

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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  2.0  3.0 | 01/20/2025  02/09/2025  02/23/2025 | Finnegan Thomas | First round of document specification  Second round of document specification (Evaluation)  Third round of document specification (Recommendations) |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has requested a web-based game that serves multiple platforms based on their current game, “Draw It or Lose It”. CTS will transform their application by maintain their existing application and implementing functionality to manage teams and players; implement unique identifiers to ensure each team and player are unique; utilize the singleton pattern to ensure only one instance of the game exists at any given time.

## Requirements

**Business**

* Expand application beyond Android users
* Manage multiple teams and players

**Technical**

* Utilize singleton pattern
* Create unique identifiers
* Implement identifier checks to ensure unique game and team names; and to ensure only 1 instance of the game exists

## [Design Constraints](#_2et92p0)

Ensure browser compatibility, application must be functional across different browsers which can affect frameworks and more. Manage memory as application must only have a single game instance at any given time.

## [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 diagram below depicts an entity class with game, team, and player classes inheriting from the entity class. The inheritance relationships illustrate the inheritance principle from the OOP principles. We also see that the classes contain private attributes that can only be manipulated by their respective methods. This aspect demonstrates encapsulation. In the UML diagram, we see that a gameservice class has an association with the game class of a 0 to many meaning that the gameservice class interacts with 0 to many game objects. Finally, we have a program driver which contains the main method used to drive the application and it has an association with the singletontester class which verifies the singleton pattern is being uses correctly. This UML Diagram also illustrates the other two principles of OOP. Modularity exists within this class structure as methods and attributes are attributed to their respective classes. Polymorphism is also observed within this class structure as multiple objects can be created through a common interface (the entity class). This class structure is utilizing all OOP principles which allows us to create an application that is scalable, maintainable, and reusable by clearly separating classes into modular components, illustrating proper separation, and readable dependencies. These characteristic are what helps the program utilize singleton and create unique identifiers that can be checked.

**"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** | Mac has a Unix based OS making it compatible with many tools, however, Mac is not optimized for high-traffic web hosting which is a requirement for The Gaming Room and its application. Mac also includes a built-in server which is beneficial for local development, but for large applications, servers such as AWS and Azure are industry standard servers for high-traffic applications. One benefit of Mac is that it’s relatively more secure than windows, but this doesn’t necessarily translate to better security for high-traffic applications. Additionally, there is also a lack of support for hosting on Mac and combined with expensive hardware and software fees—which may be charged by the hour or monthly—, Mac is not recommended for web hosting compared to Linux and Windows who both have high scalability and strong support. | Linux is also Unix based making it compatible with a wide variety of development tools. It has no licensing cost and is completely open source, allowing for full control of the OS. Linux is also viewed as an industry standard for web hosting as most the web’s servers run on Linux due to its stability, resource efficiency, and high performance optimization. Additionally, Linux has strong security measures and application support such as popular webservers (Apache, Nginx), databases (MySQL, PostgreSQL), and cloud platforms (AWS, Azure). However, Linux faces issues with hardware compatibility and a lack of professional support can become an issue. Fortunately, there are some enterprise versions such as Red Hat Enterprise Linux and Ubuntu Server which offer commercial support. However, these additional third party services have their own associated fees. Regardless of its limitations, Linux remains the standard for web server hosting and is highly recommended for The Gaming Room. | Windows is compatible with Microsoft technologies such as Internet Information Services, .NET framework, and Microsoft SQL Server. In addition, Windows has a vast network of professional support, making it reliable for large-scale applications relying on Microsoft-based infrastructure. However, Windows has expensive licensing costs and poor system resource management in contrast to Linux. Lastly, Windows is consistently targeted by cyberattacks requiring regular security updates. Windows is still a good choice for web server hosting; however, Linux may be a better option for The Gaming Room due to its cost and performance optimization. | Mobile devices are highly portable and accessible, which are useful for temporary servers during development and testing. Some mobile applications offer web server capabilities such as Android and iOS. Mobile devices are also more energy-efficient due to their low power consumption and they incorporate built-in security features such as firewalls and encryption to enhance protection. However, they are limited in their performance and resources making them less than ideal for high-traffic applications like The Gaming Room’s. Mobile devices are also unreliable as they require frequent updates and maintenance. Additionally, there are no direct licensing fees, however, third party applications are needed for hosting which may incur additional costs. Therefore, mobile devices are not recommended for hosting The Gaming room’s application. |
| **Client Side** | Mac clients naturally support Java which is beneficial for our gaming application. However, additional costs may incur by hiring Mac-experienced developers for safari optimization and MacOS UI behaviors. Testing also adds additional costs as multiple MacOS versions and hardware will be required. Lastly, third-party tools may be needed for optimizing Mac performance, potentially introducing additional fees. There are also time considerations as ensuring HTTP communications between Safari and the backend are smooth. In addition, safari compatibility testing can be time consuming from cross-browser debugging, deploying, and updating. Finally, there are expertise considerations as backend developers must ensure that server-side responses are optimized for Safari’s unique networking behavior and frontend developers will need expertise in MacOS UI responsiveness, debugging, and performance optimizations. | Linux distributions vary greatly across environments, increasing initial costs in development by requiring testing and optimization of testing platforms to account for various distributions. This also means testing and development time may take longer to ensure proper system dependencies, UI behavior, and network configurations. Finally, experts in dropwizard are needed to ensure server-side efficiency and compatibility. Alongside, experts in frontend will be needed to handle Linux specific UI behavior and performance optimization. Lastly, those knowledgeable of Linux specific components, such as security configurations and distribution-specific dependencies are vital to this application’s success. Ensuring a seamless integration across different Linux environments will be paramount to The Gaming Room’s success. | Dropwizard works well on Windows and may reduce initial costs significantly. However, testing Windows’ various hardware infrastructures may add additional costs alongside cost for frontend compatibility. Development time will also increase to ensure smooth communication with Windows’ networks, while testing various browsers and hardware configurations will also increasing testing time. Finally, backend developers experienced in Windows will ensure efficient performance and network communication. Skilled frontend developers will also be required to ensure browser compatibility and UI responsiveness, and Windows-specific specialists guarantee stability and performance with their knowledge in hardware configurations and system optimizations. | Mobile devices vary significantly making optimization a critical element for success. Development costs may remain low depending of if third party tools will be needed. However, testing costs can be expensive as extensive device compatibility is necessary to ensure performance optimizations across various screen sizes, operating systems, and hardware configurations. Development time will increase to implement responsiveness, network communication, and control optimizations for mobile devices. Testing will also incur time costs as it requires significant time to account for variations in hardware, OS versions, and device behaviors. Finally, expertise in backend will be needed to ensure efficient performance with lightweight logic and frontend developers with experience in mobile gaming UI/UX to create responsive interfaces. Mobile-specific specialists will also be helpful in optimizing hardware and performance on mobile platforms. |
| **Development Tools** | Since we are utilizing dropwizard, we’ll be using Java for backend development. However, front end languages should include JavaScript and WebGL which are essential for web-based applications. A common database used with dropwizard is MySQL which we can utilize or PostgreSQL and MongoDB depending on the nature of the database. As for IDEs, IntelliJ IDEA is recommended, however, Eclipse works fine for backend and as for frontend VS Code is an ideal IDE for its java support. Additionally, there are testing tools such as safari Web inspector and browser stack for testing Mac compatibility. Finally, AWS or GCP can be used for deployment alongside Apache Tomcat for local deployment. In addition to these tools, experienced developers, QA testers, and more will be required. These teams will add to cost in addition to licensing fees and for IDEs and testing tools. | Again, we’ll be using Java for the backend and frontend can also be JavaScript and webGL as these are widely used. Databases are similar as we are utilizing dropwizard, so MySQL is a prominent choice depending on the nature of the database. IDEs are similar as well with IntelliJ IDEA or VS Code being among the recommended for their Java support, but Eclipse works as well. As for development tools, Firefox Developer Edition, Chrome DevTools and Selenium are all good choices for cross-browser testing and debugging. Similar deployment tools can be used such as AWS and GCP or Apache Tomcat. Finally, firewalls and performance optimizer tools can be used as well to increase performance and security. These tools may need experienced developers in front and backend with QA testers and more. This team in addition to the various fees of third party servers will add to the cost. Fortunately, many Linux development tools are open-source and free, cutting the costs significantly. | Windows is no different with backend and frontend languages or databases. We can use Java, JavaScript, WebGL, and MySQL. IDEs are similar with IntelliJ IDEA and VS Code being some of the most popular, but Eclipse works too. We’ll have to utilize Edge Developer Tools, Chrome DevTools, and BrowserStack to test across different versions of Windows. As for deployment we can still use AWS or Microsoft Azure and Apache Tomcat. Lastly, understanding Windows’ Firewall rules and ensuring secure connections are crucial to deploying on windows. Deploying on Windows will demand high-functioning and cross-functional teams who are both experienced and provide specialization in their respective scope. Licensing fees are also present with testing and deployment tools each have their own pricing models. The team and tools used will both affect costs for deploying. | Mobile devices will also utilize Java for backend and JavaScript and WebGL for front end languages. As for IDEs they remain the same with VS Code and IntelliJ IDEA being popular, however, Android studio and Xcode are crucial for mobile devices as they are essentially required to develop and test on mobile devices. AWS works for hosting; however, firebase hosting may be a better option for mobile devices for its authentication and real-time database. Testing tools also include Chrome DevTools, Safari Web Inspector, and BrowserStack. There are also additional automated test tools such as GitHub Actions and Fastlane to help automate and simplify testing. Mobile devices are particularly unique and challenging requiring specialized individuals and could require dedicated teams for mobile devices. In addition to a dedicated team, licensing fees for will also add to the total costs of deploying. |

## 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 would highly recommend using a Linux operating platform due to its stability/reliability, performance & efficiency, cost-effectiveness, and compatibility/scalability. Linux servers are widely known for their uptime and stability which is ideal for hosting a game such as Draw It or Lose It because we need not worry about system updates or crashes as frequently. Linux is also optimized for performance due to its lightweight build and CPU management meaning that Draw It or Lose It can handle a high number of users without resource issues causing lagging or crashing of the application. Since Linux is free and open source, there are no licensing costs which will reduce initial expenses and make scaling more cost effective in the future. Lastly, Linux is widely used meaning it supports various server technologies (Apache, Docker, Nginx) and integrates seamlessly with cloud providers (AWS, Google Cloud, Azure) which are important aspects for deploying a web based game such as Draw It or Lost It across various operating platforms.
2. **Operating Systems Architectures**: Since Linux is my recommended choice for an operating platform, I also recommend using the ARM64 architecture which is takes advantage of a simple instruction set that’s ideal for low-power applications such as Draw It or Lose It. Linux supports more architectures than both Windows and Mac, which can be a benefit as we can choose other architectures that may fit our needs unlike Windows and Mac. ARM64 uses RISC (Reduced Instruction Set Computing) meaning it consumes less power making it ideal for mobile devices and is extremely scalable as ARM64 has been used from watches to computers. This is highly attractive to Draw It or Lose It as we are focused on expanding the game to multiple operating platforms including mobile devices. ARM64 also offers licensing so that we may purchase its architecture and modify the design to fit our specific needs and desires making this architecture very personalisable. This is a stark contrast to Windows and Mac which both have limited and controlled architectures. Lastly, many cloud technologies take advantage of ARM architectures as they reduce power and increase performance which means we can integrate seamlessly with these technologies as we share a common architecture. Windows and Mac do not share that luxury as Windows requires more resources and Mac is highly optimized for Apple’s ecosystem which gives it a limited flexibility.
3. **Storage Management**: ZFS (Zettabyte File System) is a file system which is used to store and manage files (collection of similar data) on a disk (physical storage media) and also includes storage pools which are a combination of physical disks into a single logical storage unit and virtual devices (individual or multiple discs) which are the building blocks for storage pools. It’s a powerful file system that’s suited for gaming servers because of its reliability, scalability, and data protection measures. ZFS automatically detects and repairs data corruption and includes snapshots (a captured point in time) which can return a previous version in case of crashes or failures. ZFS also handles large amounts of data via compression and minimizes redundant data all while working well in distributed environments which is ideal for Draw It or Lose It as we’ll have a high number of users across various operating platforms.
4. **Memory Management**: Linux utilizes virtual memory management which uses swap space (temporary storage area in a disc) when RAM is full. This ensures a seamless experience when users are in a game as the additional temporary space prevents lags and delays. Linux also takes advantage of page caching which stores frequently accessed data into RAM making retrieval times faster. This is ideal for Draw It or Lose It as page caching can speed up image rendering and reduce load times. Additionally, Linux implements demand paging which only loads necessary data into RAM instead of unused data which optimizes memory usage. As we’ve discussed before, memory and storage appear similar but differ in drastic ways. Memory is used for temporary data such as data for active programs, RAM, and Caches. Short term data is frequently used and stored within memory for fast access, however, memory also serves to prevent programs from interfering with each other, deallocates unused memory for additional processes, prevents crashes via prioritizing programs and much more. Storage is similar; however, storage is for long term data such as files, databases, and more. Storage utilizes file systems to manage and organize files within directories and prevents corruption by efficiently allocating storage and restricting access. Storage is designed to protect against hardware failures, ensure recovery of data, and utilize permissions to restrict access from unauthorized processes.
5. **Distributed Systems and Networks**: A client-server architecture would work best for Draw It or Lose It. The game would run on the centralized Linux server while clients (accessing on various platforms) can connect through network requests. This ensures a real time gameplay for all players which is crucial to the nature of Draw It or Lose It. Since the game is web-based we can take advantage of running on a browser which will use WebSockets to maintain interactions and reduce delays. In addition, we should look at load balancers to prevent the server from overloading when a large amount of traffic joins and containerization to scale resources based on player demand to avoid crashes. Cloud based databases will also prove beneficial as they can sync game data across different devices allowing players to switch operating platforms without losing their progress. However, to ensure fair gameplay and an enjoyable experience, we should think about reducing image quality or enabling offline mode for connectivity issues and utilize distributed servers (AWS, Azure, Google Cloud) to prevent a single point of failure
6. **Security**: Firstly, there are various ways to secure authentication and control access. Third-party authentication (Google, Apple, etc.) can prevent credential leaks and JSON Web Tokens can secure token-based authentication with expiration and encryption, alternatively, we should also use multi-factor authentication that adds an extra layer of security using email or authenticator apps. Data encryption will be needed to encrypt communication between clients and servers, encrypt user data storage, and protect in game communication. Additionally, we should ensure secure storage and cloud protection by hashing passwords, using role based access controls to limit data access, and follow cloud security compliance. There are also cyber threats we should be cognizant of and utilize firewalls, ddos protection, rate limiting and captcha to prevent brute-force attacks and ensure continuous verification before granting access. Lastly, API gateway authentication such as API keys can secure game APIs and secure WebSockets can prevent against man in the middle attacks. We should also deploy regular security audits to find and fix vulnerabilities.