

# **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 |
| --- | --- | --- | --- |
| 1.2 | 10/22/23 | Jonathan Boschke | Added to Evaluation |

**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 engaged Creative Technology Solutions (CTS) to develop a web-based version of their game, Draw It or Lose It, currently available only as an Android app. The game involves teams guessing puzzles represented by stock drawings. This software design document outlines the development process and addresses the client's software requirements.

## Requirements

The client's requirements include the ability to support one or more teams, assign multiple players to each team, ensure unique game and team names, and maintain only one instance of the game in memory. These requirements will be met through careful design and the implementation of software design patterns.

## [Design Constraints](#_2et92p0)

Developing the game application for a web-based distributed environment comes with certain design constraints. One major constraint is the need to ensure that game and team names are unique to avoid naming conflicts. This constraint will affect the application's user interface and data storage design. Additionally, only one instance of the game should exist in memory, which necessitates the use of design patterns like the Singleton pattern. These constraints emphasize the importance of careful planning and design.

## [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 UML class diagram provided in the project materials outlines the domain model of the application. It includes classes for Game, Team, Player, and Entity, with relationships between them. The Entity class serves as a base class holding common attributes, demonstrating the inheritance principle. The GameService class uses the Singleton pattern to ensure only one instance exists, managing games and their identifiers. The use of the Iterator pattern helps in ensuring unique names for games and teams. Object-oriented programming principles, such as inheritance and encapsulation, are leveraged to fulfill the software requirements efficiently.

**"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** | Macintosh operating systems offer a stable and robust environment with a Unix-based architecture, ensuring reliability and security. Mac servers are known for their scalability and seamless compatibility with web technologies. However, the potential drawback here is the cost associated with Mac hardware, which might impact the client's budget. | Linux, being an open-source platform, is renowned for its stability, security, and scalability, making it a popular choice for hosting web applications. It aligns well with the client's budgetary constraints. Nonetheless, Linux might require a higher degree of technical expertise for initial setup and configuration. | Windows provides a user-friendly environment for hosting web applications, offering robust support for various web technologies and development frameworks. However, it might not be as cost-effective as Linux, and Windows servers can demand more resources. Additionally, licensing costs could pose a concern for the client. | Mobile devices, such as smartphones and tablets, offer the advantage of portability, enabling users to access the game from anywhere. However, developing web-based software for mobile devices presents challenges related to varying screen sizes, device fragmentation, and the imperative for responsive design. It necessitates expertise in mobile development frameworks and tools. |
| **Client Side** | Catering to Mac users requires meticulous consideration of software development tools and user experience. Mac users are known to have a preference for aesthetically pleasing and responsive user interfaces. Therefore, ensuring a visually appealing and seamless user interface is paramount. Development for Mac may entail access to macOS for testing and may necessitate expertise in Xcode and Swift, Apple's preferred development tools. | Developing for Linux clients necessitates an in-depth understanding of open-source development tools and libraries. Linux users appreciate software that seamlessly integrates with their system and adheres to Linux conventions. Achieving cross-platform compatibility might entail utilizing technologies like GTK or Qt, and expertise in languages such as C or Python is imperative. | Windows users have an expectation of user-friendly installation and ease of use. Developing for Windows clients often involves leveraging Microsoft's development tools, such as Visual Studio and languages like C# and C++. Ensuring compatibility with different Windows versions and architectures is pivotal. Windows also provides a wide array of programming languages, affording flexibility in development choices. | Supporting mobile devices mandates expertise in mobile app development frameworks like React Native, Flutter, or platform-specific development using Swift or Kotlin. It also requires a deep understanding of responsive design to adapt the game's user interface to various screen sizes and orientations. Rigorous testing on different devices and operating system versions is essential to ensure a seamless user experience. |
| **Development Tools** | Developing web-based software for Mac can involve using development tools like Xcode, which provides a comprehensive integrated development environment (IDE) for macOS and iOS. Languages like Swift and Objective-C are commonly used for Mac app development. | Linux development often relies on open-source tools and languages such as Python, C, or C++. IDEs like Visual Studio Code, PyCharm, or Eclipse are popular choices for Linux development. Linux offers a wide range of development libraries and tools for web applications. | Windows development can benefit from the robust Visual Studio IDE, which supports multiple programming languages, including C#, C++, and JavaScript. Web-based applications can be developed using technologies like ASP.NET, React, or Angular, among others. Windows also provides robust server-side technologies like IIS. | Developing for mobile devices requires proficiency in mobile app development frameworks such as React Native, Flutter, or native development for iOS and Android. Commonly used IDEs include Android Studio and Xcode. |

## Recommendations

**1. Operating Platform:** We recommend adopting a web-based operating platform to ensure cross-platform compatibility for Draw It or Lose It. A web-based system allows the game to run seamlessly on various computing environments, including Windows, Mac, Linux, and mobile devices. This approach ensures that the game can reach a wide user base without the need for extensive modifications to the underlying codebase.

**2. Operating System Architectures:** The selected operating platform should be designed with scalability and flexibility in mind to accommodate the expansion of Draw It or Lose It. To achieve this, a modular architecture is recommended. This modular approach allows for easy integration with various operating systems, ensures maintainability, and supports future enhancements. The web-based platform should be built using a microservices architecture, where each component (e.g., user management, game logic, and database interactions) is encapsulated as a separate service. This architecture promotes scalability, resilience, and ease of management.

**3. Storage Management:** We propose using a relational database management system (RDBMS) as the primary storage management solution. An RDBMS provides a structured and efficient way to manage game data, user profiles, and scores. Relational databases offer strong data integrity and consistency, making them ideal for applications like Draw It or Lose It. The choice of the RDBMS should be driven by factors such as scalability, security, and cost-effectiveness.

**4. Memory Management:** The recommended web-based platform should implement robust memory management techniques to ensure smooth gameplay and responsiveness. This includes efficient memory allocation and deallocation, cache management, and effective garbage collection. To achieve this, the platform should employ advanced memory management strategies like heap and stack management to optimize memory usage. Additionally, it's crucial to minimize memory leaks and fragmentation to ensure a stable gaming experience.

**5. Distributed Systems and Networks:** To facilitate communication between various platforms and devices, we recommend adopting a distributed architecture. The Draw It or Lose It application will consist of multiple interconnected services that communicate through a well-defined API. To handle network-related challenges such as connectivity and outages, the system should implement mechanisms like message queues and asynchronous communication. Load balancing should also be considered to distribute network traffic evenly across multiple servers, ensuring high availability and reliability. The network connecting the devices should be designed to support both HTTP and WebSocket protocols, allowing real-time interactions in the game.

**6. Security:** Security is a top priority for Draw It or Lose It, and the chosen operating platform should provide robust security features to protect user information and game integrity across various platforms. This includes data encryption in transit and at rest, user authentication mechanisms, and authorization controls. The platform should support role-based access control to ensure that only authorized users can access specific game features. Additionally, it's essential to conduct regular security audits and implement security best practices to address emerging threats and vulnerabilities.

By implementing these recommendations, The Gaming Room can ensure the successful expansion of Draw It or Lose It to different operating systems and platforms, offering users a seamless and secure gaming experience across a wide range of devices.