

<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 |
| --- | --- | --- | --- |
| 2.0 | <11/26/2023> | < Mohamed Elnarzougui> | <Revision and constraints for the Draw It or Lose It game program> |

**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 software design challenge at hand involves expanding The Gaming Room's Draw It or Lose It game application to multiple platforms, including Linux, Mac, Windows, iOS, and Android.

## [Design Constraints](#_2et92p0)

Developing Draw It or Lose It game in a web-based distributed environment brings challenges, such as responsive design, scalability demands efficient architecture also security measures, device diversity, and backward compatibility add complexity

## [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 Entity class establishes a connection between the Game, Team, and Player classes, indicating that they all inherit information from the Entity class. In UML, this relationship is visualized through inheritance, designating Entity as the superclass. Analyzing their interconnection, we observe that Team and Player exhibit a "has a" relationship type. Specifically, Game has a Team, and GameService has a Games. In UML, this association is referred to as aggregation (HAS-A), signifying that one class instance holds a reference to another class instance. In practical terms, when a user "has a," it means they possess an instance of one class while having a reference to an instance of another class. Examining the UML diagram, we notice that GameService has a reference to Games, Games a reference to Teams, and Teams references of Players.

**"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** | Robust capabilities, smaller-scale usage.  Advantages: Integration with macOS tools.  Weaknesses: Not common for large-scale setups, potential macOS Server costs | Various deployment options, widely used  Advantages: Open-source, scalability  Weaknesses: Deeper expertise needed, limited support for certain software | Robust with IIS support, common in enterprise  Advantages: User-friendly, integration with Windows development.  Weaknesses: Licensing costs, higher resource requirements | Mobile devices demand server-side architectures capable of handling high levels of user interaction.  Advantage: Server-side infrastructure enables the implementation of push notifications.  Weakness: Server-side components must address security concerns related to data transmission and storage |
| **Client Side** | Align with macOS design principles, use Xcode and Swift | Accommodate various desktop environments, open-source compatibility | Consider diverse Windows versions, utilize Visual Studio and C# | Implement responsive design, use cross-platform frameworks like React Native or Flutter |
| **Development Tools** | Languages: Swift, Objective-C Tools: Xcode | Languages: JavaScript, HTML, CSS Tools: VS Code, Sublime | Languages: C#, JavaScript, HTML, CSS Tools: Visual Studio | Languages: JavaScript, Dart (Flutter), Swift (iOS), Java/Kotlin (Android) Tools: Platform-specific IDEs (Xcode, Android Studio), VS Code, Flutter/Dart SDK |

## 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**

Recommendation: Linux

Justification: Linux provides a cost-effective and scalable solution for hosting web-based applications. The platform's stability and proven scalability align well with the expansion goals of Draw It or Lose It.

1. **Operating Systems Architectures**:

Linux supports various file systems, ensuring compatibility with diverse storage needs. Additionally Linux architecture simplifies deployment and maintenance,

1. **Storage Management**:

Implement a combination of local and cloud-based storage solutions, local storage on servers for quick access to frequently used data and employing cloud storage for scalability and redundancy ensures optimal performance and reliability for Draw It or Lose It.

1. **Memory Management**:

Linux utilizes virtual memory management techniques, allowing efficient use of both physical and virtual memory. Draw It or Lose It can benefit from this

1. **Distributed Systems and Networks**:

Achieving communication between various platforms can be accomplished through the development of a distributed system using web services or APIs. RESTful APIs can facilitate seamless communication between the Draw It or Lose It server and different clients.

1. **Security**

Linux provides a robust foundation with built-in security features. To protect user information across platforms, implement secure communication protocols such as HTTPS for data in transit.