

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 | 01/22/2022 | Lina Kalala | Build a software design document for Gaming Room. |

**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 client Game Room would like to have a current game developed on a web-based platform, the game is called Draw It or Lose It. The purpose of the game is to reach as many people as possible on a multiple web-based platform, the game is currently available on Android only. Game Room’s developer don’t know how to build an environment to host the game on a browser. The game should have multiple teams with each multiple players going on four rounds each minute. The game will use a library’s images to show the team until one person guesses what the world (puzzle) is. If none has the answer within 15 minutes the adversaries will have the opportunity to guess the image instead.

## [Design Constraints](#_2et92p0)

The client’s requirements are:

1. The game must involve more than one team
2. Each team must have more than one player assigned to it.
3. The game and team names must be unique (we can’t have duplicate name)
4. We can only have one game play at any given time.

The client’s requirements are a must follow to design the code the way they want. They only design constraint would be to make the application adaptable to multiple operations system platform. We will need to write the code into probably more than one language so that it can work properly on iOS for example and windows machine.

## [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 main function is on the ProgramDriver class. Entity class has attributes and method that are shared among Game, Team and Player classes; they inherited from Entity class. Inheritance is the first relationship we observe, Game, Team and Player are all inherited attributes and methods from Entity class. The Team class has an aggregation relationship between Team, player, and Game. They are related to each other with a 0 to many relationships. Team and Player classes have a 0 to many relationships which means that a team can have one or more players, but player might not have a specific team. GameService class has a reference of Game with a none to many relationships with Game. ProgramDriver class has a composition relationship with SingletonTester because if we remove SingletonTester, we don’t have a main function in ProgramDriver class.

**"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 is an easy to access and make changes web server. For advantages, Mac is a new platform and it’s constantly updating and adding new features every year to make human life easier.  Disadvantage is many people don’t prefer to user Mac for web server. | Linux is preferred for its memory and security.  Advantages are of course security, it’s more secure compared to other OS.  Disadvantage is sometimes it’s difficult to learn for beginners but also, it’s hard to find apps to support its requirements. | Most popular OS by most people around the world.  Advantages many documentations to help and easy to use.  Disadvantages easy to hack since the platform is very vulnerable to viruses. | Easy accessibility in mobile devices with today’s technology.  Advantages: Anyone can use from anywhere.  Disadvantages: Mobile devices use different OS unfortunately. |
| **Client Side** | It will take more tie to develop the software using Mac rather than windows because Mac has different requirement compared to others and it is more complex so only expert in Mac will be able to code. The cost will be a bit higher than other. We just need to make sure that the web will be compatible with other browser and mobile phones as well. | Linux is harder to learn compared to windows so expertise is absolutely required which can take more time to build. It will be cheaper to use Linux compared to Mac. Again, we need to make sure that it can work on other web browser platforms or devices such as phones and tablets. | Windows is a popular OS so expertise will be minimum, and it shouldn’t take much time. The cost will be minimized as well. | It will be easier to send an update to every device for developers. Easy to maintain.  It will be just challenging to make similar codes to inheritance some other languages to other devices. |
| **Development Tools** | Swift language.  HTML/CSS/JavaScript for frontend development.  We can use Java or Phyton as well | HTML/CSS/JavaScript.  We can also use text editor such as visual studio to code. | Easy to use.  HTML/CSS/JavaScript. We can use Java, Python, PHP, or Ruby | Infinite number od apps we can use. HTML/CSS/JavaScript. We can use Java, Python, PHP, or Ruby |

## 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**: The operating platform will be determined by the developer because each game have their own characteristics that makes it. I believe windows would be the best choice for our client because it will require minimum expertise and won’t take that much time as well. Plus, it won’t cost much to use. Windows is a popular operating platform that makes it easy because we can find many documentations in case we run into an error in our development.
2. **Operating Systems Architectures**: A good hardware computability and support would be the x86. We can use the windows-based application that enables applications to show Graphical User Interface so that the engine will be able to show a show slide of images for the user to choose.
3. **Storage Management**: For the storage, we wouldn’t need that much space because we will only need it for images. So, I would recommend the use of SSD. In windows you can manage file on the hard drive with their feature called storage sense. We can also use the cloud version or the driver to manage our files.
4. **Memory Management**: Windows usually has it own virtual space of 4 gigabytes. Since we only have images for the memory this wouldn’t be a big deal. We will need a folder to contain our library of images to show the user. We can keep the folder inside the same directory for easy access in a secure area in our machine.
5. **Distributed Systems and Networks**: We can use Develop 4 which ca be used to create a game and cross platform it after creation. We can just use the file to upload the game in any type of OS. For this game, we will need a shared database for the players to use the same server to across multiple platforms. But since this is an online game, we can use a cloud database to share data.
6. **Security**: Windows are not good at security so I would recommend other sources to secure our data from viruses or threats. I would use Avast to help us with that. For this game we can use encrypted password to keep the data protected even under attack. The security is more on the user side, we can require the player to have a long password. Or we could use a two-factor authentication.