

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 | 11/17/2024 | Cody Stuart | Initial version |

## [Executive Summary](#_sbfa50wo7nsh)

Creative technology solutions (CTS) have reached out to request support with a request for support in facilitating the development of their game Draw it or Lose it. Teams will compete to guess what is being drawn over a 30 second period. Each game will have one or more teams, and each team will have multiple players assigned. Game and team names must be unique to allow a team to check if a name is in use. Only one instance of a game can exists in memory at a time.

## Requirements

Technical Requirements:

* Unique games and teams
* Timed Rounds
* Web based game

Business Requirements:

* Support setting up game environment
* Similar functionality to previous existing versions of the game

## [Design Constraints](#_2et92p0)

* Unique Games and Teams
  + A direct request from CTS for game and team names to be unique has been made. To achieve this we will start with the singleton design principles for the games to ensure only one instance of game exists. Additionally there will be checks made to ensure that A team name is not currently in use before being allowed to be assigned to a new team.
* Timed rounds
  + Each game or match will consist of four rounds. Each of these rounds should last exactly 30 seconds. Over the time period of the round an image chosen at random from a library of stock images, each image will be slowly rendered to the screen completing the render at the 30 second mark. If the guessing team fails to correctly guess the image during the time period each of the remaining teams will be able to offer one guess in a 15-second time limit.

## [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 for this program shows several Object Oriented Principles. Based on the three connected lines with the arrow pointing to Entity Game, Team, and Player all inherit from the Entity class. The entity class exists in this case to avoid duplicate code being written for each of the inheriting classes. The diagram also has 0..\* associations for GameService to Game meaning a GameService can have more than one game (not actively in memory though). The same association exists for the game to team class each game can have multiple teams and each team can have multiple players. GameService exists for the Singleton pattern serving as the only instance to manage games, teams, and players.**

**"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** | Mac based servers can use common web hosting applications such as Apache or Nginx. As well as other tools that provide binaries that can run on Darwin. The server operating system is Unix Based so Unix tools and utilities should be able to run on it. Apple discontinued support for the MacOS server in 2022 so it’s not likely to even be a capability at this time. | Linux is an excellent choice for the server side of the web hosted service. Not only is the Operating system open source but there are a multitude of server-side web sources that are open source as well (i.e. Nginx, Apache). Linux is a common choice among cloud environments as well due to its ability to be customized easily. Heavy support for virtualiztion and container technology on this platfrom. | Windows based servers have full support for web hosting with the built in Internet Information Services (IIS). It is known for robust security. Much more beginner friendly than Linux. Server licenses include support. | Server side hosting is not applicable for mobile devices. They lack the software as well as hardware requirements necessary for large scale web application hosting. |
| **Client Side** | MacOS is secure, as well as supports common, and modern web browsers. | For a web based application, the client side Linux machines would be compatible for the majority of distributions as most of them support the common and modern web browsers like Chrome, and Firefox | Windows client machines are the most popular client device for non-mobile devices. 73% of all devices are windows. Windows web clients are easy to use. | Andriod: Android devices are Linux based and support modern, and common web browsers. The android app stores are easy to put applications on, as well the devices support side loading via installing APK files. Mobile optimizations will need to be made due to the change in device usage and standard screen sizes (compared to most computers).  iOS: Similar to Android devices the Apple mobile devices have the support of the App store and modern, common web browsers, although there is a cost to applications on the Apple App Store. Additionally UI/UX and mobile optimizations will need to be considered. |
| **Development Tools** | MacOS is written in Swift mostly, Some of the older code base is written in Object C.  Licensing cost are quite expensive: $499/server $50/App  Xcode is the development tool and is free. | The licensing costs would be minimal to 0 for this platform. As most languages are available for it and do not require licensing. Additionally web hosting tools such as Apache and Nginx are free/opensource tools designed by the community, for the community.   Licensing for the servers themselves are free. Although if support contracts are desired licensed servers could be purchased from companies like Red Hat. | Uses C#, and .Net for development languages. Additional languages can be used to support like Powershell and Python.   Visual Studio is the most common IDE used on windows and requires licensing. Other IDEs could be used such as Visual Code however they are not as functional.  Licensing can be costly based on scale for both servers, and visual studio code. | Android:  Officially supported languages are Java and Kotlin. These languages are great as they support a write once, compile everywhere mentality. These languages are also free. Android studio is free to use and is a great tool for developing android applications.  iOS: Swift is used for mobile devices that run iOS. Similar to the MacOS devices they are developed on Xcode which is free. An apple developer subscription may need to be purchased. |

## 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 recommended operating system for the servers is Kubernetes on Linux.
   1. **Distribution**: Ubuntu (The latest stable release is 24.0.4.1)
   2. **Cost**: Ubuntu does not require a license and therefore server software acquisition can be free if no additional software is needed.
   3. **Benefits**: The low cost and open source backed communities make the Ubuntu Linux server a good choice. The servers can be run headless (without a graphical interface) to ensure max performance from the hardware. Ubuntu is also a great distribution to use as the OS for Kubernetes nodes.
   4. **Scalability**: Kubernetes on Linux is a great way to only use the resources needed and maximize server output. Kubernetes provides automatic scaling methods that allows the server to increase the number of instances running based on the user counts and also reduce the running instances when user counts lower.
   5. **Portability**: Containerizing Draw It or Lose It allows it to be run on any server capable of running containers.
2. **Operating Systems Architectures**: It is recommended to use amd64 for the OS Architecture.
   1. **Kubernetes**: requires a 64-bit OS for Control Plane and Worker Nodes
   2. **Support**: amd64 has more support and a broader usage than arm64.
   3. **Cloud**: Should the desire be to use a public cloud provider, amd64 is the OS Architecture in use there.
3. **Storage Management**: An NFS backed storage solution should be used.
   1. Monitoring: Kubernetes and NFS storage providers offer built-in monitoring tools like Grafana.
   2. Kubernetes: Within Kubernetes Rook Ceph can be used along with a storageClass that has Dynamic Provisioning enabled. This allows the storage to scale with the application when user counts are high.
   3. Backups: NFS Backups provides a backing for a disaster recovery plan. Backups can and should be automated to ensure they’re as up to date as possible.
   4. Compression: Static files that are only accessed and not being written should be compressed. Only being uncompressed in memory.
4. **Memory Management**: Memory can be used on an as needed basis with Kubernetes.
   1. **Scaling**: Kubernetes will only use memory as it scales. Additionally, if the Kubernetes nodes are virtualized memory will only be allocated as it’s used.
   2. **Redis**: Using Redis to cache frequently accessed items will reduce the strain on the storage. Redis can be scaled in Kubernetes by increasing the deployment count making the high-speed application scalable.
5. **Distributed Systems and Networks**:
   1. **Kubernetes**: Various networking tools are available in support of Kubernetes. For example, Nginx is an ingress object that allows you to control the flow of network traffic in and out of a Kubernetes cluster.
   2. **LoadBalancing**: A load balancer can be applied to spread the management of network traffic across each instance of the application. This would help as the service scales up the load balancer would be able to keep the load even across all instances.
6. **Security**: Security will be practiced at each step of the way
   1. **Encryption**: Data in transit will be encrypted between the user’s device and the server using TLS. The data will only be un-encrypted using the TLS certificates signing key when it reaches the Kubernetes Cluster. Additionally, data at rest and not in use should be encrypted.
   2. **AAA**: Authentication, Authorization, and Access should be enforced on an as needed basis. Password policies should be enforced. Users should be encouraged to use 2FA. Roles should be created based on a permissions schedule and only assigned on a needed basis.