

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)4

[**Evaluation**](#_2o15spng8stw)5

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/19/2022 | Nicolas Blaisdell | Initial setup – added exec summary & constraints |

**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 a gaming company, whose current game “Draw It or Lose It” is available only on Android phones. The company is looking to develop a web-based version of the “Draw It or Lose It” game, which can be served on multiple platforms besides Android.

## [Design Constraints](#_2et92p0)

One technical constraint will be the need for knowledge of web development so that there is a consistent experience across all platforms. Across operating systems, there is a consistent system to host the game, through a web browser. However, there are many web browsers that can be used, and the game experience should be as close as possible on each of those different browsers.

Another technical constraint will be the need for knowledge of web development specifically on a mobile web browser. While the game already exists as an app on the Android platform, using the built-in mobile web browser to access the game once it’s available on the web is a whole different experience, and the development of the mobile web version of the game will need specialized knowledge and experience with that platform.

One business & technical constraint are the software requirements for the game. Certain programming patterns can be employed to handle some of their software requirements, such as the need for only allowing a single game to exist in memory at a given time. Knowledge of the Singleton pattern, as well as other programming pattern will need to be employed, as well, to make sure the game is functioning as intended.

## [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 “Draw It or Lose It” game will be run from a single main() function, found in the ProgramDriver class. There is also a SingletonTester class which is used within the ProgramDriver class, but is not necessarily part of the game’s functionality, and is more for just testing. From there, the “Draw It or Lose It” game will be contained within the “Game” class, which is provided to the program by the “GameService” class, which represents a Singleton pattern. The “Game” class cannot be accessed directly and can only be provided by the GameService. By doing so, this will allow for unique games and no duplicate games being made. The Game class can consist of 0 or many Teams, represented by the “Team” class, and each Team can contain 0 or many Players, represented by the “Player” class. Each “Game” has functionality to add a team to it, and each “Team” has functionality to add a new player to it. Lastly, the Game, Team, and Player classes are all sub-classes which inherit from the “Entity” class. Since each of these classes contain an ID and a name property, this parent class “Entity” will contain this data and functionality, and then the Game, Team, and Player classes will inherit from it. Then, those sub-classes will only have to contain properties that pertain to the Game, Team or Player 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** | For a Mac, the underlying system that runs on a Mac is a UNIX or Linux-type system, so there would be similarities between Linux and Mac. However, Mac computers are usually used for user applications, especially for art, and not so much for hosting web applications. | Linux would likely be the best route for hosting the server side of a web application or game. Linux distributions are completely free, and only require the development time and skills to set up the server environment. | The 2nd best option for hosting the server side of our application would be a Windows machine, with the tradeoff being additional cost. Windows is very user-friendly with its interface, has many options for hosting web applications, and has the additional benefit of live support from Microsoft, if needed. | When it comes to the server side of the application, I don’t think it would make sense to host the code on a mobile device. With the number of players expected to access the game at the same time, a single mobile device, or even a suite of mobile devices working together, would likely not be powerful enough to handle the number of requests and the workload required for those requests for the game to run smoothly. |
| **Client Side** | Since the game will be hosted on the web, if the same browser is used, the experience across OS’s should be very similar. Since “Safari” is a popular built-in browser for Mac’s, additional support may be added for “Safari”, to make sure the experience is consistent. | Since the game will be hosted on the web, if the same browser is used, the experience across OS’s should be very similar. Popular browsers like Chrome & Firefox should be supported, at least. | Since the game will be hosted on the web, if the same browser is used, the experience across OS’s should be very similar. Additional support may be added to make sure the experience is consistent on Microsoft’s “Edge” browser, as well. | Unlike the previous 3 options, which will have a very similar look and feel within the browser, mobile devices work very differently, even when using the same browser. For this reason, additional work will be required for the mobile web version. The client side on mobile browsers will likely either be limited or an inferior experience. |
| **Development Tools** | For Mac, a program like VS Code to be able to edit text files should be sufficient to start developing. Other programs like Apache & MySQL can also be installed to create the web server, very similar to Linux. | For Linux, tools such as Apache & MySQL (which can be pre-installed on some Linux distributions), can be used to quickly and easily set up a web server, to start serving web pages and content to users. | On Windows, the best development experience can be found using Visual Studio, since it’s so highly integrated with the rest of the Windows ecosystem. | For mobile, Java and Android Studio are the best when working on mobile apps. The game is a web game, and not an app, but the development will likely be easier still using Android Studio. For iPhones, Swift can be used for development. |

## 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 System I would recommend would be the Linux operating system for developing the game, while utilizing the built-in web browsers for house the actual game, since all OSs usually contain a web browser, or can easily obtain one.
2. **Operating Systems Architectures**: Linux would be the best OS for starting development on the new game, as the cost is very low since Linux distributions are free and come with tools built in for building web applications. As for the browser, nearly all OS’s come with a built-in browser, allowing for users of all OS systems to be able to access the game.
3. **Storage Management**: For storage, the main component to consider will be the 200 HD images for the game. These images should either be stored on the server itself, along with the game, or be stored in a cloud-based storage solution, and then have those images be provided to the different clients.
4. **Memory Management**: The memory needed for the game will consist of the memory needed to hold at least one, perhaps more, of the 8 megabyte HD images, as well as the memory for any other necessary data for performing the algorithm to make the images increasingly more recognizable.
5. **Distributed Systems and Networks**: Initially, this game will be moved to a web-based solution. However, if other clients were to be considered in the future, it would be wise to design the game in such a way that the functionality of the game is completely abstracted away from the clients running it. Make the browsers implement the game in such a way that the server has no idea that it’s working with a browser, and that will make implementing future systems and clients much easier and quicker in the future and allow for the game to interact with other systems besides just users of a web browser.
6. **Security**: Lots of different users will be accessing this game, and there will be lots of resources exposed to the game, in the form of high-definition images, which should only be accessed by the game and not any of its users. For this, and many other considerations, access to the data and various elements of the game should be locked down using REST principles and user, groups and roles should be employed to define what users are able to access what parts of the game.