

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 | 12/07/2021 | Marques Lucas | Finished table for each side and development tools |

**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 wants to develop a web-based game that can run on multiple platforms based off the game “Draw It or Lose It” which is only available for android users currently. The purpose of this game is to guess the right puzzle as a team within the time permitted. With multiple teams if your team cannot answer in the time permitted. The opposing team member gets 15 seconds to answer the puzzle.

## [Design Constraints](#_2et92p0)

Needs one or more teams involved and each team has multiple people on it. Game and Team names must be unique to allow users to check whether the name is in use or free to use. Only one instance of the game can exist at any time, and it must run on multiple platforms. These are requirements for writing code and software. Since this is the game aspect, we still need the application development. The Gaming Room would like this to run on all devices. To do this we will need to find a way to either re-write the code in swift for (Apple devices) or come up with a way to use existing code to be run on other devices by inheriting other languages.

## [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)

Entity creates a relationship between Game, Team, and Player class. This means game, team, and player class all inherit information from Entity. In the diagram it’s showing inheritance where each class will share common references like “name” and “id” making Entity a superclass. When we look at their relationship, we see Team and Player is a “has a” type. While Game has a Team and GameService has Games. When we use UML, we call it aggregation (HAS-A). When a user “has a” I mean it's an instance of one class and has a reference to an instance to another class. When we look at this diagram, we see GameService has a reference of Games, Games a reference of Tea, and Team a reference of 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** | Very Flexible terminal commands to configure server, access, or make any changes. | Cost effective, more secure than mac, Free to use, when working and run into problems less support, Open source. | User-friendly platform, more secure than Linux, you must pay to use these services, more expensive, and you also need more support. | Specs are much better on other devices, immobile server so you can track at single pace. |
| **Client Side** | No expertise is required, and time is not a big deal. Cost is like Windows. | Maximum time needed; Expertise of Linux is required. | Cost is like Mac, not much expertise is required and time, Hard to develop network apps but it can be done. | Max time needed to support different devices, Flexible to clients. |
| **Development Tools** | Common languages include HTML, CSS, JavaScript, and supporting libraries to support Python, Java, and Ruby on rails | Relevant languages include Java, Python, CSS, and HTML, Development tools include Gedit, Vim, Bluefish and many more. | Languages utilized have mostly been in C. There are leaked copies of windows research kernel on github. | Every language and tool are accessible on OS platforms. They run locally on the user’s 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**: Windows has more software options so the Gaming Room should start here. Since windows is less expensive than other operating systems people tend to use it more. Plus, it’s easy to work with other integrated development environments.
2. **Operating Systems Architectures**: Windows allows applications to show a (GUI) Graphical User Interface while accessing different operating systems. This includes graphics, multimedia, messaging, and networking. All the services can be used with a user account or server specifically.
3. **Storage Management**: Since windows 10 comes with a silent assistant that works with OneDrive called storage sense. This feature can free up disk space by making files that are not used as much to online-only files. Other features are easy save locations to find later down line. Cloud saving is still an option if you choose and the built-in storage system allows for easy file creation on large projects, so they won’t get deleted.
4. **Memory Management**: Draw it or lose it will need a database created or you can put pictures into the library. The memory allocation reserves memory space for the execution and process of programs. This will make sure your project secure and put together well. This includes working with any integrated development environment (IDE).
5. **Distributed Systems and Networks**: Some things that are the same would-be queuing repercussions, routing, and some congestion problems. Since its different networks spread throughout, it passes messages to communicate with one another. The challenges that might come up are failed components, absence of global clock, lagging computing performance.
6. **Security**: Windows has a built-in anti-virus protection that comes installed directly on your system. By using other sources, it can secure data and other information from unwanted users. This system will scan for malware, viruses, and security threats. This all happens in real-time, and always stays up to date. Making sure to keep up with all the different threats as they change daily.