

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 | 09/22/2024 | Kyle Simonetti | Added Executive Summary, Requirements, Design Constraints, and Domain Model. |
| 1.0 | 10/06/2024 | Kyle Simonetti | Updated Design Constraints, added Evaluation |
| 1.0 | 10/20/2024 | Kyle Simonetti | Completed Recommendation |

**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 client, The Gaming Room, would like a web-based game developed for multiple platforms based on their current game, Draw It or Lose It. The client needs help to streamline the process. The application renders images from a large library of stock drawings and renders drawings at a steady rate. There are multiple teams with multiple players assigned to them, game and team names must be unique, and there should only be one instance of a game in memory at any given time with unique identifiers for each instance of a game, team, and player.

## Requirements

* One or more than one team involved in a game
* Team’s can have multiple players assigned to it
* Game and team names must be unique
* Only one instance of a game can exists in memory at any given time
* Unique identifiers for each instance of a game, team, or player
* Renders images from a large library of stock drawings
* Renders drawings at a steady rate
* Available on multiple platforms

## [Design Constraints](#_2et92p0)

* Multiple platforms require multiple development, testing, and production
* Web-based relies on a network
* Latency issues
* Mouse and phone require different ways to draw
* Security challenges as web-based relies on a server
* Increased storage to hold stock images, and enough memory is required to render the photos at a serviceable speed
* Secure authenticators needed to keep user information secure
* Restrict users from being logged into multiple different games at the same time
* Interface needed to manage many users and teams

## [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 UML class diagram provided shows 7 different classes, 2 of which function separately from the other 5. First, we have the ProgramDriver class, which uses the SingletonTester class. The ProgramDriver class has a public function Main(), which is the driver of the application, while the SingletonTester class has one function called testSingleton(), which is just a function that runs a test to make sure that there is only one instance of the game service running. We then have a class Entity, that has 3 children classes Game, Team, and Player that inherit from Entity class, shown by the open arrow. We also have a GameServive class that may contain 0 to multiple Game objetcs, a Game class that may contain 0 to multiple Team objects, and a Team class that may contain 0 to multiple Player objects. These relationships demonstrate the software requirements that one or more teams can be involved in a game and teams can have multiple players on a team by using a list of Team and Player objects. Before adding a player to a team, or adding a team to a game, the objects being added are checked using an iterator design pattern to make sure each team name and game name are unique. To satisfy there being only one instance of a game that can exist, the GameService class uses a singleton design pattern. Each player, Team, and Game object are also given unique id and names, inherited by the Entity class, and are private attributes with getter and setter methods. The Entity, Game, Team, and Player classes also contain a method toString() which overrides Java’s toString() method to provide a string representation of an object to obtain its name and id.

**"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** | Mac has access to web hosting through tools such as Apache, which provides fast and reliable web hosting. Multiple websites can be hosted on a single server with SSL and realm-based authentication. Database’s such as MySQL can be linked to a website. Servers are easy to host and customizable and has an easy-to-use graphical interface. Is based on Unix, so it has an easy-to-use terminal with well-established commands. Disadvantages of Mac include high cost for hardware and to upgrade hosting plan as player size increases and limited software compatibility, requiring workarounds. | Linux is open-sourced, flexible, and has many tools readily available, while also being cost-effective as it is free to use. Linux has strong security features, as vulnerabilities are quickly identified and fixed as it is open-sourced. Highly flexible, with access to Apache, MySQL, and PHP. Access to tools for cross-platform development, and virtual machines to simualate other operating systems. Disadvantages include its steep learning curve, file format compatibility issues, limited support for software, and lack of support for several programming languages such as ASP.NET. | Windows has extensive software compatibility and a user-interface that is incredibly easy to use and understand. There are many different development tools and software that is offered by Windows, allowing for a wide range of choice when developing a web-based application. Windows is used by a large amount of people, so it has a vast amount of support and resources. The cost for Windows is affordable compared to macOS, but more expensive than open-source Oss like Linux. The disadvantages of Windows are that system configurations and updates can hold widely different performance and stability and contains some tools that may not be as optimized as operating systems such as Linux. | Mobile devices are easily accessible, as most people have phones, and can be range in pricing depending on the mobile device. Mobile devices are not very capable of hosting or creating web-based applications, especially ones that will need many users. Mobile devices also have many different OS and web browsers to choose from, which may make it harder to make optimize the application for each one. |
| **Client Side** | Easy to use once familiar with MacOS. Not accessible on devices other than Apple. For people unfamiliar with MacOS, it will take time and training to get familiar with how it works. | Incredibly cost efficient, as Linux is free to use and install. Linux uses a command-line interface, which takes time to learn and use effectively. No technical support, as Linux is open-sourced, and user controlled. | Windows is cost efficient, easy to use and understand, requiring little time to understand how it works. Windows is highly available and offers different types of plans for different types of needs. Windows is also able to virtualize other OS, which helps in cross-platform development. | As mobile devices are very common and almost everyone has access to one, they are widely available, have a large range of pricing options, and can be easy to use and learn as most people are already familiar with them. Mobile devices have many different screen sizes, especially with foldable phones that have a couple different screen sizes, so getting the website to work properly with each type of phone might be harder to do. There is also no mouse and keyboard, so a touch interface has to be adapted. |
| **Development Tools** | Swift programming language which is mainly used to develop mobile apps and is used by apple products. Other programming languages include C and C++. Xcode is an IDE designed for MacOS with tools provided to create apps for apple products such as iPhone, iPad, and Mac, and supports a variety of programming languages, with a user-friendly interface. SwiftUI is another IDE available, which uses declarative syntax. VS code is also available and contains smart code completion and built-in debugging tools. API testing applications such as Postman can help to simplify, develop, test, and document APIs. | Linux has many development tools that can be used to build this type of software on Linux. One example would be Docker, which can be incredibly useful when developing cross-platform applications. IDEs such as Eclipse and VS code can be used to accommodate a wide array of languages. Has access to Git which is useful for teams to work on a project together. Tools like Qt can help with cross-platform development and has access to virtual machines to emulate different operating environments. | Windows has a wide variety of programming languages and tools to choose from. Ide’s such as Visual Studio and .Net framework offer many tools that can help in web development. Windows also has access to ASP.Net, a framework for building web apps and services, with native compatibility. ASP.NET can also be used to create REST APIs for a large range of clients, including browsers and mobile devices, and is easily scalable for large amounts of users. Microsoft Azure can also be used to host apps through the cloud, along with cloud-based REST APIs. | Mobile devices have access to different programming languages and tools, such as QT IDE, Xcode for iOS, Visual Studio code, and CppDroid for android. They also have access to many different programming languages, the main ones being Java, C++, and JavaScript through the different IDEs. IDEs like QT IDE can help with cross-platform app development and is scalable. |

## 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**: The operating platform that is most appropriate to allow The Gaming Room to expand Draw It or Lose It to other computing environments is Windows. Windows is incredibly easy to use, has a large user base with many different IDE’s and tools readily available for most needs, is relatively cost efficient, and has access to virtualizing other operating systems such as Linux or Android for testing purposes using Hyper-V, which is incredibly important when developing for multiple platforms.
2. **Operating Systems Architectures**: Windows is divided into two main parts, that being the user mode and the kernel mode. User mode includes what the user interacts with regarding the operating system, such as user and system processes and the command line. The kernel mode deals with the hardware side of the operating system, and controls what hardware can and cannot be accessed. These two parts work together to provide better security.
3. **Storage Management**: Microsoft Azure is a tool available to Windows that can help with storage management. Firstly, Microsoft Azure has strong data security, allowing for encryption of data to protect private data. Azure is also scalable, allowing for Draw It or Lose It to continue to grow its player base. Azure is cost efficient, allowing you to pay for the resources and services used. Azure is able to stage environments, allowing for bugs to be caught before they are implemented into the live website and affecting user’s.
4. **Memory Management**: As this application will be cross platform, and many different types of phones and computers will be using the application with wildly different available RAM, it is important that memory paging is used to ensure the application’s memory requirements do not exceed the user’s available RAM and prevent crashing. With Windows, paging can be easily managed and configured.
5. **Distributed Systems and Networks**: I believe .NET and would be an invaluable tool to create this cross-platform application. Firstly, apps can be built for all platforms and supports a wide variety of coding languages. .NET contains components such as common language runtime, which can help to control memory management. .NET has many different security features to prevent vulnerabilities, such as encryption and authentication. .NET can also interact with Azure, which as mentioned above can help to improve storage management.
6. **Security**: Windows provides different security features to protect private user information. To protect from viruses and threats, Windows includes Microsoft Defender antivirus, LSA protection to authenticate users, attack surface reduction to prevent a device or network from being compromised, and controlled folder access. Windows also provides several network security features, such as transport layer security to provide security over networks through encryption, DNS security, and Windows firewall. Lastly, Windows provides encryption and data protection through BitLocker to encrypt and protect data at rest. To protect data on multiple platforms, Microsoft Azure can again be used to encrypt data of user information.