

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 3/21/2024 | Jacob Virgilio | Initial software design |
| 1.1 | 4/6/2024 | Jacob Virgilio | Client, Server, Tools |
| 1.2 | 4/19/2024 | Jacob Virgilio | 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](#_35nkun2)

The Gaming Room wants to develop a web-based game that can run on multiple different platforms. The game is named “Draw It or Lose It”, and currently is only available on Android. This game consists of multiple teams of people guessing a picture from a library of images until a timer runs out. Each team member of all the teams gets to answer for 15 seconds.

## [Design Constraints](#_1ksv4uv)

1. One or more teams
2. Each team needs multiple players
3. Names must be unique for instancing
4. Only one instance at a time
5. Must run on multiple different platforms

The Gaming Room would like this game to run on multiple different platforms. We currently already have the game on Android only. Windows, Linux, and Apple compatibility will be needed.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

Entity creates a relationship between Game, Team, and Player class, meaning they inherit information from Entity. Each class shares common references. Game has a Team and GameService has Games. When UML is used, it's an instance of one class, and has a reference to an instance in another class. As shown in the diagram, GameService has a reference to Games, which has a reference to Team which has a reference to 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](#_z337ya)

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** | Relatively easy to use console commands, and popular enough as a server. Server is upgradable, but can be costly, and also consists of proprietary code. | Easy to use console commands, and extremely malleable. Not as approachable as other OS, due to high skill entry. | More available total software to use, most widely used OS. Less overall loading times due to optimizations. Easier victim to viruses. | Portable, more affordable and popular then other options. Wider range of compatibility. Poor security, multiple different mobile device systems. |
| **Client Side** | Moderate cost, time, and expertise required. Some proprietary code sometimes shown on Mac systems. | Greater time and expertise is required due to the nature of Linux. Cost should remain about the same as Windows and Mac, since Linux is able to use different languages effectively. | Moderate cost, time, and expertise required. Windows is the most popular computer system OS, maybe requiring less client side time investment. | Most mobile OS, for both clients and developers. Limited processing power due to size, Moderate cost, time and expertise required due to the multiple different iterations of mobile OS. |
| **Development Tools** | Swift is the main programming language on Mac, although Macs can still run most languages. | Python is very popular among linux users, and is very user friendly. Linux systems are able to use most languages. | Windows has access to multiple different coding languages such as Eclipse, C++ and python. They are all easily accessible, and easy to run. | Swift, Kotlin, Java, and C++ are all good options for mobile development, based on which type of OS you are on. |

## 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**: I would personally recommend that The Gaming Room starts with Windows, as it is one of the most popular platforms with less experience required than other operating systems.
2. **Operating Systems Architectures**: Windows provides programs and services that give the developer and software tester information on different types of system information. This includes resource usage, and other types of multimedia. These services can be used by both a user account and a server.
3. **Storage Management**: Windows provides a cloud-based storage system, allowing saving and backups at a much easier pace, as long as internet access is kept. This cloud system also provides an intuitive way of showing storage resources drawn from different parts of a program or development.
4. **Memory Management**: This game will require a database with a multitude of pictures. Memory allocation is a good solution to this, since it allows pictures to be stored outside of their default picture folder. This prevents the project drawing from a folder outside of its development scope.
5. **Distributed Systems and Networks**: Develop 4 is an IDE that can be run on any device. Once the game is created, it can be exported into the web, iOS, Android, and many more options that will allow cross-play. This will help with dependencies.
6. **Security**: Windows comes with a built-in antivirus, alongside additional security protection software. More than one layer of protection is better, and the Windows OS has a large variety of malware protection to choose from.