

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 2.0

## 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  2.0  3.0 | <mm/dd/yy>  2/9/2024  2/25/2024 | Alexandra Delarosa  Alexandra Delarosa  Alexandra  Delarosa | Added executive summary, characteristics of each OS and OS recommendations.  Added more characteristics of different operating systems and their advantages and disavadantage.  Expanded upon 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)

The software design problem that CTS is facing is developing a web-based game version of “Draw It or Lose It” for their client “The Gaming Room”. The client has requested various software requirements. The solution is to develop a Java application and implement software design patterns such as the singleton pattern and iterator pattern.

## Requirements

* *Ability to have one or more teams in a game.*
* *Multiple players assigned to each team.*
* *Game and team name must be unique.*
* *Only one instance of the game exists in the memory.*

## [Design Constraints](#_2et92p0)

* The game must be developed in a web-based environment. This implies that the game needs an internet connection to access the game and experience network latency.
* The game should be designed to handle a large number of uses at any given time and should be scalable.
* The game must be compatible with various web browsers and operating systems to be accessed.

## [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)

In the UML class diagram, the Entity sets a foundation for the Game, Team, GameService and Player class, which is where they inherit information such as “id” and “name”. This is an example of the principle of inheritance. The GameService class contains list of games and methods such ass addGame and addGame to manage the games, which displays the principle of polymorphism. It follows the singleton pattern which ensures an instance is accessible in the application. The Game class represents the game instance and contains information about the game’s state and teams. The team class represents a team in a game and manages a list of players. This class displays the principle of aggregation because it aggregates multiple Player objects. The player class represents an induvial player within a team.

**"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** | Provides a reliable and stable web hosting environment.  Has a limited range of hardware options for servers compared to other platforms.  Flexible terminal however, less preferred.  Higher hardware costs | Most preferred for server-side hosting.  Known for being secure.  Affordable with no licensing fees  Offers a wide range of distributions.  Highly flexible and customizable.  Requires higher level of expertise. | Mose commonly used for server-side hosting.  Higher licensing costs compared to other platforms.  Constantly needs updates, may cause disruptions. | Not recommended for server-side hosting.  Mainly used for client-side interaction.  Not very secure.  Limited hardware and power. |
| **Client Side** | Requires medium level of expertise and investment of time.  Provides a user-friendly interface, however slightly more expensive compared to other platforms. | Varies depending on distribution and desktop environment.  Cost friendly and cheaper compared to other platforms. | Slightly cheaper than Mac but more expensive than Linux.  Provides a user-friendly interface.  Requires minimum expertise and time for development. | Provide a responsive and user-friendly interface.  It is portable and cheaper. |
| **Development Tools** | Primary tools include Xcode (IDE), Swift and Objective-C (programming languages).  Requires a moderate level of expertise. | Supports various programming languages and tools such as VS Code and Eclipse.  Wide range of development option due to it being open source  Command line driven | Most common tool is VS Code.  May require more expertise in .NET and C#. | Requires in specific tools such as Xcode or React Native.  Requires expertise in specific languages such as Swift and Kotlin.  Tools and languages can be run in Linux, Mac and windows. |

## 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**: I would recommend a Linux-based operating platform. Linux will be helpful in expanding Draw it or Lose it because it provides scalability and flexibility. Additionally, due to it being open source, it is highly customizable and can cater to the client’s needs of different environments.
2. **Operating Systems Architectures**: Linux follows a monolithic architecture. This is because the entire operating system lives in the kernel. It contains core functionalities such as memory management and file usage.
3. **Storage Management**: SSDs provide great advantages such as speed and performance and are beneficial to applications that need quick retrieval of data. It also comes in various capacities and provides flexibility for the storage system. SSDs are common is modern storage trends which can be provide more efficiency in the long run.
4. **Memory Management**: Linux utilizes the “multi-threading” model to manage memory, where each process has its own address space. This allows for efficient use of memory and can lead to optimized performance.
5. **Distributed Systems and Networks**: Implementing a RESTful API would be ideal in this situation because it allows for communication between the client and the server using HTTP requests. This is accomplished with distributed software because RESTful provides stateless communication. This communication is also important for dependencies such as game data including teams and players.
6. **Security**: Linux is known for its security and being less vulnerable to security breaches. To ensure to protect user information, I would implement encryption on communication between the client and the server using HTTPS. This will ensure that data being transmitted securely. Encrypting the data even while it is at rest will also help ensure to protect information using SSL/TLS protocols. This will also provide authentication to ensure that only trusted users can access the data.