

The Gaming Room’s Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 7/15/2022 | Brianna McCollum | The Document has been written and updated to contain information on the first iteration of the Gaming Room. Sections Document Revision History, Executive Summary, Design Constraints, System Architecture View, and Domain Model were added. |
| 1.1 | 7/29/2022 | Brianna McCollum | The Evaluation section was added, containing information about the server-side and client-side development, as well as the development tools available for the given platforms. |
| 1.2 | 8/12/2022 | Brianna McCollum | The final section, Recommendation, was added. This section contains my final decision on what Operating System matches the Draw it or Lose it project the best, as well as explanations for my thought process. |

**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 publicize a new game called Draw It or Lose It, with the target platform being a web-based application. Multiple teams with multiple players will be able to play games guessing a prompt based of stock drawings pulled from a library. The company lacks personal development experience and has therefore outsourced to us so that we may create an environment suitable for Draw It or Lose It.

## [Design Constraints](#_2et92p0)

1. Any game will have the ability to have one or more teams present. This is a simple software constraint, as a linked list within the game class can handle this request.
2. Each team will have multiple players assigned to it. There is currently no cap value to this, but eventually this would cause storage issues if hundreds of players flood into a game to intentionally cause issues. This problem also extends to the previous constraint, which means number caps for games, teams and players should be discussed with The Gaming Room before the next iteration.
3. All names, for both game instances and teams, must be unique to allow users to check whether a name is in use when choosing a game or team name. If the names were not unique, it could cause identification issues and have the code reference sections it shouldn’t. The Iteration pattern was used in each object’s code when inserting a new game, team, or player to avoid this issue.
4. Only one instance of the game manager 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. We use the singleton pattern to achieve this, avoiding potential errors but sacrificing a bit of efficiency having unique instances for everything but the Game Manager.
5. More of a general technical restraint but doing a web browser-based game requires programming for several different types of browsers. There’s Firefox, Edge, and Google Chrome just to name a few, let alone if the game is supported on lesser used internet browsers. This can be solved by thoroughly testing different web browsers but will drain on the project’s time. Another option is to limit compatible browsers, but this cuts on our potential player base.
6. Security being a massive concern of the company is a hefty business and technical restraint. It’s difficult to enforce security measures through a web browser, becoming even more tenuous if people are using certain plugins to crack their web browser. Maintaining security is an ongoing effort, that will require time and money long after the initial project has been completed. It will also take a lot of experience from the developers to make the core program secure.

## [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 is the base class for the objects in the game, sharing some basic logic between all of them such as the id and name and their related getters and setters. The Entity class is the parent class, while the classes Game, Team and Player inherit from it. There can only be one GameService, and that GameService has a one-to-many relationship to the Game class, which means that for each GameService there can be infinitely many Games. This one-to-many relation continues for the other entity-inheriting classes, where for each game there can be many teams, and for each team there can be many players. The ProgramDriver and SingletonTester are simple, as the ProgramDriver contains our main and uses the SingletonTester class during our tests to ensure there is only one GameService.

**"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** | Able to run all major browsers for testing due to in built virtual machines. Best OS for compatibility, not just for other computer OS but for mobile devices as well. MacOS is a Unix-based OS, which gives programmers a good amount of control through the built-in terminal. | Linux doesn’t tend to play well with certain machines, which could cause compatibility issues. But the development is flexible and powerful, with most websites hosted from Linux servers. As Linux is open source, we would have complete control, with no imposed restrictions. | Most popular OS by far due to its simplicity, with web development equally as simple. This is both a pro and a con, as simpler makes developing easier but the product more basic. Auto-updates and other Microsoft actions might make development tedious, as well as the security issues like malware that may disturb Server safety. | Web development allows any mobile device to access the game. The UI will have to be customized for viewing on a phone screen, adding some extra work. Wi-Fi is questionable on the go, so the server will have to adapted to running the game with high latency. |
| **Client Side** | Mac is the most expensive option when it comes to OS, since it only comes on Apple products. Programming on Macs takes decent time investment too. Web development is like Linux due to the Unix-based OS, making swapping between easy. | Linux is incredibly difficult for novice programmers to get into, due to the large amount of difference between Linux and other OS. However, due to being completely open source, using Linux is free and would require no licensing. | Windows is the cheapest option, as there’s thousands of PC options that can host the OS, but that doesn’t mean the OS lacks power. It’s also the simplest to learn, allowing both novices and experts to help. | Programming for Mobile devices takes a unique set of skills, requiring developers that understand the limitations of mobile development. Time wise, mobile web pages are flexible and quick to program. |
| **Development Tools** | Mac does not have great tools for web development. The OS lacks a lot of functions at base and needs outside tools to help. However, once those tools are acquired, the compatibility features help for checking results. | Linux is considered the “freest” system for development. You can install any tool, making it flexible but possibly very difficult. For example, you will have to use virtual machines a lot, for things as simple as Photoshop. | Windows has a large variety of IDEs and development tools to help new developers as well as veteran ones. There’s a lot of public resources due to Window’s popularity, most of which are free to use. | Developing for mobile requires using mobile specific coding environments. This can be a hassle no matter what OS you’re programming on. Language wise, several languages work for mobile, making it incredibly varied and flexible. |

## 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 believe the best Operating Platform for the Gaming Room would be Linux. It simply has the most power and flexibility, while also being completely free and open source requiring no payments for usage. However, Windows is a very strong secondary with it’s own points to support it, such as having more public resources and overall being easier to develop for due to the lower learning curve. While Linux is my final recommendation, if the development team is struggling to adjust to it or the pros aren’t worth the extra work, then Windows would be the next best choice.
2. **Operating Systems Architectures**: Linux is built upon giving full control to the user with open-source code down to the very core of the OS. Its architecture is structured with a vide variety of kernels and has the usual hierarchical file system. Windows on the other hand, aims for simplicity and reflects that in the structure of its architecture. Windows runs with a hybrid kernel, keeping system services contained and away from the user but allowing the drivers to open to the system. Linux gives full control of the OS to the user, while Windows takes some away for simplicity’s sake. I believe the full control to be greatly beneficial for the server’s system, as we can cut pretty much everything that isn’t necessary for the server to save time and space.
3. **Storage Management**: Linux is completely controlled by the user and portable to any sort of device, therefore having incredible storage management. The OS struggles a bit with hardware when compared to Windows, but it still has plenty of options, allowing Linux to run on most devices while compatible with any sort of external storage device The Gaming Room may use. Linux has access to many tools to help manage hardware devices, while will help with controlling and monitoring the external storage devices. Windows may be able to run on basically any hardware, but it lacks the synergy with external storage devices in terms of free control and possible compatibility issues.
4. **Memory Management**: The Server will need to handle the requests of a high density of users to the Client-Server pattern it has been built on. There are 200 high-definition image files that need to be sent to users quickly and efficiently, which is a large task for any system’s memory management, let alone all the account information and UI elements that will also need to be processed and sent out. Again, the control of Linux is a huge help here, as there are many tools and commands that will be useful in keeping an eye on and managing the memory usage. Linux also has a system called Virtual Memory, where memory is “reserved” to be used but doesn’t necessarily have to be used or even made into physical memory. This allows Linux to process a ton of information at once, even more than the physical memory in their system. This makes Linux the clear superior for memory management.
5. **Distributed Systems and Networks**: Distributed systems connect several different computers into one network to be able to share information. Connecting several different types of devices is difficult to do, but since their relationship with the server is only one way (Client-Server relationship), requesting data and receiving it back, that streamlines a lot of the interaction. If the data sent can be processed by the different OS, everything should be able to play nicely. If there is a needed dependency that is required in the server, that may be a problem for Linux, as the available option for typical programs is shorter than Windows. Linux can’t use Microsoft programs like Photoshop or Excel and needs alternatives for a lot of typical programs. Therefore, if a certain external program is needed for distribution, it may be beneficial to use Windows to ensure compatibility. But at the same time, Linux is significantly more stable, allowing it to maintain connections even in emergencies. It also is simply quicker to processing than Windows, while also having more configurable settings for connection and networking. It can process data faster and send it out to any system in a very configurable way.
6. **Security**: Linux is significantly more secure than Windows for several reasons. The main one being the full control you have over Linux, which allows you to monitor all the device’s data and put in any security features you want. Meanwhile, Windows comes with prebuilt security measures, and they’re decent at best. This is mostly because Windows is significantly more targeted by malware than Linux is, due to its overwhelming popularity as an OS for casual users. Therefore, malware is designed to infect Windows OS machines and not Linux ones. This is also a rare point in Mac’s corner. While I haven’t mentioned them yet, their security is incredible, baked into the core of the OS and possibly even stronger than Linux.