

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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/18/2023 | Richard Backscheider | Initial prototype supporting customer requirements *The Gaming Room* for developing a web based game called “Draw It or Lose It”. |

**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 Game Room, has hired Creative Technology Solutions (CTS), where I work as a Technology Consultant, to develop a web-based game that serves multiple platforms based on their current game, “Draw it or Lose it”. Currently the application is only available in an Android mobile operating system. The client hoping to facilitate the development of the web-based version of the gamming app. has asked CTS to streamline the development. As the Technology Consultant I will begin developing the game application, working to address The Game Room’s software requirements. The Gaming Room has requested that their game “Draw It or Lose It” will be capable of having one or more teams involved at a time, and that each team would consist of multiple players. “Draw It or Lose It” will also require a function in place to track whether a unique name is in-use or not. Finally, The Gaming Room has also requested that only one instance of the game can exist in memory at a time.

## Requirements

* 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.

## [Design Constraints](#_2et92p0)

* The company is facing a design constraint of a limited user-base due to the operating system only being available on Android.
* Network solutions will be required to achieve the customer’s required user-base load.
* Budget limitations may limit the scope of the customer’s project.

## [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 UML diagram is represented by a rectangle containing three sections stacked vertically. The top compartment represents the classes’ name. The second compartment is represented by the attributes (i.e. information, data, or properties) that belong to instances of a class (IBM). The third compartment represents the methods or services the class provides. The UML diagram consists of the ProgramDriver class, SingletonTester class, Entity class, GameService class, Game class, Team class, and Player class. The ProgramDriver class is the software component that lets the operating system and a device communicate with each other (Aviviano, n.d.). In the UML diagram the ProgramDriver contains the main method. The closed arrow with the word <<uses>> shows a direct association or strong relationship between classes (Gleek, 2021). The ProgramDriver has a direct association with the arrow pointing to the SingletonTester object class to check if an instance of GameService already exists. (Gleek, 2021). An Entity class is building blocks of the systems “an object wrapper for a database table” (IBM, n.d.). The Entity class can have attributes, operations, dependencies, inherits relations, and aggregations (IBM, n.d.). The open arrow of the object-oriented design references the inheritance or “the ability of one class (child class) to inherit the identical functionality of another class (super class), and then add new functionality of its own” (IBM, n.d.). In the UML diagram shown, the Entity class represents the superclass. The child classes are represented by GameService, Game, Team, and Player. The inheritance relationship from the child class to the superclass is represented by an open arrow, signifying that the superclass is a generalization of the child class (subclass) (Taylorial, n.d.). The solid line connecting GameService to Game is notated with “0…\*, denoting zero or more (many), indicating that every object in GameService is associated with zero or more objects in Game, and that every object in Game is associated with exactly one object in GameService (Orion Development Group, n.d.). The notation is the same for GameService to Team and GameService to Player. Each of these child classes (subclasses) are then connected to the superclass (Entity) with an open arrow.

**"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** | Designing an application for a Macintosh based server will be costly, but secure. The server will require a Mac to build to design. This will require a much higher budget than each other operating system option.  All Macintosh based servers have server-based deployment built into the operating system. Any required tools/applications will require downloading via an Apple ID through an Apple server to the server itself. Macintosh based server operating systems are free to use, minus the initial cost of the OS itself. MacOS server was discontinued on April 21st, 2022, and is no longer available to new customers. The software and its features are still available to existing customers. | Designing an application for a Linux based server will be very beneficial. The server will not require special hardware or software to design with. Linux is secure and affordable for all budgets.  Linux based servers have server-based deployment in multiple different distros of Linux. Deployments are handled through several different methods depending on the distro. For Redhat server and Ubuntu server all deployments are handled through the command-line interface. Other Linux server-based distros may use a graphical user interface to display the deployment information. Any Linux server-based distro won’t cost anything. For support from the developers, there may be a fee. | Designing an application for a Windows based server will be reliable, affordable, and not require any special hardware or software to design with. Windows is not as secure as the other platforms, but will be versatile for affordability, support and budgets.  Windows-based server deployment is built into the server operating system. Server-based deployments are handled through a graphical user-interface (GUI) control panel/web-portal and can also be handled through the command-line interface (CLI). The cost of a windows-based server operating system will cost around $1000, and support will cost an additional fee. Any additional clients will require a client access license (CAL) and will cost an additional fee. | Designing an application for an Android or IOS based server would not be optimum. The specifications of any Android or IOS device would be considerably different and under-powered for user-load base. Security would be higher on IOS, and less secure on Android. Hosting on a mobile device could lead to much a much more costly budget.  IOS-based servers are capable of receiving server-based deployments. There are two options by which an administrator can perform deployments, one by bringing their own device, or by the organization owning the devices. Each device will receive their required software through an Apple ID and by the user or administrator installing each application. The price for an IOS server-based operating system would only be for the hardware itself and support if needed. |
| **Client Side** | Software development for Macintosh would be much more costly than the other operating systems, as it will require you to own a Macintosh computer to code with. You would also need to consider support in case of parts failure which would require Apple Support. The designer would also need to understand Apple’s own programming language called SWIFT.  MAC OS compatibility testing provides confidence that the application will be completely functional across various browsers, hardware, and versions of the MAC OS operating environment. Testing for compatibility ensures complete functionality within the MAC operating environment. The many benefits of compatibility testing are detecting bugs early in the development process, ensuring successful software release, and testing to identify any security vulnerabilities. | Software development for Linux will be much more affordable than the other operating systems, due to Linux being open-source and the ability to be ran on all different types of hardware. The programmers will not require special knowledge of any unique language. Linux is also very secure and would likely not be susceptible to malicious code.  Linux compatibility testing provides confidence that the application will be completely functional across various Linux distribution and versions thereof. Testing for compatibility ensures complete functionality within the Linux operating environment. Compatibility testing includes scaling testing across a wide array of display resolutions. The many benefits of compatibility testing are detecting bugs early in the development process, ensuring successful software release, and testing to identify any security vulnerabilities. | Software development for Windows will be an affordable option. The cost will mainly be for a license to Windows, upgraded hardware, and virus protection software. No special knowledge of any unique languages is required. The biggest downside would be vulnerability to viruses.  Windows compatibility testing provides confidence that the application will be completely functional across various browsers, hardware, and versions of the Windows operating environment. Software compatibility testing checks the functionality across various web browsers and Windows versions. For example there are differences between Windows 7 and Windows 10 that could prevent 100% functionality for a given application. The many benefits of compatibility testing are detecting bugs early in the development process, ensuring successful software release, and testing to identify any security vulnerabilities. | Software development for IOS and Android will be fairly straightforward. Designers will need to understand development for mobile operating system environments. Mobile operating systems will be more susceptible to viruses due to a much wider audience of users than its static/laptop counterparts. IOS will require SWIFT development knowledge, whereas Android will not.  Mobile compatibility testing provides confidence that the mobile application will be completely functional across various mobile browsers, hardware, and operating environments (i.e. Android and IOS). Compatibility testing will include scaling testing (application/website scales to mobile and desktop display resolutions), mobile testing, operating system testing, and browser testing. The many benefits of compatibility testing are detecting bugs early in the development process, ensuring successful software release, and testing to identify any security vulnerabilities. |
| **Development Tools** | Macintosh will require Xcode as an integrated development environment, and SWIFT programming language knowledge in-order to write code on Xcode.  The technical requirements of using a Macintosh based server for your developer team will require higher cost upfront when buying the hardware. The team will also need to have an understanding of X-Code and the SWIFT programming language, as well as having a solid fundamental knowledge of MacOS. Programming won’t necessarily require multiple development teams, as programming for other languages is also possible. | Linux will require a preferred programming language to be installed. In this case, Python or Java is a good option. You would then need a capable integrated development environment, such as Eclipse, IntelliJ IDEA, or Visual Studio.  The technical requirements of using a Linux based server for your development team will not require a higher upfront cost. The main cost will be the hardware used and any support required to maintain the system. Multiple teams will not be required, as Linux supports all programming languages. The development team will require some understanding of the various Linux operating systems to get fully integrated with the environment. There are no licensing costs, unless wanting support for Linux. | Windows will require a preferred programming language to be installed. In this case, Python or Java is a good option. You would then need a capable integrated development environment, such as Eclipse, IntelliJ IDEA, or Visual Studio.  The technical requirements of using a Windows-based server for your development team will require some higher upfront costs. The server will require hardware parts, Windows Server operating system, and any support required to maintain it. Multiple teams will not be required, as Windows supports all programming languages. Any additional clients will require a client access license (CLA) and will cost an additional fee. | Android will require knowledge of the programming languages Java or Kotlin, whereas IOS will require knowledge of SWIFT. Android will require an integrated development environment such as Android Studio, Eclipse or Visual Studio. IOS will require Xcode to develop with.  The technical requirements of using an IOS or Android based server for your development team will require no higher costs beyond the price of the smartphone, and any support to maintain it. The development team will also need to have an understanding of X-Code and the SWIFT programming language, as well as having a solid fundamental knowledge of IOS. Android will not require any special programming knowledge. |

## 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**:

Ubuntu (Linux) Server would be the optimal choice to scale their web-based game. Linux is adaptable to current networking capabilities, scalable for future upgrades, secure, and budget friendly (Canonical Ltd., n.d.). Ubuntu server is configurable to the specific needs of a client, whether the client is using OpenStack cloud for resource management, Kubernetes cluster for deployment of all applications required, or routing multiple computers through nodes for rendering (Canonical Ltd., n.d.). Ubuntu server will enhance the capabilities necessary for the client to operate their needs precisely (Canonical Ltd., n.d.). Ubuntu server is affordable, and optimal to any requirement (Canonical Ltd., n.d.).

1. **Operating Systems Architectures**:

The Linux kernel is a secure and capable kernel to work from, it is “used by Linux-based operating systems and the interface between the hardware and the computer processes” (Sysdig, Inc., 2023). A Linux kernel is a special program that manages and executes the low-level functions of a computer (Sysdig, Inc., 2023). The layered operating system Ubuntu server is capable of being configured to provide resources to the processes of highest priority. The Linux kernel uses “a monolithic kernel architecture”, which means it runs as a single program (Sysdig, Inc., 2023). The Linux kernel is not new, rather it was released in 1991 by Linus Torvalds, a programmer who “wanted to create a kernel that would support a Unix-like operating system that he could run on a PC with an x86 computer chip architecture” (Sysdig, Inc., 2023). The Linux kernel features are similar to those found in any modern operating system (i.e. resource management, input/output, system calls, and device management) (Sysdig, Inc., 2023). In addition to the basic features, modern versions of Linux offer additional features and other types of functionality unique only to Linux, such as the Kernel Virtual Machine, LXC and Extended Berkeley Packet Filter (Sysdig, Inc., 2023).

1. **Storage Management**:

The best storage management system would be the logical volume manager (LVM) for Ubuntu Server. This management system is capable of tracking, encrypting, backing-up (Snapshotting hard-drive state), and consolidation. LVM, is a storage device management technology, giving “users the power to pool and abstract the physical layout of component storage devices for flexible administration” (Ellingwood, J., 2023). Linux kernel provides the device mapper a framework for mapping physical block devices onto higher-level virtual block devices, forming the foundation of the LVM, software RAIDS, and dm-crypt disk encryption, as well as other additional features such as file system snapshots (Ellingwood, J., 2023). The benefits of LVM include feasibility and flexibility of locating/merging data with tracked volumes. Constant backups provide reliability due to the added insurance that data is saved regardless of hardware failure, data breach, or power failures. Encryption will create a specific volume/partition specifically for files that require the most security. Consolidation allows the client the ability to merge data into one volume/partition quickly.

1. **Memory Management**:

The memory management subsystem is one of the most important parts of the operating system (OS) (Rusling, D. A., n.d.). Ubuntu Server memory usage is capable of being configured to match the needs of the server for any particular requirement. The memory can be set to each particular instance if necessary. Linux memory management subsystem is responsible for managing the memory in the system (i.e. implementation of virtual memory and demand paging, memory allocation both for kernel internal structures and user space programs, and mapping of files into processes address space) (Memory Management - The Linux Kernel documentation, n.d.). Within the computing world there is always a need for more memory than physically exists in a system, but strategies have been developed to overcome these limitations, such as virtual memory (Rusling, D. A., n.d.). Virtual memory allows a system to appear to have more memory than it actually does by sharing it between competing processes as needed (Rusling, D. A., n.d.). As well as making your computer’s memory go further, the memory management subsystem provides large address space, protection, memory mapping, fair physical memory allocation, and shared virtual memory (Rusling, D. A., n.d.).

1. **Distributed Systems and Networks**:

A distributed system and network will allow The Gaming Room to have bandwidth advantages as well as fault tolerance. This will allow for continuous operation regardless of server outages. The distributed system and network utilizes geographically dispersed, yet synchronized cloud-based resources to provide the Gaming room with bandwidth that is sufficient to support peak user demand periods with a minimum of latency and lag.  Additionally, distributed systems/networks can very easily scale to meet demand dynamics.  Operational availability is also optimized as there is no single point of failure that can cause the network to fail.  Rather, when a server or regional failure (or outage for any reason) occurs, the mesh interconnection characteristics allow alternate network resources to provide a seamless experience for users.  Data integrity and reliability are also enhanced as the probability of data loss is extremely low.

1. **Security**:

Linux is open-source, making it “one of the most secure operating systems available” (Gomez, B., n.d.). Even though it is open-source it doesn’t mean that it is perfect, like any other software Linux is open to “exploits” (Gomez, B., n.d.). There are many simple things that can be done to ensure that Ubuntu Linux is more secure (Gomez, B., n.d.). This can include encrypting your hard drive, updating your operating system regularly, disabling root access, disabling unnecessary services, ensuring safe internet browsing, using Adblock to disable advertisements on websites, the use of Hypertext Transfer Protocol Secure (HTTPS) which encrypts traffic between the user and the site and using NoScript to disable scripts that are not from sites the user declares trustworthy (Gomez, B., n.d.). Security privileges will allow for a controlled and secure server environment with Ubuntu Server. Setting rights will disable the wrong hands from having access to high-level information. One other consideration is using encryption whenever possible. Linux is a secure operating system by default, but given the ability to be customized ensures for a host of ways to make it more secure (Gomez, B., n.d.).

Works Cited

Aviviano. (n.d.). *What is a driver? - windows drivers*. Windows drivers | Microsoft Learn. https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/what-is-a-driver-

Canonical Ltd. . (n.d.). *Ubuntu server - for scale out workloads*. Ubuntu. https://ubuntu.com/server

Ellingwood, J. (2023, January 5). *How to use LVM to manage storage devices on ubuntu 18.04*. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-use-lvm-to-manage-storage-devices-on-ubuntu-18-04

Gleek. (2021, March 2). *Diagram maker for developers*. Gleek. https://www.gleek.io/blog/class-diagram-arrows

Gomez, B. (n.d.). *The most important things you can do to quickly secure ubuntu linux...* Linux Security. https://linuxsecurity.com/features/the-most-important-things-you-can-do-to-quickly-secure-ubuntu-linux#:~:text=Thanks%20to%20the%20fact%20that,done%20to%20improve%20its%20security.

IBM. (n.d.). *Attributes*. in class diagrams. https://www.ibm.com/docs/en/idsa?topic=classifiers-attributes

IBM. (n.d.). *Entity Classes*. Entity classes. https://www.ibm.com/docs/en/spm/7.0.4?topic=reference-entity-classes

IBM. (n.d.). *The UML 2 class diagram*. IBM developer. https://developer.ibm.com/articles/the-class-diagram/

Mavrogiannopoulos, N. (2021, November 23). *Cloud-optimized linux kernels – what makes ubuntu the top OS across the clouds*. Ubuntu. https://ubuntu.com/blog/cloud-optimized-linux-kernels

Memory Management - The Linux Kernel documentation. (n.d.). *The linux kernel*. The Linux Kernel documentation. https://docs.kernel.org/admin-guide/mm/index.html

Orion Development Group. (n.d.). *Data Representation Model Notation*. Model notation. https://www.sedris.org/dm\_notn.htm#:~:text=in%20Class%20A.-,The%20%220..,one%20object%20in%20Class%20A.

Rusling, D. A. (n.d.). *Chapter 3 memory management*. The Linux Documentation Project. https://tldp.org/LDP/tlk/mm/memory.html

Sysdig, Inc. (2023, March 27). *Understanding the linux kernel*. Sysdig. https://sysdig.com/learn-cloud-native/container-security/understanding-linux-kernel/#:~:text=Like%20most%20modern%20kernels%2C%20the,kernel%20into%20smaller%2C%20discrete%20programs.

Taylorial. (n.d.). *UML class diagrams*. taylorial.com. https://taylorial.com/oo/uml/#:~:text=The%20inheritance%20relationship%20is%20shown,a%20generalization%20of%20the%20subclass.