

<The Gaming Room>

# **CS 230 Project Software Design**

**Draw It or Lose It**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <07/17/22> | <Joshua Perez> | <Original submission of project 1> |
| 1.1 | <08/16/22> | <Joshua Perez> | <Original submission of project 3> |

## [Executive Summary](#_sbfa50wo7nsh)

The gaming room wants to make a web-based game that serves multiple platforms. The game that they want to develop is based on their currently existing game, Draw it or Lose it, which is available only on the android platform.

## [Design Constraints](#_2et92p0)

The design constraints for developing a game application in a web based distributed environment fall with the limitations of server-based configuration as everything must be hosted and must have scalability considered. On non-PC based platforms such as IOS and the Android platforms, there will be a client-side app that accesses the web-based application. There should be a standard of uniformity and theme present across all versions of the game. On computer-based platforms, there will be a HTML component. All versions of the game will have to contact the web server and sync data and possible microtransactions that the game may include.

## [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 entity clasts created in this most recent iteration is the parent / super class of the Game, Team, & Player classes. These classes inherit the attributes of the super class, Entity. In other words, these classes will all include the attributes “id” and “name”. The Game Service class references Game. Game references the Teams class, the Teams class references the Player. This means that Game Service can contain multiple games, and each game can then have its own teams, and each team can have its own players. This is a factor of aggregation.

**"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 OS servers have the advantage when it comes to managing servers and mac clients. It is also quite a clean GUI that is relatively easy to decipher | Linux is particularly scalable and less susceptible to cyber-attacks in general. Since Linux is open source it requires a higher skill level to use in general. | Windows servers are particularly easy to use and have great support as they are widely used. | Mobile applications require hosting and to pass certain requirements to get on the respective app stores. |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.

While I personally game on Windows and prefer the platform over any others for development, for the product “Draw it or Lose it” I would instead opt to go with mobile gaming platforms as it is very apparent that is where the product will do the best and produce the largest amount of monetary gain. Since the product already exists on Android operating systems, I think the next logical leap would be to apply for Apple’s IOS Platform and put the game on there.>

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.

Apple’s IOS is designed in a layer-based structure. First, we have the core layer, aptly named “Core OS Layer” which contains basic Frameworks crucial to the operation of the system. Then we have the Core Services Layer which helps the OS provide better functionality. This layer contains many of the crucial framework components that helps the platform provide much quality-of-life features. Next comes the media layer, which manages all graphics, video, and audio technology present in the system. This layer is crucial in bringing to life the sensory components of the system and contains the Frameworks that will be where the graphics for the game are rendered. The next and last layer is known as the Cocoa Touch layer aka the application layer, where the Frameworks for the applications are stored and utilized. This is the layer that the user actually interacts with and is our gateway to interaction with this operating system architecture.>

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.

An appropriate storage management system that can be used with Apple’s IOS is SQLite, currently the most popular solution due to its open-source relational database. It’s user-friendly programming interface is written in ANSI-C. SQLite has several advantages in the form of safe access from multiple processes and threads, and the ability to make simple but powerful queries that can manipulate, add, and delete data. >

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.

Apple’s IOS used to be non-arc, but now it supports arc and Xcode automatically processes this in real time during the compiling stage. When it comes to Draw It or Lose it, Apple’s IOS will use ARC’s functionality and will release anything no longer in use in order to save memory and use what is available more efficiently. By using a retain count, or reference count to assign a value to an object, that object will only exist until the code reaches the end of a code block unless told to do otherwise in some form. We can use a similar method to represent the images in Draw It or Lose it, only loading them into cached memory when they become essential for the program, keeping them in reserve on the HDD until they are needed. >

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).

Luckily this is something that can be solved via relatively simple methods on mobile platforms, and readily applies to Apple’s IOS platform. When I say relatively simple, it is a bit of an understatement, but it is doable enough that I’ve been able to do it on my own with only a little bit of knowledge and IONIC’s documentation. IONIC is a particularly popular cross platform development tool and framework that is based on AngularJS. I can absolutely recommend it and note how user friendly and easily implemented it is. I used it somewhat late into the development of a game I was working on in Unity and it took very little time to implement it in and have functioning multiplayer. >

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.

This is one of the areas where Apple’s IOS succeeds highly. It is a platform known for its security prowess and it is often noted as being more secure than most of the other popular operating systems available. Security is built into the hardware itself with their own proprietary boot ROM, dedicated AES engines, and a Secure Enclave. Apple’s IOS starts the security from the ground up with a Secure Boot every time the system starts up and makes sure that the next step in the process is running properly and secure before moving on and handing over control. We can also use further security on the application itself, likely a third party security solution like most companies use. >