

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 | 01/24/2024 | Edward Garcia | Addressed software requirements, outlined object-oriented design principles for multi-platform support. |

**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 Gaming Room’s focus is to advance the ‘Draw it or Lose it’ game from just being Android based to being web-based that has compatibility with various platforms. This will aid in the streamlining of development of multi-platform functionality, allowing teams to compete in real-time engaging in the nostalgic dynamics of the original 1980’s ‘Win, Lose, or Draw’ game. The design will ensure that the ‘Draw it or Lose it’ game will only have one instance of the game in active memory at the given time, it will have the ability for one or multiple teams incorporated and will also have unique team and game names to allow users to verify if a name is being used when they are picking a team. We will create a creative designed web application that maintains the game’s integrity across numerous devices and web browsers.

## Requirements

There are many business and technical requirements that need to be fulfilled in order for this implementation to be successful. For business requirements they will need multi-platform accessibility, meaning the game should be accessible to users across various platforms such as desktops and mobile devices and also offer access to operating systems such as Windows, MacOS, and Linux. Another business requirement is user engagement. Having the users interact and compete amongst themselves with the same fluidity and ease of use that the original Android application had. For technical requirements it will involve multi-team support that assists gameplay involving one or more teams. Another technical requirement will be unique identifiers, meaning the names of game and team should be unique in order to avoid conflicts of having users pick the same team name. Another technical requirement will be the single instance in memory needed so that the only one instance of the game exits in memory at a time. Security is also a technical requirement because invasive security measures are needed to protect users and their personal data.

## [Design Constraints](#_2et92p0)

The game’s transition into a web-based environment is one of the design constraints involved with the newly required implementations. This means that the application should be accessible to users over the internet and on various devices while also being able to support the users with any issues they may encounter. This also requires designing the application to be able to withstand a major increase in web traffic from multiple devices while being able to provide security for the user’s data. Another design constraint will be the coding languages involved with the various multi-platform devices such as Apple and Android. Both devices use different coding syntax and development environments which will need to be tailored into the ‘Draw it or Lose it’ game programming, to allow it work seamlessly between multiple platforms. Having to alter the code to monitor unique names chosen by the users and verifying if the names were already used, is another design constraint that will need to be examined.

## [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 Gaming Room UML Diagram shows an object-oriented design of the game application that has various classes such as; GameService, Game, Player, and Team class. The Entity class is the base class that encapsulates attributes and utilizes a common relationship with the Game class, Player class and Team class. This UML diagram class structure uses inheritance by sharing data with the separate classes in the diagram. The Entity class provides attributes like ‘id’ and ‘name’ to specified classes. Each game object can contain multiple teams, while team includes several player objects which is known as a one-many-relationship. The Team class characterizes a specific team in the game while monitoring the list of teams being used. The GameService class is designed following the Singleton pattern which is a pattern used to ensure that only one instance of GameService is running at any time. The GameService class manages a list of Games and provides the methods required to add games and access information by the specified Id. The SingletonTester class is used in the diagram to ensure only a single instance is accessed, while the ProgramDriver class is used to verify and test the singleton pattern. The Game class represents the concept of a game within the app. It extends the Entity class and inherits id and name attributes. The Game class can show the teams playing and it can also manage the teams by adding them in.

**"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 servers can offer a great amount of security, while offering smooth integration with Apple products and web hosting services. MacOS is also known for being user friendly for having an easy-to-use interface. Mac servers are also known for their reliability. They have a reputation of consistent performance with minimal downtime. The downside of this is that Mac servers have higher costs, hardware limitations and have smaller market presence. While macOS can serve web applications, it’s not commonly used for large-scale server deployments due to its cost compared to Linux and Windows. | Linux is highly known for being an open-source operating system. This allows the source code to be freely available for anyone to modify and distribute. It has lots of customizability allowing developers to utilize the server towards their needs. Some issues with Linux are its complexity, due to distributions being hard to set up and manage. Linux is widely used for server environments with distributions suited for server use, like Ubuntu Server or CentOS. | Windows is known for being user friendly making it easier for users to set up and use. It also offers extensive support from Microsoft. Another advantage of Windows is its use of integration with other Microsoft products and services. One issue with Windows is its security. Windows has been known to be more vulnerable to attacks compared to other systems. Windows Server is a well dedicated OS for server management and hosting web applications. | Mobile devices are known for their portability which allows users to have access to web-based applications wherever they are. Mobile devices also have an accommodating feature allowing for touch interaction. This allows the devices to be touch friendly for the user allowing them to breeze through applications faster than other operating systems. A disadvantage of Mobile devices is performance constraints, some user device may be outdated and no longer support the application. The Mobile Web Server application allows mobile devices a means for hosting personal web applications, including, web pages and server-side control. |
| **Client Side** | Clients of Mac can benefit from a stable and secure operating system that offers stability across MacOS devices. Mac offers many devices that work seamlessly together, allowing data to be shared in a more efficient way. The disadvantage of this is development cost due to the licensing and high-cost maintenance of Apple hardware. Developers may need to understand specific features and design patterns preferred by Mac users to ensure the application feels intuitive. | Linux for clients can be cost effective due to its open-source nature. They offer many distributions available at no cost to the user. Linux is also known for being able to run on a wide range of hardware from older systems to the most current. Developers of Linux should be familiar with the permissions and security model of Linux OS to optimize the web application's performance and compatibility. | Developing for windows may involve cost with purchase licenses for environments. Some tools are free to use like Visual Studio. Time is important for Windows apps because they must work for specific Windows technologies like .NET. Developers will need expertise in system integration with Windows services like authentication. Windows has a large developer community, with a wide range of resources and tools that can speed up development time. | When planning software development for mobile devices costs can include licenses for IDEs, and subscriptions for mobile platforms. There are also fees associated with the manufacturers app store. Testing will require a range of devices to ensure the app functions which can be costly. Time in development may be hindered due to the need to handle multiple frameworks. Mobile browsers also support modern web standards, but testing on actual devices is important due to the diversity of mobile hardware and OS versions. |
| **Development Tools** | The development tools for mac include languages such as Swift which is used for native MacOS applications. The IDEs mac uses consist of XCode, Visual Studio Code, and JetBrains’ suite. Mac also uses Homebrew as a package manager. | Linux used programming languages such as C#, C++, Python, Java, PHP, Ruby, and GO. Linux has many IDEs and Editors such as Eclipse, Visual Studio Code, IntelliJ IDEA, NetBeans, Sublime Text, and Atom. Other tools used by Linux are Git, Docker, Make, and Bash Scripting. | Windows uses programming languages such as C#, Visual Basic, PowerShell, and Java. Some IDEs Windows uses are Visual Studio, SQL, Windows SDK, and Microsoft Azure Tools. | Mobile devices use programming languages such as Swift, Objective-C, Kotlin, Java, and Dart. Some tools and IDEs used by mobile devices are; XCode, Android Studio, and Flutter. Some other tools used are React Native, Gradle, and Fastlane. |

**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**: Due to the need for muti-cross platform support, I recommend Linux as the primary operating server platform for Gaming Room’s ‘Draw it or Lose it’ game. Linux is a versatile open-source server platform that allows for many methods of customization and scalability. It is also cost effective due to it being open-source which is suitable for a gaming environment. Linux is also a well-known platform in server environments and is known for its stability, reliability, performance, security, customizability, and community developer support. Linux also comes with many hardware architectures that are well accommodating to the client’s needs.
2. **Operating Systems Architectures**: Linux offers many details when it comes to architectures. It is a flexible architecture that supports numerous microservices that are scalable for web applications and games, including a wide range of hardware and software. Linux is highly modular and customizable and grants developers more options to pick and choose from when it comes to which specific components they may need to create an optimized environment. Linux can scale each service independently, preventing issues when ‘Draw it or Lose it’ undergoes an increase in the number of active games. Linux is also able to adjust the game to various computing environments which is beneficial for the client.
3. **Storage Management**: Linux has many tools available for storage management. An appropriate storage system such as ext4 or XFS can be used to help with storage tasks. They are highly scalable and also have many options for storage. A combination of relational databases and NoSQL can be used to help maintain storage management as well. PostgreSQL can monitor relational data needs and MongoDB can be used for session management and real time analytics. This offers flexibility when handling different data types and transaction loads.
4. **Memory Management**: Linux has the ability to efficiently manage memory with techniques like demand paging. This allows only necessary parts of an application to be loaded into memory. This will benefit ‘Draw it or Lose it’ as it is able to optimize resource usage during times of large player counts and graphic rendering. Linux also utilizes many other memory management techniques like virtual memory, The kernel’s memory management unit (MMU) can handle memory tasks and ensures optimal memory usage which will benefit the game’s performance.
5. **Distributed Systems and Networks**: ‘Draw it or Lose it’ can utilize communication through various platforms using the distribution and networking tools in Linux. We can implement a service-oriented architecture and adopt RESTful APIs for communication between services. To manage real time aspects, WebSocket can be used to implement connections between the client and the server. To reduce latency a Content Delivery Network (CDN) can be used to distribute content closer to users globally. Linux also supports distributed applications using technology such as Docker and Kubernetes. These can benefit the client by utilizing containerization to deploy the game across various platforms and manage the communication between them.
6. **Security**: To protect and secure user information across various platforms, several tools and Linux security strategies will be implemented to ensure that the user is protected. Utilizing secure communication by using SSL/TLS to encrypt the data between clients and servers. This will prevent any data exchanged being tampered with or intercepted. Linux also has powerful firewall tools like iptables to control incoming and outgoing network traffic. User authentication is another security measure that can be used which includes two-factor authorization for user accounts that provides an extra layer of security. Linux also has a feature called Security-Enhanced-Linux (SELinux) for advanced access control. SeLinux is a security architecture integrated into the Linux Kernel using Linux Security Modules(LSM). This feature will implement Mandatory Access Control (MAC) to restrict program capabilities and access amongst files, network ports, and users.