

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

Alexander Flood

21st, April 2024

CS-230 Operating Platforms

## 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 | 03/24/2024 | Alexander Flood | Initial release of the software design document for “Draw It or Lose It” |
| 2.0 | 4/07/2024 | Alexander Flood | Assessed the characteristics of each platform and how it would perform on the client side, server side, and the available development tools and considerations for each. |
| 3.0 | 4/21/2024 | Alexander Flood | Provided recommendations for the operating platform, operating system architectures, storage management, memory management, distributed systems and networks, and security. |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions is transforming “Draw It or Lose It” from an android-exclusive application to a complete web-based gaming experience that will support multiple platforms. This transformation aims to widen the game’s accessibility, allowing it to reach a broad audience and enhance player engagement on desktops, tablets, and smartphones.

The game is inspire by the classic TV show, “Win, Lose, or Draw,” and challenges teams to guess words or phrases based on drawings as they are rendered over 30 seconds. This proposal outlines the software requirements, constraints, and aims to ensure an engaging and accessible game. We also look at some of the anticipated challenges related to cross-platform support and user interaction.

## Requirements

Business Requirements

* Market expansion. Transition the game to a web-based platform to increase its accessibility and reach a wider audience.
* Accessibility. Ensure the game is accessible on various devices, including desktops, tablets, smartphones, and web browsers as well as the top application stores such as the iOS store, Android app store, and Microsoft marketplace.

Technical Requirements

* Cross-platform compatibility. Ensure the game functions seamlessly and predictably across different operating systems and platforms, as well as web browsers.
* Unique identification for game instances, teams, and players. This will allow instances to be managed efficiently.
* Real time interaction is required to facilitate impactful and engaging gameplay among players.

## [Design Constraints](#_2et92p0)

Primary design constraints include cross-platform compatibility, ensuring the game operates predictably across different devices and operating systems, such as the Android and iOS platforms. The development approach should leverage existing web technologies and potentially use cross-platform libraries or frameworks to help maintain consistency and performance between platforms.

Real time data handling will also be another major design constraint, as managing unique instances of games, teams, and players, as well as real time rendering in a distributed environment will require a robust backend architecture. Low latency solutions for networking will need to be assessed to ensure smooth and uninterrupted gameplay.

Database management will also be a crucial design constraint. The database selected must manage concurrent sessions and ensure data integrity and uniqueness across all games while allowing for fast and efficient queries.

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram for the application represents an object oriented approach, centered around an Entity base class that imparts attributes like ‘id’ and ‘name’ to the Game, Team, and Player classes through the principle of inheritance. The inheritance streamlines the creation of unique identifiers for each entity without unnecessary duplication of code. The relationships between Game and Team, and Team and Player, display a clear hierarchy and the assembly of the game structure.

The diagram also demonstrates encapsulation with each class managing its own data and behavior, while polymorphism is used through methods like ‘toString’ that is tailored to each class context. The singleton pattern is shown in the GameService class, which ensures that only one game service instance controls the game operations, preventing duplicate game instances. The iterator pattern is also used to navigate through collections of games and players, allowing the application to manage and traverse these entities.

**"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** | Macs are unix based and known for their reliable, secure, and user friendly interfaces.  Continued platform support and security updates can be expected for many years.    However, server hardware options are limited and costly compared to other platforms. Commercial licensing fees must also be considered. | Linux is a popular choice and regarded for its community support and security.  Open source software can be leveraged for many backend requirements. While this can save money on licensing fees, technical expertise are required to manage and scale services efficiently. | Windows servers are well integrated with existing Microsoft services while offering a user friendly interface.  Licensing the Windows platform for commercial use can be expensive and the platform is often targeted by malware, which should be factored into development costs.  Routine security audits and updates are neccessary to protect server data. | While mobile devices offer a diverse ecosystem of applications, limitations on processing power and storage will pose particularly challenging for hosting a web-based application when compared to more traditional server platforms.  Mobile devices are also often restricted to wireless communications which can be unreliable for hosting. |
| **Client Side** | Mac OS X has a rich user interface that is favored by designers, developers, and creative professionals alike.  A world class SDK is provided by Apple for native development.  Consideration should be given to the premium cost and limited selection of hardware. | Linux applications must be developed with variability in mind.  There are many distributions of Linux based operating systems which must be considered when building the application to ensure predictable behavior across the many distributions.  Many open source graphical user interface libraries are available on Linux platforms for building rich user interfaces. | The Windows operating system is widely used and provides a consistent and predictable environment to run graphically accelerated applications.  The platform also allows for customization, but is frequently targeted by malware due to the extensive user base.  Ensuring secure client connections to the server is essential for safe and functional web services. | Developing mobile clients for platforms like Android and iOS will let us tailor the experience to each platform, considering different screen sizes and processing power.  While tailoring the client for each mobile platform will increase development time and cost by a considerable amount, it should be considered given the wide reach of Apple and Android mobile platforms. |
| **Development Tools** | Mac supports modern IDEs and programming tools such as Xcode with rich support for languages like Swift and Objective-C. Mac also supports other standard languages such as C, C++, Java, as well as many other commercial and open source editors.  Mac also provides exceptional documentation for their products and languages such as Objective-C which can greatly aid in development. | Linux supports a wide range of programming languages and tools and is preferred by users who need deep customization or integration with server-side technologies.  Open source editors such as Eclipse are available, as well as paid solutions like IntelliJ IDEA.  Support is well established for standard languages such as C, C++, Java, and many others. | Windows supports many development IDEs such as Visual Studio as well as an inclusive suite of technologies to build applications in languages like C# with the .NET framework.  Users can also find commerical and open source editors for various other language such as C++, Java, Javscript, and many more. | Tools for mobile platform development are well supported and mature. Android Studio can be leveraged for the Android platform while Xcode can be used for the iOS platform.  Cross-platform frameworks such as React Native or Flutter can allow developers to create applications for multiple platforms from a single code base, but these frameworks often have performance trade-offs.  Decisions on how many platforms to to target when developing mobile products can have a big influence on development cost. |

## 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 using a Linux based operating system such as Ubuntu Server or Fedora Server. This will allow flexibility in deployment across many host providers and ensure a secure environment to manage the game data. This will allow the project to save money on costly licensing fees associated with Windows or Mac based servers. Another added benefit to the project is support for open source technologies which can further reduce development costs and time.
2. To ensure the game server is properly secured and routinely updated, the development team should have members familiar with Linux based operating systems. Team members who are not familiar can be quickly brought up to speed with the abundant documentation available online; another benefit of the open source approach to Linux.
3. **Operating Systems Architectures**: The Linux kernel is stable and mature, making Linux platforms a great choice for the deployment of our game server. Linux is also supported by many web-based cloud providers such as Amazon Web Services and the Google Cloud Platform, providing the scalability and stability that can serve many users regardless of their underlying operating system or location.
4. Another useful platform feature is the support for cron jobs. Cron is a scheduler used to run scripts or commands at scheduled times. This feature can help support our game servers by automating tasks like backups of critical game files, system updates, and system monitoring scripts.
5. **Storage Management**: The game servers should be deployed on hardware that includes solid state drives (SSD), which will allow for fast and efficient file storage. A robust RAID (Redundant Array of Independent Disks) configuration should be established, where multiple drives are used to distribute and duplicate data. In particular, a RAID 1 configuration which duplicates the same data on two disks is recommended. In this case, if one disk were to fail, the system can continue to operate with the remaining disk without data loss. This feature is very useful for protecting sensitive game and player files.
6. For database management, I recommend either of the popular open source relational databases PostgreSQL or MySQL which can be hosted directly on the server or in a cloud solution such as Amazon Web Services or the Google Cloud Platform. While hosting the database with a cloud provider does come with cost considerations, these solutions often provide automated security updates and backups that can also save development time.
7. **Memory Management**: Memory resources can monitored in real time easily on a Linux based platform, allowing servers hosted on cloud solutions to be dynamically scaled up based on player demand, allowing for greater efficiency and cost savings. This can also be useful when planning for big player events, where many players may connect in a short period of time without crashing the server.
8. Another feature that can be implemented at the client level to help memory management is lazy loading. This is a technique where resources are loaded into memory as they are needed. Using this technique with our large image files can help the client run faster, giving players a more enjoyable and smooth experience. This technique is especially useful for mobile devices where memory might be lacking.
9. **Distributed Systems and Networks**: Hosting our game server in a cloud solution such as Amazon Web Services or the Google Cloud Platform will allow us to leverage an existing global network of highly efficient servers across the globe, while reducing latency and improving response times. By utilizing this existing global cloud infrastructure, Draw It or Lose It can achieve the greatest market penetration.
10. This distribution will also improve resilience against server outages. With load balancers, traffic can be redirected from servers experiencing issues or high load to more performant servers. We can also replicate the servers in regions where player demand is particularly high, ensuring broad market expansion is possible. The server will also host a RESTful (Representational State Transfer) API, a web services model that will allow the game server to communicate with different clients and mobile devices.
11. **Security**: When it comes to security concerns, the application should implement SSL encryption for sensitive game and player data to ensure fair and fun play. SSL encryption will make it difficult for bad actors to intercept and tamper with sensitive game data. Role based access control can also be implemented to ensure separation between admins, games, and players. Amazon Cloud Services and the Google Cloud platform provide role management and encryption solutions to help facilitate this process and reduce development time while building a secure product.
12. Regular security audits should be implemented to ensure security between various platforms. Because security is not just a concern at the software level, it is also important for developers and support staff to be briefed on secure password practices as well as current phishing trends.