

Draw It or Lose It WEB

# **CS 230 Project 3 Software Design**

Version 2.0

## 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 | 01/27/2024 | Jacquelin Kitcher | Executive Summary & Design Constraints |
| 2.0 | 2/9/2024 | Jacquelin Kitcher | Requirements, Domain Model, Evaluation |
| 3.0 | 2/20/24 | Jacquelin Kitcher | Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has requested a web-based version of their existing game, Draw It or Lose It, which is currently only compatible as an app on Android devices. They have tasked CTS to develop an environment that is web-based and works across multiple platforms. A thorough examination of the operating platform elements, client considerations, and development tool options has allowed us to make our recommendation: Linux will be the best operating platform to translate the exiting app to a web-based program that can be used across many platforms and devices. Creating this environment will allow The Gaming Room to reach users who play games on more devices, increasing their games’ reach to larger audiences.

## Requirements

The following requirements have been requested:

* The game should allow for teams to play against each other.
* The teams should allow for multiple players to be added to it.
* The user should be able to create a unique name for their game so other teams can join the specific game and their team, which can be achieved by allowing the user to check if their name choice is already taken and allow them to try different names until the game is unable to find an existing matching name.
* By creating unique identifiers, the game shall only allow one instance of the game to be stored in memory at any time, including game, team, and player.

## [Design Constraints](#_2et92p0)

CTS will need to develop this game environment adhering to strict timeline and budget guidelines proposed by The Gaming Room. Therefore, we will prioritize the requirements set forth by our client to commit to design and development excellence, before focusing on additional items favored by the development team or extra features requested by the client, which will require a budget and timeline negotiation. Additionally, must consider the technical requirements we are asking of the development team who will be working to streamline the existing game to work across other platforms, so we will need to utilize members of our team who are comfortable with many platforms, including Android, first and foremost to help with translating the game environment.

## [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 following UML class diagram is a visual representation of what CTS will be creating the software from, which outlines the requirements of our client. In addition to a Main and the singleton tester, there will be a general class called Entity for which unique Games and User IDs can be created. Inheriting from the Entity class will be 3 subclasses which contain the same user ID and name fields as the Entity class but have their own individual attributes aside from Entity and each other. The Game class has a one-to-many relationship with both the GameService class (which satisfies the single instance of a game at one time) and the team class. The Team class has a one-to-many relationship with both the Player and Game classes. This ensures that each Game, Team, and Player are unique, creating one instance.

**"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** | MacOS is popular with users, but less used by developers. | Ideal for programmers, supports multiple programming languages, is quick and easy to use for the developer. Low web hosting cost. | Well known and widely used. It also works with Linux. Licensing fees, uses more resources. | Creating a web-based app for use on a mobile device will allow anyone to be able to use it, however functionality may not translate as well on a mobile device, causing increased allocation |
| **Client Side** | Mac is high quality for high cost(hardware), simple for users, but there are licensing limitations. However, because XCode is the main IDE, its free which saves on software licensing fees. | Linux can handle a high volume of users and produce quick-running software simultaneously, which is a feature users favor. | Windows is well known by most people who use technology, so tends to be more widely used, however there are costs associated with Windows programs and security can be an issue. | Can reach anyone who has a mobile device (aka everyone), is not as user friendly as a mobile app. |
| **Development Tools** | Apple uses XCode as their main IDE. Supports Python, Objective-C and Swift languages. | Supports C, C++, Python, Java languages. Tools allowed include Visual Studio Code, Eclipse, and PyCharm. | Supports mangy languages including Java, C/+, Python and several IDEs, including VSC and Eclipse. | You can use any of the development tools, but with the limitations listed above, best to use app development tools vs. web based. |

## Recommendations

**Operating Platform**: There are benefits to each platform, depending on the goals of the project. Linux seems to be the most well-rounded when compared to the others we analyzed for this project during the evaluation portion. Linux It is affordable to produce, our programmers are comfortable using it, and it would be the quickest way to complete the project. Additionally, there are benefits to the user of a program running on a Linux OS, as it can handle multiple users and maintain its reliability and speed.

**Operating Systems Architectures**: Linux uses a kernel design which is ideal for multitasking when creating web-based applications, and there is a large support network for developers, as well as many developing tools and libraries. Linux support multiprocessor systems, that use symmetric multiprocessing(SMP), where all processors function equally, or asymmetric processing which works using a master/slave relationship. A client-server architecture allows users to send inputs to the game, which in turn produces outputs.

**Storage Management**: We recommend using a cloud-based storage system for use with Linux. Unlike Apple who has their cloud, or Google has their filing system, you can use any cloud-based storage program with Linux, such as their standard extended file system. You can integrate a cloud based storage system SDK when creating a web-based program using Linux that is compatible with multiple programming languages. This gives flexibility to choose what works best for our program to store and retrieve the game files is a simple and cost-effective way.

**Memory Management**: The kernel structure (KSM) of Linux optimizes system resources, using virtual addresses to store information, demand paging to only access and utilize memory for the portions of the program that are actively being used, and swaps memory as needed. It also catches and prevents duplicate usage of memory. Multiprocessing allows the addition of CPUs, and the more CPUs, the more memory is available for use. Linux can cache and buffer which optimizes input/output operations, so storage and retrieval of the game images can be done seamlessly.

**Distributed Systems and Networks**: Linux distributes to multiple CPUs,

which communicate with each other allowing for use across many devices and platforms, which is our main concern with expanding this program. A RESTful API is ideal for web-based applications, because it is easily scalable, easy to implement and can be customized to fit the needs of the program. Linux also has systems that utilize redundancy and replication which can handle issues with connectivity or network outages that could impact the user experience when playing the game. Load balancing can be used to handle multiple players at once.

**Security**: Linux uses file encryption, has a built-in firewall, and administrators can write security levels to protect information that is private, while allowing information that is needed for the user to run the program. When configured properly, Linux can be more secure than other operating systems. This is an advantage of using an open-source OS, as it has built in security in its architecture, and is regularly audited by the Linux community. Also, Linux offers features that use the least privilege principle, which ensures that only those who need access, have it. Much like in the healthcare field, which uses the minimum necessary rule, to only allow those who need access to important health information.