

# Draw It or Lose It

# **CS 230 Project Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design**](#_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)5

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/17/2022 | Michael Steenkamp | Added the Executive Summary, Design Constraints, System Architecture View, Domain Model, Evaluation, and Recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game. The game “Draw It or Lose It” should be able to serve multiple platforms, and it is currently only available in the Android app store. As a frame of reference, the game is loosely similar to the 1980s television game [Win, Lose or Draw](https://en.wikipedia.org/wiki/Win,_Lose_or_Draw). There are four software requirements, described in “Design Constraints” on Page 2.

<https://en.wikipedia.org/wiki/Win,_Lose_or_Draw>

## [Design Constraints](#_2et92p0)

* A game needs to have the ability to have one or more teams involved, because the game cannot be played with a single team.
* Each team needs to be able to have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game 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.
* The game should be a web-based application that can be accessed by multiple platforms, such as Mac, Linux, Windows and Mobile Devices.

## [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 games body is contained within a single file named “com.gamingroom”. The classes that exist within the body are “ProgramDriver”, “SingletonTester”, “GameService”, Game”, Team”, “Player”, and “Entity”. The main function, which runs the application, is contained within the “ProgramDriver” class and this class uses the “SingletonTester” class by sending synchronous messages to it to prove there is only one instance. The classes “Game”, “Team” and “Player” inherit from “Entity” and the three leaf classes are related to each other, as well as with the “GameService” class, with a zero to many relationship. The inheritance fulfills the software requirements efficiently by only having one declaration of an “id” and “name”, and their respected accessors and mutator. This minimizes duplication of code and increases the implementation of object-oriented design in the application.

**"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** | Mac supports small client specific, such as resource forks. It is easy to create features for Win clients. A disadvantage is that the hardware is limited thus limiting its capabilities. | Linux systems are stable and reliable, they were designed to be powerful. A disadvantage is that it might be a difficult learning curve. | Windows systems support hypertext mark-up language files and are not limited to their hardware. A disadvantage is that it has some stability issues with server hosting. | Server-side development is only recommended on a mobile device if the team is large enough and have experience. Mobile devices are also limited by their hardware. |
| **Client Side** | The user is required to by a mac system and is generally more costly than a Windows system. | Linux has a minimum cost and it’s only financial / time drawback is the time that it takes to learn how to use it. (Linux is open source) | Windows can be installed onto a mac device (to use VS, VSC etc. for non-apple features) and can then use Mac OP when wanting to work on apple specific features.  Cost is higher than Linux | Mobile devices are generally less expensive; however, they have a small screen and so the UI have been designed to fit / perform better on smaller screens. |
| **Development Tools** | Generally objective-c. An IDE that is commonly used is Visual Studio Code and Xcode. | Linux based OS support all common programming languages, such as C, Python, Java, etc. IDE’s that are used are Visual Studio Code, Eclipse, and Geany. | Windows also supports a wide range of programming languages, but the main language is C++. A common IDE that is used is Visual Studio. | For iOS Swift, Objective-C and Xcode are mostly used, and for Android App Development Java is mostly used. |

## Recommendations

1. **Operating Platform**: I would recommend that The Gaming Room uses a Linux based operating system to expand Draw It or Lose It to other computing environments. Linux is a very secure, reliable, and well-established operating system for server-side development. It is also financially beneficial because it is open source.
2. **Operating Systems Architectures**: Linux has a layered architecture; it consists of four layers. The inner most layer is the hardware, outside the hardware is the kernel layer, outside the kernel layer is the shell, and outside the shell layer is the applications. Hardware is all of the physical devices connected to the system, such as RAM, CPU, etc. The kernel is the core component of the operating system that interacts with the hardware. The shell is the interface that receives input and delivers an output. The applications are utility programs that run on the shell.  
     
   Linux architecture. TecAdmin. (2020, October 29). Retrieved March 17, 2022, from <https://tecadmin.net/tutorial/linux/linux-architecture/>
3. **Storage Management**: When working with a Linux based system, there are two main ways to partition information onto a drive (storage management), Master Boot Record and GUID Partition Table. The storage management system that we should be using is the GUID Partition Table (GPT), this is because it is becoming more popular, it supports distributed data which allows for it to store multiple copies so that it is more robust and can recover data if it becomes corrupted. We can also use an xfs file system to arrange files onto a drive.  
     
   Victor. (2021, January 15). Storage management in linux explained with examples. TekNeed. Retrieved March 20, 2022, from <https://tekneed.com/storage-management-in-linux-explained-with-examples/>
4. **Memory Management**: Linux uses a Memory Management Unit (MMU) to handle virtual memory, which is useful in ensuring that the application cannot be corrupted by rogue programs. The memory can be moved so that we can ensure the application is stable and running. Memory can also be swapped with a disk so that we can ensure that we have enough memory to sustain the program and its memory requirements.  
     
   TheLinuxFoundation. (2017, April 4). Introduction to memory management in linux. YouTube. Retrieved March 19, 2022, from <https://www.youtube.com/watch?v=7aONIVSXiJ8>
5. **Distributed Systems and Networks**: Draw It or Lose It is a web-based game, which means that it can be accessed through the internet. If a user wants to access the game, then they would need access to a device, such as a computer or smartphone, and the device needs to be able to connect to the internet. As a server is running and hosting the application on the web, it is essential that the server is connected to the internet and has power. If the device(s) that run the server loose connection to the internet or do not have power, then the application will be offline. A device is primarily dependent on power in order for it to operate; however, internet connection is essential for the device to be able to be connected to the web. A distributed system can help the game communicate between various platforms because it limits each platforms dependency. This allows an application to be accessed by devices that are the same network, no matter where they are located.
6. **Security**: Linux is a very secure operating system; it has built in kernel security defenses. Linux has an “array of built-in security defenses” which include firewalls that uses filters in kernel and the “UEFI Security Boot Firmware verification mechanism”. Since Linux is open source, its security is constantly being improved by members of the community. Linux makes use of various security features such as, encapsulation, encryption (symmetric, asymmetric and authentication), Service Access Points, and virus protection methods (firewalls).

Day, B. (n.d.). How secure is linux? Linux Security. Retrieved March 19, 2022, from <https://linuxsecurity.com/features/how-secure-is-linux>

Work Cited:

. Inc., A. (n.d.). Xcode - ide. Apple Developer. Retrieved April 2, 2022, from <https://developer.apple.com/xcode/ide/>

BrainStation. (2022, January 5). What tools do IOS developers use? (2022 guide). BrainStation®. Retrieved April 2, 2022, from <https://brainstation.io/career-guides/what-tools-do-ios-developers-use>

-, J. W., By, -, &amp; Wallen, J. (2018, August 3). 5 essential tools for linux development. Linux.com. Retrieved April 2, 2022, from <https://www.linux.com/training-tutorials/5-essential-tools-linux-development/>

Name Name\*, & Raffi Says: (n.d.). Mac vs windows: Advice for web developers. Retrieved April 2, 2022, from <https://createdbycocoon.com/post/mac-or-windows-for-web-development>