

<Gaming-Room-Drawit-Loseit>

# **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 | <09/25/24> | <Noah-Khomer> | Added everything in the template as needed |

**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 currently has an Android game called Draw or Lose It which needs to work on multiple platforms. However, they also need a web-based version for their Android game. The objective of this executive summary is to develop a solution in which multiple teams and player can utilize unique identifiers for their entities so that duplicate names for team and games are not allowed. Also, the application requires that there is only one instance of the game in the memory at any time. To design and solve this solution. The most appropriate formula to use is the Object-Oriented Principle and the design pattern such as Singleton and iterator. The benefit of using these patterns and techniques with object principles is that it will efficiently manage resources where we can fulfill the clients’ requirements. Therefore, by the end, the web-based application of the Android game will function over different platforms while also maintaining the system performance with a strong emphasis on security.

## Requirements

There are a total of 4 requirements for the application.

1. The application in which the game takes place must allow a lot of teams and players.
2. The application in which the game takes place must also have game and team names unique where they are able to check for all the duplicates each time, they create a new game or the team.
3. The application should always check that there is only one instance of the game existing at a Single time.
4. Lastly, the application of the game requires that the game, team, and player must each have unique identifiers.

## [Design Constraints](#_2et92p0)

There are a lot of design constraints for this project. The reason for that is because it's a web-based version of the same game. Therefore, with games on webs, there are issues with scalability, data consistency, memory, resource management and security. Scalability becomes issue because the game should be able to support multiple users across different platforms. It is something very common currently because not everyone uses the same platform to access the same game. Therefore, it should be made sure that the server and the client resources are met effectively to scale the game. The second issue is with the data consistency, as we can see that the game requires that we have a unique identifier for the game, teams and players. Therefore, implementing strict rules and checking for unique name. In every single user session is important. Memory and resource management would normally not be a problem. However, this application requires that only one instance of the game is existing in the memory at any time. Therefore, the application will need to manage memory more efficiently. Lastly, there should be a strong emphasis on the security because the game is being distributed across multiple platforms such as Android and web based. Therefore, it is important for us to secure the communication and user data thoroughly on each platform.

## [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 given below gives a relationship between different classes such as Entity, Game, Team and Player. The entity class has a strong focus on holding the common attributes like ID and name which is giving a strong foundation for the game, team and player classes which are taking which are inheriting directly from the entity. The Game class is focusing on extending the entity class by representing the single game instant. By doing this, we're able to have a game where multiple teams can exist. The Team class is also extending the Entity class just to make sure that each team can have multiple players and team names can be unique. Lastly, the player class is focusing a strong inner emphasis on individual players who belong to the teams. The player names also must be unique, which is being accomplished in the player class attributes. In this UML class model, there are a lot of object-oriented principles like inheritance and capsulation and polymorphism. The usage of inheritance is making sure that there is no code duplication, while encapsulation is making sure that class can manage and is privately only giving the necessary methods to each other. Lastly, the Singleton pattern is being used in the Game service class which is making sure that there is only one instance of Game existing in the memory which was the main requirement of the project.

**"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 can provide more cloud-based solutions like AWS however it is less common so scaling it is going to be difficult | Linus is very popular for web application because it can provide open-source flexibility and low cost overall | Windows give a lot of options when it comes to hosting because it relies on .Net, however it becomes costly because of licensing fees | Mobile devices are usually clients and so not used to host applications. However, cloud services such as firebase can assist in server side |
| **Client Side** | Mac requires a very high-end proficiency in programming language like Swift and specific frameworks which can make the application responsive, but it utilizes Safari not google so more issues there | Linux requires that you have knowledge on Docket and Kubernetes to scale the software. However, the main benefit is that there are no licensing costs | Windows give access to tools like visual studio for development so .Net framework expertise is needed and this reason cost and time is always going to be higher than Linux but more balanced | Expertise in Swift, Kotlin and other frameworks like Flutter and React is needed which can cut down on costs and time. However, making sure these frameworks are compatible with Safari and Chrome can increase them. |
| **Development Tools** | The main IDE used to configure Mac programs is Xcode where Swift and objective C is used, and they are proprietary software which means Apple developer account is needed | Linux gives access to a wide range of development tools such as Eclipse, IntelliJ, and Vim. Ability to use Java, python and PHP will surer site is responsive and scaled. | Visual studio code because the main Ide with focus on .Net, JavaScript etc. However, the licensing fees are very high because windows are seamless when it comes to integration. | Development can be done using Xcode or Android studio depending on the OS used. The solution is to use unified development like Xamarin. |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**:

The operating system that should be used to make the server-side hosting game is going to be Linux. The reason for this is because Linux is open source which is leading to a lot of flexibility and cost effectiveness as seen in the chart. Linux can focus on a lot of client-side development which can lead to utilizing frameworks like Flutter or even React Native for both iOS and Android. It is important to remember that the main takeaway is to ensure compatibility between different platforms while maintaining the costs to be low. Therefore, I believe Linux is going to be the perfect operating system to utilize server-side hosting.

**Operating Systems Architectures**:

operating system architecture of Linux can provide access to utilizing containers like Docker which can lead to a lot of scalabilities which is also needed by the main business partner. Scalability can then help focus on responsive front-end technology for compatibility. Aside from that, another thing that scalability provides is to make sure the growing numbers of users can be handled correctly. Scaling the software now will not only manifest growth but also show preparations to adopt heavy user load. This flexibility will allow improved user experience.

**Storage Management**:

There are a lot of cloud-based solutions such as Amazon Web Services or Google. Cloud which can offer scalable storage throughout the Linux server easily. The main thing these solutions provide is redundancy. For example, data is stored securely and then easily accessible whenever necessary. The integration of cloud solutions such as AWS or Google Cloud will help the game be scaled for a higher user base and keep its performance high and consistent.

**Memory Management**:

Linux has an open-source service in place which offers virtual memory system. This virtual memory system acts as a solid utilization between were. Memory should be used highly and where it should be used less likely. For example, a lobby with 10 players should use the memory less. Meanwhile at the same time a lobby with 100 players should use the memory more. The main thing a system like this ensures is that resources are used properly where they are needed the most without wasting memory and overloading the system.

**Distributed Systems and Networks**:

To streamline the communication between the game and the web clients and even the mobile devices, one of the main things is to utilize API with Amazon Web Services or even Azure. The main takeaway is that these platforms can support the distributed systems in a way that ensures minimal latency. The minimal latency will help make sure the game is working smoothly over multiple platforms. Seamless gameplay and communication across different platforms or devices will be easier to manage.

**Security**:

Since the game generalizes different types of communication across different platforms, one of the main things to utilize is encryption. The encryption system will make sure that the user data is secured. In this sense, utilizing Linux is one of the best things because Linux has a built-in firewall configuration and strong security protocols. For example, the use of iptables can be very effective when it comes to blocking unauthorized access. At the same time implementing SSL/TLS protocols for data encryption can be very beneficial with Linux. Lastly, Identity and access management can provide granular control over who can access the different types of resources and when. This way only authorized users who have the proper permissions are able to access the software, which leads to a higher sense of security. In conclusion, iptables for the firewall protection, SSL/TLS for the layers of added security and IAM for access management control for authorized users are all impressive ways in which Linux can provide a comprehensive security system for the game while protecting user data over the different domains of platforms by utilizing proper compliance with security standards.