

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc94801767)

[Table of Contents 2](#_Toc94801768)

[Document Revision History 2](#_Toc94801769)

[Executive Summary 3](#_Toc94801770)

[Design Constraints 3](#_Toc94801771)

[System Architecture View 3](#_Toc94801772)

[Domain Model 3](#_Toc94801773)

[Evaluation 4](#_Toc94801774)

[Recommendations 10](#_Toc94801775)

[Sources 12](#_Toc94801776)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/22/2022 | Connor Wallace | First draft of document. Original Summary based on proposal from The Gaming Room. Requirements and Constraints based on original summary. |
| 1.1 | 2/6/2022 | Connor Wallace | Updated the Evaluation section. |
| 1.2 | 2/20/2022 | Connor Wallace | Updated the Recommendations section. |

## [Executive Summary](#_sbfa50wo7nsh)

The Game room is seeking to develop a web-based game. This game must be accessible to multiple players on different devices and guarantee that the game remains accessible to all players. Each game must allow for multiple players on a team, and multiple teams in a game. It is also necessary to ensure that only one instance of a game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

1. The Game class must be able to handle any number of teams playing.
2. The Team class must be able to have multiple players
3. Game and team names must be unique.
4. The GameService class must be implemented using the Singleton design pattern. This will ensure that the requirements “game and team names must be unique to allow users to check whether a name is in use when choosing a team name,” and “only one instance of the game can exist in memory at any given time” can be met by ensure there is only one instance of GameService managing all games, players, and teams.

## [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 Game, Team, and Player classes will all inherit from the Entity class. The Entity will serve as a base class implementing all basic information and methods that al the subclasses will need. Each Entity subclass has an association with another subclass. Said simply: each Game can have any number of Teams, and each Team can have any number of Players. The GameService class does not inherit from the Entity class but has an association with the Game class (the GameService can have any number of Games). The GameService implements the Singleton Design pattern by having a private constructor, a static variable of its own class (service in the UML), and a static getInstance method.

This UML demonstrates several Object-Oriented Programming Principles (OOPP). The Game, Team, and Player classes demonstrate inheritance by being subclasses of Entity. Each class in the UML is designed with encapsulation by having all class variables private with accessor and mutator methods.

**"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** | The Mac OS is not often used for servers. While it is possible to do so, its cost is greater than other options, and there will be less tools and support for it. | Linux is the most popular choice for a web server OS. Linux is cost effective, the base OS is free, as are many relevant tools. Linux servers are setup for multiple users, and remote administration. Linux experiences security errors less often than Windows. Linux servers are more complex to operate and update. Not all versions come with long-term support. | Windows is a more beginner-friendly and intuitive server OS than Linux. Windows comes with guaranteed long-term support. It is simpler to update Windows servers, and install drivers for hardware. Windows servers come with high licensing costs, and are not designed as multi-user systems. This leads to more potential for user error and security issues. Windows is also more vulnerable to malware than Linux. | It is technically possible to use a mobile device as a server. It is not a feasible solution for our project due to performance and a number of other limitations. |
| **Client Side** | Because we are not running any software that interacts with the client’s operating system the only operating system concern for a web application is that it is able to run on browsers supported by that operating system. To accomplish this, a team of developers fluent in HTML, and CSS will be necessary. CSS frameworks, and the reset stylesheet are recommended for making website compatible with all browsers. HTML expertise is necessary to ensure the Doctype is defined and compatible with all browsers, and that there are fallbacks for features unavailable in certain browsers. Conditional statements are necessary in both CSS and HTML to handle older or more difficult browsers. Testing in multiple browsers is essential, so a team must be put together and given adequate time to perform tests in multiple browsers.  Popular browsers for MacOS are: Safari, Opera, Chrome, and FireFox. | Because we are not running any software that interacts with the client’s operating system the only operating system concern for a web application is that it is able to run on browsers supported by that operating system. To accomplish this, a team of developers fluent in HTML, and CSS will be necessary. CSS frameworks, and the reset stylesheet are recommended for making website compatible with all browsers. HTML expertise is necessary to ensure the Doctype is defined and compatible with all browsers, and that there are fallbacks for features unavailable in certain browsers. Conditional statements are necessary in both CSS and HTML to handle older or more difficult browsers. Testing in multiple browsers is essential, so a team must be put together and given adequate time to perform tests in multiple browsers.  Popular browsers for Linux are: Tor, Brave, Chrome, and FireFox. | Because we are not running any software that interacts with the client’s operating system the only operating system concern for a web application is that it is able to run on browsers supported by that operating system. To accomplish this, a team of developers fluent in HTML, and CSS will be necessary. CSS frameworks, and the reset stylesheet are recommended for making website compatible with all browsers. HTML expertise is necessary to ensure the Doctype is defined and compatible with all browsers, and that there are fallbacks for features unavailable in certain browsers. Conditional statements are necessary in both CSS and HTML to handle older or more difficult browsers. Testing in multiple browsers is essential, so a team must be put together and given adequate time to perform tests in multiple browsers.  Popular browsers for  Windows are: Microsoft Edge, Safari, Chrome, and FireFox. | Developing a web application that functions well on both desktop and mobile devices has two possible solutions. The first is to develop a stripped back web application with few images, no pop-ups, and carefully considered button and font sizes. Media queries and meta tags in HTML are also recommended for responsive performance on mobile devices. This first option would require the development team to be familiar with adapting webpages for mobile device performance. The second option is to develop a mobile app to better showcase the game in the mobile environment. A mobile app would be a better environment for image rendering (a core aspect of our game). Making the webpage mobile friendly would require us to hold back performance of the desktop clients so the mobile clients could keep up. This second option would require another team of developers to create the application, which would be more expensive. A decision would also have to be made to either refactor the customer’s current Android application and write an application for iOS, or to create a new hybrid application. Either plan will require additional testing time. Either a mobile emulator, or a suite of devices for testing will be necessary. |
| **Development Tools** | As stated above, there is no actual difference in developing a web application for any specific desktop OS. Unless the customer chooses to develop an app for mobile instead of a mobile friendly website, only one development team will be necessary.  Our backend has been implemented using the Java language in the Dropwizard framework. The top IDEs for Java web development are Eclipse, IntelliJ, and NetBeans. I recommend staying in Eclipse because this is what the team has been working in.  For the frontend we will need to write HTML, CSS, and likely some Javascript. Our development team will need to be comfortable using HTTP protocol and using the JSON text format. We will want to choose a toolkit/plugin for writing, debugging, and testing our frontend code in Eclipse. There are many different options at differing costs. The Eclipse web developer tools seem to be a comprehensive free option from the Eclipse Marketplace (https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0). | As stated above, there is no actual difference in developing a web application for any specific desktop OS. Unless the customer chooses to develop an app for mobile instead of a mobile friendly website, only one development team will be necessary.  Our backend has been implemented using the Java language in the Dropwizard framework. The top IDEs for Java web development are Eclipse, IntelliJ, and NetBeans. I recommend staying in Eclipse because this is what the team has been working in.  For the frontend we will need to write HTML, CSS, and likely some Javascript. Our development team will need to be comfortable using HTTP protocol and using the JSON text format. We will want to choose a toolkit/plugin for writing, debugging, and testing our frontend code in Eclipse. There are many different options at differing costs. The Eclipse web developer tools seem to be a comprehensive free option from the Eclipse Marketplace (https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0). | As stated above, there is no actual difference in developing a web application for any specific desktop OS. Unless the customer chooses to develop an app for mobile instead of a mobile friendly website, only one development team will be necessary.  Our backend has been implemented using the Java language in the Dropwizard framework. The top IDEs for Java web development are Eclipse, IntelliJ, and NetBeans. I recommend staying in Eclipse because this is what the team has been working in.  For the frontend we will need to write HTML, CSS, and likely some Javascript. Our development team will need to be comfortable using HTTP protocol and using the JSON text format. We will want to choose a toolkit/plugin for writing, debugging, and testing our frontend code in Eclipse. There are many different options at differing costs. The Eclipse web developer tools seem to be a comprehensive free option from the Eclipse Marketplace (https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0). | In this section we will assume the customer has chosen to develop an application for mobile devices (if the customer opts to have a mobile friendly website, this section will mirror the others to its left).  If we opt to make a hybrid app for Draw It or Lose It, this would be accomplished using HTML, CSS, and Javascript. Dropwizard seems to have the capability to be a framework for mobile applications, but other frameworks such as React Native are preferred by the professional community. Eclipse is a viable IDE for Dropwizard, but a different IDE would be necessary for React Native.  Creating a native app for iOS would require some code in Swift, and potentially Objective-C. This would require its own IDEs, framework, and plugins, many of which have licensing costs.  If we chose to develop a native app for iOS, we will need to ensure we have a team that is comfortable with these technologies, because they are not similar to the work this team has already done. |

## 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**:

Linux is the most flexible and trusted server OS. While it may require the most expertise to utilize, the security, and community support make it the correct choice for our project. The recommended architecture for this project will be a Linux server that handles client requests and uses a database to get files by communicating with a Linux file storage server.

1. **Operating Systems Architectures**:

This project requires two servers for multiple reasons. Keeping the files in a separate server will improve performance of the application server handling client requests. Keeping the files separate also improves security because the application server will be the only client of the file server. Beginning the project with two databases will also allow for greater scalability. Below is a summary of the Linux operating system and its advantages from *Operating System Concepts*:

Linux is a modern, free operating system based on UNIX standards. [. . .] It provides a programming interface and user interface compatible with standard UNIX systems and can run many UNIX applications, including an increasing number of commercially supported applications. [. . .]

The Linux kernel is implemented as a traditional monolithic kernel for performance reasons, but it is modular enough in design to allow most drivers to be dynamically loaded and unloaded at run time.

Linux is a multiuser system, providing protection between processes and running multiple processes according to a time-sharing scheduler. Newly created processes can share selective parts of their execution environment with their parent processes, allowing multithreaded programming. Interprocess (sic) communication is supported by both System V mechanisms—message queues, semaphores, and shared memory—and BSD’s socket interface. Multiple networking protocols can be accessed simultaneously through the socket interface.

The memory-management system uses page sharing and copy-on-write to minimize the duplication of data shared by different processes. Pages are loaded on demand when they are first referenced and are paged back out to backing store according to an LFU algorithm if physical memory needs to be reclaimed.

To the user, the file system appears as a hierarchical directory tree that obeys UNIX semantics. Internally, Linux uses an abstraction layer to manage multiple file systems. Device-oriented, networked, and virtual file systems are supported. Device-oriented file systems access disk storage through a page cache that is unified with the virtual memory system.

1. **Storage Management**:

The recommended two server setup is a storage-area network (SAN). It is “a private network (using storage protocols rather than networking protocols) connecting servers and storage units” (“Operating Systems Concepts” 2011). This architecture will scale well with the ability to support multiple servers. SANs also have the security benefits of being a private network. This is the recommended set up for our Draw It or Lose It project.

Our storage servers will implement C-SCAN scheduling because, due to the timed structure of the game and the nature of multi-platform applications, uniform wait time is desirable for Draw It or Lose It. “Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip” (“Operating Systems Concepts” 2011).

Our storage servers will be a disk drive, implemented in a specific RAID structure. RAID (redundant arrays of independent disks) refers a collection of disk-organization techniques, separated into different levels. Our disk drive will implement RAID level 2. Level 2 implements memory-style error-correcting-code organization, which allows single bit errors to be caught and fixed. Level 2 does all this while avoiding the significantly lower performance of RAID levels 3 and higher.

1. **Memory Management**:

Memory management is important to our Draw It or Lose It server because it is how all the client requests are processed. Linux has an efficient memory management system that we will take advantage of with well-structured code and files. Below is a summary of Linux’s memory management system.

Our Linux server’s “memory-management system uses page sharing and copy-on-write to minimize the duplication of data shared by different processes” (“Operating Systems Concepts” 2011). Pages load on first reference “and are paged back out to backing store according to an LFU algorithm if physical memory needs to be reclaimed” (“Operating Systems Concepts” 2011). Memory management in Linux breaks down into two parts: allocating and freeing physical memory, and mapping address spaces of running processes that will not fit into physical memory (this is often referred to as virtual memory).

For physical memory, the kernel keeps a list of free pages, which are given out based on requests. A page allocator uses a “buddy system” to combine free memory into larger groups and breaks these groups into smaller regions based on the size of requests. This ensures that physical memory is allocated and freed efficiently.

“The Linux virtual memory system is responsible for maintaining the address space visible to each process. It creates pages of virtual memory on demand and manages loading those pages from disk and swapping them back out to disk as required” (“Operating Systems Concepts” 2011). Linux virtual memory implements paging exclusively, over the slower technique of swapping.

1. **Distributed Systems and Networks**:

Because Draw It or Lose It is a RESTful web application, it is a distributed system by design. The REST API ensures that our application will run on any device that can run a modern internet browser. With our client server architecture, all communication between clients takes place within the server. This means our main concern to support communication between various platforms is to ensure stable network connections and ensuring smaller mobile devices can keep up with the much faster desktop users. One of the advantages of distributed systems is load sharing. By leveraging load sharing for specific user operating systems, we can speedup performance on less powerful mobile devices. We should implement smaller sized images to be utilized by clients on mobile devices. We could also consider sever-side rendering of images or moving additional processes to the server if a client is on a mobile device if performance has been an issue for Draw It or Lose It as an Android application. To handle connection issues, we will want to implement dynamic routing, and packet switching to best accommodate various platforms.

1. **Security**:

Our system architecture and Linux server have security features built into it. Earlier in the document we mentioned that our SAN structure will encapsulate many of our project’s files. The Linux OS comes with authentication and access control security mechanisms. Linux adds random “salt” values to password files, so that the original password “cannot be deduced from the password file except by trial and error” (“Operating Systems Concepts” 2011). Linux also has robust access control, given the root user is correctly protected, which is standard operating procedure for Linux Admins.

Additional security will be necessary to keep our project safe. We will incorporate a robust security policy implementing intrusion detection, virus protection, and application firewalls to proactively combat malware. Auditing, accounting, and logging of processes, performance, and messaging is an essential reactive part of our security policy. We must also ensure proper user access and authorization. Any user data that is stored must also be encrypted to protect our customers.

## 

## Sources

*Browser Market Share Worldwide | Statcounter Global Stats*. (2022). StatCounter Global Stats. <https://gs.statcounter.com/browser-market-share>

Choudhury, A. (2020, September 25). *6 Best IDEs Developers Can Use For Swift*. Analytics India Magazine. https://analyticsindiamag.com/6-best-ides-developers-can-use-for-swift/

Clark, J. (2021, September 22). *Top 10 IDEs for React Native*. Back4App Blog. https://blog.back4app.com/best-ide-for-react-native-app-development/

Doyle, M. (2019, March 20). *10 Tips for Building Cross-browser Websites*. Matt Doyle | Elated Communications. <https://www.elated.com/cross-browser-website-tips/>

*Eclipse Web Developer Tools*. (2018). Eclipse Plugins, Bundles and Products - Eclipse Marketplace. https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0

GeeksforGeeks. (2020, October 3). *Top 10 Most Popular Java Frameworks for Web Development*. https://www.geeksforgeeks.org/top-10-most-popular-java-frameworks-for-web-development/

*If MacOS is claimed to be so good, why is it rarely if ever used for servers?* (2019). Quora. https://www.quora.com/If-MacOS-is-claimed-to-be-so-good-why-is-it-rarely-if-ever-used-for-servers

January 30, A. M. (2019). *8 Essential Tips for Building a Cross-Browser Compatible Website*. SmartBear.Com. https://smartbear.com/blog/cross-platform-website-development/

*Linux vs. Windows: a comparison of the best web server solutions*. (2021, January 14). IONOS Digitalguide. <https://www.ionos.com/digitalguide/server/know-how/linux-vs-windows-the-big-server-check/>

Olanrewaju, O. (2022, February 5). *Top 15 Swift Frameworks*. Dunebook. <https://www.dunebook.com/swift-frameworks/>

*Operating System Concepts*. (2011). [E-book]. Wiley