

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 | 12/10/2023 | Mikaela Spence | Recommendations section updated for the development of the Draw It or Lose it application. |

**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 is looking to move their Android game “Draw It or Lose It” to a web-based platform. “Draw It or Lose It” is like the game “Win, Lose, or Draw” from the 1980’s, except there will be a library of stock clues. Players are on multiple teams, and there are four rounds of gameplay that are a minute in length. After the time limit, if the puzzle is not solved by the active team, the other teams are allowed one guess each within 15 seconds to earn the point.

## Requirements

Business Requirements

1. Available on a web-based platform.
2. Stays within The Gaming Room’s budget.
3. Marketed to a larger customer base by being on the web.

Technical Requirements

1. Web-based functionality that mirrors current Android app.
2. Must support multiple teams with multiple players.
3. Game and team names must be unique.
4. Only one instance of the game can exist at a time.

## [Design Constraints](#_2et92p0)

1. Project should mirror current Android app in functionality and style.
2. Web-based application to be used across multiple platforms.
3. There must be the capability of having multiple teams with multiple players.
4. Only one instance of the game should exist at a time.
5. Game and team names must be unique with a check to see if names are already in use.

## [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 diagram shown below displays the classes for the application. The Entity class is the base superclass and includes variables that are inherited by the other classes. The Game, Team and Player classes are the sub classes to Entity. The Game class consists of one or more Teams, and each Team class can have one or more Players. The GameService class manages the lifecycle of a Game instance and references multiple Game objects, which is a compositional relationship. The Game class has a compositional relationship with the Team class, which has a compositional relationship with the Players class. The Program Driver class holds the main function. A singleton instance is used to make sure that only one instance of the GameService class exists at a time. The Program Driver adds games, teams and players and is dependent on the Singleton class.

This diagram shows several OOP principles. Inheritance is present in that the other classes inherit from the Entity parent class. Encapsulation is present with the GameService class, where only one game instance exists. Abstraction is present because the details of implementation for teams, games, and players are not publicly available.

**"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** | **Characteristics:**  A stable and secure Unix-based environment  **Advantages:**  Unix based  Simple, easy to use GUI  Easy to access and use command line features  **Weaknesses:**  Expensive to maintain servers  Not as commonly used as Windows, so fewer available applications | **Characteristics:**  An efficient web-server platform  **Advantages:**  A cost-effective option  Wide range of software available  **Weaknesses:**  Can be more difficult to navigate than Mac or Windows | **Characteristics:**  A more expensive, but easy to use platform  **Advantages:**  Simple user interface  Supports more applications than Linux or MacOS  **Weaknesses:**  More expensive servers and licenses | **Characteristics:**  Easy to use but varied in abilities and has limitations with server hosting  **Advantages:**  More flexible and mobile access to server-side applications  Integration with server-side platforms with Software Development Kits  **Weaknesses:**  Limited resources which may affect performance  Reliance on network connectivity  Less secure |
| **Client Side** | **Cost:**  Macs tend to cost much more than the other options  **Time:**  If not familiar with the GUI, may take a little time to learn but is designed to be user-friendly  **Expertise:**  Designed to be user-friendly | **Cost:**  Very low cost comparatively  Open-sourced and free  **Time:**  **Expertise:**  Requires a lot of knowledge and skill compared to other platforms, however more user-friendly distributions such as Ubuntu can help | **Cost:**  Cheaper than Mac, but more expensive than Linux  **Time:**  Lower than Mac or Linux  **Expertise:**  User-friendly interface makes it easy to pick up  Developers may want to know Visual Studio and .NET | **Cost:**  Varies. iOS tends to be more expensive compared to Android  **Time:**  May take more time to integrate  **Expertise:**  Typically low, as mobile devices are designed with users in mind  Development requires knowledge of platform-specific languages |
| **Development Tools** | **Languages:**  Helpful to know HTML, CSS, Swift, C++/C and Javascript  **Tools/IDEs:**  Xcode is the official IDE | **Languages:**  C/C++, Python, Java, JavaScript  **Tools/IDEs:**  GCC/G++, Visual Studio, Ruby | **Languages:**  JavaScript, C#/C++, Python, Java  **Tools/IDEs:**  Visual Studio, .NET, Eclipse, Pycharm | **Languages:**  Swift, Objective-C, Java, Kotlin, Javascript  **Tools/IDEs:**  Xcode, Android Studio, Flutter and Dart, React Native, PHP |

## 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 operating system I would recommend for this project is Windows. Windows is widely used and is a cost-effective option. There are many development tools available for the operating system as well.
2. **Operating Systems Architectures**: Windows has a user mode with a user-friendly GUI, as well as a kernel mode that users do not need to access which is involved in memory management, processing, and networking.
3. **Storage Management**: Windows consists of a hard disk and Storage Sense to help manage storage of files. There is also the option for a connection to a cloud storage system such as Microsoft Azure.
4. **Memory Management**: Windows has built-in Memory Management as a tool. There is also a Disk Cleanup tool that can help with memory management. Windows uses its physical memory, RAM, and can also connect to cloud services to store information and save physical memory space.
5. **Distributed Systems and Networks**: There will need to be a strong server connection in order to handle things such as outages and maintain connectivity among thousands of users. Using a cloud service can assist with dependencies and helps handle the networks and distributed systems. Using a cloud-based platform such as Microsoft Azure would be useful.
6. **Security**: Windows has built-in security software. In addition to this, added security would be provided by using Azure. It has VPN capabilities, and you can set access to the IP as Whitelist only, adding further protections. Finally, encryption can be used to protect data such as user information and passwords.