

Win, Lose, or Draw

# **CS 230 Project Software Design Template**

Version 4.0

## 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 |
| --- | --- | --- | --- |
| 4.0 | 10/14/23 | Michael Henderson | The gaming room needs recommendation on which platform to use, as well as storage/memory management and security. |

**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 seeking our help. Now that we have created the code for Draw it Or Lose It, they want us to expand their gaming app on multiple platforms such as mobile, Linux, Mac, and windows. The company wants us to evaluate each OS to help them inform their decisions.*

## Requirements

* *Ability to have one or more teams involved.*
* *Each team will have multiple players assigned to it*
* *Game and team names must be unique, and the game must be able to check if a name has been taken.*
* *Only one instance of the game can exist in memory at any given time.*

## [Design Constraints](#_2et92p0)

* *The app needs to function on to OS platforms*
* *The app can not exceed budget*
* *Remain in budget*
* *App needs to run on multiple platforms.*
* *Checks to make sure your user has a unique name.*
* *Only one instance of the game allowed at one time.*

## [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)

Program driver allows (main) is where the code is run and can be tested. Entity is a parent class that GameService, Game, Team, and Player inherit from. This allows attributes in the Entity class to be used by each subclass.

**"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** | -Unix based system, like Linux. Can be used for web hosting, but less common in enterprise environments. macOS server may have licensing cost. Mac hardware is more expensive, and has limited scaling options. | Open sources cost effective hosting solution. Scalable and reliable for web applications.  - May require more technical knowledge for set up. Limited commercial support. No licensing cost.> | - User friendly interface. Extensive commercial support. Integrated development tools.  -Higher licensing cost, and can face some scalability challenges. | Hosting a full server on mobile might not be the best option since they do not have the best power, However is the most advantageous in terms of cost as there is none to get started. |
| **Client Side** | -Cost would be like a windows setup, as the OS is not open source. Time would depend on expertise, as someone with experience with Mac would need less time. | Cost would be low if there even is a cost. Maximum time and experience needed to use. Linux is not common so would need someone who is good with it. | Windows is more user friendly and does not require as much expertise to use the basics. | Mobile devises are the easiest to work with as many are made user friendly. |
| **Development Tools** | -Swift, C, Xcode Vs code. Development team should include macOS and iOS development expertise. Mac hardware may be needed for iOS development. Xcode is free. | -Relevant tools are Python and Java.  - Linux does use eclipse vs code.  -Requires a team with expertise in Linux-compatible technologies, but is open source so no licensing cost. | - Relevant tools are C#, .NET.  -Windows uses Visual studios.  -Development team would need expertise in window compatible tech. Integration with windows tools may be required and there is licensing cost. | For iPhones the development tools are like the ones for mac and iOS apps. They are typically written in swift |

## 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**: Considering the requirements for scalability, cost-effectiveness, and ability to support a web-based application, Linux is recommended. Linux provides a robust, open-source environment, suitable for hosting web applications and has extensive support for various hardware architectures.
2. **Operating Systems Architectures**: Monolithic Kernel: Linux follows a monolithic kernel architecture where the kernel provides core services and executes within the same address space as the OS.

-Multiuser System: Linux supports multi users simultaneously ensuring that Draw it or Lose it can accommodate numerous players.

- Network Stack: Linux has a well-developed network stack facilitating communications between distributed components.

1. **Storage Management**: Linux uses PostgreSQL. This is a robust, open-source relational database system, that provides ACID compliance for data integrity. It also supports complex queries and can handle the expected load for draw it or lose it.
2. **Memory Management**: Linux has virtual memory, and this helps it manage efficiently. It utilizes segmentation for organizing memory spaces and has page replacement algorithms for optimal memory usage.
3. **Distributed Systems and Networks**: **Distributed Communication:**

**RESTful API:** Implement a RESTful API to facilitate communication between different platforms.

**Message Queues:** Utilize message queues for asynchronous communication between distributed components.

**Dependencies and Considerations:**

**Connectivity:** Ensure robust connectivity between servers and clients.

**Outages Handling:** Implement fault-tolerant mechanisms to handle outages gracefully.

**Load Balancing:** Distribute incoming traffic across multiple servers for better performance.

1. **Security**: **User Protection:**

**Authentication:** Implement secure authentication mechanisms for user access.

**Authorization:** Employ role-based access control to manage user permissions.

**Platform Security Capabilities:**

**Linux Security Features:** Leverage Linux security features like SELinux for access control.

**Firewalls and Encryption:** Implement firewalls and encryption protocols to secure data in transit.

**Regular Updates:** Keep the operating system and software components up to date to patch security vulnerabilities.