

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 | May 22 | Danielle Williams | <Brief description of changes in this revision> |

**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 client, "The Gaming Room," aims to develop a web-based game that can be accessed on multiple platforms. The company’s goal is to ensure the game runs on other operating systems, including macOS, Linux, and Windows. They want to base this new game on "Draw It or Lose It," which currently only operates on Android.

## Requirements

  - Broader audience reach

  - Higher sales and increased user engagement

Requirements:

- Video Editing Software

  - Must be high-quality

  - Available only for Macs

  - Purpose: Create video clips for training and educational use

## [Design Constraints](#_2et92p0)

Game Structure:

- Support for Multiple Teams:

  - Each team consists of several players

  - Names of games and teams must be unique

  - Only one game can exist in memory at any given time

Implementation:

- Incorporate unique identifiers and IDs for:

  - Games

  - Teams

  - Players

## [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 class diagram for "Draw It or Lose It" illustrates key object-oriented programming principles. Each class (Game, Team, Player) manages its own data and behaviors. These classes represent high-level concepts without detailing the gameplay logic. A Game object aggregates multiple Team objects, and each Team object contains multiple Player objects. Each Player and Team has a unique ID to ensure their uniqueness.

**"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** | macOS is user-friendly for developers and based on Unix, which provides stability and good development tools. However, it is more expensive and does not support some server-side technologies natively. | Linux i is open-source, flexible, and efficient with resources. Many people use it in cloud and production settings. | Windows is widely supported and has a user-friendly interface, but may require more resources and licensing costs for enterprise hosting. | Mobile devices have limited storage but are great for interacting with clients. Android is more flexible and customizable, while iOS requires strict rules for apps in the App Store. |
| **Client Side** | For macOS development, you typically need Xcode and knowledge of Swift or Objective-C. Testing can take longer because of Apple’s rules. | Linux support is less common but can work through web-based delivery. It offers flexibility, but you may need extra testing across different Linux versions. | Windows provides the widest compatibility. Development is well-supported with Visual Studio and other tools, but quality checks must consider many Windows versions. | Kotlin or Java for Android, and Swift for iOS. Alternatively, you can use cross-platform tools like Flutter or React Native. It’s also important to test on various screen sizes. |
| **Development Tools** | Xcode, Swift, RubyMine, and cross-platform environments like VS Code or IntelliJ | Eclipse, NetBeans, VS Code, and command-line tools like gcc/g++ and Git | Visual Studio, Unity, and JetBrains IDEs | Android Studio and Xcode, Flutter, React Native, and Unity |

## 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**: Windows works well with many devices and supports important development tools like Visual Studio. It also connects easily with Microsoft services. Many developers already know how to use Windows, which makes it easier to learn and work on different platforms.

**Operating Systems Architectures**: The Windows operating system has a hybrid design that works well with .NET applications, making it suitable for multi-tier web applications.

1. **Storage Management**: A relational database like Microsoft SQL Server or PostgreSQLshould be used to manage structured data such as game sessions, teams, and player statistics. These databases work well on Windows and provide strong features like recording transactions, indexing data, and ensuring high availability through clustering or replication.
2. **Memory Management**: Windows manages resources effectively by using virtual memory, demand paging, and process prioritization. We can analyze and improve memory usage with tools like Task Manager and Performance Monitor.
3. **Distributed Systems and Networks**: To let players on different platforms play together, we can use an API. You can put this software on Azure, which is Microsoft's cloud service. The Windows servers help manage everything, making sure that players can easily connect and play the game together.
4. **Security**: - Utilize Windows Security features- Implement secure logins with OAuth 2.0 or JWT. Encrypt data both in storage and during transfer. Establish security policies to guard against SQL injection, CSRF, and XSS attacks.