

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 5**

[**Recommendations 8**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/26/2025 | Angel Lopez | 1st Revision of Project One Template |
| 2.0 | 02/09/2025 | Angel Lopez | 2nd Revision of Project one Template |
| 3.0 | 02/23/2025 | Angel Lopez | 3rd Revision of Project one Template |

**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 project aims to create a web-based game inspired by the existing mobile game “Draw It or Lose It,” which is currently exclusive to Android devices. The new version will be accessible across multiple platforms, enhancing its reach and user engagement. The game will consist of multiple teams, each comprising several players. The gameplay is structured into four rounds, with each round lasting one minute. During these rounds, a picture is drawn from a curated library of images.

## 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)

**1. Cross-Platform Compatibility**  
The game must be designed to operate seamlessly across multiple platforms, ensuring that players can engage with it regardless of their device or operating system.

**2. Team Composition**  
Each team participating in the game should consist of multiple players, allowing for collaborative gameplay and strategic team dynamics.

**3. Single Instance Limitation**  
Only one instance of the game can be active at any given time, which means that players cannot create or join multiple simultaneous games.

**4. Unique Naming System**  
Both game and team names must be unique within the game’s ecosystem. This requirement enables users to verify the availability of a name before selecting it for their team, preventing duplication and confusion.

## [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)

In this diagram, we have a structure that involves three main classes: **Game**, **Team**, and **Player**. These classes are related through an overarching class called **Entity**, which serves as the superclass. This relationship can be effectively illustrated using Unified Modeling Language (UML), where we denote inheritance and composition relationships.

**Inheritance Structure:**

**The class Entity acts as a superclass from which the other classes inherit properties and behaviors. This means that both Team and Player derive from the Entity class, gaining its attributes and methods.**

**Aggregation Relationships:**

**The relationship between these classes can be characterized as “has-a” relationships, which in UML terminology is referred to as aggregation.**

**Specifically, we can say:**

**A Game has a reference to one or more Teams.**

**A Team has a reference to one or more Players.**

**A GameService has references to multiple instances of Games.**

**Instance References:**

**When we say that a user “has a” reference to another class, it indicates that an instance of one class contains or points to an instance of another class. For example:**

**An instance of the GameService class will hold references to instances of the Game class.**

**Each instance of the Game class will maintain references to instances of the Team class.**

**Each instance of the Team class will keep references to instances of the Player class.**

**"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** | Macs provide a stable and reliable environment for hosting web-based applications, their higher cost and limited software compatibility may be drawbacks depending on specific project needs. | Linux offers numerous advantages such as cost-effectiveness, security, and flexibility in application support. It also presents challenges like case sensitivity and limited compatibility with Windows applications that should be considered when hosting a web-based software application. | Windows offers a user-friendly environment with strong development support for hosting web-based software applications, its cost and security vulnerabilities may pose challenges that need careful consideration. | Mobile devices offer significant advantages in terms of accessibility and user interaction, their limitations in performance and hardware capabilities make them less suitable for hosting demanding web-based software applications. |
| **Client Side** | When developing software for multiple client types on Mac, key considerations include cost (budgeting for development tools, licenses, and third-party services), time (ensuring compatibility and thorough testing), and expertise (requiring a skilled team in macOS development with knowledge of Swift or Objective-C and various client frameworks). | The cost includes budgeting for development tools, infrastructure, and ongoing maintenance. The project timeline may extend due to the need for extensive testing across different client environments and configurations, which affects the time consideration. Additionally, developers must possess a deep understanding of Linux systems, various client architectures, and potentially different programming languages or frameworks that are compatible with these environments, which relates to the expertise consideration. | When developing software for multiple Windows clients, key considerations include cost (budgeting for tools, licenses, and maintenance), time (project timelines may extend due to compatibility testing), and expertise (requiring a skilled team familiar with various environments and frameworks). Additionally, developers must address user experience variations to ensure seamless functionality. | When developing software for multiple mobile platforms, key considerations include **cost**, which encompasses budgeting for diverse platform development and ongoing maintenance; **time**, as the timeline may extend due to cross-platform compatibility testing; and **expertise**, requiring a skilled team in various programming languages and frameworks. Additionally, understanding user experience across devices necessitates investment in design and usability testing. |
| **Development Tools** | To develop software for macOS, developers typically use Swift and Objective-C for native applications. For cross-platform apps, languages like Python, JavaScript (with frameworks like Electron), and C++ are popular. Integrated Development Environments (IDEs) such as Xcode are crucial for native app development, while Visual Studio Code and Atom serve well for web-based or cross-platform projects | To build software for deployment on Linux, several programming languages and tools are used. The relevant programming languages include C, C++, Python, Java, and Go. Common Integrated Development Environments (IDEs) used are Eclipse, Visual Studio Code, and JetBrains IntelliJ IDEA. Tools such as GCC (GNU Compiler Collection) for compiling code, Make for build automation, and version control systems like Git are essential. | To develop software for Windows, popular programming languages include C#, C++, and Python due to their compatibility with the operating system. Integrated Development Environments (IDEs) like Microsoft Visual Studio and JetBrains Rider aid in coding, debugging, and testing. Tools such as Windows Installer and WiX Toolset are crucial for creating installation packages, while version control systems like Git manage code changes. | The relevant programming languages and tools used to build software for deploying on mobile devices are: **Java or Kotlin for Android, Swift or Objective-C for iOS, React Native, Flutter, Xamarin, JavaScript, Dart, C#, Android Studio, Xcode, Visual Studio, Gradle, CocoaPods, and Jenkins**. |

## 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 optimal operating platform for The Gaming Room to expand “Draw It or Lose It” to additional computing environments is Windows. This choice is advantageous due to the abundance of Integrated Development Environments (IDEs) available for Windows, which facilitates development and enhances productivity.
2. **Operating Systems Architectures**: Microsoft Windows is a graphical operating system created and distributed by Microsoft. It offers users the ability to manage files, execute software applications, engage in gaming, view videos, and access the internet.
3. **Storage Management**: Windows allows you to save files directly to the cloud on some of their newer operating systems. You may also save all your files onto local drives like a hard drive or a solid-state drive on your computer directly.
4. **Memory Management**: Windows 10’s Storage Sense feature facilitates the organization and management of photos from the game “Draw It or Lose It,” as well as those belonging to game players. This functionality enables users to consolidate these items in a secure memory space.
5. **Distributed Systems and Networks**: Network-based multi-user interaction systems, such as network games, typically feature a shared database that allows players, who are geographically dispersed, to interact with one another over a network. At present, developers of network games must create both the shared database and the inter-player communication mechanisms from the ground up.

1. **Security**: Windows includes integrated security protection software designed to safeguard user data and information. However, it is advisable to consider additional security solutions for enhanced protection.