

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

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

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

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

[**System Architecture View**](#_heading=h.3dy6vkm)3

[**Domain Model**](#_heading=h.1t3h5sf)3

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

[**Recommendations**](#_heading=h.3rdcrjn)5

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/13/2021 | Lucas de los Santos | Initial design document for web-based app |

**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.2et92p0)

The client requests a web-based application for a game called Draw It or Lose It. The requirements for this game are the ability to have multiple teams that allow multiple players assigned per team. The game and team names need to be unique and the program must be able to check for existing names and not allow duplicates. Only one instance of a particular game can exist at a time in memory. By using unique identifiers for each game, team, or player this can be solved.

## [Design Constraints](#_heading=h.tyjcwt)

The viewing area of a web application is not a fixed space. Between mobile, PC, monitors, etc the resolution is different thus the application needs to be flexible for all these various situations.

Due to the requirement of being web-based, it requires active internet whereas a standalone application may not require. On top of that, it requires backend architecture to support multiple users in various regions.

## [System Architecture View](#_heading=h.3dy6vkm)

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.1t3h5sf)

Within the package com.gamingroom, ProgramDriver class has a dependency on the SingletonTester class in order to prove there is only a single instance. GameService depicts multiplicity in relation to Game, Team, and Player by displaying the concept of cardinality of zero to many. Here this means GameService can create zero to many Games, Teams, and Players. The Entity class is a parent class and utilizes inheritance making the Game, Team, and Player child classes. Entity uses the concept of encapsulation by setting a protected access modifier.

**"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](#_heading=h.2s8eyo1)

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** | Overly expensive and more suited for Mac client management. While MacOSX Server does allow for web hosting it’s not really scalable or cost-effective in a larger environment. | The staple of most companies when it comes to hosting and deployment. It’s open-source, scalable, easy to automate large infrastructure, and ultimately more secure than most other OS’s for server-side deployment. Takes comfortability with terminal due to generally headless servers. | Most of the world is comfortable with Windows from Client-side development or general desktop use, but the management of IIS services in a scalable, automated way is not for the faint of heart. It can be accomplished but the ease of it requires familiarity in Powershell and even that is limited in scope. | Not advised as it’s not powerful enough in general to host any form of service that would require multiple users in concurrency. |
| **Client-Side** | From a hardware perspective, it’s a more expensive option but the setup of a development environment is simplistic. A learning curve exists for native windows users but is fairly easy to navigate and learn. | Relatively inexpensive hardware requirement as it can be run on anything. Setup and expertise for learning to use as a development environment could be more advanced since it’s unfamiliar territory for most. | Equal expense for hardware as Linux since most laptops/desktops are bundled with Windows and generally not sold without it. Most commonly used operating system and fairly straightforward. | Not necessarily a development platform. Most mobile development occurs within SDKs that live on Windows, Mac, or Linux and work in a virtualized environment. |
| **Development Tools** | Visual Studio Code would be best as it works well with JavaScript and HTML5. It’s free and works on all platforms. | Visual Studio Code would be best as it works well with JavaScript and HTML5. It’s free and works on all platforms | Visual Studio Code would be best as it works well with JavaScript and HTML5. It’s free and works on all platforms | Web-based development would occur on Mac, Windows, or Linux. HTML5 and Javascript would suffice to render a mobile webpage. At this point, it’s just a matter of accounting for screen size in the coding. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

To note due to the overall vagueness of language it’s unclear if this is meant to be the development environment or the server environment to run on since they are two separate animals.

1. **Operating Platform**: From a development perspective Mac would be a preference due to the ability of Unix terminal and easier to mock a server for local development. For the server-side, Linux is recommended as it’s tried and true for the world around us. The images can be made very slim in docker containers to reduce the overall attack vector, boost performance, and reduce noisy neighbors such as services running on the system that isn’t needed. Both development and server deployment platforms work well together in terms of developing an automated pipeline.
2. **Operating Systems Architectures**: Linux is portable in that it works on numerous hardware and can run with a very small footprint. It is a security-oriented system and easily hardened to the user’s requirements. One of the main features that is most attractive is the use of the terminal which allows remote management and streamlined commands that will work across multiple versions types of Linux within the same Distro family.
3. **Storage Management**: Kubernetes, the container management system that the Linux servers would run on, can provision static or dynamic volumes should data need to be stored persistently. In general most of the data would live within a database and it would connect to the various applications that serve the website. In this, the volumes can be specified specific access based on the need of the systems running within the Kubernetes pods.
4. **Memory Management**: This is kind of a multi-layered process since Kubernetes manages all its pods, Linux containers, on workers nodes that have their own assigned memory. Pods are assigned to nodes by Kubernetes scheduler and migrate pods between nodes depending on the total load of CPU and memory. The internal system Linux which is running inside the pods has its own complex system of memory management by taking its physically allocated memory specs and creating it into virtual memory. The virtual memory abstracts the details of the physical memory and allows bits of info to be kept in the physical memory. This facilities protection and controlled sharing of data between running processes. It uses other concepts such as swap space should the RAM space fill up and in turn, it utilizes disk space to substitute the missing RAM being used. In most cases, however, this is insecure and best to disable.
5. **Distributed Systems and Networks**: Distributed systems can be accomplished, using AWS EKS as an example, one can create multiple EKS clusters across various regions to make a higher availability. Now you’ll have a scalable number of Linux clusters with a database and application layer that is highly available while solving latency issues for loading times. Each cluster can communicate with a regional database and sync that data via aggregation to a centralized database to keep users all in sync regardless of region. All data transmission would be done via encrypted channels using https protocol. A benefit of liveness and readiness probes based on health checks of your Linux pods allows quick recreation of pods should an issue arise and thus adds fault tolerance.
6. **Security**: Linux is an open-source project thus anyone can review the code, do vulnerability scans, and patch any bugs on a much faster and broad cadence than proprietary code bases such as Windows and MacOSX. By limiting the overall services to just what’s needed and being mindful of recommended practices when setting up an application server on Linux it has the capacity to be more secure than its counterparts. With built-in strict firewall rules and features such as SELinux or AppArmor, the environment is locked down almost to the point of madness.