

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

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 8**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/27/24 | Emily Haymond | Added the Executive Summary, Design Constraints, and Domain Model sections |
| 1.0 | 8/13/2024 | Emily Haymond | Completed the table to compare Server, Client, and Developer Tools considerations for macOS, Windows, Linux, and mobile platforms |
| 1.0 | 8/18/2024 | Emily Haymond | Submitted the recommendation for the system’s operating platform, architecture, and storage management |

**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 would like to create a game called Draw It or Lose It, based on the tv game Win, Lose, or Draw, where a series of images are rendered to a screen and teams and/or players try to guess the puzzle before the time runs out. We will create a web-based version of this game, which will support multiple users and teams playing in a single instance of the game. There will be unique team and player identifiers to avoid conflicts. A single game instance will make sure there is only one game in memory at any given time, preventing duplicate sessions.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* The system must be accessible on both mobile and desktop platforms, including various operating systems such as Windows, macOS, and Android. The system must be able adapt to different screen sizes and input methods.
* The system must also be able to work on multiple different browsers, such as Chrome and Safari, to ensure consistent behavior and functionality for all players.
* There should only be one instance of the game in memory at any given time. We will use a Singleton pattern to enforce this constraint, preventing duplicate game sessions.
* The system must also have high performance, so that when images are rendered to the screen it is quick and engaging for all users.

## [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)

Each Game, Team, and Player are a kind of entity, and each will have their own id and name that they inherit from the parent Entity class. This is an example of Inheritance in OOP, each child class will inherit the name and id attributes of the parent Entity class. Games use the Game Service, where a list of Games are stored, and Games can be added or retrieved, there can be many Games in the Game Service. Games can be retrieved from their id or their name, a count of total Games is stored, and there are methods to show the next available player id and team id so that it ensures each team and player are unique. This shows an association between the Game and GameService classes, as the Game Team and Player uses the GameService. Each game will also have a list of Teams, and the ability to add new Teams. Each Team has a list of Players, and ability to add new Players. The Entity id and name fields, as well as the lists of teams and players are all private members of their classes, showing encapsulation, the ability to hide their attributes from outside sources so that they are protected and not modified outside their scope. The Game Service also has many private attributes, the list of games, the next game, player, and team ids, and an instance of itself, that also show encapsulation. The use of the Game Service inside of itself shows the Singleton pattern used to ensure there is only one instance of a game at any given time in memory, fulfilling the software requirements.

**"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** | macOS is generally not used for large-scale server deployment, but does do well with mac and ios ecosystems. Integration with Apple devices is superb, and is very user friendly for people already familiar with macOS. There is less support for enterprise-level roles, and fewer cloud hosting options for macOS. The licensing comes free with any Mac purchase, but the Server app is $20. | Linux is open-source and highly customizable, making it very stable and flexible with lots of community support. Linux has strong security features, extensive support for cloud services, and many tools for server management. The OS does require more technical expertise to set up and manage. There are free versions such as Ubuntu or CentOS, or a Red Hat Enterprise Linux starting at $349/year and will vary depending on support and features. | Windows has server editions specifically designed for enterprise-level deployment. It is easy to use for those already familiar with Windows, and has strong integration with Microsoft services and compatibility with many enterprise software. Although, because of Windows popularity, it is often a bigger security risk. It also has a higher recourse usage than Linux, and is less customizable and flexible. Windows Server Standard starts licensing around $1069 per license, and Windows Server Datacenter starts licensing around $6155 per license. | Mobile operating systems are generally not used for server deployment. There is excellent integration with mobile apps and strong support for mobile-specific services. Although, it is not suitable for server deployment and it requires sticking to platform-specific guidelines. There are no server licensing costs, as it is not suitable to deploy. Web developers can work across all platforms, but there may be a need for a specialized ios or Android developer or team. |
| **Client Side** | Mac uses Safari, which fully supports responsive web design, and allow various screen sizes. Webkit will need to be used to test rendering. Frameworks such as React can be used that support all desktop browsers. Backend communication is smooth, and will interact with REST API services. Can support more complex UI with a mac’s resources. Security is mature, using features such as HTTPS. Safari has it’s own debugging and developer tools, and automated testing is available. Will need to test with latest browser versions, ensuring compatibility. | Linux can use modern browsers, such as Chrome, Firefox, or Edge, and support responsive web design and various screen sizes. With different rendering engines, testing will need to be done across all browsers. Frameworks such as React can be used that support all desktop browsers. Backend communication is smooth, and will interact with REST API services. Can support more complex UI. Security is mature, using features such as HTTPS. Each modern browsers has it’s own debugging and developer tools, and automated testing is available. Each browsers will need to be tested with latest version, ensuring compatibility. | Windows users generally use Chrome or Edge, and each support responsive web design and various screen sizes. With different rendering engines, testing will need to be done across all browsers. Frameworks such as React can be used that support all desktop browsers. Backend communication is smooth, and will interact with REST API services. Can support more complex UI. Security is mature, using features such as HTTPS. Each modern browser has it’s own debugging and developer tools, and automated testing is available. Each browser will need to be tested with latest version, ensuring compatibility. | Mobile usually have smaller screens, so responsive design is important for the various screen sizes. Design also needs to support rotation, with portrait and landscape modes. Testing will need to be done on both Safari for ios and Chrome for android to ensure consistent behavior. Frameworks such as React Native or Flutter can be used for both ios and android, and performance needs to be more closely considered as mobile devices don’t have as much resources available. API will need to be optimized for the different network connections of mobile devices, such as ones with lower bandwidth and higher latency. Offer offline support for when network is poor or unavailable. Battery life needs to be considered by optimizing. Simpler UI is used, with the addition of touch interaction as a primary design. Security features are stricter and more unique, and will need to use platform-specific features, such as biometrics. Will need more thorough testing and debugging, using emulators, physical devices, and additional tools such as Xcode for ios and Android Studio for android. |
| **Development Tools** | macOS uses Swift and Objective-C for mac and ios development, but also uses JavaScript, HTML, and CSS for cross-platform web development. Xcode is the official IDE for mac and ios, and can also use Visual Studio Code – which are both free, with the addition of a $99/year Apple Developer Program membership to deploy to the App Store. | Linux uses Python for scripting, web development, and backend services; C/C++ for system-level programming and performance-critical applications; JavaScript, HTML, and CSS for cross-platform web development; and Java for server-side applications. Eclipse is and IDE used for Java, C++, and Python; PyCharm is used for Python; Visual Studio Code; and Vim/Emacs are text editors used for their flexibility and customization. The IDEs and text editor are all free, with the exception of a PyCharm Professional version. | Windows generally uses C# for desktop applications and backends; C++ for high-performance applications and system-level programming; and JavaScript, HTML, and CSS for cross-platform web development. Visual Studio and the simpler Visual Studio Code are IDEs used for development in Windows. Visual Studio has a free community editon, and also a Professional and Enterprise editions that require licenses; and Visual Studio Code is free. | Swift is used for ios, Kotlin/Java is used for Android, and JavaScript, HTML, and CSS are used for mobile web development. Xcode is the IDE used for ios, and Android Studio is used for Android. React Native or Flutter is used for cross-platform mobile development, and those frameworks use Visual Studio Code for the IDE. Xcode is free, with the addition of a $99/year Apple Developer membership to deploy to the App Store. Android Studio and React Native/Flutter are both free. |

## 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**:

The recommended operating platform is Linux due to its many benefits. Because Linux is open-source, you can customize the system to meet your needs and there are no licensing fees, which makes it less expensive. Its strong performance and stability are essential for gaming and ensuring a high server uptime. Along with strong security features, Linux has a vibrant community that quickly fixes security flaws. The platform's scalability allows it to manage operations on both small and large scales, and its compatibility with different hardware and cloud environments further increases its flexibility. Additionally, Linux offers an excellent development platform.

1. **Operating Systems Architectures**:

Linux is superior when comparing the architectural support with its flexibility and community-driven solutions and support**.** It supports some of the most commonly used architectures, such as x86\_64 and ARM64, as well as many other popular options for legacy systems, workstations, and mobile devices. The recommendation for the architecture to use with this Linux system is the x86\_64 for its compatibility, high performance, and mature software support.

1. **Storage Management**:

Storage technologies can offer many options for storage management, data integrity, scalability, and advanced features. LVS is one of those technologies that has flexible volume management and partitioning. ZFS and Btrfs are two others that offer more advanced features, and Ceph and GlusterFS are for distributed and scalable storage. For this project, the recommendation is for ZFS for its advanced features like snapshots and data integrity checks to ensure a scalable and reliable game application. ZFS also offers both a file system and volume manager, which will be useful as the application grows and there will be no need for a separate volume manager component.

1. **Memory Management**:

For this project, efficient memory management is essential to ensure smooth performance across all platforms. On the server side, the choice of ZFS as the storage technology provides robust memory management features, including advanced caching mechanisms that optimize data retrieval and reduce the load on system memory. ZFS's support for dynamic memory allocation and its built-in data integrity checks help prevent memory-related issues. On the client side, the application will be optimized to handle varying memory capacities across mobile and desktop platforms, ensuring a responsive experience by minimizing memory usage and preventing leaks.

1. **Distributed Systems and Networks**:

Using the recommended architecture, implementing a distributed system is key to scaling the application and ensuring high availability. Using ZFS, which integrates well with distributed storage solutions like Ceph, allows efficient management and replication of data across multiple nodes, ensuring redundancy and fault tolerance. The network configuration will be optimized for low latency and high throughput, enabling smooth communication between distributed nodes. This setup ensures that the system can handle a large number of concurrent users while maintaining performance, with the ability to expand by adding more servers to the network as needed.

1. **Security**:

To ensure the security of the application, we will leverage Linux's robust built-in security features, including SELinux (Security-Enhanced Linux) for mandatory access control and AppArmor for application-level protection. These tools will be configured to enforce strict policies, limiting access to sensitive files and processes. Additionally, the project will implement data encryption both at rest and in transit, using industry-standard protocols such as disk encryption and secure communications over the network. By integrating Linux’s native firewall, we will further secure the system by defining specific rules that limit inbound and outbound traffic. Regular security audits and updates will be conducted to ensure ongoing protection against vulnerabilities.