

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 11/8/23 | Daniel Collins | |  | | --- | | Initial creation of software design document for Draw It or Lose It web-based game development. | |
| 1.1 | 11/24/23 | Daniel Collins | Evaluation and Recommendations updated. |
| 1.2 | 12/7/23 | Daniel Collins | Evaluations and Recommendations updated to reflect feedback received. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to evolve their Android game, Draw It or Lose it into a multi-platform web-based game. We are tasked with delivering an intuitive multi-platform game that users can access regardless of their device or platform. We will design a system that creates a singular game instance with unique identifiers for games, teams, and players.

## Requirements

## [Design Constraints](#_2et92p0)

There are several constraints that must be addressed; firstly, the game must handle concurrent user sessions across different platforms, so we need to ensure data consistency. The unique naming requirement requires us to create a centralized management system to handle this. To enforce the singleton pattern, the design must prevent multiple game instances. These constraints will dictate the design patterns and technologies required to deliver this product.

## [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)

“Entity” is the base class where “Game”, “Team”, and “Player” inherit from, displaying the OOP principle of reusability by allowing shared attributes. “GameService” class manages game instances and enforces the singleton patter, which makes sure there is only one instance of the game. “0…\*”, or one to many, indicates that “GameService” can manage many “Game” instances and can have many “Teams” which can have many “Players”. Encapsulation is also shown where attributes are kept private in classes.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | UNIX provides reliability due to having fewer viruses and malware, but the Mac hardware is more expensive. | Cheaper, open-source, and can be molded to fit specific needs but can be more difficult to use. | Widely used, so its widely compatible, but can be subject to security threats so it requires more maintenance. Licensing costs $6000+ | Do to their limited processing power and storage capacity, mobile devices are not suitable for this role. |
| **Client Side** | Mac’s hardware is more controlled which should simplify software development but we may have higher costs for their programs. | More open and flexible, but there are many Linus distributions and we need to develop with this variability. | Larger user base means more resources are available, but we must make it compatible with different versions of Windows. | Almost everyone uses a mobile device so we can reach a wide user base but need to design around different platforms. |
| **Development Tools** | Mac using their own IDE called Xcode so we need to become familiar with this program. The program is free, but costs $99 to publish the app. | Open source OS means open source IDEs, so we will be using Eclipse, PyCharm, and Code::Blocks. All of these programs are free. | The most common IDE for Windows is Visual Studio, as well as Visual Studio Code, and .NET SDK. Visual Studio Enterprise costs $6000. | Xcode for Apple and Android Studio for Android platforms, there are cross platform IDEs that can be used to speed up development time. |

## Recommendations

1. **Operating Platform**: Linux would suit our needs as it is a cross platform OS. The OS is stable, has many resources available for development, and can run on different hardware systems.
2. **Operating Systems Architectures**: The architecture is modular, allowing components to be removed or added based on needs, providing the flexibility needed to develop across platforms.
3. **Storage Management**: Linus uses ext4 for file systems management and LVM for disk space optimization.
4. **Memory Management**: This OS memory management system uses cache memory to store commonly used data and slab allocation for kernel objects.
5. **Distributed Systems and Networks**: Linux can use the networking protocol TCP/IP for reliable data communication and APIs can manage communication across systems.
6. **Security**: Encrypted data protocols like TLS/SSL can be used to secure communications as well as security checks that will review code and test for vulnerabilities. Using two-factor authentication and controlling assess will also improve security. TLS/SSL certificate costs can range from $8 to $1000, depending on the number of domains and validations.