

<Name-of-Software-Application>

# **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 |
| --- | --- | --- | --- |
| 3.0 | 16/4/2024 | Samuel Rincon | Draw It or Lose It is an application that uses stock drawings as clues in a puzzle-solving game, with four rounds lasting one minute each. |

**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)

The Gaming Room created the well-liked gaming app Draw It or Lose It, which can be downloaded on Android smartphones. The customer wants to make the program available on more platforms, such as Windows, Linux, Mac, Android, and iOS. Our suggested course of action is creating a web application with a contemporary, responsive user experience that can operate on all of the intended platforms. This method makes use of the scalability and flexibility of web technology to guarantee interoperability across devices.

## Requirements

The commercial and technological needs of the customer include making the Draw It or Lose It gaming application compatible with desktop and mobile devices, supporting thousands of players at once, and increasing its availability to numerous platforms. The program must also follow security guidelines while preserving a flawless user experience on all platforms.

## [Design Constraints](#_2et92p0)

The gaming program must be developed with cross-platform compatibility, scalability to support a high number of players, and strong security measures to safeguard user data in a web-based distributed environment. Because of these limitations, deployment tactics, development tools, and architecture must be carefully considered.

## [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 links between the many classes in the Draw It or Lose It program are shown in the UML class diagram that is included. Important elements inside the game's domain are represented by classes like Player, Game, and Drawing. successful modeling of these things makes use of object-oriented concepts like inheritance and encapsulation, which promote successful program development and maintenance.

**"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** | Although Mac is a dependable server, it might not be as web hosting friendly as Linux or Windows. The client's prices may increase due to the licensing of macOS server software. | Linux is notable for its affordability and adaptability when it comes to hosting web applications. There are several deployment options and low license fees. | Compared to Linux, Windows Server has greater license prices but offers more complete server solutions. | Web-based apps can run on mobile devices, but factors like screen size and performance constraints need to be considered. |
| **Client Side** | It takes knowledge of macOS-specific technology and consideration of development tools to support various client types on Mac. | Like Mac, Linux support for many client types requires knowledge of development tools and technology unique to Linux. | It requires knowledge of Windows-specific technology and evaluation of development tools to support various client types on Windows. | Understanding cross-platform development frameworks and mobile-specific technologies is necessary to support various client types on mobile devices. |
| **Development Tools** | Xcode and frameworks for iOS programming such as UIKit or SwiftUI are pertinent development tools for Mac. | Programming languages such as Python and Java, as well as web development frameworks like Django or Flask, are examples of development tools for Linux. | Visual Studio and.NET frameworks are examples of development tools for Windows that are used in desktop and web development. | Mobile development tools include cross-platform frameworks like React Native or Flutter, as well as Xcode for iOS programming and Android Studio for Android development. |

## 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**:

In order to surpass competency and cater to The Gaming Room's varied demands, I suggest implementing a serverless computing platform like AWS Lambda or Google Cloud Platform (GCP) Cloud Functions. Because of the unmatched scalability of these platforms, Draw It or Lose It may easily extend across several operating systems without requiring explicit server management. Utilizing serverless architecture guarantees cost-effectiveness and maximum resource usage while also lowering operational overhead.

1. **Operating Systems Architectures**:

The serverless paradigm abstracts away the underlying architecture so that developers may concentrate only on developing and distributing applications. These systems run on a microservices-based architecture, which allows for the independent deployment and scalability of individual processes. The agility and resilience that this design fosters are essential for a game as dynamic and fast changing as Draw It or Lose It.

1. **Storage Management**:

An event-driven, serverless database, such as AWS DynamoDB or Google Cloud Firestore, would be a suitable option for storage management. These databases may be easily integrated with serverless computing systems, allowing for autonomous scaling according to demand and real-time data synchronization. By using a serverless database, Draw It or Lose It can effectively manage a range of workloads by doing away with the requirement for manual provisioning and maintenance.

1. **Memory Management**:

Memory management in a serverless system is done automatically by the platform according to the needs of individual functions. The serverless platform of Draw It or Lose It optimizes speed and reduces latency by dynamically allocating memory resources to each function execution as it scales. The game can easily adapt to variations in user traffic thanks to its auto-scaling feature, all without sacrificing responsiveness.

1. **Distributed Systems and Networks**:

A serverless messaging service such as AWS SNS (Simple Notification Service) or Google Cloud Pub/Sub can be used to facilitate smooth communication across different platforms. By enabling asynchronous communication between the various Draw It or Lose It components, these services provide fault tolerance and dependable message delivery. Through serverless messaging, communication is separated from the underlying infrastructure, allowing the game to remain available and responsive in dispersed contexts.

1. **Security**:

Any distributed system must prioritize security, but this is especially true when managing user data that is sensitive. Advanced security features like OAuth 2.0 authentication and end-to-end encryption should be used in order to surpass proficiency. Additionally, granular access control and threat detection throughout the whole application stack are guaranteed by utilizing cloud-native security services like AWS IAM (Identity and Access Management) and Google Cloud Identity-Aware Proxy. Draw It or Lose It may achieve unmatched levels of security and compliance across a variety of operating systems and platforms by adopting a defense-in-depth strategy.