

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 |
| --- | --- | --- | --- |
| 1.1 | 04/01/24 | Joseph Ruiz | Updated version of the software design template |
| 1.2 | 04/20/24 | Joseph Ruiz | Updated version of the 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)

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 not as commonly used for server environments compared to Linux and Windows.  Advantages: Mac may offer ease of integration with other Apple devices and services.  Weaknesses: Mac may not be as optimized for server hosting compared to Linux and Windows.  Mac, while less commonly used for server hosting, may provide ease of integration with other Apple products | Characteristics: Linux is commonly used for server environments due to its stability, security, and scalability. It has a wide range of server software available.  Advantages: Linux is cost-effective for server deployments as it is open-source and often used in high-traffic websites.  Weaknesses: Linux may require more technical expertise to set up and maintain server configurations.  Linux is often preferred for its cost-effectiveness and scalability | Characteristics: Windows Server is a popular choice for hosting web-based applications due to its compatibility with Microsoft technologies.  Advantages: Windows Server has a user-friendly interface and is well-suited for organizations familiar with Windows environments.  Weaknesses: Windows Server may have higher licensing costs compared to Linux.  Windows may offer a more user-friendly interface for organizations familiar with Microsoft technologies | Characteristics: Mobile platforms are not typically used for server hosting of web-based applications.  Advantages: None specific to web-based server hosting for mobile platforms.  Weaknesses: Mobile platforms may not have the necessary server software and configurations for hosting a web-based application.  For mobile platforms like iOS and Android, there are typically no licensing costs for the server operating system. However, if the web-based application needs specific server-side technologies that require licensing fees |
| **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 Mac development, the relevant programming languages are Objective-C and Swift, with Xcode as the main IDE and tool.  Mac development using Xcode requires knowledge of Apple's development ecosystem and design guidelines, potentially requiring multiple development teams for specialized knowledge in iOS/macOS app development.  Xcode is free to download and use for Mac and iOS development. | For Linux development, the relevant programming languages include C, C++, Python, with IDEs and Tools such as Eclipse, IntelliJ IDEA, and Visual Studio Code.  The impact on the development team for Linux development is a requirement for a good understanding of system programming and working with command-line interfaces, potentially necessitating multiple development teams for different expertise levels.  Most Linux development tools are open-source and free to use, reducing licensing costs for the development team. | Windows development involves programming languages like C++, C#, Java, with IDEs and Tools including Visual Studio, Visual Studio Code, and Eclipse.  Developing for Windows may require knowledge of different programming languages, potentially necessitating multiple development teams for different language requirements.  Visual Studio has licensing costs for commercial use, but Visual Studio Code and Eclipse are free to use. | For mobile platforms (iOS and Android), the relevant programming languages are Swift for iOS, and Java, Kotlin for Android, with IDEs and Tools like Xcode for iOS and Android Studio for Android.  Mobile development requires knowledge of mobile app development frameworks and design patterns, potentially requiring multiple development teams for iOS and Android platforms.  Xcode is free for Mac users, and Android Studio is free for all users, with potential licensing costs associated with publishing apps on the Apple App Store and Google Play Store. |

## 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**: To expand Draw It or Lose It to other devices and operating systems, I suggest using cloud-based servers like Amazon Web Services or Microsoft Azure. These platforms make it easy to scale the game and ensure it works smoothly on Windows, Mac, iOS, and Android without needing major changes. Cloud servers also provide security and backup features to keep the game running smoothly. This will help The Gaming Room reach more players and keep the game running well on different devices.
2. **Operating Systems Architectures**: Operating platform architectures, like Amazon Web Services or Microsoft Azure, use layers of services to support games. They provide virtual servers to run the game, storage and security services to manage data, and tools to monitor and manage performance. These platforms make it easy to run games like Draw It or Lose It on different devices and operating systems.
3. **Storage Management**: For the storage management system with the recommended operating platform, it's best to use Amazon S3 or Azure Blob Storage. These are cloud storage services that can handle lots of data for a game like Draw It or Lose It. They keep data safe and organized, and work well with the operating platform. Plus, they have flexible pricing options based on usage.
4. **Memory Management**: The recommended operating platform for Draw It or Lose It software uses memory management tricks to handle memory effectively. This includes getting and giving back memory as needed, cleaning up unused memory bits, and using disk space as extra memory when needed. These tricks help keep things running smoothly and make sure the software works well for users.
5. **Distributed Systems and Networks**: To make Draw It or Lose It work on different devices, we need to connect them through a system of servers, clients, and networks. If the network has issues, it could disrupt communication, so we need backup plans. We also need to set up communication rules using standard protocols like HTTP. Adding security measures like encryption and firewalls will keep data safe. By managing these connections and ensuring everything stays connected and secure, Draw It or Lose It can work on different platforms smoothly.
6. **Security**: To keep user information safe on different platforms for Draw It or Lose It, we can use encryption to scramble data and control who can access it with secure passwords and authentication methods. Making sure APIs are secure and choosing a safe operating platform that gets regular security updates will also help protect user information. Regular security checks and monitoring for any suspicious activity can further enhance the overall security of the application. By implementing these measures, Draw It or Lose It can ensure user data is protected across various platforms.