

<Name-of-Software-Application>

# **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 | 11/10/2023 | Brett Nottmeier | First edition software design template |

**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 has contacted CTS to develop a web-based game that will be available on multiple platforms. The App will be based on their current game, Draw It or Lose it which is currently only available on Android.

## Requirements

1. The game should be able to support multiple teams.
2. Each team will consist of multiple players.
3. The game should be available across multiple platforms and have cross platform functionality.
4. Team names should be unique to avoid naming conflicts.
5. Only one instance of the game can exist at any given time, there must be unique identifiers for each instance of team, game, or player.

## [Design Constraints](#_2et92p0)

1. Ensuring cross-platform capabilities for web-based applications.
2. Making sure the UI and experience is the same across all different platforms.
3. Secure data management to protect users information.
4. Ensuring unique identifiers for team, game, or player so multiple cannot exist.
5. Synchronizing data for users across multiple platforms.

## [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)

**There is a relationship between Entity and other classes Game, Team, and Player class. All these classes inherit common attributes and methods from the Entity class. This demonstrates OPP principles of Inheritance and Encapsulation. This allows us to design an efficient and organized system ensuring that the game data, players, and teams promotes code reusability and allows for better readability.**

**"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** | Max servers are known for reliability and performance.  They have limited hardware options and not as popular. | Linux is open-source and highly customizable, enabling us to reduce costs. It may require more expertise for setup and maintenance | Windows is known for its ease to use and supports .NET technologies which are good for hosting web-based applications.  Requires more cost for licensing. | Would create easier accessibility for mobile users. Mobile devices don’t typically host web-based application and might be limited than other platforms. |
| **Client Side** | Developing for Mac requires expertise in MacOS development and may have additional time and cost to set up. | Would require more expertise and setup time that may have additional time and cost to setup. | Would require less expertise but would be a higher cost due to licensing and use of .NET. | Would require more expertise and could take more time to setup and accrue additional costs. |
| **Development Tools** | IDE: Xcode,  VS code  Languages: Java, Swift, Objective C, C# | IDE: VS code, PyCharm, Eclipse.  Languages: C/C++, Python, Java | IDE: Visual Studio  Languages: C# | IDE: Xcode (IOS)  Android Studio  Languages: Swift, Java, Koltin. |

## 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**: The Gaming Room should choose a Linux Ubuntu-based server. It provides a stable and secure environment and is cost-effective. It is open source and has a large community and support. Using Linux will ensure compatibility across various platforms and create seamless distribution to other platforms.
2. **Operating Systems Architectures**: Linux is lightweight and modular, allowing easy customization and efficient resource management. This operating system will ensure that it will fit Draw it or Lose it future scalability, specific needs, and efficient resource management. The modular design will ensure that the OS architecture will fit the web application design and scalability.
3. **Storage Management**: I recommend setting up cloud-based storage such as Google Cloud Storage for future scalability, accessibility, and cost. This storage management will limit the need for hardware purchases and have the adaptability to fit the company-specific needs with the option of growing the application in the future, allowing for unlimited scalability if they add expansions to the game. Faster deployment times are another benefit since they do not have to worry about managing a storage system, and developers can focus on the game's development.
4. **Memory Management**: Leverage Linux's built-in memory management techniques for efficient distribution and clearing of resources. Ensuring we can ensure the system efficiently allocates and deallocates memory, guaranteeing optimal performance and minimizing our memory usage.
5. **Distributed Systems and Networks**: I recommend an API-based communication protocol to create a seamless environment across different platforms to create efficient communication across platforms. Redundant network configurations will maintain constant connectivity to prevent any outages. Load balancing and fail mechanisms will be implemented to ensure a reliable service across different platforms.
6. **Security**: Linux is known for its security and has had a reputation for being hard to infiltrate and hack. I recommend setting up encryption using SSL/TLS to secure data transmission and enable full disk encryption to store sensitive data safely. A secure authentication system should protect user data across different platforms. Helping prevent unauthorized users from accessing or tampering with other users’ information, creating a safe and reliable gaming experience for all users.