

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 | 01/22/2024 | Kameron Jackson | Updates to Executive Summary, Requirements, Design Constraints, Domain Model, Evaluation and Recommendations. |

**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 current design problems include having an application which the client would like to have on multiple platforms being available on a singular platform. There is also a lack of knowledge in setting up an environment for the web-based version of the game. The solution is to discover a way to make the application functional across multiple platforms and design the environment for the web-based version.

## Requirements

Requirements include the ability to have one or more teams involved, having multiple players assigned to each team, unique game and team names and unique identifiers.

## [Design Constraints](#_2et92p0)

What needs to be accomplished entails transforming an existing application that is available on Android and converting it to a web-based application. The potential design constraints include the application not running at the same capacity on the web-based version versus the current version, making it frustrating for current users to utilize. Another challenge is scalability and making sure that the database is able to withhold the amount of information across both platforms, this could potentially compromise the functionality of the cross platform ideation.

## [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 below UML class diagram demonstrates the classes needed to make the current application of Draw It or Lose It functional. The ProgramDriver class contains the “heart” of the functionality by possessing the main() method. The SingletonTester class and the ProgramDriver class relative by unidirectional association which is shown by the <<uses>> meaning that the ProgramDriver class uses the SingletonTester class. The Entity class is used as a base class to inherit from in the Game, Team and Player classes and contains the commonly used attributes and methods, eliminating the need for redundancy in the code. The GameService class serves as a singleton service for the game, checking to make sure that there is one instance in memory. The game, team and player classes contains the lists of teams and players. The object oriented principles used are inheritance as stated above, encapsulation and association which is what the diagram depicts.

**"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** | The advantages of Mac on the server side from experience is the user-friendly ability that it has, as well as the security in comparison to Windows. The disadvantages can be compatibility of some applications that are not interchangeable with other operating systems, making Mac a lesser option in this scenario. If this ended up being a larger scaled application, extra hardware costs could be involved. | The advantages of Linux are it is a cost-effective operating system with flexibility in its server’s environment. The disadvantages are hardware compatibility and the lack of commercial support. Linux has several secure options when looking at server-based hosting. Cost associated would lie within application support if necessary due to Linux operating under open-sourced licensing. | The advantages of Windows are the user friendliness and with it being a widely used operating system, Windows offers strong server-based deployment and supports multiple technologies. Weaknesses include possible not being the most cost-effective option. | The advantages of Mobile Devices are mobility and the ability to have your device anywhere with connectivity. The disadvantages are security concerns and network dependence. iOS and Android historically are the least used in handling platforms for the actual application. Typically, it is hosted by a pc server, then made to be cross compatible for other servers. |
| **Client Side** | The software considerations necessary to ensure the application Is compatible with Mac, we would be looking at developers that are proficient in Mac specific languages, the time considerations would include what timeline GameRoom is looking at. There would potentially be specific frameworks applicable to make allowances for cross platform compatibility. | When looking at software development considerations there are factors of the cost it takes to employee developers that are specific to Linux, as well as the software licenses, etc. The time would depend on the specifications based on the hardware, updates, if there are any issues in the development process etc. It Is required that the features and capabilities, as well as limitations are tested during the development process. | When looking at software development considerations there are factors of the cost it takes to employee developers that are specific to Windows, as well as the software licenses, etc. The time would depend on the specifications based on the hardware, updates, etc. The application would need to go through a series of tests to ensure the response is accurate and the user does not lose any features that is currently experienced. | When looking at software development considerations there are factors of the cost it takes to employee developers that are specific to mobile devices, as well as the software licenses, etc. The time would depend on the specifications based on the hardware, updates, etc. There would need to be tests ensuring the application response time is accurate and the graphics are translated. As well as runtimes to ensure quality user experience. |
| **Development Tools** | The relevant coding languages for Mac are Swift and Objective-C. The official IDE for MacOS is Xcode. Git is used for version control. Seeing as Windows is the most used operating system, it would be necessary to employ developers that have expertise in these languages. There would be licensing costs and potentially distribution costs to create and deploy the software. | The relevant languages for Linus are C, C++ and Python. IDEs that are used are, Visual Studio Code and Eclipse. Git is used for Version Control. Similar to developing on Mac, developers specific to this platform may be necessary. There are not usually licensing costs associated with developing a program within Linux due to it being an open-source OS. | The relevant languages for Windows are C#, VB.NET and C++. IDEs are VS and VS code. Git is used for Version Control. There would be a need for specific developers that exceed in these languages/tools. There are licensing costs associated with developing an application within Windows. | The relevant languages for mobile devices are Java, Kotlin and Swift. IDEs are Android Studio, Xcode and React Native. There would be a need to have developers specific to the languages being used. Java can also be used in other platforms, so developing in Java and ensuring that it has cross-compatibility would lower the cost options and might be an efficient option. There would be licensing costs for both Android and iOS |

## 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**: The operating platform I would recommend is Windows. Particularly, Microsoft Azure. This recommendation is based on Azure being a cloud-based operating system.
2. **Operating Systems Architectures**: The most commonly used architecture within the Windows desktops are x86(32 bit) and x64(64-bit). Microsoft Azure offers virtual machines, PaaS which would be used for scalability and integration services.
3. **Storage Management**: The storage management system to be used is Storage Spaces. If choosing to use the Azure platform, Azure Blob Storage would be the best storage management.
4. **Memory Management**: Windows uses virtual address space that enables addressing up to 4 gigabytes of memory. Microsoft Azure offers virtual memory management.
5. **Distributed Systems and Networks**: The best way that I’m aware of to communicate between various platforms would be to develop a language that is able to be processed by multiple platforms, also by being able to be accessed by multiple types of networks. I.e. Wifi and cellular.
6. **Security**: The best way to ensure security within the application would first be to use security promoting techniques within the code such as encryption where necessary, secure configuration and the use of secure frameworks.