

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/19/2024 | Vy Huynh | Address the client’s problem and develop a design constraint |
| 2.0 | 02/03/2024 | Vy Huynh | Broke down different operating system |

**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](#_heading=h.35nkun2)

The client, The Gaming Room, currently has a game called “Draw It or Lose It” which is an Android-based game. They want to develop a web-based game that serves multiple platforms, attracting many users and increasing profit.

## Requirements

***Business***

* Develop the game within the budget
* Increase revenue
* Attract a wide range and variety of users
* Continuous update the game to prevent bugs
* Collab with influencers to help the game reach more people

***Technical***

* Multiple teams and multiple players can participate in the game
* Unique identifier for team name and player name
* Maintain a single instance of the game
* Compatible with many different platforms
* Allow users to play with friends
* Allow users to sync their profile across different platform

## [Design Constraints](#_heading=h.1ksv4uv)

***Compatible Issues***

* The game must be able to run across many different browsers and devices
* This means there will need to be many cross-platform tests to make sure every user has a satisfied experience
* Some browsers/platforms will need to be modified for the game to function properly

***Latency***

* The various internet speeds could potentially cause lag and delay of the game
* Develop a latency migration technique to prevent a complete disadvantage for users with slower Wi-Fi
* Optimize the game for lower-end devices to avoid minimal delay and latency

***Scalability***

* The need to be able to handle a growing base of users
* Implement things like load balancing to handle heavy traffic
* Make sure the database is able to handle a large amount of users

***Updates/ New Events***

* New content and events should be regularly updated
* Adding new events keeps the user base engagement high
* Ensure that new updates are compatible with existing platforms
* Make sure that every update is thoroughly tested and there are no bugs

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

**"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.**

The ProgramDriver class ‘uses’ the SingletonTester class, which means that the ProgramDriver will create an instance of ‘SingletonTester’ and use it to test the singleton pattern. These two classes represent OOP through association, as ProgramDriver is associated with SingletonTester by creating an instance of ‘SingletonTester.’

The ‘GameService’ class serves as a singleton for managing functionalities in the game application, the ‘getInstance()’ retrieves the instance of ‘GameService’ unless GameService is null, this is when it will create a new instance. The private constructor within ‘GameService’ ensures that no class can be instantiated outside of the class. ‘GameService’ represents the principal encapsulation through its private method. The private games, nextGameId, nextPlayerID, nextTeamID, and service means that these data will not be accessible outside the class.

After creating an instance ‘GameService’ is now able to use ‘addGame’ to start a new game, through addGame() a new Game object is being created before adding to the list Game. Then the ‘Game’ class can call a new method called ‘addTeam’ which can add teams into a list called ‘Team.’ Finally, the class Team can call an addPlayer() method that will add a player to a list called ‘Player.’ The (0..\*) relationship between GameService to Game, Game to Team, and Team to Player represents a composition relationship, which is a part of OOP. Each GameService consists of multiple ‘Game’, each ‘Game’ consists of multiple ‘Team’, and each ‘Team’ consists of multiple ‘Players.’

Entity class represents the base class in the application, The three classes ‘Game’, ‘Team’, and ‘Player’ inherit ‘id’ and ‘name’ from the entity class. By inheriting the attribute from the ‘Entity’ class could help promote code reusability and maintain consistency across the application program. The ‘Entity’ class represents the OOP principle or inheritance since all three classes ‘Game,’’ Team,’ and ‘Player’, inherit ‘id’ and ‘name’ from the ‘Entity’ class.

## [Evaluation](#_heading=h.z337ya)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS is an operating system that Apple develops. The OS is known for its simplicity and ease of use. One of the advantages of using macOS is its compatibility with Unix tools; macOS also offers seamless integration with popular tools like Xcode and Docker. Another pro is that macOS is known for its in-house security feature, which can help protect the application and its data. One disadvantage that will come with macOS is the cost of Apple products, which is a lot more expensive than other products. Another disadvantage is limited support for web hosting software on macOS. Regarding deployment, macOS has already pre-installed Apache, which can be used for web sharing; another alternative option is to install and use NGINX. The potential cost of a macOS will include hardware; another cost could be if the client wants to buy support for their servers, like updates and troubleshooting, which could be an additional cost. | The most popular aspect of Linux is that it’s open-source, which means it can be modified based on the user’s needs. Another advantage of Linux is its scalability; a Linux-based server can easily be scaled to accommodate an increasing user base. One of the biggest disadvantages of Linux is that there is learning. Linux is not a common operating system, unlike Windows and macOS, which many people grew up using; therefore, there will be a learning curve when it comes to using Linux. Another potential disadvantage of Linux is the lack of support for Linux compared to Windows. Regarding potential licensing costs, Linux is open-source; therefore, there will typically be no cost for the server. However, depending on the client, there might be costs associated with support for the server. In terms of server deployment, there are many options for Linux; the most popular ones are Debian and Ubuntu, which are both free to use. | Windows being the most popular desktop operating system. Microsoft develops it, the operating system that comes with most off-the-shelf laptops. Windows OS offers a wide range of support for many different hardware. One advantage of Windows is that it comes with Microsoft features like Framework, which makes it easier to host servers. Another pro of Windows is that it has excellent commercial support, and there are many tools for managing web applications. Lastly, since Windows is the most popular desktop OS, there won’t be as much of a learning curve as Linux or Mac. A negative of Windows is the expensive licensing cost; depending on the size of the servers, the cost overall could be extremely high. Finally, unlike Linux, Windows is not an open source therefore, there will be less Flexibility and freedom with customizing the server to tailor to specific needs. | The characteristic of mobile devices for hosting is that mobile devices have a touchscreen interface, unlike other OS, which has a mouse and keyboard. Another characteristic of mobile devices is that there are many devices that come in different shapes and sizes; some even fold. One advantage of mobile devices is that they’re portable and easily accessible; therefore, the web application can be built on the go. Mobile devices have many disadvantages, including limited screen size, performance capability, and not being meant to host servers. The biggest challenge for Mobile Device development is that most mobile devices use different OS (iOS, Android), which means there could potentially be compatibility issues. |
| **Client Side** | Regarding development tools, the famous Mac development tool Xcode is free to use, but depending on usage, there could be additional costs for third-party libraries. Ensuring compatibility across different macOS versions could include buying other devices, which could be expensive. Another cost is the maintenance cost to continuously keep the application functional and bug-free. Since macOS is its operating system, there will be a greater development time since everything must be made specifically for macOS. In terms of expertise, there need to be specific developers that specialize in macOS technology /language like Swift. | Regarding development costs, Linux tools are usually free; this includes Dieban and IDEs, which means development costs will be much cheaper on Linux than on Windows or macOS. However, to test compatibility around different versions/OS, there will need to be an investment in virtual machine software, which will cost money. Theoretically, it doesn't offer any support when using Debian as a server deployment. Therefore, third-party support is needed; the support will ensure that the server is up-to-date and that there are no problems with the server. In terms of the timeline for development, it will take around the same amount of time as developing a server on Windows or macOS. There will be platform-specific development and compatibility Testing maintenance testing and deployment. In terms of expertise, unlike macOS, which requires Swift, Linux uses C/ Python, which is a standard programming language, making it easier to find developers. | There will be costs for development tools like IDE and compatibility test costs, which are tests to ensure that the server is compatible with different versions like Windows 11, Windows 10, etc. Finally, there will be the cost of supporting and updating the server. The time it takes will be similar to any other platform, which will bring development time and compatibility testing time. For expertise, they will need to find people who have Windows development skills like NET frame and WIN API and experts who know about Windows deployment and packages. | The cost for mobile devices will include development tools, which are Xcode for iOS and Android Studio for Android. There will need to be testing where many devices are tested to ensure compatibility between different operating systems and platforms. It will take a lot more time than other platforms because Mobile devices require at least two different platforms (iOS, Android). After the development, there will be time for testing and ensuring compatibility, regardless of the device or screen size. For expertise, there will need to be experts who specialize in Swift for iOS and Java for Android. There will also be the need for UI designers to develop an application that can be seamless no matter the screen size or resolution. |
| **Development Tools** | In terms of development tools, the most important is the language, which is Swift. Swift is the primary language for all macOS development. Another language famous for macOS is Objective-C, an extension of C. For IDEs, the official IDE for development is Xcode; however, depending on the preference, there are also IDEs like IntelliJ or Visual Studio Code, which can code in Swift; however, both Visual Studio Code and IntelliJ will cost licensing fees for employees. Another essential tool is a repository, where the team can store their code and collaborate on the project. | When developing in Linux, the programming language will likely be C, C++, or Python, so the IDE will likely be either Visual Studio Code or Clion, which has free and paid versions. Another tool will be CMake, which is used for building projects on Linux. Lastly, there will be a Git, which the team can use to share code and collaborate. | The most popular programming languages for Windows servers are C, C++, and C#. Therefore, the IDE will likely be either Visual Studio Code or a JetBrains IDE. Other tools needed for Windows development will be an SDK, which provides tools for Windows applications, and a Framework and API, which help developers build Windows applications. | For programming languages, it will be Swift for iOS and Java for Android. The most popular iOS IDE is Xcode, which is included with Apple Mac purchases. For Android, it will be Android Studio, which is free to use. Another tool that is needed is React Native. React Native is a framework that allows developers to write one set of code and deploy it on iOS and Android, potentially reducing development time. The last application is Firebase. Firebase is a popular tool that provides backend infrastructure for web applications; it has many useful features like a database and cloud storage. |

## 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 best operating system for the web application will be Linux. The best thing about Linux is that it’s open-source, which means that it’s extremely flexible and will allow for future growth. Linux is also cost-effective when compared to other operating systems, this can help with lowering cost. Furthermore, Linux is compatible with a wide range of platforms, this will be much better than macOS or Windows.
2. **Operating Systems Architectures**: At the heart of a Linux operating system is the kernel, the kernel manages the communication between the software and hardware of the OS. When using Linux it’s important to choose a distribution platform, and the most popular ones are Debian or Ubuntu, both are different in their ways with different packages. The choices between Debian and Ubuntu will come down to preferences and application requirements. Linux shell is a CLI, like a terminal on macOS or PowerShell on Windows, it’s a way for users to execute and run commands. It’s a way to interact with the OS and perform many different tasks.
3. **Storage Management**: For the game, the file system that should be used is ext4, it’s a popular file system that many applications use for Linux due to its reliability, it’s also the default file system for many distribution systems. Additionally, use LVM (Logical Volume Manager) for volume management along with ext4. LVM is extremely flexible, which makes it easier to manage storage and allocate resources. LVM allows for partitioning, which will make expanding and shrinking storage much easier. Furthermore, migration of data with LVM is straightforward as LVM enables mobility and movement between different data. The combination of ext4 and LVM ensures efficient resource allocation and proper storage management that aligns with the game.
4. **Memory Management**: For memory management on Linux, one option could be compression. This means compressing items/pages to reduce and save memory, which improves efficiency. The way to compress memory is through zRam/zSwap. Another option is through the Linux virtual memory. Linux virtual memory can manage the space for different aspects of the program like code and data. The division is good for the organization of the memory, and it can also perform something called a memory balloon, which means it can dynamically adjust the memory space based on the current demands. Memory management can help improve the overall gaming experience, by improving response times and preventing lags and buffers.
5. **Distributed Systems and Networks**: To communicate with many platforms, APIs can be used for this. APIs can be used for communication between the server and the client. Clients can make requests using the API, the API can then send the request to the server, the server then processes the request and returns it to the client. The server can have many types of APIs for many needs like creating a new game, fetching a profile, creating a new user, and more. To allow for APIs to happen, the application will need to be broken down into many smaller tasks, which means implementing something called Service Oriented Architecture. The basic concept of this is that the components of the game are broken down into smaller parts, like matching, login, leaderboard, and friendly, they are all developed as separate parts before combining as a part of the full applications. By breaking down these aspects of the game, they can communicate with each other through APIs and HTTP. Breaking down an application into smaller parts can help with managing the applications across various platforms, it also allows for better flexibility and future expansion.
6. **Security**: For security for game management it’s always important to have two-step authentication, two-step authorization helps verify that it’s the actual client that is logging in to the game, not a hacker. For security purposes, it’s important to encrypt all sensitive information and data, this prevents potential breaches and data leaks when sending information across different platforms. To make sure there are no potential data vulnerabilities that can be exploited, it’s important to regularly update the applications and the operating system, this can help protect against potential exploits. Finally, for security and safety purposes, it’s essential to monitor and log all activities and changes to the system, this can help quickly identify the problem when there is a leak or hack. Monitoring can also help detect unusual activity, which could potentially stop the attack before it happens.