

Draw it or Lose It

# **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 | 03/23/2025 | Robert Chappell | Initial version for Project One submission |

**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)

This project aims to provide a web-based, cross-platform application for Creative Technology Solutions (CTS), therefore augmenting the current game "Draw It or Lose It". Using a shared codebase, the customer wants the game to run on desktop environments and mobile devices among other platforms. This software design document presents a scalable and effective method applied with object-oriented design ideas for development. While satisfying technological and financial needs, the suggested method guarantees the game stays interesting, user-friendly, and compatible with several devices.

## Requirements

The customer wants the game to let users log in and participate in teams and support several platforms. The game has to guarantee that every name is unique and save usernames. Using kept credentials, players should be able to start their sessions once more. The program has to be scalable, safe, and keep great performance on several systems. Business needs call for expanding the audience and updating the game experience.

## [Design Constraints](#_2et92p0)

The application has numerous restrictions since it must operate in a distributed web-based environment. Development calls for consideration of network latency, browser compatibility, and device diversity. The program must additionally safely save and access game data and user credentials. Furthermore, separating the game core and user interface will help to facilitate modular development and cross-platform distribution. These restrictions call for careful design to guarantee a dependable and responsive user experience on several running systems and devices.

## [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 basic elements of the Draw It or Lose It Web game are shown on the UML class diagram. Linking with the Team, Player, Round, and WordBank classes, the Game class oversees the whole game session. Every game features two or more team objects in addition to many player items. The Round class manages the timer and chooses a word from the WordBank, therefore handling the gaming process. Used during every round, the WordBank class maintains a list of words that one can sketch.  
  
This approach keeps data and behavior contained inside every class by means of object-oriented ideas including encapsulation. By dividing duties, including keeping game logic in the Game and Round classes and player data in the Player class, it also facilitates abstraction. This framework facilitates growth, updating across several platforms, and maintenance of the application.

**"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** | Although Mac is not often selected for large-scale installations, it can be hosted for a web-based software application. It lacks native support for various server technologies and is not as adaptable as Linux, even if it presents a user-friendly interface and robust development tools. Moreover, less scalable and more costly than alternatives is hosted on macOS. | Web-based applications are hosted on Linux, the most often used platform. It is stable, open-source, and quite customizable. It supports a great spectrum of programming languages, databases, and server tools. For scalable installations, their minimal weight and robust security make perfect fit. Furthermore, reasonably priced, Linux is fit for CTS's objectives. | Especially for applications created in Net environments, Windows Server is a dependable platform for hosting them. It calls for more resources and licenses yet offer good support for business applications. Less often used for open-source web programs, it is less customizable than Linux. | Though they are not usually used to host web apps, mobile devices are crucial for consumers. Power, performance, and network limits restrict hosting on mobile devices. The game must be refined, though, to fit mobile users on both Android and iOS platforms. |
| **Client Side** | Testing the web application with Safari and Chrome will help Mac users to be compatible. There is a modest development time; cost relies on the developer's Apple system knowledge. Users of Mac computers anticipate responsive, high-quality interfaces, hence UI/UX design should get particular attention. | Linux users might run Chromium or Firefox among browsers. While less popular for general users, Linux compatibility is crucial for developers and tech-savvy consumers. Although testing across several distributions is a development issue, generally cost and time influence are negligible. | Most often used desktop client platform is Windows. Essential is making sure Chrome, Edge, and Firefox are compatible. Development is easy; yet, testing across Windows versions could be required. It delivers dependable performance and a broad user base. | Mobile assistance is essential. Responsive design, touch compatibility, and effective performance should take front stage in development. Supporting iOS and Android calls either a cross-platform framework like React Native or Flutter or native development for every one of them. Though it lengthens development time, this is required to satisfy user expectations. |
| **Development Tools** | While web programming may be accomplished with IDEs such as VS Code or IntelligJ, Xcode is valuable for iOS development. Homebrew's tools help to install development packages more easily. Mac runs Java, JavaScript, Python, and more. | Along with IDEs as Vers Code, Eclipse, and IntelligJ, Linux provides potent tools including GCC, Java, Node.js, and Python. For scripting, automation, and full-stack development especially, it is perfect. APT and YUM streamlining program installation helps package managers. | Windows runs a large spectrum of programs like Visual Studio, Eclipse, and VS Code. It works with Java, C#, Python, and several web technologies. Running Linux tools straight on Windows is made possible by Windows Subsystem for Linux (WSL). | For Android, mobile development depends on IDEs like Android Studio; for iOS, Xcode. With a shared codebase, cross-platform tools like Flutter or React Native assist create apps for both platforms. Mobile development depends critically on testing and debugging tools. |

## 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 advise using a Linux-based server platform, such Ubuntu Server, for Draw It or Lose It to spread across several computer environments. Perfect for scalability, security, and cost-effectiveness, it is dependable, open-source, and generally supported.
2. **Operating Systems Architectures**: Linux, the suggested platform, runs in the same address space for efficiency as its monolithic kernel design dictates most operating system functions. High performance and responsiveness follow from this design supporting process scheduling, memory management, and file systems all inside the kernel. It also provides great fit with virtualization systems (such as Docker), so enabling flexible deployment across platforms.
3. **Storage Management**: Media files, stored game states, and user-generated content all better fit a cloud-based storage solution like Amazon S3 or Google Cloud Storage. These systems are scalable, fault-tolerant, and integrated with security elements including in-transition and at rest encryption. On Linux machines, ext4 or XFS file systems are reliable and efficient solutions for local storage.
4. **Memory Management**: Advanced memory management mechanisms include paging, demand paging, and swapping let Linux effectively employ virtual and real RAM. For multi-core CPUs, it promotes memory overcommitment, shared memory segments, and non-uniform memory access (NUMA) optimization. This guarantees the game runs effectively under different workloads, particularly during high demand.
5. **Distributed Systems and Networks**: Draw It or Lose It should use a client-server architecture with a RESTful API housed on the Linux server to permit interaction between several platforms. Managing game logic, data storage, and real-time player interactions, the server will be the hub from which all else flows. Low-latency communication for multiplayer elements can be enabled with WebSockets or gRPC. Reverse proxies (e.g., Nlinux) and cloud-based scaling solutions (e.g., AWS Elastic Load Balancer) help to create redundancy and load balancing therefore lessening the effect of network failures or outages.
6. **Security**: The suggested Linux server platform would use numerous integrated security techniques in order to guard user data across platforms. TLS will encrypt all of the client-server correspondence to stop sensitive information from being intercepted. Safe methods including OAuth 2.0 will be used for user authentication to guarantee that session data and credentials stay under protection. Industry-standard techniques including AES-256 will encrypt data kept on the server; the system will enforce input validation to stop data corruption or injection attacks. Routine server backups, firewall setups, and active monitoring using Fail2Ban or auditd will assist guard against unwanted access and system breaches. These techniques taken together offer a robust security basis for protecting user information and preserving platform trust.