

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 | 07/16/2023 | Ryan Hoskins | Initial draft, added Executive Summary, Requirements, Design Constraints, and Domain Model explanations. |
| 2.0 | 07/25/2023 | Ryan Hoskins | Second draft, added Evaluation section for Server side and client side details for each criteria. |
| 3.0 | 08/12/2023 | Ryan Hoskins | Final draft, included last section for Recommendation for each criteria. |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

As the finished product will be a platform in which the “Draw It or Lost It” game can be played, there are a few design problems that present themselves. As the game functions by allowing teams of players to compete against each other in games in which they attempt to guess a phrase, object, etc. as a related image is rendered, the application must allow several instances of unique games, each with different unique teams, and each of those teams with unique players.

The game is currently only available on Android, which presents the need to make a web-based platform that the game can be played on across multiple devices. Making the game available on a web platform resolves the primary issue of the game not being accessible to those without an Android device.

To facilitate seamless game and team management, the application will enforce the uniqueness of game and team names. Users will be able to check name availability before proceeding, preventing any confusion or overlap. Furthermore, this design will implement the Singleton pattern, ensuring only one instance exists in memory at any given time for a specific and unique game. This centralization of game and team management will prevent conflicts when multiple users interact with the application concurrently.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

Requirements for the project can broken down into a few categories, including Game and Team Management, Uniqueness of Game and Team Names or Identifiers, and Game Functionality.

To start, requirements for Game Functionality include:

* **The Rendering of stock images as clues**: The application must have a library of images that can be used as clues through the game. Each round will present a new image for the teams to use. Additionally, the drawings must take about thirty seconds to render completely.
* **Game Rounds and Time Limits**: Each game will consist of four rounds with each round lasting one minute. One team will have the opportunity to guess correctly during the first thirty seconds while the image renders. If they are unable to guess correctly in this time, the opposing team has fifteen seconds to guess correctly.

The requirements for the uniqueness of game and team names or identifiers can be broken down into the following:

* **Unique Game/Team Names:** The application must enforce the requirement for no two game instances to have the same name. Similarly, team names must be unique within the context of a game instance, preventing teams from having identical name, as well as players.
* **Unique Identifiers:** The application must utilize a Game Service Management that can be implemented as a Singleton Class to ensure that only one instance of GameService exists in memory at any given time, which makes sure only one instance of a unique game can be in use at one time.

Lastly, Game and Team Management requirements can be listed as:

* The application must allow the creation of one or more game instances, each with its own unique name and identifier.
* Each game instance can have multiple teams, and each team must have its own unique name and identifier.
* Teams must be able to have multiple players assigned to them.

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

Developing the application to be a web-based platform presents several unique constraints to take into consideration or implications to be mindful of. One the primary implications to take into consideration is cross-browser compatibility. Each web browser (Chrome, Firefox, etc.) can behave differently with web platforms. This requires the application be tested across all major browsers throughout the development cycle to ensure the platform functions correctly regardless of what browsers are used.

Since a web platform can be accessed across a number of devices, the platform must be responsive. This means that various devices will access and play this game, including mobile phones, tablets, laptops, and desktops with varying monitor sizes will access the game, which requires it be compatible with the display of all of them. This is a crucial constraint to be mindful of when considering accessibility.

Additionally, as a web-based application, this program will be susceptible to potential network limitations or issues. This includes potential latency while playing the game – which could interfere with the already narrow windows of time for each team in a round.

The images used in the game must be properly sources or licensed for use as well. Whether they are Creative Commons Zero Attribution images or licensed for use, the images may require citations, credits, or have restrictions on whether they can be used for commercial purposes or not – which is something to be mindful of when sourcing the images.

Lastly, the development of a web platform is a large constraint. Whether players will need to have an account to play the game will present other design constraints as well. If a player creates an account, appropriate data privacy and GDPR laws will need to be abided.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

The provided diagram represents the components that will make up the application and determine how it functions. The ProgramDriver class essentially serves as the entry point to the application and is responsible for starting the game. It does not have any direct relationships with other classes in the domain model, but its “main” method acts as the starting point for the game.

Next is the “SingletonTester” class, which is connected to the “ProgramDriver “ class and is used by it to ensure only one instance of “GameService” exists at any time. This is accomplished by using the “Singleton Pattern”, which is an object-oriented programming principle that restricts the instantiation of a class to one object. By using the Singleton pattern, the application ensures there is only one GameService instance managing game instances, teams, and players.

The “GameServices” class is what manages many parts of the application and game. It’s the class that will manage the game instances and their respective teams and players. It contains several attributes, such as a “List” of games. It has a relationship with the “Game” by, which is visualized using the line that connects them. In this case, it shows that the GameServices class can manage several instances of the “Game” class. This class is what utilizes the Singleton Pattern, which allows the application to make sure only one instance of GameServices can exist at any given time, which allows multiple users to interact with the application at one time without any disruptions of unintended behavior.

The “Entity” class functions as a base class for the “Game”, “Team”, and “Player” classes. This allows it to hold attributes that are shared across those classes, which in this case are “ID” and “Name”. This allows those three classes to inherit shared attributes and methods for accomplishing certain tasks.

Lastly, the “Game”, “Team”, and “Player classes are how information pertaining the a specific

instance of a game, team, and player can be stored or created. The Game class can have multiple teams participating and has methods of either retrieving or creating instances of “Team”. The Team class has a relationship with the Players class, indicating it can have multiple players participating and that it has methods of either creating or retrieving instances of Players.

**"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** | <Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Linux for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

**Server Side:**

**Mac:** MacOS can be a compelling option for end users, as it is typically designed for. MacOS Server was previously an Unix based option up until 2022, when it was discontinued. However, MacOS itself does come with many of the features previously available on MacOS Server already built into it. One such feature is that is supports server-based deployment using web servers such as Apache – which is commonly used across many systems. Another advantage is that it provides solid integration with other devices in the Apple ecosystem. However, it does lack server-specific tools that other options have and some of its biggest strengths are limited to only Apple devices.

**Linux:** Linux has by far been one of the strongest contenders for hosting web-based software. Linux is a powerful and versatile operating system known for its stability, security, and scalability. It is well-suited for hosting web applications due to its well-rounded server-side capabilities and extensive server administration tools. Another positive is that Linux offers a wide range of server-based deployment methods, including one of the most popular web servers - Apache. In addition to this, it is also open-sourced, meaning there plenty of options that can eliminate a licensing fee. Adding onto the fact that it is open-sourced, this also indicates the widespread community support ensures consistent updates.

**Windows:** Microsoft Azure has been an up-and-coming contender for web-based application needs. It is a well-established brand that has been known for having user-friendly design and strong support for hosting web applications. Like Mac, this also provides the added benefit of seamlessly connecting to other Windows devices. In addition to this, Azure typically provides MS SQL Server – an alternative to MySQL for database needs. Microsoft also provides some the added strength of being able to run applications on their .NET Framework. However, the biggest disadvantage is going to be licensing fees. A Windows solution is going to cost substantially more than the Linux alternative would.

**Mobile:** This would be the most complicated choice by far. There are options for a Mobile Web Servers such as an open-sourced software known as i-jetty. This is a powerful tool that utilizes Java and is designed for hosting web applications on Android devices and is like related tools such as Apache Tomcat. However, a mobile device has limited resources and varying network conditions, which can lead to several issues.

**Client Side:**

**Mac:** Mac clients require web applications that are compatible with macOS and Safari, the default web browser on Mac. Developers must have expertise in web technologies such as HTML, CSS, and JavaScript, as well as knowledge of macOS-specific APIs and features if the application requires integration with native functionalities. One thing to be mindful of is whether developers have experience using some tools commonly seen with Mac development, such as Swift or Cocoa.

**Linux:** For Linux, most browsers used on this system will be Firefox, Chrome, or other Chromium based browsers – which are some of the most commonly used browsers. As such, developers will need to be proficient in HTML, CSS, and JavaScript. Additionally, Linux is a widely favored system for many software developers. There is also a wide assortment of open-source technologies that Linux is compatible with.

**Windows:** As other systems, the primary browsers used for this system are going to be Firefox, Edge, and Chromium based browsers. Windows does have the added benefit of a wide variety of integrations available as well, including Microsoft Office. Additionally, there is the added benefit of Microsoft Visual Studio and VS Code.

**Mobile Devices:** This, again, is going to represent the most complicated of the four systems. Mobile devices can utilize any number of screen sizes ranging from small phones to large tablets. There’s the added obstacle of needing to not just develop for Android, but for iOS and any other variation that exists – Samsung, LG, etc. With that in mind, it’s crucial that any developer be very familiar with the differences between the many systems that exist. Additionally, mobile devices and tablets do not utilize input devices such as a mouse and keyboard. As such, the entire method for navigating through the game will need to be created in such a way that it takes this into consideration and upholds UI/UX standards across every device used. Knowledge of cross-platform development frameworks is a must as well for any developers involved in the project at hand for the game.

**Development Tools:**

**Mac:** The primary languages used for MacOS and iOS are Swift and Objective-C. Swift is a high-level language that is also open-source. Objective-C is an object oriented language similar to C. The primary IDE for MacOS is called Xcode – which is developed by Apple and includes a graphical interface builder, which allows developers to build user interfaces. Some frameworks available for MacOS includes Cocao Framework and Cocoapods. Cocoapods is a dependency manager for Swift or Objective-C projects.

**Linux:** Developing software for Linux offers many different options, a variety of programming languages, and tools. Regarding programming languages, C++, Python, Java, Go, HTML, PHP, CSS, JavaScript, and many other options are available. To add onto this, several IDE are also available including VS Code, Eclipse, JetBrains, and several others. There are Package Managers available as well, one of which is Advanced Packaging Tools. Linux development also supports the use of Docker – which can be particularly helpful for distributing across different platforms. It’s important to note many of these tools are available for free as developers. However, Docker does cost around $25 per month.

**Windows:** Windows, like Linux, offers support for a variety of programming languages such as Python, C#, C++, and Visual Basic .NET. It also has some of the best IDE options as well which include VS Code and Visual Studio. One of the strongest benefits is that it has the .NET Framework, which is a comprehensive platform for building Windows applications and offers many libraries. Git can also be used for version control as well. It’s important to note the Visual Studio will likely cost money. There are options for Enterprise or Business solutions for both Mac and Windows OS, but that will cost around $6,000.

**Mobile Devices:** When building software for deploying on mobile devices, developers have various programming languages and tools at their disposal. As mentioned earlier, Java is the primary language used for developing Android based applications and Swift/Objective-C can be used for iOS devices. The two primary IDEs are Android Studio and Xcode, for Android and iOS respectively. There are frameworks that are provided for Android specifically and for iOS specifically, such as Android SDK and iOS SDK. However, there are also options such as Flutter – that offer a cross-platform framework. Version control with Git, testing with emulators/simulators, and distribution through Google Play and the Apple App Store complete the mobile app development workflow.

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

While there are several potential options when selecting an operating platform, there is one in particular that stands out as the most feasible and compatible with this application – Linux. A Linux based operating platform offers many advantages. One such advantage is that it is actually free – which greatly reduces cost that could otherwise be put into other aspects of the application, such as storage management and security. In addition to the cost, Linux can be considered more secure than alternative platforms such as Windows and is a much more stable platform that allows the user to control what updates are pursued and when they are installed.

Selecting Linux as the operating platform also provides much more control over the operating platform, offering choices such as Ubuntu or Fedora. For “Win it or Lose it”, I would recommend choosing Ubuntu. Ubuntu is a flexible, secure, and scalable option for a Server OS that is particularly powerful for web-based applications. This option is rich in community support, offers consistent security updates, Standard Security Maintenance for five years, and fairly low resource requirements for installing and maintaining the platform. One of the most appealing features of this is its compatibility with Apache Server.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

Linux, or particularly Ubuntu Server, can be viewed as having five main layers to its operating system architecture. These layers consist of hardware, kernel, shell, applications, and users. As expected, the hardware consists of the physical equipment such as the hard disk drive, RAM, CPU, etc. The Kernel layer can be considered as the core of the operating system that provides critical and essential services for applications. The Kernel layer is what directly interacts with the hardware and is responsible for system memory management, software program management, hardware management, and filesystem management as well.

The Shell is a layer what allows the system to take inputs from the user and relays it to the kernel, as well as taking outputs from the kernel and sends it to the shell. This is the layer that allows a user to perform command executions, process control, set or manipulate environmental variables, and navigate the file system. The applications layer is what users typically interact with the most and are the programs that run on the Shell layer. Lastly, the user is simply someone that is currently logged into the system and performing various tasks.

As a whole, Ubuntu Server operates in a “monolithic kernel architecture”, which is an operating system design that has all core system functions and device drivers integrated into one module – which in this case is the kernel. This ensure the kernel operates as a single unit without separation between itself and the users. This design offers several advantages such as increased performance, simplicity, and low latency – which can be crucial for a system hosting a web-based application, such as “Draw It or Lose It” in this case.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

Storage management could be handled in few different ways. One such way is to utilize an “all-in-one” approach where all storage management processes are taken care of by a single system. One such system is Ceph – an offering from Red Hat. However, the cost of using this and technical know-how could be difficult. Another option is to utilize a combination of systems, one for structured data such as profiles, permissions, etc. and another for unstructured data such as images. This could be accomplished by using several tools such as MySQL and Amazon S3. MySQL could handle the structured data such as user data and Amazon S3 could handle content delivery and storing images. Another option is to utilize a system such as OpenStack. OpenStack is a great option as it is compatible with Linux and Ubuntu Server and has a wide variety of offerings for different needs. A major advantage of it is that OpenStack is considered a modular architecture, meaning there is full control over how it functions with the application. Additionally, it has great security and provides a completely distributed storage that is accessible by API, providing an effective way to incorporate it into the application.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

Linux and Ubuntu Server utilize several memory management techniques that are applicable to the Draw It or Lose It application. One such technique is referred to as virtual memory. Virtual memory enables the device to act as if it has more memory than it actually does. This is accomplished through the use of Memory Paging – which essentially breaks memory into smaller components. In the case of Linux, the system creates these “pages”, or blocks of memory, that consist of contiguous memory. Contiguous memory allows the system to allocate a piece of physical memory on launch for crucial services to ensure things operate smoothly. It also enables the system to allocate that block of physical memory to other services that may demand it while running. Linux also utilizes page caching, which allows the system to store data in a cache and quickly retrieved later. This is incredibly helpful for improving the performance of the application.

1. **Distributed Systems and Networks**: Apache

Draw It or Lose It functions as a distributed application, which operates on principles of distributed systems. The core idea of this is that an application can be distributed to many devices that can then be tied together using a network. This allows Draw It or Lose It to be distributed to clients across a range of devices over a network that allows it to communicate with crucial services that enable the game to function. One advantage of this is that it’s incredibly efficient. The back-end software is housed in the server that manages the application on a system, which in this case would be Ubuntu Server and the Storage Management service (OpenStack). The back-end software is able to manage shared resources and processes the application. The front-end would be the client that is running the application, which in this case would be a web browser. The front-end is how a user would play the game.

These two communicate over a network, which means a connection is required to play this game as that is how the client communicates with the server to play the game. One benefit of this is that if one of the clients experiences an outage, any other clients playing the game will be unaffected. However, if the server were to experience an outage, all clients previously connected to it and playing the game would also experience an outage. The network enables the server and client to exchange messages, which can be requests or data that provides instructions – such as to share an image, retrieve user data, or authenticate a user’s login information they’ve entered.

<Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

Security can be implemented across various components of the distributed system, the server, how it communicates with the clients, and among the client’s users as well. Starting from the server, which in this case is Ubuntu Server, full-disk encryption could be implemented to ensure any data is secured. In addition to this, Ubuntu Server allows an admin to utilize Secure Shell and set up a SSH key pair. A firewall should be set up to establish a set of rules that can be applied to incoming internet traffic – which is essential to be safe.

Regarding the security of the storage management system OpenStack, API restrictions can be put into place to only authorized users are accessing it – which would make sure access to crucial data and user information is limited to authorized users allowed. A firewall can also be implemented for OpenStack and its data encrypted.

The network communication can also be made secure by implemented SSL/TLS encryption to ensure communication between the clients and servers are safe. The network communication can use a protocol such as HTTPS to further protect any data that is transferred from server to client. There are also options such as Apache that could further bolster any network communication that occurs for the game.

Lastly, there are several security features that can be implemented within the game itself that ensure user security. For example, role-based access control can be implemented by registering users when they first sign up and setting their role as either a user or admin. Users can log into their account and play the game, but they won’t be able to make system level decisions that affect the whole game as admins would. To add further security, multi-factor authentication can be required as well to bolster security on a user level as well. Incorporating these security features ensure the game application will be well-secured, just as the team has requested it.