

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 | 05/16/2023 | Arica Bryant | Updated the Executive Summary, Requirements, Design Constraints, and Domain Model sections |
| 1.1 | 05/31/2023 | Arica Bryant | Updated the Evaluation section |
| 1.2 | 06/14/2023 | Arica Bryant | Updated the Recommendations section |

**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)

Our clients, The Gaming Room want to expand their game, Draw It or Lose It, which is currently only available as an Android app, into a web-based game that will be compatible with multiple platforms. The game is inspired by the 1980s TV game Win, Lose, or Draw.

## Requirements

* The game must be a web-based application and compatible with multiple platforms.
* A single game must have the ability to include one or more teams.
* Each team should include multiple players.
* The names of the Game and Team must be unique.
* Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

1. Since the application will need to be compatible on multiple platforms, testing will need to be done for the specified platforms to make sure the game is cross-compatible.
2. The project is specifically a web-based application and will need to be designed so that it can run in browser windows.
3. Only one instance of the game can exist in memory at a given time which means a Singleton pattern should be used in the design process.

## [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 is the base or “parent” class that holds the main methods and attributes used by the other classes. From the diagram, the white arrow leading away from the Game, Team, and Player classes shows that they are all inheriting from the Entity Class. The “0…\*” symbol connecting some of the classes indicates that one class can have 0 or more instances related to instances in the other class. The ProgramDriver class on the diagram is shown to use and require the SingletonTester class. This helps to test that only one instance of our game is running as required in the design constraints. This UML Diagram will be used to construct the program.

**"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** | MacOS Server, which used to be available as a separate product, has been discontinued, and server features are now integrated into new MacOS versions. However, using alternative server applications to replace the removed features may result in additional costs.  MacOS has hardware limitations and is not commonly associated with game development.  It is not designed for high scalability, making it challenging to maintain a server with thousands of players. On the positive side, MacOS is generally considered more secure and reliable compared to systems like Windows. | Linux is a popular choice for hosting webservers. It is customizable, stable, less susceptible to viruses and offers high scalability.  Linux offers a free open-source software called Ubuntu that can be utilized for server applications.  Some disadvantages of using Linux include that not all versions offer longtime support and learning to use the software requires time. | In terms of security, windows systems are less secure than other operating systems.  The Windows OS is compatible with Windows Server and Azure. Up-to-date documentation is always readily available.  Windows Server is not a free application so the costs of the product and server hosting should be noted. | Mobile devices are not typically used for server hosting web-based applications. Compared to OS systems like Mac, Linux, and Windows, there are less resources available and the website will struggle with scalability.  The costs for building a web-based application are more affordable than building an app.  However, web-based applications are less secure than mobile apps and tend to run slower. |
| **Client Side** | MacOS is more expensive than operating systems such as Windows and Linux.  Compared to OS systems such as Linux and Windows, Macs are not commonly used for development. Developers may need to learn how to utilize an unfamiliar system. | Linux is an open-source free operating system so the costs of development are low and affordable. | Various versions of Windows may be required for testing, and these must be purchased.  Windows systems are very user-friendly, and the interface is easy to use. | Care will need to be taken when working with different screen sizes and orientations for mobile devices such as tablets and phones. |
| **Development Tools** | Knowledge of HTML, CSS, and JavaScript is required for developing an application for MacOS. | Knowledge of HTML, CSS, and JavaScript is also required for developing an application for Linux. Linux also has PHP and MySQL support. | Windows is a popular choice when it comes to game development.  Knowledge of HTML, CSS, and JavaScript is required for Windows development.  Windows is compatible with most development tools and programming languages. | Knowledge of HTML, CSS, and JavaScript is required for mobile web-based applications. |

## 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**: Based on characteristics, strengths, and weaknesses of various operating platforms, I recommend the use of the Linux Ubuntu server operating system based on its affordability, stability, and security.
2. **Operating Systems Architectures**: The Linux operating systems architecture is made up of five different layers. There is the Kernel, System Libraries, System Utility Programs, Hardware Layer, and the Shell. The Kernel is the core of the Linux OS and is responsible for carrying out essential functions. The system libraries hold the required functions for the operating system such as system calls or memory management processes. System utility programs utility holds actions required for specific jobs such as command-line prompts entered by the user or program. The hardware layer consists of devices such as the CPU, HDD, and RAM. Lastly, there is the shell that acts as an interface between the kernel and the user.
3. **Storage Management**: For storage management, cloud storage options such as AWS should be considered. AWS provides multiple options for cloud storage and is a more reliable option for keeping data preserved. Storage costs can be optimized based on how much space is needed and how the data is used. Cloud storage can be accessed anytime as soon as it is needed.
4. **Memory Management**: The Linux operating system utilizes memory management techniques such as demand paging and virtual memory to ensure applications are functioning smoothly. Demand paging ensures that only necessary data is brought into memory if the executing process requires it. Virtual memory can be utilized as backup when the main memory runs low improving system performance.
5. **Distributed Systems and Networks**: Serverless cloud providers such as AWS Lambda can be used alongside other AWS services to ensure effective client/server communication across multiple operating platforms. A user accessing the Draw It or Lose It game application makes a REST API call to the Amazon API gateway. This triggers AWS Lambda which runs the necessary code required to retrieve the required data and returns this information back to the user from an Amazon database.
6. **Security**: I recommend that developers take advantage of the security features offered by the Linux Ubuntu OS. Linux OS implements authentication features such as password protection, data encryption, and access control to specific files to provide users with proper security. Linux Ubuntu also features automatic updates and 10 years of support for LTS (Long Term Support) versions. System updates for Linux should be applied when available because they may contain vital security patches.

**References**

*About macos server 5.7.1 and later*. Apple Support. (2023, May 8). <https://support.apple.com/en-us/HT208312>

JavaTPoint. (n.d.). *Linux memory management*. www.javatpoint.com. <https://www.javatpoint.com/linux-memory-management>

*Linux architecture - detailed explanation*. InterviewBit. (2022, June 10). <https://www.interviewbit.com/blog/linux-architecture/>

Ortiz, P. C. (2022, September 7). *Pros and cons of mobile application vs. Mobile Web Application*. Flexisource. <https://flexisourceit.com.au/resources/blog/pros-and-cons-of-mobile-app-vs-mobile-web-app/>

Tucakov, D. (2021, December 9). *Linux vs. Microsoft Windows Servers, the ultimate showdown*. phoenixNAP Blog. <https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers>