

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 |
| --- | --- | --- | --- |
| 1.2 | 04/15/2023 | Harshilkumar Jayswal | Updated Recommendations Section |
| 1.1 | 03/29/2023 | Harshilkumar Jayswal | Updated Evaluation section |
| 1.0 | 03/17/2023 | Harshilkumar Jayswal | Initial Version release |

## [Executive Summary](#_sbfa50wo7nsh)

“Draw It or Lose It” is a popular android game by the Game Room based on the 1980s television game, *Win, Lose or Draw.* The team behind the app would like to expand the success of the game by developing a web-based game of the same app. This web based version will serve multiple platforms such as Windows, Mac, Linux and Mobile Devices. The game will have the ability to pull in one or more teams with unique players assigned to the respective teams. Game and team names must be unique to allow users to check if the name is already in use. Only one instance of the game can exist in memory at any time. Staff at the game room is looking to set up the development environments and are reaching out to CTS to facilitate the streamlined development.

## Requirements

* Servers to host the games and user data.
* Networking infrastructure to provide ample bandwidth to the players with minimal latency.
* Development machines so that programmers can develop the game.
* Software for design, coding, version control, monitoring and security.
* Human resources management to hire new staff and assign roles, as well as manage budget and accounting of the project.
* Skilled professionals to develop the game for multiple platforms.
* Marketing/Sales team to promote the game once its released.

## [Design Constraints](#_2et92p0)

* Current game version only exists as Android app, so team would need to migrate the game and its features to iOS and web for various PC operating systems.
* Game must allow one or more team to play in the same session.
* Game must have verification check for unique player IDs and team IDs during sign up.
* Players should be able to run the game regardless of the platform, game should be cross platform compatible
* Developing for various operating platforms require different programming experts.
* There should not be any difference in UI or functionality of the game so player can enjoy same experience regardless of the platform they are using.
* Enough servers to host all player data and enough bandwidth to allow players to play the game comfortably without any noticeable latency.
* Consider the different types of web browsers running on different frameworks.
* Restricting only one game session execution in the memory of the player

## [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 ProgramDriver class has the main method for game execution. ProgramDriver class uses SingletonTester class to insure only one instance of GameService exists within memory. The GameService class contains the list of games, the game’s identifiers such as gameID, PlayerID and TeamID. The class has the ability to create new games. It contains an iterative pattern to verify if a unique instance of the game already exists. GameService class exhibits a logical association of zero-to-many relationships with Game Class.

Entity class is a parent class to Game, Team and Player class exhibiting inheritance. The Game class contains the list of teams for the unique game and can create a new team if the team id does not exist yet. The class inhibits a zero-to-many relationships with Team class. The team class extends from Entity class and contains a list of players for that team and has functionality to add players. The team class exhibits zero-to-many relationships with Player class. The player class extendeds Entity class and contains the player’s unique identifier and name. Thus showcasing that one Game can have many teams, and one team can have many players.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | **Characteristics:**  - UNIX Based OS, Known for user friendly UI  **Advantages:**  - Seamless integration with large userbase Apple client devices such as iPhone and iPad.  - Privacy and Security focused environment out of the box  - Less targeted malicious attacks due to market share  **Weakness:**  - High hardware cost  - Limited upgradability since many components are not replaceable  - Limited virtualization features | **Characteristics:**  - Free and Open source  - Plethora of modified versions available with focus on hosting  **Advantages:**  - Free, secure, and highly scalable  - Widely used in web development that guarantees more compatibility with other systems  - Easy CLI deployments  **Weakness:**  - Can be more complex to setup and maintain  - Experts in field needed that can solve complex issues | **Characteristics:**  - Most used OS in the world  - Well integrated with other MS based business solutions  **Advantages:**  - Offers wide support with global solution leaders such as Azure, AWS and Google Cloud  - Developers are familiar with the environment  **Weakness:**  - Could be more vulnerable to security threats due to popularity  - Expensive licensing from Microsoft | **Characteristics:**  - Portable devices with rechargeable batteries  **Advantages:**  - Can be hosting from anywhere that has internet  - Good for very small-scale hosting great for experimenting  - Low cost  **Weakness:**  - Not suitable for web hosting for large scale due to very low performance  - Not many web hosting tools available  - Connections could be intermittent due to network issues  - Small screen and limited input options that can hinder development |
| **Client Side** | - Could be costly to hire developers with Mac expertise  - Could be costly to obtain hardware for testing  - Ensure compatibility with various browsers such as Safari, Chrome, and Firefox  - Could require special third-party plug-ins | - Not a wide user base but it would take more time to ensure compatibilities with many variants of Linux OSes  - Requires high level of expertise to develop  - Ensure compatibility with various browsers such as Firefox and Chrome | - Large userbase with this client  - Basic web-based framework might come pre-installed  - Supports wide range of browsers such as Edge, Firefox, Chrome, Opera that we need to ensure compatibility | - Almost everyone uses this client  - Needs complete design and gameplay overhaul to support touch-based input  - Need optimization to support various low end performance hardware, screen sizes and networking issues  - Players might use this client more than others, so any post-release issue would need immediate resolution |
| **Development Tools** | **Programming Languages:**  - HTML, CSS, JavaScript, Java, SQL  **IDE Tools:**  - Xcode, Visual Studio for Mac  **Technical Requirements:**  - Need expertise on Mac Development and IDEs  **Cost:**  - Costly Licensing costs | **Programming Languages:**  - HTML, CSS, JavaScript, Java, SQL  **IDE Tools:**  - Visual Studio for Linux, Eclipse  **Technical Requirements:**  - Need expertise on Linux Development and IDEs  **Cost:**  - Relatively low licensing cost if used open-source projects | **Programming Languages:**  - HTML, CSS, JavaScript, Java, SQL  **IDE Tools:**  - Visual Studios for Windows, Eclipse, Notepad++  **Technical Requirements:**  - Need expertise on Windows Development and IDEs  **Cost:**  - Costly Licensing costs | **Programming Languages:**  - Swift, Java, HTML, CSS, JavaScript, SQL  **IDE Tools:**  - Xcode, Android Studio, Eclipse  **Technical Requirements:**  - Need expertise on Mobile Development and IDEs  **Cost:**  - Costly Licensing costs |

## 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 recommend the Linux Operating System for Draw It or Lose It. Because Linux is an open-source operating system, anybody may use, change, and distribute its source code without restriction. As a result, there are more customization and system integration options available. A server that must be accessible round-the-clock needs to be able to run for extended periods of time without needing to be rebooted. Because of its open-source nature and lack of high licensing costs, Linux is a cost-effective alternative for server platforms. I advise using server-specific Linux distributions like Fedora, CentOS, Ubuntu Server, or Debian.
2. **Operating Systems Architectures**: Its kernel is open source. It has a command-line shell that can be utilized to automate server chores. All files and directories are arranged in a tree-like layout in Linux's hierarchical file system. All other directories and files are arranged beneath the root directory (/), which is at the top of the tree.
3. **Storage Management**: I advise using Google Cloud or Amazon for cloud storage. On-premises storage infrastructure is not necessary with cloud storage, saving money on upkeep and upgrades. You only pay for the storage space that you really use with cloud storage, which can save your company money. It is very reliable and available, which is essential for servers to run efficiently. To safeguard your data against theft, unauthorized access, and other security risks, these providers have strong security procedures in place.
4. **Memory Management**: Memory management is a critical aspect of the operating system that ensures the efficient use of available memory resources. Linux employs virtual memory to make it appear as though there is more memory than there is. The system divides memory into pages, and only the pages that are needed are loaded into physical memory. The system can switch out physical memory pages to hard drive pages when the gaming software demands extra memory. The kernel controls how much memory is given to each process, and it does so according to the needs of each process. Linux protects the system's memory by using memory protection techniques to stop programs from accessing one another's memory.
5. **Distributed Systems and Networks**: Linux can be used to host the critical data and computing for the program. We can leverage SQL databases to store user data and session information. Cloud services can be utilized to host the images for the application. And the various browsers on client devices that can consume the content.
6. **Security**: There are many security options available with Linux, such as encryption, secure protocols, access controls, and frequent security updates. Linux includes dm-crypt and LUKS as built-in encryption mechanisms. The data at rest can be secured using these programs, which can encrypt the entire hard disk or a selected partition. Additionally, it supports a number of authentication methods, such as biometric authentication, password-based authentication, and key-based authentication. The package managers included with Linux distributions make it simple to repair and update applications and packages.