

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 05/23/25 | Riley Eichenour | Initial Submission of Document; Added Executive Summary, Requirements, and Domain Model descriptions |
| 2.0 | 06/06/25 | Riley Eichenour | Add Evaluation Section |
| 3.0 | 06/21/25 | Riley Eichenour | Add recommendation section |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has a desire to create a web-based version of their game Draw It or Lose It based on their android app. They would like the functionality of the web-based version of the game to reflect their mobile version and need assistance setting up the environment.

To address the issue faced by The Gaming Room this project will aim to set up the infrastructure and application to support a web-based version of the Draw It or Lose It Android application.

## Requirements

From the business side of the requirements the client would like the game to be available through a web-based environment. The application will need a library of stock images to create the game rooms. There will need to be a server application and client application to support the intended functionality of their game.

There are no explicit technical requirements outlined by the client at this time; however, a server-based application will need to be created to host the game and client-based application to allow players to join games. The server-based application will need to be hosted on an onsite or Cloud-based server, with access to the public internet for outgoing and incoming traffic. The client application will need to be stored on another server or Content Distribution method to allow users to download the client. A real-time communication method will need to be utilized to readily transmit data between the server and other clients.

## [Design Constraints](#_2et92p0)

The design constraints are as follows for Draw It or Lose It:

* **Real-Time Communication**: Drawings are incrementally rendered, which means an active and stable connection must be in place to handle quick, consistent transfer of data
* **Synchronization of Server and Clients**: In addition to the active transfer of data, the game state needs to be managed and validated on the server, as well as handling distribution of resources to prevent games becoming unresponsive or unexpectedly crashing.
* **Scaling**: As time shifts through the day the application should be capable of scaling up or down based on the number of active or forecasted games for the day to ensure high uptime and prevent poor user experience due to low availability/stability.
* **Adaptive Content Delivery**: The delivery of the stock images or renderings must adapt to the user connection to prevent high latency for users on lower quality connections, preventing the ability to effectively play the game with other users due to waiting for the image to load.

## [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 provided presents the class structure for the Draw It or Lose It web-based game application. The GameService depicts a singleton instance to ensure that only one instance of the service is running at a time. Notably the service attribute on the GameService class and the constructor being set to private provides this information. A GameService can have 0 or many games associated with it, which is handled through the game getter methods and addGame method. A Game can then have 0 to many teams associated with it, which is managed through the addTeam method on the game class. A Team can have 0 to many players associated with it, which is handled through the Team addPlayer method. With all of the relations in mind the service can have many games running, with many teams associated with the games, and the teams can have many players associated with them.

The Object-Oriented Programming principles clearly indicated by the UML are: Inheritance, Polymorphism, Abstraction, and Encapsulation. Inheritance is represented with the Entity class, as a parent class, and the Game, Team, and Player classes and the open white arrow that goes from the children classes to the parent. Polymorphism is implied through the naming of the Entity class being generic, which indicates that the entities should be readable without their specialized classes, and the entities of the application can take on different behaviors. Abstraction is most notably represented through the game service where the management of the state of the service only has functionality exposed that is necessary to perform the operations of other classes, without the other classes being aware of the underlying implementation, specifically the getGameCount, getNextPlayerId, and getNextTeamId methods. Finally, encapsulation is represented again through the GameService, the Game class, and Team classes, which maintain their relationships, but do not expose data other than what is strictly necessary for the other classes in the application to perform their operations.

The Object Orient Principles utilized within the application effectively create a representation of the real-world use case by hosting the single service, which can have multiple games running with the teams composed of players participating in the 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Mac operating system is not common as the server-side platform for enterprise applications as often as other platforms. The user interface utilized for the services is often intuitive for users. While it is good for utilization in macOS based stacks, it does struggle outside of these stacks in comparison to other platforms. | Very common in the enterprise market for various applications, including web servers, database managements systems, and game servers. Open source and many distributions offer many choices for a system that is tailored to the needs of the application. High support in cloud-platforms and familiarity by system admins. | Not uncommon in enterprise level systems to support applications, especially Windows based applications. Offers a familiar user interface to the Desktop operating systems. Integrates well with the Azure cloud platform. Licensing is expensive for the servers, especially when considering enterprise applications. | Limited hardware resources due to the space limitations on these devices can limit the scalability and utilization of the resources. Good for prototyping in smaller testing or development environments, but limited exposure in an enterprise environment. |
| **Client Side** | The interface is friendly to users and has a relatively large market share, in both mobile and desktop-based applications. It integrates very well within devices in the iOS ecosystem making utilization across multiple devices simpler. The hardware is often more expensive than other alternatives and prevents changes for the client. | The interface, or lack of one, makes it slightly less user friendly, but it offers an extensive amount of extensibility and customizability. There is support for a lot of applications utilized for daily user, however proprietary software often doesn’t have high availability or open-source alternatives. Hardware manufacturers often don’t provide as much support for these platforms in comparison to other operating platforms. Not often utilized by low-tech knowledge users. | One of the most common clients based on market share. Offers a large amount of proprietary applications and often has very early support for most software applications due to the wide user utilization. Maintains a consistent UI that is familiar to users from one generation to another. | Common availability for average users with the continuing rise of smart phones, which includes devices that utilize other operating platforms. Limited hardware due to the footprint of the devices limits the potential for the functionality of the clients. It’s difficult to support all devices due to the wide variance in client hardware and lack of consistency in the supported resolutions. |
| **Development Tools** | Geared toward iOS development and integrates well with stacks that are intended to develop on the iOS platform. Being based on a Unix operating system provides some form of familiarity with developers who commonly work on the Linux operating platform. Many of the tools are closed source and controlled by Apple. The support for Windows based applications is more limited. | Offers many tools out of the box for C and C++ development. Package managers simplify the process by adding support for additional languages. Inclusion of the Git VCS can also make retrieving and building source from source. The terminal support and tools are familiar to developers and systems admins, such as Vim, for utilization across systems, even if they only share the common base of the Linux platform. | There is a lot of support for technologies commonly worked on the platform including .NET, C#, and Powershell, however other common languages are often not supported out of the box and require a lot of user setup. The offering of WSL reduces the frustration with Linux users developing on a Windows environment. | Development and mobile platforms is often dependent on SDKs offered by the manufacturers of the hardware or operating system on the device. Due to the variety of devices supported by varying operating platforms, there is little standardization between tools and stacks utilized for development on these platforms. |

## 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 Server based distribution through the AWS Cloud Provider
2. **Operating Systems Architectures**: The Ubuntu Server distribution is a widely supported Linux based operating system, which is very common as part of a distributed system environment. The modularity of the system and open-source nature of the Linux kernel allows designing a system that meets the needs of the application and allows flexibility to add customized functionality to the system. The licensing costs associated with Linux operating systems are cheaper than those offered under Windows and macOS based systems. Docker and Kubernetes are two modular, portable services that are simple to utilize through a Linux distribution with no required work arounds, which leads to a scalable application automatically as opposed to manual spinning up additional devices. Finally, utilization of a Cloud based environment will enable the Draw It or Lose It application to have low latency, effective regional availability, and high reliability.
3. **Storage Management**: Due to the possibility of multiple servers and clients needing to access common storage systems, the client should utilize a Network Attached Storage with RAID-1 to prevent catastrophic failure with data redundancy between drives and concurrent access by the servers for distribution to clients. The user information should be stored in a relational database with an SSD for the data storage for fast access for frequently read information from a Postgres server running on the server side of the Draw It or Lose It application.
4. **Memory Management**: For efficient operation, the frequently utilized data will be available as part of the caching on the operating system, and current game drawings to reduce latency reaching out to the slow NAS storage. Ubuntu utilizes the Linux kernel with flexible memory management available to system administrators and game developers. Virtual memory is utilized to give each process and isolated memory space, which prevents exposure of data between processes during runtime. Paging enables moving parts of the application that are not in use to be swapped due to inactivity to reduce the application memory footprint to maintain low system response and request processing time.
5. **Distributed Systems and Networks**: With the utilization of an AWS cloud provider an secure API gateway can be setup to control the traffic entering the server side of the application and provide protection through encryption of the data when in transit. RDS can be utilized to handle the Postgres database recommended without having to worry as much about the underlying configuration of the storage. Utilization of platform agnostic communications, such as the HTTP protocol or REST specification would decouple the reliance on an individual architecture on both the client and server side of the application. The Elastic File System would be useful alongside the Amazon File Cache to enable high availability for the images utilized in the application. The high availability and multiple availability zones would ensure the Draw It or Lose It infrastructure would be available, even if shifts in the infrastructure may increase user latency slightly.
6. **Security**: The configuration for the user groups on the Ubuntu server instances needs to be handled in a way that the Game application only has access to the policies and access control that is needed. The root user will not be utilized as the user that runs the game application to prevent unintended access to the system. The connection between the clients and server needs to be handled with the utilization of SSL with a certificate generated by the Draw It or Lose It team that is signed by a certified authority. Certificate management and signing can be handled through the AWS Certificate Management system. The data stored about the users of the application should be appropriately encrypted to prevent data leaks or breaches from resulting in exposed user information, such as names, emails, or payment information. Users should have their identity verified through an OAuth 2.0 system, which in the recommended case would be handled through AWS Cognito. AWS IAM would be utilized to provide restrictive roles to the users and allow only the access strictly needed by the users. AWS Shield will provide potential for denial of service due to high volume traffic. The design of the application should include rate limiting to prevent clients from flooding the server with traffic unintentionally. Scheduled maintenance windows should be utilized to verify the Ubuntu Server system utilized has appropriate security patches and scans.