

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 |
| --- | --- | --- | --- |
| 1.0  1.1  1.2 | 6/1/2024  6/5/2024  6/22/2024 | Eric Davison  Eric Davison  Eric Davison | Updated executive summary, design constraints and domain model.  Updated evaluation.  Updated recommendations section. |

**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 (The Gaming Room) would like to create a game called “Draw It or Lose It” for other platforms. Currently the application is created only for Android on mobile. They would like the application to be web based and available on multiple platforms. This may require an adaptation of the code to better work for other systems. They request that only one instance of the game exist in memory at a time. This can be achieved by creating unique identifiers for each instance of the game.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

Designing an application for the web may require thought on which browsers to support. Crome for instance uses a different framework than Microsoft Edge so compatibility is a concern. It may require tests or a trial period to verify the application works as intended for the most common browsers. One solution may be to focus on a single browser and recommend the user play the game on that browser if they experience any issues. The game would also need to utilize a single instance in the memory at a time, this may look different between Windows, iPhone and MacOS.

## [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 program uses Singleton methodology and includes a singleton tester that is linked to the main () program body in class ProgramDriver. This is to test to ensure that the application works as intended. ProgramDriver uses SingletonTester and both methods are public. Entity is the framework for 3 other classes Game, Team, and Player. All three classes inherit from Entity. The only class that does not inherit from Entity (other than Singleton tester and main()) is Game service which is associated with game with 0 or more instances. Game has an association with Team for 0 or more instances. Player and Team also have an association with 0 or more instances. All associations are done for 0 or more, allowing each class to either be utilized or ignored as needed for the game session.

**"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 easy to set up for server connections. The user experience is very user friendly. Does not play well with applications outside the apple ecosystem. May be the more expensive OS to maintain. The OS has rigid support but is very reliable. | Linux is basically a black sheet of paper; it is difficult to learn to use but costs little to nothing to use. This makes it very useful in terms of support as the OS can be more heavily configured. | All around has the most support of all the other OS’s. Can be run through input on command prompt or PowerShell. Has continuous support for other devices and will play nice with servers. | Android is written with Linux, so it has many of the same rules. IOS is run with a modified version of MacOS so it will run similarly. |
| **Client Side** | Mac is very expensive to purchase but less so to maintain. Integration with other systems may be difficult due to their rigid support system. Those unfamiliar with the OS may find it challenging to navigate. | Linux will be difficult for the user. Has a potential for various configurations leading to more risk with compatibility. | Windows is the easiest to integrate and many have a knowledge base in it. Like Linux it can use various hardware that the user can change at any time. This makes it difficult to predict based on the end user’s settings and hardware. | Different mobile devices have different specs. Other than IOS devices, android devices tend to use proprietary versions of the android OS. The operating system may work differently between Samsung, Google or LG for instance. |
| **Development Tools** | Can use swift (Apples own programing language), JavaScript, HTML or Python. Good tools are Visual studio, swift or pycharm. | Can use python, C++ and Java. Tools are Visual Studio, pycharm and eclipse. | Works well with most programing languages such as HTML, JavaScript, Python, C++, C, C# among others. Visual Studio is made by Microsoft and may be the best choice. | Will require the most support as it does not natively support languages as well as the others. Can be written in any language and ported over. |

## 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**: For an operating platform I recommend either using Windows or Linux. Windows would allow for the most innate compatibility with other operating systems. This is good as a general option but may be slightly less efficient in terms of storage and memory constraints. Linux, however, would allow for a more tailor-built program with less storage and memory usage. The downside to this approach is that it would need to be built and designed from the ground up, taking additional time and resources that may not be beneficial for a rather small gaming project. This is also because the application as it stands will likely not need much storage space or memory.
2. **Operating Systems Architectures**: I believe cloud-based architecture would be the best choice. This would allow for most systems to be present at the server side, letting the user access and interact with the application remotely. This also would lead to less compatibility issues as the core program would not be on the user’s side.
3. **Storage Management**: SSD is the standard for storage space and allows for maximum utilization while eliminating the potential risk for fragmentation and corruption.
4. **Memory Management**: The user will receive multiple images from the host while playing draw it or lose it. It’s important to keep in mind that the uncompressed form of images generally sent is about 10x in size (if .jpeg is the compression method). This may overload the user’s local RAM and I recommend sending packets of data as the user needs it to prevent this, perhaps sending 2 or 3 images at a time.
5. **Distributed Systems and Networks**: Relying on cloud-based computing would lead to less issues with compatibility but they would still need to be considered. For instance, if I was using Windows to play the game, then later switch to MacOS, would the system be able to handle that? I would recommend using server based save data as the use in various platforms that store data differently would be more difficult to accommodate.
6. **Security**: The application could use various security systems including user access systems. It would be beneficial to use a username and password system to access user information on the server side. Using cloud-based software would allow for the most efficient means of preventing access. Then the admin would be able to set their own username and password for deletion or other restricted access using method of least access.