

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 3.0 | 02/25/23 | Kemal Cankurt | Finalizing the design document |

**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 wants to develop a web-based game that can run on multiple platforms. The game will be called “Draw It or Lose It” and is currently only available on android. The purpose of this game is multiple teams consisting of several people going four rounds at a minute each. When a picture is pulled from a library of images one team guesses till time runs out. If not answered each opposing team member gets to answer till 15 seconds runs out.

## [Design Constraints](#_2et92p0)

* Needs one or more teams involved
* Each team has multiple people
* Game and Team names must be unique to allow users to check whether the name is in use or free
* Only one instance of the game can exist at any time.
* Must run on multiple platforms

## [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 Entity class creates a relationship between the Game, Team, and Player classes. The inheritance indicates that all of them are inherited from Entity. Therefore, each class will share common information such as “name” and “id.” Also, they are making Entity a superclass. In terms of their relationship, the Team and Player are a “has a” relation type which means a team can have multiple players. Also, the Game class uses “has a” relationship with the Team, which means a game contains numerous teams, and the GameService has multiple Games.

**"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 computers are trusted and known for their reliability. The client would need to purchase MacOS for each device, which could be expensive for a large number of devices. Additionally, MacOS is only available on Apple hardware, so if the client is using other hardware, it would not be an option. | There are many software options to host a web application on a Linux server. Likewise, open-source operating systems would be beneficial in lowering the overall cost. For example, Ubuntu, Fedora, or CentOS. However, if the client wants support and enterprise-level features, they may opt to purchase a commercial distribution such as Red Hat Enterprise Linux, which can be costly. | Hosting a web application on a Windows machine is easy as there is a serious amount of support. Windows even has dedicated operating systems to build applications. The cost would depend on the specific version of Windows and the number of licenses required. | Mobile devices can host an application’s server side with the support of a web server software. The cost would depend on the specific platform being used. For iOS devices, the client would need to pay a fee to Apple to be able to deploy their app on the App Store, and for Android devices, the cost would depend on any licensing agreements with Google |
| **Client Side** | Using a Mac will be beneficial for the client side as they are equipped with better screens. Similarly, providing a good user experience will likely increase the number of users that are using the application. Expertise level will be moderate as Mac computers offer different experiences than others. Lastly, the cost is more expensive than others. | Linux can offer cheaper development experience thanks to the open-source support. However, expertise level may be higher than others. Therefore, the time may be needed more than other operating systems. | Windows is the most common operating system that can support the development process. Time and expertise levels will be at minimum. However, there are higher licensing costs. | The cost of a mobile device is like the other options. The need for expertise will be at minimum as almost everyone is familiar with mobile devices. |
| **Development Tools** | Programming languages: Swift, Objective-C  IDEs: Xcode  Other tools: macOS Developer Library, Cocoa  Licensing costs: Xcode and macOS Developer Library are free for developers, but there are fees for publishing apps on the Apple App Store. | Programming languages: Python, C++, Java  IDEs: Eclipse, NetBeans, Visual Studio Code  Other tools: Apache Tomcat, MySQL  Licensing costs: Most of the tools used for Linux development are open source and free, although there may be fees for support or commercial use. | Programming languages: C#, .NET, Visual Basic  IDEs: Visual Studio, Visual Studio Code  Other tools: Microsoft SQL Server, Microsoft Azure  Licensing costs: Visual Studio and SQL Server require a Microsoft Developer Network (MSDN) subscription or a perpetual license, while Microsoft Azure has a pay-as-you-go pricing model. | iOS:  Programming languages: Swift, Objective-C  IDEs: Xcode  Other tools: iOS Developer Library, Cocoa Touch  Licensing costs: Xcode and the iOS Developer Library are free for developers, but there are fees for publishing apps on the Apple App Store.  Android:  Programming languages: Java, Kotlin  IDEs: Android Studio  Other tools: Android SDK, Android Emulator  Licensing costs: Android Studio and the Android SDK are free, but there may be fees for support or commercial use of some components. |

## **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**: I recommend using Linux as the operating system for the game servers.
2. Linux offers robustness, security, and flexibility, which are essential for managing large-scale game servers. Additionally, Linux is highly customizable, allowing The Gaming Room to optimize the operating system to meet their specific needs. Finally, Linux is open-source, which means it is freely available and can be modified to suit the needs of The Gaming Room.
3. **Operating Systems Architectures**: The architecture of Linux systems can be divided into three main components: the kernel, user space, and system libraries.
4. The kernel is the core of the Linux operating system. It is responsible for managing hardware resources such as CPU, memory, and input/output devices. The Linux kernel is designed to be modular and flexible, with support for different architectures such as x86, ARM, and PowerPC. The user space is the area of the system where user-level applications run. It includes various utilities and tools that are used to manage the system, such as shell scripts, compilers, and text editors. The user space is designed to be modular and can be customized to meet the needs of different users. The system libraries are a collection of pre-compiled code that provides functions and services to user-level applications. These libraries are essential for running applications on Linux systems. They include standard C libraries, graphics libraries, and database libraries.
5. **Storage Management**: There are several storage management systems available for Linux that can be used to manage and store data efficiently.
6. The file system is an integral part of the Linux operating system, and it is responsible for organizing and storing data on disk. Linux supports several file systems, including Ext4, Btrfs, and XFS. These file systems are designed to provide high performance, reliability, and scalability. Logical Volume Manager (LVM) is a storage management system that allows Linux to manage disk space more efficiently. It allows its consumers to combine multiple physical volumes into a single logical volume, which can be resized as needed. This makes it easy to manage storage for large-scale applications and databases.
7. Based on the characteristics and needs of The Gaming Room, I recommend using a combination of the Ext4 file system and Logical Volume Manager (LVM) for storage management in Linux. This will provide The Gaming Room with a robust and scalable storage solution that can be easily managed and optimized. The Ext4 file system is widely used and has been proven to be reliable, high-performing, and easy to use. LVM will provide The Gaming Room with the flexibility to manage disk space efficiently and optimize storage performance as needed.
8. **Memory Management**: Linux uses virtual memory management techniques to provide a stable and secure environment for running applications. This allows applications to access more memory than is physically available on the system.
9. Also, Linux uses memory swapping techniques to move unused memory pages from RAM to disk. This frees up memory for other applications and helps to prevent memory exhaustion. When Draw It or Lose It requires more memory than is available in RAM, Linux can swap out unused memory pages to disk to make room for the application. Having these memory management techniques, The Gaming Room should allocate sufficient memory to the application to ensure optimal performance.
10. Additionally, monitoring memory usage will ensure that the application does not exceed available memory and trigger swapping or compression, which can impact performance. By following these recommendations, The Gaming Room can ensure that Draw It or Lose It runs smoothly and provides an optimal gaming experience for its users.
11. **Distributed Systems and Networks**: A distributed system is a collection of independent computing devices that communicate with each other to achieve a common goal. Each device in the distributed system is called a node, and nodes communicate with each other through a network.
12. Using web-based APIs and messaging protocols is highly recommended to enable communication between various platforms. This is because web-based APIs will provide a common interface for all platforms to interact with Draw It or Lose It. In contrast, messaging protocols will provide a reliable and efficient way for nodes to communicate with each other.
13. To be more specific, RESTful APIs provide a standard interface for different platforms to access Draw It or Lose It. The Draw It or Lose It application can be hosted on a central server, which exposes RESTful endpoints that different platforms can access. Also, message-oriented middleware (MOM) provides a reliable and efficient way for nodes in a distributed system to communicate with each other. In this approach, Draw It or Lose It can be designed as a set of loosely coupled services that communicate with each other through a message queue. The message queue can be hosted on a central server, and nodes can subscribe to the queue to receive messages. This approach is highly scalable and fault-tolerant, as messages can be stored in the queue even if nodes are offline.
14. Furthermore, the network should be designed to provide high availability, low latency, and high throughput.

Similarly, the distributed system should be designed to handle network outages and node failures. In the case of network outages, messages should be queued and delivered once the network is restored. In the case of node failures, other nodes should be able to take over the responsibilities of the failed node.

1. **Security**: Protecting user information is crucial for any software application, and Draw It or Lose It is no exception.
2. Several measures could be followed to ensure the security of user information on and between various platforms. First, all communication between the application and multiple platforms should be secured using secure communication protocols such as HTTPS and SSL/TLS. Doing so will ensure that user data is protected against interception and tampering. Additionally, users should be authenticated before accessing any sensitive information, and access to data should be restricted based on the user's role and permissions. Authentication and authorization can be implemented using OAuth, OpenID Connect, and JSON Web Tokens.

Furthermore, sensitive data such as user credentials, payment information, and game data should be encrypted to ensure that it is not compromised in case of a data breach.

Linux has several security capabilities that can be leveraged to protect user information. For example, Linux provides firewalls, access controls, and auditing, which can help secure the system. Linux also has several security tools, such as SELinux, AppArmor, and grsecurity, that can harden the system against attacks.

Finally, to protect user information on and between various platforms, The Gaming Room should follow best practices for security and privacy. The development team should perform regular security audits, penetration testing, and code reviews to identify and fix vulnerabilities. The team should also stay updated with the latest security threats and implement security patches promptly.