

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)4

[**Evaluation**](#_2o15spng8stw)5

[**Recommendations**](#_m8aleynsvzvc)9

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/21/2022 | Craig O’Loughlin | Initial Version: Summary, Constraints, and Domain Model |
| 1.1 | 02/06/2022 | Craig O’Loughlin | Updated: Evaluation (Server, Client, Dev Tools) Table by Platform |
| 1.2 | 02/20/2022 | Craig O’Loughlin | Added recommendations and evaluation formatting |

**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)

“Draw It or Lose It” is a game application that is currently available only on the Android smartphone operating system platform. This availability limits the potential audience who may use this application. To address this, The Gaming Room would like to build a web-based version of this application that may run on multiple platforms, with a service that can run many unique game instances at a time.

## [Design Constraints](#_2et92p0)

Web-based application development

This constraint restricts the development tools and environments we may use for this project. The end result must be an application that can be run on any of the most popular web browsers (Chrome, Edge, Firefox).

Multi-Platform availability

The outcome of this project must be able to run on any OS, mobile or PC, that may attempt to access this web application. Therefore, it must be ensured by choice of development tools that the outcome is platform agnostic.

Multiple unique game instances, teams, and players

This constrains the way that the software is built at the domain level in order to include proper object security so that no instance of a game can be run multiple times. Game and team names will similarly be uniquely identified to prevent duplicates.

Multiple team and player support

Each game instance will be built to allow many teams, with each team allowing multiple players per instance. This also constrains the architecture of the game build and therefore the development. Object oriented programming principles will be applied to ensure safe object instantiation to meet this parameter.

## [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 proposed UML class diagram is pictured below.

**"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 primary components of this domain are:

An Entity class which each Game, Team, and Player are built off of. The Entity class encapsulates the unique id and name of each component object along with providing public read-only access to this information. The Game, Team, and Player class each inherit this functionality along with adding their own expanded functions. This inheritance affords each of these objects the ability to hold their own unique id or name per the design constraint. As an added benefit, the Entity base methods can be accessed from each sub-class via polymorphism as needed.

Each Game class can use zero to many Team objects that will participate in that game instance. Each Team object can similarly instantiate zero to many Player objects that each represent a single game player (user), fulfilling the requirement for multiple team and player support.

A GameService class acts as a manager for all the current game instances and through abstraction provides a simple interface for creating and managing the required multiple game instances, along with automatically assigning game instances. It is a singleton entity where only one GameService is allowed to exist, which ensures no duplicate instances of a game can be run.

A ProgramDriver class which provides the main game loop and gets the service instance of the GameService class. The ProgramDriver also uses the SingletonTester which is used to ensure that the GameService class is properly built by testing it.

## [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 | - Typically, the most expensive hosting solution.    - Requires proprietary (but top of the line) hardware and paid software. Server specific Mac OS price is $999 as of 2021.  - Offers great uptime and security features built in.  - Hardware + software ready-built solutions are available for a monthly fee. | - Open source and ability to run on weaker hardware make this less costly.  - Only the hardware is needed to be purchased. OS software is generally free, however there are paid enterprise versions such as RHEL server at $349 as of 2021 (some versions of RHEL are now free).  - Rarely needs software updates.  - Requires implementation of your own or third-party security package.  - Widely used and supported with many options and proven implementations. | - Closed system like Mac.  - Must pay for OS license and upgrades. Windows Server 2022 starts at $501 for ‘Essentials’ version and $1069 for ‘Standard.  - Stricter hardware requirement, server can crash easily without appropriate CPU and RAM.  - Can run Microsoft proprietary systems such as ASP and .NET.  - Some available hosting services provide hardware and software with monthly pricing and ‘grid hosting’ to reduce downtime during updates. | - Not typically used for hosting although it is possible.  - Limited processing power means suitable only for simple and low traffic applications.  - Limited tools for hosting environments, such as Tiny Web Server for Android.  - Extremely low cost. Android OS is free. iOS requires the purchase of an apple device. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Development Requirements | Mac | Linux | Windows | Mobile Devices |
| Client Side | - A multi-platform client porting to Mac requires extra development effort.  - Additional costs include a development licensing fee, purchase of Mac computers for development, and possibly recruiting a specialized development team due to the more limited available languages and programming environments.  - Less cost incurred from testing as the hardware on this platform is more uniform and of higher quality.  - Apple software approval process required before application will be made available for use.  - Safari is the default web browser and compatibility with this browser should be a concern for the server platform. Other browser options on Mac exist such as Firefox. | - Linux is a flexible open-source operating system, and many development tools or languages exist which allow parallel development to this platform, such as C++, Java, and Python. This indicates a lower development cost to platform.  - Linux is not common at client-side usage but may be found running on especially old hardware as it has a small footprint and low system requirements. This may introduce further testing considerations and increased testing costs.  - There is no default web browser for this platform, but many distributions are packaged with Firefox which is widely adopted on many platforms. Other browser options exist here such as Google Chrome. | - Windows is the most widely used consumer platform and should be a primary target for client development.  - Additional costs for development include the purchase of Windows licenses to host the development suite and testing environment.  - Many developers are familiar with programming in a Windows environment and a specialized team may not be needed. Widely utilized and free development tools exist.  - Microsoft Edge browser is the default for this platform. Modern compatibility has improved compared to its predecessor Internet Explorer, but care should still be taken for compatibility. Other browser options exist on the platform such as Firefox. | - The mobile development environment is more fragmented in general than PC and additional development efforts may be needed to reach this platform. As an advantage, this application is already live on the Android mobile platform.  - Developing for the iOS platform incurs similar costs as Mac, such as developer licensing fees, purchase of Mac computers, and recruiting specialized developers, and possibly having to rewrite a significant portion of the codebase depending on its current structure.  - Safari is the default web browser, but other options exist such as Firefox or Google Chrome. Special consideration should be given to the mobile browser display vs. a PC in the server software. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Development Requirements | Mac | Linux | Windows | Mobile Devices |
| Development Tools | - C/C++/Objective C or Swift can be used along with the Xcode IDE.  - Development, at least compilation, testing, and deployment, must occur in a Mac computer environment. Virtual environments can be used to assist in cross-platform development.  - An Apple Developer Program yearly license is required. | - Wide range of tools and development frameworks are available.  - Offers developer friendly tools such as a built-in package manager and a BASH shell.  - As an open-source environment, no licensing needed for development. | - Proprietary freeware tools such as Visual Studio with C#, a modern, Java-like programming language can be used along with the .NET library for simple high-level development.  - Wide range of other tools and cross-platform support.  - Windows OS development highly recommended which requires a one-time license purchase for one or multiple machines. | - Separate programming tools between platforms.  - Android uses mostly Java and Android Studio or a general-purpose IDE such as Eclipse.  - iOS uses Swift or Objective-C and exclusively the Xcode IDE.  - May require multiple development teams as developers tend to specialize in iOS or Android. Cross-platform tools exist to ease development such as Xamarin and React Native. |

## 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 Linux operating system is recommended for this server application. It has great flexibility in development tools and frameworks that can accommodate this project. The operating system is open source and there are no licensing fees to consider. It is an effective, safe, and low-cost option with a proven record of use in this application field.

The option is available to either build a Linux server or rent/lease the use of a ready-made Linux server and hardware for a low monthly cost. Services exist starting at ~$10 monthly.

1. **Operating Systems Architectures**:

The Linux OS is described as a ‘Unix-like’ system, that is, it was built based on the secure Unix OS that came before it. It was, however, mostly built from original code and does not duplicate or share Unix code.

The Linux OS is made up of the kernel, which contains the main functions of the OS that operate the computer hardware, and a shell, which is a layer that provides application access to the kernel functions. Many Linux distributions use Bash (Bourne-Again Shell) as a default.

There are many distributions of the Linux OS currently available which offer a different range of default configurations and feature sets. In the context of a server-based game, Debian is recommended due to its proven stable performance record and the fact that it comes pre-equipped with the tools required to get started. However, if a distribution with commercial support for server development is desired, RHEL (Red Hat Enterprise Linux) is a common choice. This distribution includes a paid subscription option which provides technical support and security resources.

1. **Storage Management**:

Modern servers are beginning to adopt the use of SSDs (solid-state drives) due to the increased performance, reliability, and declining prices vs. traditional hard drives. An SSD is recommended for the storage system of this server. IO speeds of SSDs are on the order of 20x faster than the traditional HDD (hard drive disk), contain no moving parts, and consume significantly less power in operation.

There are many potential choices for a software file system management on Linux, however, the older Ext4 (Extended file system 4) (currently the default for Debian) remains a dependable choice for this server application. It has excellent read and write times and is among the most dependable choices for Linux. Also, it contains SSD specific management tools and optimizations such as TRIM, which is an SSD optimized data block management module, and the ability to disable journaling, extending the longevity of an SSD system.

1. **Memory Management**:

Memory on the Linux OS is abstracted into ‘virtual memory’ due to the wide variety of CPU and memory management hardware configurations that are possible. An application running on this platform therefore will not deal directly with physical memory addresses and instead work through the kernel, which keeps track of the physical memory through demand paging and tables.

The kernel designates a certain amount of virtual memory space, which may exceed the physical capability of the computer system, and further allows a program to allocate memory and reserve its use. The operating system can be configured to either reject allocation requests that exceed the available amount, or instead grant all requests which is faster but risks the system running out of memory. In this case OOM (out-or-memory) protocols will run and some tasks may be ended abruptly by the operating system.

Memory that is not currently in use is used for page caching. The operating system stores file reads so that subsequent read requests can be completed faster, directly from memory. The OS keeps track of when to refresh these caches to accommodate file changes.

A primary concern with this server application is uptime therefore the available memory of the server should be carefully considered in order to meet the requirements. Further, safe practices should be used such as configuring the system to check availability before making memory reservations.

1. **Distributed Systems and Networks**:

A uniform, RESTful API should be used to facilitate communication between this server application and the various user platforms. This type of API eliminates many considerations for communication and compatibility between software platforms. The data transfer should use a uniform object representation such as, commonly, JSON strings.

The Linux OS provides a stable platform for a server with low downtime due to outages. The Linux kernel is proven and reliable. There are no forced updates and many component updates do not necessarily require a restart. As for connectivity the server should remain on a network at all times in order to facilitate data transfer requests. HTTP protocols and URLs can be used per RESTful principles in order to allow this software to communicate directly over the internet without needing a private network.

1. **Security**:

Securing the system begins with the physical server location. Precautions should be in place to prevent unauthorized access to the server. The Linux OS has security features to further prevent unauthorized access. For example, the widely used PAM (Pluggable Authentication Modules) suite is an authentication library that is supported out of the box on many Linux distributions such as Debian and RHEL.

The network access to the server should also be secured using a firewall that can monitor and control the type of activity that is permitted on any connection port. Most Linux distributions include ‘firewalld’ by default which is a firewall program configurable for a wide range of needs; many third-party firewalls exist for Linux as well.

The server application itself should also be designed with its own authentication system. This can be implemented in the program API to determine which connections have proper authorization to perform certain sensitive actions such as deleting or copying information.