

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/18/2025 | Steven Nave | Full drafting of Executive Summary, requirements, design constraints and the Domain Model. |
| 1.1 | 04/03/2025 | Steven Nave | Addressed concerns in the Evaluation Table |
| 1.2 | 4/15/2025 | Steven Nave | Made Final Recommendations for the Software Design at the end of the Document |

**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)

This document outlines the design for the web-based version of the game Draw It or Lose It for the client, The Gaming Room. Draw It or Lose It (DILI) is a team-based guessing game, and the goal is to create an application that has the ability to support multiple teams and players in a single game, featuring unique team and game names to ensure only one instance of the game exists in memory at any given time. Additionally, the game will feature real-time gameplay including four timed rounds of play one minute each. Drawings must be rendered at a steady rate and fully complete 30-seconds into a round. If the time expires before the team guesses the puzzle, there will be an opportunity for the other teams to offer one guess a piece to solve it within 15 seconds.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The constraints for this software project include the strict adherence to a single instance for a game, team and player at any given time, so game sessions are unique, protecting against data duplication.

The game must be compatible across platforms including web browsers and mobile devices, while it is currently only available on Android, we want to expand this to all web capable browsers. Furthermore, the game must be able to support real-time game interactions for the timing to be synchronized across all teams and players.

Finally, there must be a system in place to render images without latency and delay, so everyone has a fair chance in the timeframe allowed in-game.

## [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)

As demonstrated in the Gaming Room’s provided UML Diagram, there will be a ProgramDriver containing the main application which uses the SingletonTester to ensure the GameService class follows the Singleton pattern, meaning it will be constructed to have only one instance at any time. There is a base class named Entity that encapsulates the id and name attributes, their getter methods, and the toString() method from which Game, Team, and Player all inherit, abstracting out the common functionality of the three. Now, in order from left to right the diagram shows the relationships between these classes. GameService has a zero-to-many relationship with Game, Game has a zero-to-many relationship with Team, and team has a zero-to-many relationship with Player. As game service could have zero or any number of *different* games operating, a game can have none or any number of different Teams, and a Team can have any number of different Players at a time.

**"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 servers are reliable and performant for web-based applications. Integrate well with Apple products and services, although the cost of hardware and limited scalability may be deterrents. | Highly customizable, scalable, and highly performant for web-based applications. Cost-effective with a strong community support system. May require a higher level of expertise or the recognition of a steeper learning curve. | Windows servers are user-friendly and still offer good performance for web applications. It features extensive software support and is easy to set up and manage. Can be more expensive than Linux, but less than Mac, and may have more security vulnerabilities than either. | Mobile Device Operating Systems are usually not the best for servers due to limited compute power and storage. Highly limited scalability and reliability when compared to the other three. |
| **Client Side** | Considerations for Mac client-side application development requires the knowledge of Swift or Objective-C. The cost for development tools and hardware in this area can be high, and development times can be a bit long depending on testing. | Linux client development requires expertise specific to Linux distributions and knowledge of C++ or Python. Generally lower cost than Mac, although the development time can be drawn out due to the large amount of Linux distribution environments that must be tested. | Windows is a straightforward enterprise for client development considering the moderate cost of development tools and hardware, and vast software support. Development time is relatively short considering the comprehensive testing environments and support. | Mobile development requires expertise in Swift, Objective-C for iOS and Java, Kotlin for Android, for targeted development, although cross-platform solutions exist now – addressed below.  The cost of development tools can be high and development time may be significantly longer due to extensive testing requirements for the variety of devices being considered. |
| **Development Tools** | Mac development usually employs Swift and Objective-C programming languages. The IDE for macOS is Xcode and tools like Homebrew for package management. | Relevant programming languages for Linux include C++ and Python. Developers may use Eclipse, Visual Studio Code, and Linux-specific tools like apt-get. | Windows developers utilize languages such as C#, Java, JavaScript, and C++. Visual Studio is Microsoft’s primary IDE. Chocolatey is used in the Microsoft Powershell for package management. | Swift and Objective-C for iOS, Java and Kotlin for Android, although Kotlin Cross-platform is available now to address web, Android, and iOS development simultaneously. React Native also addresses cross-platform development. IOS frequently uses Xcode and Android Studio for Android IDEs. Flutter is an additional cross-platform development framework. |

## 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**:

For Draw It or Lose It, I recommend a cloud-based web application platform using Linux as the server-side operating system. This combination provides the optimal balance of cost-effectiveness, scalability and cross-platform compatibility. By implementing a responsive web application architecture hosted on AWS, the game can run on virtually any device with a web browser, including Windows, PC’s, Macs, and mobile environments.

1. **Operating Systems Architectures**: The Linux-based AWS architecture offers several advantages for Draw It or Lose it. The current AWS-optimized Linux distribution, Amazon Linux 2023, provides enhanced performance, security, and compatibility with AWs services while maintaining the benefits of Linux’s kernel architecture. AWS supports containerization services for consistent deployment across development, testing, and production. Additionally, AWS offers Auto Scaling alongside Linux’s efficient resource management to handle player loads of various size automatically, as well as provides the ability to run redundant deployments of servers across multiple physical locations for reliable uptimes.
2. **Storage Management**:

For the storage requirements of Draw It or Lose It, I recommend Amazon S3 with CloudFront CDN allows for the storage of the 200 HD image files in S3 buckets that will be distributed via CloudFront, ensuring fast global delivery of game images with reduced latency, which is paramount for the 30-second rendering requirement, regardless of platform or location of players in a game. Additionally, implementing the database for game state information, player profiles and team configurations with AWS DynamoDB, which provides millisecond-level performance at any scale with automatic scaling.

1. **Memory Management**:

The Linux-based AWS platform enables sophisticated memory management techniques that will work for Draw it or Lose It. These include EC2 Instance memory configurations, lambda function memory allocation which automatically scales, container memory limits to prevent any game from using too many resources, and CloudWatch Memory monitoring to help track usage patterns and trigger alerts or scaling action when needed.

1. **Distributed Systems and Networks**:

AWS API Gateway offers a WebSocket API for real-time communication between clients and servers to help synchronize gameplay timing and image rendering. This along with application load balancers to distribute traffic among multiple server instances will help maintain performance across platforms during high usage times. Also, as mentioned earlier, the use of the CloudFront CDN will help ensure that users receive the images in a timely manner regardless of their geographical location.

1. **Security**:

AWS comes ready-made with robust security features that appropriately meet the needs of Draw It or Lose It. AWS Certificate Manager handles free security certificates for HTTPS encryption for all communications between clients and servers. Amazon Cognito can be used for user authentication and authorization, using OAuth token management. Also, AWS offers services that protect against DDoS, malicious activity, unauthorized behavior, access controls and sensitive data encryption.