

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 | 10/20/2024 | Heather Moore | Changes client is requesting are to make a web-based version of the application that is already in use. |

/

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is requesting a web-based version of their game, Draw It or Lose It, based on their current application. They have provide a list of design constraints.

## Requirements

Technical Requirements:

-serves multiple platforms

-only one instance of a game can exist in memory any given time.

-create unique identifiers for each instance for a game, team, player

Business requirements:

-market their consumer base

-stay within budget

-increase revenue

## [Design Constraints](#_2et92p0)

-must be compatible for multiple platforms

-have the ability to have one or more teams involved

-each team will have multiple players assigned to it

-game and team names must be unique to allow users to check whether a name is in use while choosing a team name

-only one instance of memory of the game can exist in memory at any given time

-create unique identifiers for each instance for a game, team, or player

## [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)

This UML class diagram outlines the Domain Model for The Gaming Room’s game “Draw it or Lose it”.

-ProgramDriver: entry point

-SingletonTester: tests singleton design pattern

-Entity: base class for common behaviors and attributes

-GameService: handles operations, instances (game, team, player)

-Team: team-related operations and players

-Player: individual users

OPP:

Inheritance: Entity

Association: GameService > Game> Team> Player

Singleton: Singleton tester

This model demonstrates structure within the entities and the interaction of the entities.

**"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** | Dependable servers, developer friendly environment, and high-cost maintenance | High-security standards, extremely customizable, lack of software compatibility | Wide compatibility, user-friendly interface, security vulnerabilities | Instant access, data synchronization, security concerns |
| **Client Side** | Easy to learn using interface, cost depends on the requirement for the client, different expertise needed for different clients | Easy to navigate web development, cost depends on the requirement for the client, different expertise needed for different clients | Test across multiple browsers, cost depends on the requirement for the client, different expertise needed for different clients | Must have mobile app design knowledge, cost depends on the requirement for the client, different expertise needed for different clients |
| **Development Tools** | Languages:  Python, Java, JavaScript.  IDEs:  PyCharm, Jupyter, Visual Studio Code | Languages:  Python, R  IDEs:  Jupyter | Languages:  Batch, HTML, CSS  IDEs:  Visual Studios, NetBeans, PyCharm | Languages:  Objective-C, Swift, Java, Kotlin  IDEs:  Android Studio, Eclipse, Komodo Edit, NetBeans |

## 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 is highly recommended for the Draw It or Lose It game. It has software that is compatible with multiplatform. It has a wide range of capabilities, first party support, and is great for web-based applications.
2. **Operating Systems Architectures**: kernel with low-level device drivers, Realtime (RT) APIs
3. **Storage Management**: < Storage Widows had a cloud available for storage, manage devices through Storage Spaces.
4. **Memory Management**: < Windows is highly functioning and offers a variety of memory management. RAM temporarily stores data allows quick access data by utilizing its resources
5. **Distributed Systems and Networks:** By using a web-based application, for example a cloud, you can connect accounts across various platforms without having to meet certain requirements. You can access this web-based application from any device, including but not limited to, mobile phone, desktop, tablet or laptop.
6. **Security**: Creating a multilayer login system for users will ensure they are not at risk for a security breech. Implementing various restrictions on character and symbols, capitalized letter and using number will make it more difficult for an account breech. You could also have the user change their passwords once a week and never use the same password more than once.