

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 |
| --- | --- | --- | --- |
| 3.0 | 06/20/2023 | Chase Barnes | Third Revision |

**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 is developing a web-based game called "Draw It or Lose It" that can be played on multiple platforms, the current version of the game is only available on Android. The game is a party game where multiple teams of several people compete to guess the most pictures in four rounds of one minute each. When a picture is pulled from a library of images, one team has one minute to guess what it is. If they cannot guess it, the opposing teams have 15 seconds each to guess. The team with the most correct guesses at the end of the game wins.

## Requirements

* The game needs at least one team to play.
* Each team should have more than one player.
* Game and team names should be unique, only the chosen names should be allowed to be used.
* The game can only be running one instance at any one time.
* Must be able to run on mobile platforms, as well as IOS, Linux, and Windows

## [Design Constraints](#_2et92p0)

* Game needs to be web based
* Data should be shared across all platforms
* API should be able to access all platforms
* Leverage different development kits for different OS’s

## [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 Entity class creates a relationship between the Game, Team, and Player classes. This means that all three classes inherit from Entity. In UML, we can show this with inheritance. Each class will share common references like "name" and "id". This makes Entity a superclass.

When we look at the relationship between the three classes, we see that Team and Player are "has a" relationships. This means that a Team "has a" Player and a Player "has a" Team. Game is a "has a" relationship with Team. GameService is a "has a" relationship with Game. In UML, we call this aggregation (HAS-A).

When a user "has a", it means that the user is an instance of one class and has a reference to an instance of another class. In this diagram, we see that GameService has a reference to Games, Games has a reference to Team, and Team 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](#_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** | Offers flexible terminal commands that can be configured, accessed, and changed using terminal commands which makes it easy for admins to manage the server.  Server is popular in web hosting due to scalability and reliability.  Pros: Easy to upgrade and versatile  Cons: Not as popular as other servers for web hosting as its more expensive and requires more technical experience to manage | Like mac with security, but more cost friendly. Less bloat and more customization available.  Pro’s: Security issues can often be caught before becoming an issue. Preferred choice for web hosting  Cons: Harder to find apps to support the needs of web hosting | Has way more software available compared to other operating systems.  Closed platform  Pro’s: High resource requirements, easier to use, lower load times  Cons: susceptible to viruses, cost to keep server up, setup can be difficult | Could prove to be efficient due to lack of necessary hardware to run.  Pro’s: wider reach, offers better portability, cost effective  Cons: Poor security, battery drain could cause the whole server to shut down |
| **Client Side** | Doesn’t play well with other operating systems but can work with web apps. Takes more time and has a steeper setup. Can have a higher cost for scale, hardware, and software. | Can take more time and experience for the developer to set up server, but due to free os and low hardware requirements, could be a viable option. | Development time would be lower than other options, the cost would be relatively average and an easy task for a developer to set up. | With a mobile development specialist, it would be pretty quick to develop, and a relatively low cost compared to enterprise computers. |
| **Development Tools** | Swift is the popular language to use with MacOS. Mac can run all languages fortunately but doesn’t support all IDEs. | Linux can work with most popular IDE’s, as well as notepad. Like Mac, Linux supports all languages. | Easier to use than Linux, but runs the same languages and IDE’s. | Using Swift and Android, you can create tons of applications, though it does not support IDE’s. It still will support all languages. |

## 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’d suggest Draw It Or Lose It start on Windows devices due to the large amount of software available as well as a typically minimum amount of effort and cost to get the project going. Not to mention there isn’t any lack of available IDE’s.
2. **Operating Systems Architectures**: Windows provides services used by all Windows-based apps. These services enable things like graphical user interfaces, access to system resources, and tons more. These can be used by bother user accounts and servers. Users can leverage these services to run apps and access the system, while servers can access other computers on a network.
3. **Storage Management**: Windows has a great storage system that allows you to section and manage files on the hard drive, as well as control how much space they take up. It also allows you to save app locations where you’d like, as well as access cloud data.
4. **Memory Management**: With this particular game, you’ll need to have a database with lots of images. The memory allocation allows for easy storage of images outside of the default folder. This helps to keep the whole project together and more secure, including when you’re working in an IDE.
5. **Distributed Systems and Networks**: Develop4 is a cross-platform IDE that can be run on any device. It enables developers to create apps that can be used on a variety of platforms. It’s user-friendly, so it’s easy to learn and use. With the included features, it allows the user to create high quality games using it’s large libraries and tools.
6. **Security**: Windows comes with built-in security that scans for malware, viruses, and security threats in real time. However, due to constant updates, it’s recommended to use 3rd party services as well. The native security can be helpful for common threats but is not an all in one solution.