

**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 | 09/17/23 | Jeyapriya Saravanakumar | Initial version. |
| 1.1 | 10/01/23 | Jeyapriya Saravanakumar | Evaluation details added. |
| 1.2 | 10/14/2023 | Jeyapriya Saravanakumar | Recommendations added. |

## [Executive Summary](#_sbfa50wo7nsh)

Gaming Room client wants to extend their current web-based game Draw it or Lose It to all other platforms. Currently this game is only on Android (Mobile Platform) and that needs to be available on other OS platforms like Windows, Linux, Mac, and another major mobile platform iOS. This will remove the current restriction to playing this game only in Android OS mobile and can increase the user base by providing this game in all other OS and mobile platforms.

The main requirements for this game are that one or more team(s) can play this game, and each team can have multiple players. Game and Team names should be unique and reference a single instance to avoid duplications. Also, the player's name should be unique within the team.

To develop the game in multiple platforms, we need to prepare this Software Design Document to streamline the development process. This document will address requirements, Design constraints, System Architecture, Domain Model, Evaluation, and Recommendations.

## 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.>*

## [Design Constraints](#_2et92p0)

As per the requirements, the game needs to be developed to support multi-OS and mobile platforms, so technical constraints and challenges should be evaluated separately. Because each platform has its dedicated syntax, language libraries, development software, speed, security, limitations, features, etc.

The following vital aspect to consider is the Graphical User Interface (GUI) for both mobile platforms, If the plan is to have the same GUI on both platforms, we need first to do some research to confirm if it is possible or not, because these platforms generally have different GUI features. The following constraint is the non-functional requirements like how many concurrent users can be supported and, the speed and memory required to use this app, these values may vary between these platforms.

Also, gather any issues and enhancements from the current Android game and address them in the new development for other platforms to increase game quality and have the latest gaming features.

## [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 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.**

The above UML class diagram represents all classes in the com.gamingroom package and their relationships between them. Entity class is a parent class and has common attributes id and name. Game, Team, and Player are the child classes that extend the Entity class and can inherit attributes and members from Entity class. Inheritance relationship avoids duplications and increases code maintenance. Also, Entity class has private default constructor which can avoid creating instances without required attributes, but it has overloaded constructor which takes id and name arguments to create object for Entity class. Getters and toString methods are present in Entity class but no setters to avoid changing any attribute value.

Another relationship in the diagram is an association dependency between GameService and Game, Game and Team, and Team and Player classes. All have zero to many cardinalities between them, which can be achieved through the List data type containing zero or more associated class objects. Also, Game, Team, and Player classes have toString method as well to print their attribute names along with values. Next, these classes have overloaded constructors taking id and name as arguments. The Game class has an addTeam method to create a new team or return an existing team object, and the Team class has an addPlayer method to create or return an existing player object.

GameService is a service class that will be initialized once and provides a single instance using private constructor and getInstance methods. It has a games attribute, which lists Game objects to keep all active games and can be used to check before creating a new game with the same name. GetGame methods implement polymorphism using the same name but different arguments; one is a name, and the other method has an id as an argument. The addGame method creates a new Game or returns an existing game with the provided game name and returns it. NextGameId, nextTeamId, and nextPlayerId attributes will be used to create Game, Team, and Player objects, respectively.

ProgramDriver is the main class that has the main method to start the program, and it creates an object for SingletonTester class to call testSingleton method.

## [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** | **Characteristics:**   * Macs provide a more stable and secure environment to host web applications. It generally uses high quality hardware and processors to provide the best performance. * Mac using macOS operating system developed using traditional UNIX OS. Mac supports most of the UNIX commands and features. * Mac provides the best Graphical User Interface, and it is easy to use as well.   **Advantages:**   * MacOS has high end security features, and it is very hard to break macOS security. * It offers simple setup and configuration steps to set up web servers. So, it will save setup time and complexity.   **Weakness:**   * Main issue to have server in Mac is the cost, it is so high compared to other web server hosting platforms. * It has limited software availability to support web server hosting and purchasing these needed additional tools will add more cost. * MacOS is not providing much support and enhancements/new features for server side compared to client-side applications. | **Characteristics:**   * Linux’s best characteristics are secure, reliable, and stable. * It is open source and provides extensive options to customize it based on needs. * In the market various Linux distributes are available to choose from, such as Red Hat Enterprise, Ubuntu, and CentOS.   **Advantages:**   * As it is open source, it is the most cost-effective option to host a web server. * It provides root access, so we can install only required tools and uninstall any unwanted preinstalled software. * Linux servers are highly scalable and quick, which is needed to support any type of traffic. * Larger community support available.   **Weakness:**   * Command-line interaction is not user-friendly and requires learning syntax, which is difficult and takes time. * Some of the softwares are not available in Linux distributions which will force to use available tools. | **Characteristics:**   * Windows Server OS is used to host web server in window platform. * Provides better user-friendly options to develop and deploy web applications. * If web application developed using .NET framework, then windows server provides easy options to deploy and manage.   **Advantages:**   * Windows server regularly applies patches to protect the platform. * Window server provides native support to windows-based tools, framework (.NET) and database (Microsoft SQL Server). * Using graphical user interface web server and application can be easily configured and monitored.   **Weakness:**   * Due to frequent patching, the Windows server must be rebooted, impacting web application availability. * Virus and malware attacks are more common in Windows Server, impacting application security. | **Characteristics:**   * Generally Mobile platforms (iOS, Android) are not used to host web applications. * Used to install APPs to run and provide file system management. * Not supporting web server for production ready application.   Advantages:   * Can be used to develop proof of concepts and protypes can be easily developed and tested.   **Weakness:**   * Will not provide enterprise options to host web application in web server. * Not scalable and performance impacts. |
| **Client Side** | **Cost:**   * Safari browser is built in MacOS to access gaming applications. * MacOS has various supported versions, so in each version, the app should be tested, so it requires more infrastructure to have these versions which impact cost.   **Time:**   * To test gaming app in all supported Mac versions takes more time. * To test performance, responsive features and features supports impact time.   **Expertise:**   * Generally, Safari browser compatibility to test gaming app required more experience and expertise. * Sometime common features are not supported in Safari or behaves differently. To overcome this, we need to apply specific logic. | **Cost:**   * Linux has various distributions and desktop environments. To test game apps in all these will increase the manual testing cost. * App distribution may involve licensing cost.   **Time:**   * Development time increased to support all the Linux distribution and desktop versions. * Troubleshooting via debugging also takes more time.   **Expertise:**   * It is hard to get expertise in all these distributions. To manage logic between distributions and their versions needs complex code to handle. | **Cost:**   * To provide better gaming experience in windows browser (Edge) gaming app should support latest technologies and responsive design. * As windows supports to install other browsers as well testing cost increased to test the app in all these browsers from windows.   **Time:**   * As Windows requires frequent updates, testing time was impacted.   **Expertise:**   * Strong development skills required to support all the versions of browsers. * Automated testing tools and knowledge required to test the app. | **Cost:**   * Development environment licensing cost. * Apple developer program cost.   **Time:**   * App Store review process time varies in different app stores. * Development time varies based on the features.   **Expertise:**   * Expertise required to develop apps in iOS. * Game designing skills to implement UI/UX features in iOS. |
| **Development Tools** | * Xcode is the IDE required to develop the app. * Audio editing tools (Adobe Audition, etc.) and Graphics design tools (Adobe Photoshop, etc.) * All these tools are required licensing cost and required specific hardware requirements which takes additional cost. | * Visual Studio Code (IDE), Godot (specific to gaming development). * GIMP is a graphics design tool. * Audacity is an audio editing tool. * Git version control tool. | * Gamemaker Studio 2 – Game development too. * Visual Studio, Unity – IDE * Construct 3 – helps to create 2D games. | * xCode – iOS app development IDE * Swift Playgrounds – used to prototype small code. * TestFlight – To test iOS apps. * Android Studio – Android app development IDE. * Gradle – build tool. |

## 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**:

Based on the details provided in the evaluation section, I would recommend using Linux operation platform. Below are some important factors to prefer this option,

* **Cost-Effective:** Linux is free and provides open-source operation system distributions.
* **Scalability:** Linux supports scalability very well, and it is easy to scale application instances using container-based (Docker or Kubernetes) images based on the traffic.
* **Performance:** Linux performs better, which is critical for gaming applications. It used low-latency and high-throughput approaches to achieve high performance.
* **Stability and Reliability:** Linux instances can run longer than other operating platforms with our interruption, i.e., no frequent reboots are required to free up resources. Gaming application requires uninterrupted service for 24/7 availability.
* **Customization:** The gaming application requires frequent changes and a lot of flexibility to apply customization to support user feedback and improvements. Ubuntu and CentOS are famous distributions to select. My choice is CentOS Stream because of its stability and enterprise readiness, and it is based on the Red Hat Enterprise Linux distribution.

1. **Operating Systems Architectures**:

Cent OS Stream (Linux) operation system architecture is based on x86\_64 (64-bit), the latest design to run 64-bit processors, and supports more extensive memory needs. Cent OS can support binary compatibility with RHEL binary, the main advantage of using enterprise-level features/tools without any associated cost.

Cent OS Stream is the successor of traditional Cent OS, and the latest version is Cent OS Stream 9. Cent OS Stream supports rolling release distribution, so frequent updates are distributed with the latest features and security updates. It provides more predictable and continuous releases and is designed to be in the middle of Fedora and RHEL as a platform to support development and testing.

1. **Storage Management**:

Multiple storage options are available, such as Local Storage using SSDs or HDDs, File Systems Ext4, XFS, Network Attached Storage (NAS), Object storage from Cloud platforms, and Database storage.

I recommend a database storage option for the Draw It or Lose It game. In the database, we can store both team player details and the image. Among various database options like RDBMs databases such as MySQL, PostgreSQL, or NoSQL databases such as Monga DB and Cassandra, my choice is MySQL, a commonly used free enterprise-level Database.

Install MySQL database on Cent OS Stream operating system with InnoDB engine, which supports ACID transactions. MySQL provides backups, and replication options in case the primary database got impacted to resume the service quickly by using them.

1. **Memory Management**:

Linux uses swap space as virtual memory when physical memory RAM is fully utilized. With a proper swap memory configuration, we can avoid any performance issues. Also, we can use caching options such as Redis or Memcached to speed up the resource fetching process.

Linux provides a default memory management tool, malloc, which can offer a more efficient way of memory allocation and deallocation. Some advanced memory management tools like tcmalloc or jemalloc can be used to improve memory management.

Also, in Linux, we can allocate memory in fixed-size blocks which can reduce fragmentation and improve memory allocation management. Multithreading supported in Linux can ensure each resource access is controlled and managed.

1. **Distributed Systems and Networks**:

Distributed systems and networks can be achieved using web services to provide clients with Draw It or Lose It game features. REST-based web services use JSON as input and output types. It is a modern distributed architecture that can be easily integrated with various clients, such as desktop applications like browsers and mobile platforms through apps. Web services can consist of many APIs which will provide specific functionality, in our game application to get teams, players from each team, game instances, etc.

All clients (Mac, Windows, Linux, iOS, Android) call the same REST API, so there is no need to develop back-end systems for each client. JSON datatype contains a key/value pair as an input and output mechanism, and all these clients use REST APIs to communicate with other systems.

HTTPS protocol should be used to communicate from the client system, which is secure and encrypted. Wireless networks (WI-FI) should be able to connect distributed systems to enable mobility. Network bandwidth should be sufficiently configured to handle large-size image files.

Content Delivery Networks can deliver static resources such as fixed images to avoid latency. Routing mechanisms can be used to support high and low-traffic clients, e.g., mobile platforms generally have high traffic for gaming applications compared to desktop clients.

1. **Security**:

Data encryption is the primary security control required to persist or transmit data over the network. TLS or SSL can transfer data securely over the network, and encryption/decryption algorithms can encrypt data before storing and decrypting data while retrieving it.

Linux servers should be updated with the latest security updates and patches to prevent any security attacks. Implement a solid firewall to control incoming and outgoing calls. IDS/IPS intrusion detection and prevention systems can monitor and block malicious activities.

Application code should be developed and tested with secure techniques to catch any SQL injection or cross-site scripting attacks. Mainly implement user access control logic, which can provide only required access to the user with a specific role. Also, if possible, multifactor authentication (MFA) to provide more secure access.

Network security can be achieved by whitelisting specific IP addresses and setting up Virtual Private Networks (VPN) to communicate between related and relevant systems. Code scans should be done frequently to find any latest vulnerabilities and should be fixed based on severity.