## The design of the game application must adhere to the requirements specified by the client. The use of the Singleton design pattern is suggested to ensure the single instance requirement is met. Additionally, the application should be scalable to support potential future enhancements.

## [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 domain model of the game application includes classes such as Game, Team, Player, and GameService. These classes are interconnected to represent the structure of the game and facilitate the management of teams and players during gameplay.

**"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** | Characteristics: Mac is a proprietary operating system developed by Apple, renowned for its user-friendly interface.  Advantages: With a user-friendly interface, Mac is well-suited for multimedia creation and graphic design tasks.  Weaknesses: Mac may have limited gaming options and less customization compared to Linux. | Characteristics: Linux is a free and open-source operating system known for its stability and adaptability, offering extensive configuration possibilities.  Advantages: Linux is mostly free and open-source, providing a stable platform with a wide range of configuration options.  Weaknesses: Linux may have a limited range of software and can pose significant challenges for users with limited IT knowledge. | Characteristics: Windows is a proprietary operating system developed by Microsoft, widely used on personal computers, laptops, and servers.  Advantages: Windows is versatile, supporting general productivity tasks, gaming, software development, and business applications, with compatibility for a wide range of hardware and software.  Weaknesses: Windows may be more susceptible to security vulnerabilities than Linux and may have stricter limitations on customization. | Characteristics: Mobile devices offer portability and accessibility for web-based applications, making them ideal for on-the-go use.  Advantages: Mobile devices allow for reaching a wider audience and support mobile-friendly designs for enhanced user experience.  Weaknesses: Mobile devices have limitations such as smaller screen size and processing power compared to desktops, requiring optimization for different devices to ensure smooth performance. |
| **Client Side** | Important software development factors to consider for supporting multiple types of clients on Mac include cost, time, and expertise. It is crucial to evaluate the cost of software development, ensuring it aligns with your budget. Time is another key factor, as efficient development processes can lead to quicker deployment. Additionally, expertise plays a vital role in ensuring the quality and success of the software project. By considering these factors, you can make informed decisions when developing software for various clients on Mac. | The common features of software applications include pointers, toolbars, and buttons. The three main types of software maintenance are corrective maintenance, adaptive maintenance, and perfective maintenance. When supporting multiple types of clients on a Linux platform, key software development factors to consider are cost, time, expertise, compatibility, scalability, security, and user experience. | When developing software to support different types of clients on the Windows platform, key considerations that need to be carefully evaluated include the overall cost involving licensing fees, infrastructure expenses, and personnel salaries; the timeline required for gathering requirements from various clients, completing development and testing cycles, deploying the software, and maintaining it over its lifetime; and the expertise necessary in areas such as the chosen programming languages and frameworks, the Windows development environment, security and performance best practices, understanding diverse client domains, and planning for future compatibility, integrations, and upgrades as client needs evolve. | Key software development factors that need to be considered when supporting multiple types of clients on mobile devices include the scope of the project, customization needs, support offered by the software vendor, type of business model, development platform, device compatibility, app functionalities, security, testing, hosting, maintenance, and the expertise of the development team. These factors can influence the total cost, time, and expertise required for successful mobile app development. |
| **Development Tools** | For developing applications on macOS, the primary tools used are Xcode (Apple's Integrated Development Environment), along with the programming languages Swift and Objective-C. | For developing applications on Linux, the primary tools used are text editors or IDEs like Visual Studio Code, Eclipse, or IntelliJ IDEA, along with programming languages such as C++, Python, Java, Go, and Rust, as well as build tools like GCC, Make, and package managers for installing dependencies. | For developing applications on Windows, the primary tools used are Visual Studio (Microsoft's Integrated Development Environment), along with programming languages such as C++, C#, Visual Basic .NET, and .NET Framework or .NET Core, supplemented by source control tools like Git and build automation tools. | The programming languages commonly used to create software for mobile devices include Java, Kotlin, C#, and C++. IDEs such as Android Studio and Xcode are popular tools for mobile app development. |

## 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 adopting the Microsoft Windows operating system for expanding the Draw It or Lose It game to other computing environments. Windows is cross-platform, widely available on various devices, has a large market share, and provides a familiar user interface. Additionally, it offers a range of development tools and frameworks that can simplify multi-platform deployment.
2. **Operating Systems Architectures**: Windows supports both 32-bit and 64-bit architectures, allowing it to run on a wide range of hardware configurations. The latest versions of Windows are based on a hybrid kernel architecture that combines elements of monolithic and microkernel designs, providing a balance between performance and modularity.
3. **Storage Management**: The recommended storage management system for the operating platform mentioned in the previous question could be Pure1®, which is a data storage management solution that delivers a Modern Data Experience by bringing the convenience of cloud storage to on-premises storage.
4. **Memory Management**: The recommended operating platform utilizes memory management techniques such as single contiguous allocation, partitioned allocation, and paged memory management to ensure efficient memory allocation and utilization for the Draw It or Lose It software. These techniques help in managing primary memory effectively, allocating memory to processes, and optimizing memory usage for the software's operations.
5. **Distributed Systems and Networks**: Draw It or Lose It can effectively communicate between different platforms using distributed software and network connectivity by ensuring that the software components are shared among multiple computers or nodes. This allows the systems on different networked computers to communicate and coordinate by sending messages back and forth to achieve a defined task. Additionally, utilizing a distributed network that connects all components together enables seamless communication and coordination between different platforms.
6. **Security**: To protect user information on and between various platforms, implement strong encryption for data in transit and at rest. Use secure authentication methods, like multi-factor authentication, to verify user identity. Regularly update and patch the software to fix vulnerabilities. Additionally, leverage the security features of the recommended operating platform, such as built-in firewall and anti-malware tools, to enhance protection against threats.