

<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 | 04/16/24 | Ray Schrump |  |

**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)

Creative Technology Solutions (CTS) is addressing the software design needs of The Gaming Room, a client seeking to develop a web-based version of their game "Draw It or Lose It". The game, inspired by the 1980s TV show Win, Lose or Draw, involves teams guessing phrases based on rendered images. To meet the client's requirements, we propose a software solution leveraging modern web technologies and design patterns. The solution will ensure scalability, uniqueness of team and game names, and efficient memory management.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The development of the web-based game application for The Gaming Room comes with several design constraints in a distributed environment. Firstly, the application must support multiple platforms, requiring compatibility with various web browsers and mobile devices. Secondly, ensuring uniqueness of game and team names necessitates robust data validation and storage management. Thirdly, the application's performance and scalability must be considered, especially in handling multiple concurrent game sessions. Lastly, security measures must be implemented to protect user information across different platforms and during data transmission.

## [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 provided UML class diagram outlines the domain model for the game application. The Entity class serves as the base class, holding common attributes such as id and name. The Game class inherits from Entity and represents a game instance, which contains multiple teams. Each Team and Player class also inherit from Entity, ensuring unique names through iterative validation. Object-oriented principles such as inheritance and encapsulation are employed to efficiently fulfill the software requirements, promoting code reusability and maintainability.

**"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 must 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** | Mac provides ease of use and integration, suitable for smaller-scale deployments but may incur higher licensing costs. | Linux offers robust server capabilities, often preferred for its stability and security, with minimal licensing costs. | Windows offers extensive compatibility and support but may require additional licensing fees, particularly for enterprise-grade features. | Mobile devices, while capable of hosting web applications, may pose challenges in performance optimization and device fragmentation. |
| **Client Side** | For Mac-based client-side development, the main consideration lies in ensuring compatibility with Safari, the default web browser on macOS. Developers need to focus on using web technologies like HTML, CSS, and JavaScript to create responsive interfaces that work seamlessly with Safari on Mac. | Client-side development for Linux involves catering to a variety of web browsers, including Firefox, Chrome, and others. Developers should prioritize using standard web technologies and frameworks to ensure compatibility with different browsers on Linux-based systems. | Windows-based client-side development focuses on ensuring compatibility with popular browsers like Chrome, Firefox, and Microsoft Edge. Developers need to use standard web technologies and frameworks to create responsive interfaces that work well across different browsers on Windows-based systems. | Supporting client-side development for mobile devices involves considerations for both iOS and Android platforms. Developers should focus on using platform-specific development tools and frameworks to create responsive interfaces tailored for iPhones, iPads, and Android devices. |
| **Development Tools** | Development tools for Mac encompass IDEs like Xcode, which provides comprehensive support for macOS, iOS, watchOS, and tvOS development. Xcode includes features for code editing, debugging, and testing, making it the primary choice for macOS and iOS app development. Additionally, tools like Homebrew provide package management for installing development dependencies on macOS. | For Linux-based development, a variety of tools are available, including text editors like Visual Studio Code and Vim, as well as IDEs like IntelliJ IDEA and Eclipse. These tools support a wide range of programming languages and frameworks, enabling developers to build web applications, command-line tools, and desktop applications on Linux. Package managers like apt and yum simplify the installation of development dependencies and libraries on Linux distributions. | Windows-based development primarily revolves around IDEs like Visual Studio, which offer comprehensive support for Windows, web, and mobile development. Visual Studio includes features for code editing, debugging, and testing across multiple platforms, making it suitable for a wide range of development scenarios. Additionally, tools like NuGet and Chocolatey provide package management for installing dependencies and libraries on Windows. | For mobile development, platform-specific IDEs are used, such as Xcode for iOS development and Android Studio for Android development. These IDEs provide features for designing user interfaces, writing code, and testing applications on respective mobile platforms. Cross-platform frameworks like Flutter and React Native offer alternatives for building mobile applications using a single codebase, reducing development time and effort across iOS and Android platforms. |

## 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**: We recommend leveraging Linux as the server platform for hosting the web-based game application. Linux offers robust server capabilities, cost-effectiveness, and extensive community support, making it an ideal choice for scalable deployments.
2. **Operating Systems Architectures**: Linux, being open source, follows a modular architecture with a monolithic kernel, providing stability and security for server environments. Its scalability and performance make it suitable for hosting distributed applications like Draw It or Lose It.
3. **Storage Management**: For storage management, we suggest utilizing a distributed file system such as NFS (Network File System) or GlusterFS. These systems offer scalability, fault tolerance, and ease of management, ensuring reliable storage for game data across multiple servers.
4. **Memory Management**: Linux employs virtual memory management techniques such as demand paging and memory segmentation, optimizing memory usage and performance for the game application. Memory allocation strategies ensure efficient resource utilization, enhancing overall system stability.
5. **Distributed Systems and Networks**: To facilitate communication between various platforms, we propose implementing RESTful APIs and WebSocket protocols. These technologies enable real-time data exchange and synchronization across distributed systems, ensuring seamless gameplay experiences for users.
6. **Security**: Ensuring user information protection is paramount. Linux provides robust security features such as SELinux (Security-Enhanced Linux) and iptables for access control and firewall management. Additionally, implementing HTTPS encryption and user authentication mechanisms enhances data security during transmission.