

<Name-of-Software-Application>

# **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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <02/3/23> | <Anthony Aleman > | Evaluation of Linux, Mac, Windows to better inform the client on the platform and their utility in a distributed environment |

**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 client The Gaming Room is seeking to develop a web based game which targets multiple platforms. By targeting multiple platforms The Gaming Room is actively looking to expand its customer base a considerable amount as the current base game which will act as inspiration for the current multiplatform version of “Draw It or Lose It”, which is only made for the android platform.* *The purpose of the game is for multiple teams going four rounds at a minute each, with teams consisting of multiple players. When a picture is pulled from a library of images one team guesses till time runs out. If not answered each opposing team member gets to answer till 15 seconds runs out.*

## Requirements

Software

* *A game will have the ability to have one or more teams involved.*
* *Each team will have multiple players assigned to it.*
* *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
* *Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.*

*Business*

* *Expansion of customer base by targeting multiple platforms through a web based application*

## [Design Constraints](#_2et92p0)

The business and software requirements affect the design constraints of the application as to ensure the web based application work seamlessly through the main platforms of Windows, Linux, and Apple, meaning a system must be established to handle the infrastructure of multiple code bases which all target the multiple platforms. Specifically, when it comes to Apple the programming language Swift and IDE xcode must be used in development of iOS applications.

## [Domain Model](#_8h2ehzxfam4o)

In this domain model, Entity is a superclass that is inherited by the Game, Team and Player classes. These three classes share common references to attributes like “id” and “name” of which help to implement the iterator design pattern. Team and Player have a "has a" relationship with Entity, while Game has a Team and GameService has Games.

**"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** | * High performance and stability * Excellent security features and minimal vulnerability to viruses and forms of malware * macOS is a unix-based operating system which has a considerable developer community which can act as tools and resources for web development * less preferable for hosting services | * Many similar characteristics as Mac but more cost friendly and open source * High security as there is a large developer community so security flaws are caught before they become an issue making linux a preferred choice | * Os developed and maintained by Microsoft a very big and profitable business * Widely used for personal machines and servers * Easy to find developers with Windows expertise due to wide use * Compatible with a large range of hardware and software | * They have touchscreens, which can provide a more intuitive and user-friendly interface for web applications. * Many mobile devices have built-in sensors, such as GPS and cameras, that can be utilized by web applications to provide additional functionality. * Mobile devices have smaller screens and less processing power than desktop machines * Limited storage |
| **Client Side** | * May require extra time and developers with specialized expertise to ensure compatibility with multiple versions of MacOS * May require more developers to create the necessary development resources | * May require extra time and developers with specialized expertise to ensure compatibility with multiple versions of Linux and different Linux distributions * Due to the large amount of distributions, a large amount of experience is needed | • Given wide use of Windows, competent developers will be easier to find | Provides flexibility to clients or even developers to see updates at any place. Slightly more difficult to implement than other devices. |
| **Development Tools** | For Macs, the programming language Swift would be utilized in conjunction with xCode. All languages can run on Macs but for making a native application Swift, Apples own language is recommended as it was designed to run seamlessly on Macs | Linux can work with visual studio, eclipse, along with any general purpose programming language | Easier to use than Linux in part de to the large amount of resources, but can run the same as it. So visual studio, eclipse to name a few of the IDE but also any general purpose programming languages can run with windows | You can create countless apps using android and swift. Androids use Android studio and Apple uses xCode with Swift |

**With some further research, our team documented our findings on the viability of each operating system**

| **Operating Platform** | **Server Side** | **Characteristics** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- |
| **Mobile** | **Not suitable** | **Designed for small screens and touch based surfaces`** | **Widely used for its user friendly interface** | **Limited processing and storage power, and limited compatibility with desktop software** |
| **Mac** | Possible with a proper setup | User friendly, designed for use by many different professions | High quality hardware and software, highly integrated to the Apple ecosystem | Higher costs, less customization, less compatibility outside of the Apple ecosystem |
| **Linux** | Server style setup is suitable for the Linux operating system | Open source, customizable, scalable, secure | With an open source ecosystem, it is supported by a large community of developers that use a wide range of general programming languages, and has robust security features | Steeper learning curve for newer developers and users, limited desktop app compatibility |
| **Windows** | Server Style possible | Widely used operating system, and compatible with a range of hardware and software | Large base of users with a wide range of compatibility | Malware and security issues as well as a higher cost for the enterprise version |

**Based on the requirements a Linux or windows operating system would be most beneficial for the server side as it would be suitable for the most amount of users as well as being cost effective in comparison to the choice of Mac. Also those two offer the best compatibility as Mac can have issues with non Apple software and hardware**

| **Operating Platform** | **client Side** | **Characteristics** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- |
| **Mobile android** | **HMTL interface compatible** | **Widely used and customizable, opensource** | **Wide range of web browser support and a large developer community with a low cost of ownership** | **Fragmented browser ecosystem, limited compatibility with certain software and security issues** |
| **Mobile (ios)** | **Suitable for HTML interface** | **Customizable, stable and secure** | **Wide range of web browser support** | **Fragmented browser ecosystem and limited compatibility with non-Apple software and hardware. Also higher costs** |
| **Mac** | HTML suitable | User friendly, designed for use by many different professions | High quality hardware and software, highly integrated to the Apple ecosystem | Fragmented web browser ecosystem, limited compatibility with non-apple ecosystem |
| **Linux** | HTML interface suitable | Open source, customizable, scalable, secure | Wide range of web browsers supported with a large developer community | Fragmented web browser ecosystem, limited compatibility with certain software |
| **Windows** | HTML interface suitable | Widely used operating system, and compatible with a range of hardware and software | Large base of users with a wide range of compatibility | Fragmented web browser ecosystem with issues concerning malware and security |

**In the discussion of having multiple clients, it would be necessary to create a wide range to offer the necessary support to a wider customer base. Given our android only application, we can and should make this available through any channel that is HTML interface suitable. It will be costly in time and labor as isn’t just development of the clients that need to be of utmost importance but the maintainability of their application.**

**Finally as for development tools There can be a wide range of tools used but as for IDE and programming languages, Visual studio can be used due to its wide community support and any choice of programming language that is compatible for hybrid development can be used. Though for any apple development, xCode and Swift must be used**

## 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**: Based on the requirements of expanding Draw It or Lose It to other computing environments, I recommend that The Gaming Room use a Linux-based server operating system as the primary operating platform for the game. Linux provides a stable and reliable platform for hosting web-based applications like Draw It or Lose It. Additionally, Linux has a wide range of software development tools and libraries available, which will make it easier to develop and deploy the game on different platforms.
2. **Operating Systems Architectures**: The Linux operating system is based on a modular architecture that is designed to be highly customizable and scalable. The Linux kernel provides the basic system functions, such as memory management, process management, and file system management. The kernel is complemented by a range of system utilities and libraries that provide additional functionality to the operating system.
3. **Storage Management**: an appropriate storage management system to be used with Linux is the Logical Volume Manager (LVM). LVM provides a flexible and scalable solution for managing disk storage on Linux systems. It allows disk partitions to be grouped together into logical volumes, which can then be resized, moved, or even mirrored without affecting the underlying data. LVM also provides support for snapshots, which can be used to create point-in-time copies of logical volumes for backup or testing purposes. Overall, LVM provides a robust and reliable solution for managing storage on Linux systems, making it an ideal choice for The Gaming Room's Draw It or Lose It game.
4. **Memory Management**: Firstly, Linux uses virtual memory to manage the allocation of physical memory to processes. The virtual memory system allows processes to use more memory than is physically available in the system. Each process is allocated a virtual address space, which is mapped to a physical address space in the system's memory. This allows multiple processes to share the same physical memory without interfering with each other.

Secondly, Linux provides support for memory swapping, which allows the operating system to move data between physical memory and disk storage as needed. When the physical memory becomes full, the least frequently used data is moved to the disk to free up space in the physical memory. This is known as swapping out. When the data is needed again, it is swapped back into the physical memory from the disk. This helps to ensure that the most important data is kept in the physical memory for faster access.

Thirdly, Linux provides support for memory protection, which allows processes to be isolated from each other. Each process is allocated a separate address space, which is protected from other processes. This ensures that one process cannot interfere with the memory of another process, which helps to improve system stability and security.

1. **Distributed Systems and Networks**: To allow Draw It or Lose It to communicate between various platforms, a distributed software architecture can be used. The architecture consists of multiple components running on different devices that communicate with each other over a network.

One approach to implementing this architecture is to use a client-server model, where a central server manages the game state and communicates with clients on different platforms. The server can be implemented using a variety of technologies such as Node.js, Java, or Python.

To enable communication between the server and clients, a network protocol such as TCP/IP or HTTP can be used. This protocol can be implemented using standard libraries in various programming languages.

To handle connectivity and outages, the client software can be designed to automatically reconnect to the server if the connection is lost. The server can also be designed to handle client disconnections and reconnects gracefully, ensuring that game state is preserved and consistent across all clients.

In terms of dependencies between the components, it's important to ensure that the network infrastructure is reliable and secure. The network should be designed to handle potential outages, such as network congestion or server downtime. In addition, security measures such as encryption and authentication should be implemented to protect user data and prevent unauthorized access to the system.

Overall, a distributed software architecture provides a scalable and flexible solution for implementing Draw It or Lose It across various platforms, allowing for efficient communication between clients and the central server over a secure network.

1. **Security**: Firstly, the recommended operating platform, Linux, has several built-in security capabilities that can be leveraged to protect user information. For example, Linux provides support for file system encryption using tools such as dm-crypt or LUKS, which can encrypt sensitive user data stored on the system's hard drive.

Secondly, the Draw It or Lose It game can be designed to use secure communication protocols, such as SSL/TLS, when communicating over the network. This helps to ensure that user data is transmitted securely and encrypted, preventing eavesdropping or tampering with the data in transit.

Thirdly, access controls can be implemented to ensure that only authorized users have access to the system and its data. This can be achieved through the use of user accounts and permissions, which can be managed using Linux's built-in access control tools such as the sudo command or Access Control Lists (ACLs).

Finally, regular system updates and patching should be performed to address any security vulnerabilities that may be present in the system. Linux provides a robust update and patching mechanism through its package manager system, which can be used to ensure that the system is up-to-date with the latest security patches and updates.

Overall, by leveraging the security capabilities of Linux and implementing secure communication protocols and access controls, the Draw It or Lose It game can be designed to protect user information on and between various platforms, ensuring that user data is encrypted, secure, and protected against unauthorized access.