

<Name-of-Software-Application>

# **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 | <11/17/24> | JJ Newell | Initial release |

**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 client currently has a game called “Draw It or Lose It” which is an app only available on Android. The client wishes to take this game and make it web-based so it may serve multiple platforms. The current development team does not know how to set up the environment. They require help in facilitating the development of their web-based version. There are four primary software requirements that must be met in addition to this goal. 1) A game will have the ability to have one or more teams involved. 2) Each team will have multiple players assigned to it. 3) Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. 4) Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## Requirements

There are four primary software requirements that must be met in addition to this goal. 1) A game will have the ability to have one or more teams involved. 2) Each team will have multiple players assigned to it. 3) Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. 4) Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The primary design constraint for this project is the requirement of making it multi-platform. As stated earlier, the current development team is not familar with a multi-platform project. Part of overcoming this constraint will be training the team so they are knowledgeable and can work on this project. Another constraint would be ensuring that the app is consistent across multiple platforms. It has specific criteria to meet, so consistency between platforms must be met.

## [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 ProgramDriver class uses the SingletonTester class and also contains the main function. The Entity class is the parent class of this diagram, with the Game class, Team class, and Player class all acting as Entity’s child classes. Game, Team and Player all inherit the attributes of the Entity class. The GameService class is used so that the client’s four software requirements, stated previously, are all met. In addition, the Game class has a Team list, and the Team class has a Player list. However, the Player class does not have a list, instead it simply holds the Player’s id.

**"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** | Macs have no issue working with Linux and Windows devices, making them valuable for multi-platform projects. However, Windows and Linux are both capable of doing this as well, and equivalent hardware for a pc running Windows/Linux is cheaper. | Linux is a highly scalable and versatile OS that allows for plenty of room to grow depending on possible future needs of the application. While Linux does support a wide range of hardware and software, it is more likely to run into an issue where one of these is not supported, when compared to other platforms. | Server technologies such as ASP.NET and SQL are fully supported on Windows as they are Microsoft products. A drawback of this would be the licensing costs involved with utilizing such products. | Mobile devices could be useful for testing small, low-cost apps that do not require much processing power. Besides this, there are a large number of drawbacks to using mobile-based servers. They are generally cloud-based, which poses security threats. Also, Mobile devices simply lack a great deal in terms of processing power when compared to other platforms. |
| **Client Side** | A benefit of using Mac for this is that it is easy to use once the developers understand what they are doing. The downside however is the simple fact that Mac OS is only accessible on Apple devices. And Apple devices are much more expensive than their Windows and Linux counterparts. | Linux generally has better security than both MacOs and Windows while maintaining its wide versatility. However, the drawback to this is its steep learning curve. It is much more difficult to properly learn how to utilize Linux when compared to Windows or Mac. | Windows greatest strength is its ease of access coupled with its versatility of development tools. However, the downside to this would be the various licenses that may be required when using Windows. | One of few pros of using mobile devices in this regard is that the price point is generally rather low. The primary downside is the simple fact that a mobile device cannot compete with a PC or Mac as these platforms beat Mobile in almost every regard. |
| **Development Tools** | The primary programming language on Mac that would be used for this sort of project would be Swift. The primary IDE that would be utilized would be Xcode. | The primary tools that would be used for creating this sort of web-app on Linux would be JavaScript, HTML & CSS. In addition to this a simple IDE such as Visual Studio or Visual Studio Code would be sufficient in most cases. | Windows is quite like Linux when it comes to developing this sort of app. The primary tools that would be used for creating this sort of web-app on Linux would be JavaScript, HTML & CSS. In addition to this a simple IDE such as Visual Studio or Visual Studio Code would be sufficient in most cases. | While Java is not generally used for multi-platform apps, it can still be used for it. Since the development team has created a native Android app already, chances are they may be familiar with Java. In addition to this a simple IDE such as Eclipse would suffice. |

## 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**: Windows would be the most recommended choice. It can integrate with their pre-existing Android application. Not only this but around 70% of the world uses Windows, as it is easy to learn and there are a plethora of tools and resources on it.
2. **Operating Systems Architectures:** While the latest version of Windows may not be the best in the eyes of Developers, it is the most recent. So, using it would be best as any technologies on it will be supported for longer and new technologies will almost certainly be supported in the latest version.
3. **Storage Management**: Server-based storage would be best for this sort of application. It provides an adequate amount of security, while also allowing for games to be backed-up while also ensuring duplicate games do not exist.
4. **Memory Management**: There are several ways in which Windows can manage memory, two primary ways being the use OneDrive for cloud storage and another being Azure cloud storage.
5. **Distributed Systems and Networks**: Using a cross-platform environment will help to limit the sheer amount of expertise needed for individual platforms, instead focusing on technologies that are primarily used for multiple. To address various issues in regard to connectivity and outages, the client should create enough server space according to their projected userbase.
6. **Security**: While Windows may not have the same level of security as Linux, it still has built-in protection such as Windows Antivirus and Firewall. However, in addition to Windows’ latent security, I would also suggest the use of an additional security tool such as OAuth coupled with industry best security practices in mind when developing the cross-platform application.