

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 | 03/14/2023 | Matthew Cochrane | Integrated superclass ‘Entity,’ addTeam() and addPlayer() methods |

**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)

This iteration of Draw It or Lose It provides the user with the ability to both add and view instances of games, teams, and players. The previous iteration only provided the ability to add and view games, but this latest version has more necessary features leading to the final game.

## Requirements

* *One or more teams*
* *Each team will have multiple players*
* *Game and team names must be unique*
* *Only one instance of a game may exist in memory at any given time*

## [Design Constraints](#_2et92p0)

* Must be compatible with all major operating systems (Mac, Windows, Linux, Android, iOS)
* Cross compatibility between operating systems, players on Android may play with Mac users, etc.

## [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 GameService class contains most of the logic of the program. The Entity class is a superclass with the Game, Player, and Team classes being the subclasses of Entity. The ProgramDriver class contains the main() method where requests can be made to generate instances of the different Entity subclasses. The SingletonTester class contains the testSingleton() method, which verifies that only one instance is created at any given time.

**"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 must 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:**  - Seamless integration with other Apple devices  - Less prone to viruses and malware  - Built-in web server  **Advantages:**  - Unix based, developer friendly  - Built-in development tools  - Highly portable  **Disadvantages:**  - More expensive in general  - Limited upgrade options  - Limited third-party support | **Characteristics:**  - Open-source operating system, easy to manipulate  - High security  **Advantages:**  - Completely free, no associate license fees  - High level of control and customizability  **Disadvantages:**  - Higher skill floor, requires more base knowledge than other platforms  - Smaller user base, may have compatibility or support issues at scale | **Characteristics:**  **-** Widely used, familiar user interface  - Built-in security features such as Defender and Firewall  - Highly upgradeable  **Advantages:**  **-** Easy to use  - Broad third-party support  - Highly upgradeable  **Disadvantages:**  - Can be expensive  - Less secure than others in general | **Characteristics:**  - Limited processing power and storage  - Smaller screen  **Advantages:**  - Highly portable  - Used by almost everyone  - Hardware features can enhance functionality (GPS, camera)  **Disadvantages:**  - Large variety of devices may require separate development  - May not meet some hardware requirements |
| **Client Side** | **Cost:** Mac is usually the most expensive option given the necessity for proprietary hardware and support.  **Time:** Development time can increase based on regulatory hurdles from Apple.  **Expertise:**  Mac has specialized development tools that are not easily transferrable from other knowledge bases. | **Cost:** The cheapest option given that Linux is an open-source platform.  **Time:** Development time can increase with testing compatibility on the numerous distributions of Linus.  **Expertise:**  Deep knowledge of the command line interface is required, the GUI is lacking compared to the other platforms. | **Cost:** Windows usually involves licensing fees for the OS itself as well as other tools.  **Time:** Development time can increase with testing compatibility with different versions of Windows and various hardware configurations.  **Expertise:** Must be able to use PowerShell and be able to translate C#. | **Cost:** External specialized hardware is usually needed.  **Time:** Development time can increase with testing compatibility on different operating systems and screen sizes.  **Expertise:**  Knowledge of mobile-specific development is required. |
| **Development Tools** | **Programming Languages:**  - Swift  - Objective-C  - PHP  - JavaScript  - HTML  **Tools:**  - Xcode  - Visual Studio Code  - Homebrew  - Git  - Docker | **Programming Languages:**  - PHP  - HTML  - C++  - Python  - Ruby  - JavaScript  **Tools:**  - Visual Studio Code  - Eclipse  - Apache  - Nginx  - MySQL  - Git  - Docker | **Programming Languages:**  **-** C#  - Python  - JavaScript  - Visual Basic  - C++  - HTML  **Tools:**  - Visual Studio  - Anaconda  - Git  - Docker  - Microsoft SQL Server | **Programming Languages:**  - JavaScript  - Objective-C  - HTML  - Kotlin  - Swift  **Tools:**  - Xcode  - Android Studio  - Visual Studio Code  - React Native  - Ionic  - Apache Cordova |

## 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**: Android
2. **Operating Systems Architectures**: There are multiple layers that makeup the architecture of an Android application:
   1. Application Layer: Comprises the user interface such as buttons or other screen objects that the user interacts with
   2. Framework Layer: A set of high-level APIs that the back-end of the application uses to interact with the operating system.
   3. Libraries Layer: A set of pre-built libraries that applications may use to increase functionality.
   4. Android Runtime Layer: Executes code. Includes the Dalvik Virtual Machine and the Android Native Development Kit
   5. Linux Kernel Layer: Base layer, provides memory management, process management, and device drivers.
3. **Storage Management**: Cloud storage should be used since local storage is limited and the application requires a network connection regardless.
4. **Memory Management**: Android uses many memory management techniques that other operating systems use as well, such as garbage collection, heap management, memory compression (ZRAM), and memory mapping. In addition to this, the cornerstone of memory management in Android is Binder. Binder allows for processes to communicate directly to each other as if they were the same program.
5. **Distributed Systems and Networks**: The primary distributed software that I recommend for The Gaming Room to use for Draw It or Lose It is Microsoft Azure. It is a robust cloud computing platform that has the appropriate framework to handle any kind of cross-platform applications.
6. **Security**: To protect user information, no one measure is good enough, but many together will construct a Fort Knox that I am certain The Gaming Room will be satisfied. A lot of these features are built-in to Android, but they are worth mentioning regardless. Android utilizes encrypted storage, secure boot, and Google Play Protect. Users should also be guided by The Gaming Room to enroll in two-factor authentication.