

Draw It or Lose It

# **CS 230 Project Software Design**

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)4

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/9/21 | Makayla Anderson-Tucker | I have added information for the following sections: executive summary, design constraints, and domain model. |
| 1.2 | 09/12/21 | Makayla Anderson-Tucker | I have completed the evaluation section of the design document. |
| 1.3 | 09/15/21 | Makayla Anderson-Tucker | I have completed the recommendations section of the design document. |

## [Executive Summary](#_sbfa50wo7nsh)

Our client, The Gaming Room, is looking to develop a web-based game that is cross platform to gain a larger user base. Right now, they only have an android application available. To solve this, our company will create software design document for the client. This document allows the client to better understand the development of the web-based game and will play a major part in streamlining development.

## [Design Constraints](#_2et92p0)

There are quite a few design constraints. The first requirement is that there can only be one instance of the game existing in memory at any given time. Unique identifiers for the game, team, and players must be implemented to ensure the first requirement works. To achieve this, we will use the singleton pattern. The second requirement is that that game/team names must be unique and allow users to check if the name is in use or not. This is important because there can only be one instance of a game in memory at a time and not checking the names would completely go against it. This ensures that no function is being unnecessarily overloaded as this can cause multiple problems with the program running correctly. The third requirement is that the web-based game needs to be able to support one or more teams involved in playing the same game. The final requirement is that each team is able to handle multiple players being assigned to it. This is important because if the team function is developed to only hold a limited number of spaces, users are more likely to not play the game. Allowing multiple team members ensures that users can play with all their friends.

## [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 Game, Team, and Player class all inherit from the Entity class, which is denoted by the open arrow. The GameService class is connected to the Game class with a straight line that indicates an association between the two classes. The cardinality is 0 to many instances. The GameService class will implement the singleton pattern to ensure there is only one instance of the game at one time. The Game class is connected to the Team class with straight line that indicates an association between the two classes. The cardinality is a 0 to many instances. The Team class is connected to the Player class with a straight line that indicates an association between the two classes. The cardinality is 0 to many instances. These three classes (Game, Player, and Team) satisfy the third and fourth software requirements. Both requirements state that the classes need to be able to handle multiple teams and players within the same game. The cardinality is displayed on the UML diagram to demonstrate those requirements. ProgramDriver class is connected to the SingletonTester class with a closed arrow. This means that the SingletonTester class will be deleted if the ProgramDriver class is deleted.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Mac operating system is the most secure of all four options. This is because they have a smaller user base, so there is less attack surface. One disadvantage is that in order to use the Mac OS you can only use devices purchased from Apple. This limits the different kinds of devices you can use. | The Linux operating system provides very good security. The OS even comes with protections against buffer overflow attacks. It is not as secure as Mac, but significantly better than windows. Again, this is attributed to their small user base. | The Windows operating system provides built in security features. However, it is the least secure option of the 4. This is because they have a great number of users, which significantly increases their attack surface. This makes it difficult for the OS to address all security concerns. | Users with any mobile device will be able to use the application as there is no concern about compatibility with most modern devices. Because the game application is web based, the client is not restricted to the rules and fees of app stores. It is important to keep in mind that because it works across a multitude of device OS, the attack surface for crackers is increased. |
| **Client Side** | The Mac OS is a moderately affordable web hosting option. The OS only operates on Apple devices, the client has limited options. This means the client cannot use a non-Apple device with a more affordable price. This is a disadvantage as Apple devices are some of the most expensive devices on the market. | The Linux OS is the cheapest web hosting option of the four. Linux is an open-source software, which means that the software is free and the only cost is for hosting. Linux is also known for being a very easy and convenient operating system to work with so developers will not need special knowledge of how to use it. | The Windows OS is the most expensive web hosting option. Because of licensing fees, Windows server-based deployment tends to be more expensive than Linux or MacOS. A disadvantage is that the client will have to spend more time on development to ensure that there are minimal vulnerabilities as Windows does not cover them all. | Web hosting for mobile devices can be almost as expensive as it is for Window OS hosting. It would be a good idea to look into cloud-based storage for more affordable fees. |
| **Development Tools** | The gaming application will be written with Java and Mac OS is not great for Java development. If the client decides to go this route, they will have to invest in a VM to host Linux for development. Apple offers GUI and terminal command line for users. The client will need to use HTML/CSS, JavaScript in addition to Java for the game’s graphics. | The gaming application will be written with Java and Linux is the best of the options for web server hosting. This is because of how easy it is to use and the number of languages they support. Unfortunately, it only offers the terminal/command line because there is no GUI. The client will need to use HTML/CSS, JavaScript in addition to Java for the game’s graphics. | The gaming application will be written in Java and Windows OS is somewhat suitable for the development. It works with only a few frameworks, like .NET, and languages.  Windows offers GUI and terminal command line for users.  The client will need to use HTML/CSS, JavaScript in addition to Java for the game’s graphics. | The mobile web-based application cannot use the exact same code as the desktop site. This is because the aforementioned languages perform differently based on the device. Also functions like API calls would have to be changed because mobile browsers do not support Adobe Flash. |

## Recommendations

1. **Operating Platform**: I believe that using the Linux operating system to develop Draw It or Lose It is the best option for the client.
2. **Operating Systems Architectures**: Linux operating system provides some of the best security safeguards among operating systems. This is because Linux has a small user base and as a result it reduces the attack space. It also comes with built-in protections against leaking so that the client won’t unintentionally give crackers back doors. There is no cost to license Linux because it is an open-source system, so the company would only need to pay for the web hosting. Another reason I chose Linux is that the operating system works very well with game design and the client is developing a gaming application.
3. **Storage Management**: If budget is not an issue, I would recommend using on-the-premise servers because it will allow them to personalize their storage and fix issues on demand. If the servers ever go down or there is some issue with the connection, the company will not have to wait for a third party to fix it. This method of storage management can be achieved by using the client/server pattern. However, I recognize that there may be budget restrictions. In that case, I believe that Amazon S3 is the best option for managing the storage for this application. The client has mentioned wanting to increase their user base and Amazon S3 scales up on demand. This will prevent the server from becoming slow if there is ever a significant amount of traffic. The pricing is not the cheapest, but it is not overly expensive and only based on the amount of GB that the company uses. Using the cloud provider will also allow the company to have servers in different locations without having to worry about paying storage fees. This means the users should have a relatively positive experience with the game’s load time. Finally, I would recommend including redundant servers when signing up for the cloud plan. This is so if one server is lost, the information is already backed up and ready to go on another server.
4. **Memory Management**: The Linux operating system uses a variety of memory management practices. Some of these include demand paging, allocation for kernel structures and user programs, and file mapping.
5. **Distributed Systems and Networks**: A distributed system has multiple functions on different servers to facilitate client requests while appearing as a single unit. The client-server system is the most appropriate structure of distributed systems for the game Draw It or Lose It. The structure allows for multiple client requests at the same time and works over a computer network. If there is an outage, one of the redundant servers can replace it and the flow of operations is not interrupted. It is also fairly easy to scale up servers using this model, which means the company does not have to worry about being able to handle a large number of users at once.
6. **Security**: Assuming the client goes with the Amazon S3 cloud service, there are not a lot of security features that need to be implemented. The Linux operating system has built-in security features which makes the system mostly secure. The cloud provider also offers security, and they take care of it for the client. However, it will be important to implement password hashing/salting to obscure user data if crackers ever did obtain user data. The biggest security risks are the employees. The client must provide their employees with an education on recognizing phishing emails and disposing of sensitive data properly. For example, if an employee writes their login credentials on a sticky note and throw it away, anyone intending the company harm can go through the trash and obtain the credentials. It does not matter how secure a software is if the people handling it are not also taking security measures to keep data private.