

<Name-of-Software-Application>

# **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 | 12/17/23 | Katelynn Thompson | An overview of the recommended software and operating system we should use to effectively develop Draw It or Lose it. |

**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)

Our new client, the Gaming Room, wants us to develop a web-based version of their game that can be played on multiple platforms. The game they have decided to base their project on is Draw It or Lose It, which is currently only available on Android devices. They aim to make the game available for other operating systems such as MacOS, Linux, and Windows. The game itself is a fun and engaging team-based guessing game. Players are presented with a series of images that are being drawn, and they must guess what the image is before the time limit runs out. The game also features a time limit for the drawing process itself. Each round of play lasts for one minute, and there are four rounds in total. The Gaming Room is looking to expand the existing game by adding new features, improving the graphics, and making it more accessible to players worldwide.

## Requirements

## [Design Constraints](#_2et92p0)

As we consider the design constraints for the game application, there are a few key factors to keep in mind:

1. To make the game available on different platforms, we could translate the current Android app into a web app accessible from other operating systems. We could use a REST API to communicate via HTTP to ensure platform compatibility.
2. The user interface should be similar, if not identical, to the current Android app.
3. We need to ensure that the game can support multiple teams and players per game with unique IDs.

To accomplish this, we would need to use a client-server architecture to ensure the server can handle multiple players simultaneously. It is also essential to ensure that games and team names are unique and that only one game can exist in memory at any given time. We require unique games, teams, and player IDs to achieve this. Additionally, we must determine the best approach to manage memory allocation, considering the application's availability on multiple platforms. We would also need to identify player platform IDs.

## [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)

This UML diagram provides a visual representation of the program that will be used to develop Draw It or Lose It. The Entity class acts as a superclass, forming a relationship between the Game, Team, and Player classes. This allows them to inherit information and attributes from the Entity class. The ProgramDriver uses the SingletonTester to test the code, ensuring that the program follows the requirement of having only one game per instance in the memory. All the methods that form the structure and functionality of the game will be contained in the GameService class. To ensure uniqueness in games, teams, and players, separate classes have been designed for each in this diagram. The lines connecting each class imply their association, while the numbers indicate the number of associations within each class. This diagram will guide the development process, and some adjustments may be made as needed to create the final product.

**"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 known for its easy accessibility and server configurability, with an intuitive graphical user interface and flexible terminal commands. A MacOS server can benefit Mac clients/users on a network if it offers full support for Mac applications and streamlined administration through a user-friendly graphic interface. One of the disadvantages, however, is that MacOS has a limited scale compared to other platforms due to its Apache web server, which can hinder growth and make it less suitable for hosting larger platforms/applications; it can also be expensive and isn't as widely used compared to Windows and Linux. | Although I have yet to use Linux, it's well known for its stable, secure, and flexible web hosting platform that offers a variety of server distributions to tailor to different needs. It supports various programming languages and has built-in security features to protect against cyber threats. While it can be complex for beginners, Linux's scalability, customizability, reliability, and affordability make it a popular choice for businesses of all sizes. | Windows is popular for hosting web-based software applications due to its user-friendly interface and compatibility with Microsoft devices and applications. It offers different hosting options tailored to the needs of businesses and provides robust security features. However, it may only be the best fit for some businesses due to its higher cost and resource requirements. | Mobile devices are convenient for hosting web-based software applications but have limitations. Their processors and browsers can support web-based applications but need more resources to run complex software and handle large amounts of data. Mobile devices are also more vulnerable to cyber-attacks and have limited connectivity options. They are not practical for continuous use and can overheat. It's recommended to use dedicated servers or cloud-based hosting services for hosting web-based software applications. |
| **Client Side** | Developing software for Mac clients can be a complex process that requires specialized tools, hardware, and knowledge. This can lead to increased development costs and time. Moreover, the software architecture and user interface differences may demand extra effort to ensure that the software meets Apple's design guidelines. Therefore, having developers with expertise in Objective-C, Swift, and Apple's Cocoa framework is essential for creating functional and user-friendly software for Mac clients. In conclusion, it is crucial to carefully consider these factors to meet the specific needs of end-users. | When developing software for multiple types of clients that run on Linux, it is essential to consider various factors. These factors may include the cost, time, and expertise required to support different types of clients. One of the critical considerations is the choice of programming languages and frameworks that can help other Linux distributions. It is also essential to understand the client's hardware and software environments, as well as their specific needs and requirements. Additionally, developers must ensure that the software is compatible with various versions of Linux and can run smoothly on different types of hardware. Overall, carefully considering these factors is necessary to create functional and reliable software for multiple clients running on Linux. | Developing software for multiple Windows clients requires careful cost, expertise, and time consideration. The cost may vary depending on the client's requirements and organization size. At the same time, developers need a broad range of skills to ensure software can work on different Windows environments and hardware. Overall, considering these factors is crucial for creating reliable software. | Developing software for multiple mobile devices requires careful cost, expertise, and time consideration. The cost may vary depending on the device's requirements and organization size. At the same time, developers need a broad range of skills to ensure software can work on different mobile operating systems and hardware. Considering these factors is crucial for creating reliable software that can be used across various mobile devices. |
| **Development Tools** | To develop software for Mac, commonly used programming languages and tools include Swift, Objective-C, HTML, Interface Builder, JavaScript, Git, and Homebrew. | For developing software for deploying on Linux, commonly used programming languages and tools include C, C++, Python, Java, JavaScript, and Eclipse, as IDEs. | To develop software for Windows, commonly used programming languages and tools include C++, C#, Visual Basic, .NET framework, Visual Studio, and Git | To develop software for mobile devices, commonly used programming languages and tools include Java, Swift, Objective-C, React Native, Android Studio, Xcode, Unity, and Git |

## 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**: After evaluating our project with The Gaming Room, I concluded that Windows would be the best operating system for us. It has an easy-to-understand learning system, making it suitable for both experienced and new developers and users. Additionally, it is cost-effective and user-friendly. Windows is also known for its excellent gaming capabilities, making it a perfect choice for our needs.
2. **Operating Systems Architectures**: The Windows operating system is split into two modes: User mode and Kernel mode. User mode processes are what users interact with and can modify. Kernel mode handles low-level tasks such as inputs, outputs, memory management, networking, hardware management, and routines. To store data, Windows uses a directory structure. The system also supports multiprocessing and hardware modularity, which enables customization.
3. **Storage Management**: The game, Draw It or Lose It, will be a web-based application. I highly recommend using Cloud storage for this project. Cloud storage allows us to use only the storage space required to keep the game running. It also allows for the storage to be easily upgraded without spending money on hardware as the game gains popularity.
4. **Memory Management**: Windows uses a virtual memory system that allocates memory as needed, even if physical memory is unavailable. In the case of Draw It or Lose's requirement for one game at a time, it efficiently allocates space for it and more if it's chosen to expand. If the program uses too much memory, Windows uses paging to free up physical memory for other applications. These techniques ensure smooth program operation without causing problems for different applications.
5. **Distributed Systems and Networks**: Since the game will be primarily web-based, Google Chrome would be a good option as it's widely available on most platforms and devices. This will also allow people from all over to connect without worrying if the OS supports the program.
6. **Security**: To protect user information, we need to implement secure communication channels, access control mechanisms, and encryption. Windows OS provides built-in security features such as Windows Defender and Firewall. We can further enhance security by implementing multi-factor authentication. By doing so, we can ensure that user information is protected on and between various platforms.