

Draw It or Lose It Web

# **Liam Daley CS 230 Project Software Design**

Version 1.0

## 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 | 5/18/2023 | Liam Daley | Initial revision |
| 1.1 | 6/2/2023 | Liam Daley | Project Two update |
| 1.2 | 6/14/2023 | Liam Daley | Project Three update |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to bring their Android app game “Draw It or Lose It” to the Web, serving multiple platforms. This software needs to support multiple users, show graphics, and take timed inputs from players. To be broadly accessible from many Web-connected devices, it will leverage common technologies and services that enable the greatest number of potential users to participate.

## Requirements

* Web-based app accessible from multiple platforms.
* Image display from a large library
* Round timers and limits
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique.
* Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

Platform Independence: The game needs to be developed in a way that it can run on multiple platforms, regardless of the operating system. This constraint requires using web technologies that are compatible across different platforms. It also means avoiding platform-specific dependencies or technologies that may limit the application's portability.

* Implication: The application development should focus on using web technologies and frameworks that are cross-platform compatible, ensuring the game can be accessed and played seamlessly on different devices and operating systems.

Scalability: The game should be designed to handle multiple concurrent users and teams without compromising performance or user experience. It should be able to accommodate a growing user base and handle increased traffic during peak usage times.

* Implication: The application needs to employ scalable architecture and design patterns, such as load balancing and caching. This ensures that the system can handle increasing user demands and maintain optimal performance.

Security: As a distributed web application, security is crucial to protect user data, prevent unauthorized access, and ensure the integrity of the game. The application should employ secure mechanisms to verify user identities and protect against common security threats.

* Implication: The application needs to incorporate secure coding practices, implement proper authorization mechanisms, and utilize encryption for sensitive data transmission.

## [Domain Model](#_8h2ehzxfam4o)

GameService is the singleton master class that handles overall operations of the Gaming Room app. It has a none-to-many relationship with Game, which represents a single instance of Draw It or Lose It. These Game objects have none-to-many Teams, which represent groups of none-to-many Players, which are end users. Game, Team and Player all inherit from Entity, a superclass with methods to prevent reuse of ids and names.

**"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 systems are reliable and support development, but being traditionally closed-source and difficult to extend makes Mac a suboptimal platform for server development. Additionally, licensing fees for Apple and Mac dev environments make it cost-prohibitive. | As an open-source platform, Linux is widely supported for server hosting solutions. It is lightweight and customizable, meaning physical costs can be reduced through optimization. Being open source, there are no fees or licenses to acquire whatsoever to deploy a Linux server. | Windows is a supported platform for server developers, however operating system overhead can reduce performance-per-dollar by wasting resources on system functions. Windows Server is an edition of windows explicitly licensed by Microsoft for deployment. | Mobile devices are limited in power and network connectivity, making them very poor choices for hosting server applications. In particular, their poor thermal regulation makes them unable to handle sustained workloads. Similar to Mac, there are licensing costs to work in the iOS ecosystem as well. |
| **Client Side** | Browsers on Mac have feature parity with their counterparts from other operating systems, allowing them to browse the Web in an identical manner. | The popularity of Linux with the software developer crowd means browsers are well supported and feature-rich on the platform. | The most popular PC environment, all major browsers support features on Windows. The wide user base and support make it a potent platform to develop for. | The most regularly used devices in the world, mobile devices are equipped with lightweight and performant browsers that support many modern features of the Web. If a native application is to be used instead, certain web app frameworks support being run as containered applications for a different user presentation. |
| **Development Tools** | Web development is homogenous between platforms, consisting of HTML, CSS, and an embedded language like JavaScript. Server development can be done through any language that compiles into a Mac binary. Apple’s custom language Swift is intended to run on most iOS devices, but must be learned by development teams. Apple charges licensing costs to developers to work in it’s ecosystem. | Linux is an open-source operating system supported by developers, so there is a myriad of ways to develop server software for it. Most major languages can be built into Linux program files, including C, C++, Java, and C#. These tools are freely accessible, without licensing costs. The most popular development environment, jetBrains Rider, requires corporate licensing. | As the most popular platform, Windows is the most supported in terms of languages that can be compiled into a Windows binary. Additionally, many APIs are available for Windows features, such as dotNET. Certain IDEs and software packages are free for enterprise use, while others like Visual Studio require licensing. | Mobile devices are developed for in two categories, iOS and Android. These two systems have very different languages and tools required to build for them, making it a more expensive process to deploy apps to both platforms. Apple charges users for development tools to build for it’s devices, while Google Android requires only a one-time fee to publish to the Play Store to use it’s Android Studio IDE and API. |

## Recommendations

1. **Operating Platform**: Linux is the recommended choice for server deployment in multiplatform online games. Its open-source nature allows developers to customize and optimize the servers to meet project requirements, ensuring seamless performance across various platforms. Additionally, Linux's stability and reliability ensure uninterrupted experiences for players. The array of server management tools and comprehensive documentation available within the Linux ecosystem simplifies the development process, making it easier for developers to create and maintain a scalable and efficient server infrastructure.
2. **Operating Systems Architectures**: Linux, being open source, has distributions and versions available for many common hardware architectures, including x86, x86\_64 and ARM. Primarily suggested is x86\_64 for its improved memory bandwidth compared to x86 and much wider software adoption compared to ARM. Being the consumer hardware architecture, there are x86\_64 systems widely available from server producers and from platform providers such as Microsoft Azure and Amazon Web Services. Additionally, they have developed security features built into the instruction set and chip architecture, offering greater security by default.
3. **Storage Management**: Linux servers use the ext4 file system most commonly. This system provides features for data integrity checking, efficient space allocation and support for large files. Additionally, RAID setups and Network-Attached Storage is supported, allowing for centralized storage of user data using backups and redundancy drives to protect its integrity. Volume encryption is another valuable tool supported at the file-system level, providing an additional line of defense for potentially sensitive user data.
4. **Memory Management**: Linux utilizes virtual memory to provide each game server process with a virtual address space larger than the physical memory available. It also employs various caching mechanisms to improve memory access times. Draw It or Lose It servers can benefit from these caches by reducing disk I/O operations and keeping frequently accessed data in memory. Additionally, programs running on Linux can utilize garbage collection (alongside manual memory management) to minimize their impact on hardware resources. This involves an algorithm that finds objects that are no longer referenced, freeing that space for later use.
5. **Distributed Systems and Networks**: Draw It or Lose It is best fit for a server-client distributed system. The server hosts games, selects the image to show, and manages joining/leaving users, while clients collect input from players and display the transmitted image in accordance with the game rules. Network disconnects between one client and the server disrupt the experience for that user individually, while a server disconnect removes all affected players from the game. In an environment with reliable connection between clients to servers across a large geographical area, this ensures that the fewest players are impacted by any given network anomaly.
6. **Security**: Being open-source, Linux supports many frameworks for user and application security. REST APIs, such as Jersey, provide a structure on which a secure and reliable networked application can exist. User authentication and authorization is an important step in software security, and end-to-end encryption using methods such as HTTPS preserve the privacy of data even as it travels over the Internet. Additionally, maintaining a need-to-know development style, only revealing or transmitting minimal information about users to clients, helps prevent unwanted data access.