

Draw It or Lose It Application

# **Charles Campbell CS 230 Project Software Design Document**

Version 3.0 - Final

## Table of Contents

[**Draw It or Lose It Application** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326) **- 6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/21/2024 | Charles Campbell | First version of Document; added Executive Summary, Requirements, Constraints & UML Explanation |
| 2.0 | 10/02/2024 | Charles Campbell | Created Evaluation of client, server & development tools for Mac, Linux, Windows & Mobile Devices |
| 3.0 | 10/16/2024 | Charles Campbell | Completed document by filling out the Recommendations for The Gaming Room |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has asked Creative Technology Solutions to assist in developing their *Draw It Or Lose It* game that is currently on Android. They need assistance in software & streamlining requirements to move their game from Android to other platforms like the web.

## Requirements

Taking a currently developed application mainly based for Android smartphones, and adding/editing the software to ensure similar function and capabilities to other platforms. Game must have the ability to have one or more teams involved, with multiple players in each team, game & team names must be unique (no duplicates), unique identifiers for each game/team/player so that only one instance exists in memory at a time.

## [Design Constraints](#_2et92p0)

1. Transition from an Android (smartphone)-based programming language to allow different platform users to play the game will be a major constraint. The software design could vary, making it challenging adaption to go from one platform to others like web or iOS devices.
2. Understanding and implementing different licensing restrictions/requirements for other platforms The Gaming Room wants to expand to will be a necessary constraint to review.
3. Constraints of The Gaming Room’s schedule and budget must be considered to ensure that our work & efforts to help them with their project can be met. Doing so will allow The Gaming Room to meet their deadlines & expectations of their customers, which will help Creative Technology Solutions continue to be a trustworthy and reputable business partner.

## [Domain Model](#_8h2ehzxfam4o)

The gaming room system is a package that contains the classes Program Driver (main) and it uses the Singleton Tester. The Singleton Tester checks to ensure that only 1 instance of a team, game, or player exists, and no duplicates are in memory. The superclass Entity defines essential information such as an ID or name, and the get functions (like getID() or getName()). The subclasses Game, Team, & Player are subclasses to Entity, and use inheritance for Entity’s attributes. The Game Service class is used to make a 0 to many sets of games, teams within the games, and players within the teams.

**"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)

Evaluation of the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices. Consideration of the requirements outlined by The Gaming Room to ensure their project works across all desired platforms.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | - Pro: High security from Apple.  - Pro: Automatically provides web hosting  - Con: Compatibility issues if not developed in Swift  - Con: Less web & other platform compatibility than Windows/Linux | - Pro: High flexibility & customization  - Pro: Considered best in security, due to open sourced OS  - Con: No official server support, needs technical expertise  - Con: Switching Linux versions can be elaborate and difficult | - Pro: Easy to use user interface for file explorer & settings  - Pro: Highly scalable to software’s needs & allows 3rd party tools  - Con: Any device with internet can access, high security risk  - Con: Complicated user-based licensing | - Pro: Optimized to handle multiple users  - Pro: Cloud-based allows for easy or no hassle upgrading to new server versions  - Con: Strict specifications such as bandwidth usage  - Con: Spikes in traffic can cause real-time performance issues |
| **Client Side** | - Cost: Apple’s App Store has $99 annual fee to host app software to clients  - macOS updates frequently, lots of time to ensure compatibility for user  - Must meet Apple’s security, kernel & hardware standards and agreements for clients to use an app | - Cost: Cheap monthly cost from $1.00 – 6.99  - Stay up-to-date with Linux security patches and know weaknesses  - Understand Linux Standard Base for compatible use for all client distributions  - Open source OS might need specialized expertise for different desktop environments | - Variety of application hosting services, from $1.00 – 17.00 - Ensure software is compatible with multiple versions of Windows(7/11/XP etc)  - Understanding Window’s User Account Control, and ensuring app doesn’t trigger Windows Firewall | - Google Play Store: One Time $25 fee  - Apple App Store: $99 a year  - Understanding different Software Requirement Specifications so clients can easily use software app  - Repeat app updates to match mobile OS specifications |
| **Development Tools** | - Native MacOS apps via Apple’s language: Swift. Limited use of Java.  - iOS doesn’t support Java. Apps can be transferred to iPhone via OpenJDK.  - Visual Studio Code: Free download IDE - Xcode exclusive to Apple machines (rent/buy Macintosh) | - Native language is C but easily runs Java  - Nano: pre-loaded IDE that can be converted from Python to Java  - Eclipse (free) can be downloaded and used on Linux  - Visual Studio Code: free IDE for coding java applications | - Native language is C or C++ (for apps)  - Java applications will run on Windows, with use of Java Development Kit (free)  - Eclipse IDE (free) for the current java app - Visual Studio Code: (free) can run C languages & java with installation of JDK | - Current program was made in Eclipse IDE is native to Android phones. - iPhone/iOS: OpenJDK IDE for Java code.  - Pixel: Same developers of Android (Google Play Store), can run Java applications (Eclipse IDE)  - Visual Studio Code: IDE for all mobiles |

## Recommendations

Analyzation of characteristics and techniques specific to various systems architectures for recommendation to The Gaming Room:

1. **Operating Platform**: It’s our recommendation that for a gaming application, the use of the Windows Operating System (OS) be used. It’s extremely easy to use interface, high compatibility with other along systems/3rd-party tools, and a lower cost for both hosting and hardware makes it ideal for software design & hosting. Linux is very close second, only because it’s learning curve to operate requires more time and possibly professional help to utilize effectively.
2. **Operating Systems Architectures**: Windows is a Single User Multitasking OS, meaning a single user can execute multiple tasks (such as running multiple programs such as an IDE like Visual Studio, an Internet Browser, the File Explorer etc.) without fault. The hardware processes run through the kernel mode which helps the many parts such as the memory, central processing unit, and other hardware like the graphics processing unit to coordinate and helps the system run. The second layer, called the user mode, is where the user can interact and utilize applications to operate tasks. Additionally the Windows OS utilizes object-oriented methods for it’s input/output (I/O) requests, much more than other systems. This sets it apart from other operating systems, as the data structures & design are simplified, which makes it both flexible and easy to understand.
3. **Storage Management**: Data for the system is held in physical hardware, specifically within either a Hard Disk Drive (HDD) or the newer Solid State Drive (SSD). While the HDD’s can hold a greater amount of data, up to multiple terabytes, they are slower in read & write speeds. SSD’s can double, almost quadruple read/write capabilities, while sacrificing a slightly less storage space. Either option can handle the 1.6 gigabyte (GB) worth of picture data that The Gaming Room requires for their application, with room to spare for more additions in the future. SSD’s are recommended though, as their speed can be extremely beneficial for faster and more efficient user experience.
4. **Memory Management**: All operating systems require the hardware of physical memory (RAM) to operate. It helps load applications, files, and processes. Modern computer operating systems like Windows also utilizes virtual memory, which includes software to help the physical memory by freeing up data that’s not used as much, which improves both operating speeds & performance. Windows also utilizes Paging methods, meaning that the system properly categorizes physical memory and virtual memory, and loads the ‘pages’ of virtual memory to keep reading and writing data in a uniform manner that doesn’t interrupt system performance. This greatly helps in applications like Draw It Or Lose It, as it greatly reduces the risk of fragmentation issues like resource exhaustion & system failure.
5. **Distributed Systems and Networks**: Windows is regarded as a leader in user friendly file hosting, along with it’s ability to be compatible with other operating systems & external users if the operating system is used to host a server. The Windows system utilizes a distributed file system (DFS) called namespaces which not only allows the host to share files/access to programs in an easy way, but can also easily allow the host to move the data later and allow for easy future growth. For a robust and continuously operable software that allows multiple different users on a network, a combination of both User Datagram Protocol (UDP) and a Transmission Control Protocol (TCP) can be used. This allows the users to maintain a reliable connection that can handles a range of connectivity or even outages through TCP, while users can still have quick access and reliable connection speeds through the connectionless data transfer of UDP. This will work great for Draw It or Lose It, as players can get maintain effective & fast connectivity with both the server and other users while not having to worry about loss of data transfer to complete the games tasks.
6. **Security**: Windows comes standard with protection feature called simply “Windows Security” which continuously scans for malware, viruses, and security threats which can work both individually for users or for server file protection. Further protection would be highly recommended for a multiplayer style game that Draw It Or Lose It seeks to be. Development of a security policy for both internal and external awareness is paramount. User authentication on files can be done through the Windows File Permissions Tool, which can greatly help keep files protected and only accessible by authenticated users/owners. For network security, use of external player (user) authentication through passwords would be greatly beneficial to ensure that the network stays clean and data stays secure; multifactor authentication through one-time passwords or multi-encryption greatly improves security. Further security can be helpful, such as intrusion-detection and intrusion-prevention systems (IDS & IDPs) as well as network firewalls but these options can’t be coded in the application or included in Windows and may involve extra funding. Security should also be monitored and managed through penetration tests to identify weak points and maintain low risk.