

Draw It or Lose It

# **CS 230 Project Software Design Template**

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 | 02 /020/2022 | Florangela Gomez | Analyzing the characteristics and techniques to various systems architectures. |

**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 has requested to develop a web-based game that is able to run on multiple platforms. Their game is Draw It or Lose It, and it is currently only available on Android. The purpose of the game consists of the application rendering images from a large library of images. Multiple teams with several players going four rounds a minute each turn. Drawings are rendered and complete at the 30 second mark. If the playing team does not answer within the time mark than the remaining teams have an opportunity to offer on guess to solve the puzzle within 15 second mark.

## [Design Constraints](#_2et92p0)

* Requires one or more teams
* Each team have multiple players
* Game and Team names must be unique to allow users to check whether a name is in use when choosing team names.
* Only one instance of the game can exist in memory at any given time.

The requirements above are what needs to be followed during the designing and coding process. The Gaming Room would like to run on all devices, as of right now it only runs on android. In order to accomplish running the game on devices with Windows, Linux, and Apple we would need to come up with a way where the existing code can inherit other languages depending the device.

## [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 game model has a base Entity class containing attributes ‘id’ and ‘name’ creating a relationship between Game, Team, and Player class. This means they all inherit information from the Entity class each class will share common references such as attributes ‘id’ and ‘name’, making Entity a super class. When looking at the relationship between Team and Player there is a composition type, meaning it’s an instance of one class and has a reference to an instance to another class. When looking at this UML diagram we see the GameService references Games; Games references Team; Team references Player.

**"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** | Advantages  Reliable on a Mac OS server you can run all the same software you use locally without  issues.  Disadvantages  Mac servers often cost more because they are less in demand | Advantages  In terms of security, Linux is much safer than other OS.  Web hosting is compatible with most popular web hosting software and hardware.  Disadvantages  Servers are more difficult and require more expertise. | Advantages  High resource requirements.  Offers more software availability  Disadvantages  Websites with high activity can have problems with Windows hosting.  Servers require frequent reboots to run properly. | Advantages  Scalability, cloud hosting can grow or shrink according to needs.  Transfer speeds can be faster on-site hosting services and offers more robust backup capabilities.  Disadvantages  Poor security, data is accessible virtually making any device prone to security threats. |
| **Client Side** | Moderate expertise and less time require compared to Linux. Costs more compared to other OS. | Maximum expertise and time required. Costs is much less compared to others. | Maximum expertise and time required. Similar to mac on costs. | Moderate expertise but, time required but, cost effective. |
| **Development Tools** | Common programming languages: C++, Ruby, Python, SQL, Java, HTML etc. Apple provides Swift and Objective-C APIs to do ‘Mac-specific things’ they are recommended. | Common IDEs: Visual Studio, Eclipse, Notepad++, etc. Along with many common languages supporting the frontend and general-purpose languages | Runs the same as Linux regarding IDE and programing languages but, it is much easier to use Windows vs. Linux. | Common languages: Java, C++, Python, HTML, CSS, JavaScript, etc. Android Studio and Swift can be used to make apps. |

## 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 is by far the most user-friendly, efficient option when it comes to custom applications and games. Linux does offer support for gaming in several platforms however, it is extremely complex and does not offer nearly as much support as Windows and MacOS. Apple’s Mac operating system is popular and provides stunning visuals. Unlike Windows which allows endless customization and upgradeability, Mac does not offer the same. MacOS can be limiting, exclusivity holds it back. I would recommend The Gaming Room to start on Windows machines for it is the largest platform for gaming in PC. It has more software availability, minimum expertise, and cost.
2. **Operating Systems Architectures**: The Windows operating system design includes file management that allows the user to control and manage the computer’s memory. It provides opportunity for developers to work with many different programming languages.
3. **Storage Management**: The Windows Storage Management provides a wide range of storage configurations. It allows you to scrutinize and manage files externally, and internally. You can use the cloud to save data. The built-in storage system allows for easy file creation and placement for large projects to avoid losing or an accidently delete.
4. **Memory Management**: The Windows memory allocation allows for easy storage of pictures outside of the default picture folder. While creating the game Draw It or Lose It, there will need to be a database or library with lots of images stored. Windows process on 32-bit has its own virtual address space that allows addressing up to 4GB of memory. Windows process on 64-bit has virtual address space of 8TB. All threads of a process can access its virtual address space.
5. **Distributed Systems and Networks**: Cross-platform game creation will be helpful in being able to publish the game to run on all devices. There are several cross-platform tools for app development such as React Native which interprets your source code and converts it to the native elements. Its framework is emerging enabling rapid prototyping and offers initial velocity.
6. **Security**: Windows machines come with a built-in security protection software called Windows Defender. It is pre-equipped with protection that scans the system for malware, viruses, and security threats. However, for maximum security it is recommend to have additional software such as a third-party antivirus such as Bitdefender, Kaspersky, etc. Window Defender lacks endpoint protection and full-service investigations that other anti-virus software offer.