

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 | 07/20/25 | Asia Springs | Executive summary, design restraints, domain model |

**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 purpose of this project is to develop a web-based version of Draw It or Lose It for The Gaming Room, expanding its accessibility beyond its current Android-only platform. Inspired by the classic TV game show Win, Lose, or Draw, the game involves multiple teams competing through four one-minute rounds, where each team attempts to guess an image pulled from a library of stock drawings. If the team fails to guess correctly within the time limit, opposing teams have 15 seconds to respond. This software design document outlines the proposed solution to deliver a cross-platform, user-friendly, and engaging game that meets the client’s requirements and enhances the overall gaming experience.

## Requirements

Web-based accessibility

Team participation

Unique names

Single game instance

Timed game rounds

Guessing rules for opposing teams

## [Design Constraints](#_2et92p0)

Web-based distributed environment- must be developed for a web-based platform, considering constraints like network communication, security, and compatibility across browsers and devices

Multi-platform support- should be accessible and functional across various platforms and operating systems

Team-based play with multiple players per team- each game should support multiple teams, with the ability to assign several players to each team

Unique game, team, and player names- game, team, and player names must be unique to prevent conflicts

Single game instance limitation- only one instance of the game should run in memory at any time

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram for the Draw It or Lose It game application illustrates the relationships between core classes using object-oriented principles such as inheritance, aggregation, encapsulation, and abstraction. At the foundation is the Entity superclass, which defines shared attributes like id and name and is extended by the Game, Team, and Player classes, ensuring consistency and reducing redundancy. The diagram demonstrates "has-a" (aggregation/composition) relationships, where GameService manages multiple Game instances, each Game contains multiple Team objects, and each Team includes multiple Player objects. These relationships are maintained through references, reflecting the structured management of entities within the system. The ProgramDriver class serves as the application’s entry point, instantiating the singleton GameService and facilitating the creation and organization of games, teams, and players. It also has a dependency on the SingletonTester class, indicating a usage relationship. This structure ensures that only one instance of the game service exists, supports scalability, and encapsulates internal data while providing controlled access through defined interfaces.

**"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** | Offer easy accessibility, customizable server configurations, and a balance of user-friendly graphical interfaces with flexible command-line options, making them suitable for both novice users and advanced administrators. They are not always cost-effective. Extensive software compatibility, supporting a broad range of applications and development environments. These systems are designed to work with a wide variety of hardware, from modern servers to older machines, and are backed by comprehensive documentation and active community support. Their portability allows deployment across physical, virtual, and even mobile environments, though user experience on mobile devices may vary depending on hardware specifications. | Many use a command shell—like Bash or PowerShell—for straightforward and efficient server configuration. This allows for precise control, automation, and quick access to system functions without relying on a graphical interface. These platforms are also highly cost-effective, especially open-source options like Linux, which reduce or eliminate licensing fees while offering powerful, scalable tools for managing servers. | Offer broad software compatibility, supporting everything from web servers to databases and development tools. Run efficiently on a wide range of hardware—from high-end servers to older machines—making them highly adaptable. Backed by detailed documentation and strong community or vendor support, they provide reliable performance and easy troubleshooting across various deployment environments. | Mobile device specifications can vary significantly between users, with differences in operating systems, processing power, screen sizes, and available features. As a result, server-side platforms must be adaptable to ensure consistent performance and accessibility across a range of devices. Many modern server management tools are designed with this in mind, offering responsive web interfaces or dedicated mobile apps that adjust to different screen resolutions and hardware capabilities. This flexibility enhances portability, allowing administrators to monitor and manage servers remotely from smartphones, tablets, or lightweight laptops. Whether on-site or on the go, users can access server resources reliably, making portability a key advantage of today’s server-side solutions. |
| **Client Side** | Can come with high upfront costs for users. Beyond the initial investment in equipment and licensing, ongoing expenses may include maintenance, upgrades, and support services. Additionally, effectively managing and configuring these systems typically requires a moderate level of technical expertise and time. Users often need to be familiar with command-line interfaces, networking concepts, and security practices to ensure reliable performance and system integrity. While these platforms offer powerful capabilities, the learning curve and associated costs can be barriers for beginners or organizations with limited technical resources. | Often demands a significant level of expertise and time investment. Users are expected to understand core concepts such as file systems, permissions, process management, and command-line operations. Familiarity with Linux-specific data structures, configuration files, and system behavior is essential for effective administration and troubleshooting. While the learning curve can be steep—especially for those new to Linux—the long-term benefits are considerable. One of the biggest advantages is cost-efficiency: most Linux distributions are free to use, with no licensing fees, and offer a wide range of open-source tools that reduce the need for paid software. This makes Linux-based systems an attractive option for users willing to invest the time to learn and manage them properly. | Come with a medium cost, which includes licensing fees for the operating system and, in some cases, additional software or server roles. However, this cost is often balanced by the platform's ease of use and accessibility. Windows Server offers a familiar graphical interface that simplifies system configuration, management, and updates, making it a popular choice for users with limited technical backgrounds. Built-in tools like Server Manager and PowerShell provide both visual and script-based control, catering to a wide range of skill levels. Because of its intuitive design and extensive documentation, setting up and managing a Windows server generally requires minimal expertise, allowing even less-experienced users to get up and running quickly. | Offer flexible cost options, ranging from free and open-source solutions to paid services with advanced features, making them accessible to a wide range of users. They also provide clients with the ability to access and manage servers remotely using smartphones or tablets. This mobile flexibility is especially valuable for teams working across different locations or time zones, as it allows for real-time updates, monitoring, and troubleshooting from virtually anywhere. With dedicated mobile apps or responsive web interfaces, users can stay connected to their server environments without needing a desktop setup, enhancing both convenience and productivity on the go. |
| **Development Tools** | HTML, CSS, and JavaScript are used for front-end development, supported by various libraries. development tools include PyCharm, GitHub, Visual Studio, and others | HTML, CSS, and JavaScript, JavaScript, Ruby, PHP, and Python | HTML, CSS, and JavaScript, Eclipse, Command Prompt, PyCharm, and others | HTML, CSS, and JavaScript, IDEs for programming support languages like HTML, PHP, C++, and Python |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: To help Draw It or Lose It reach more users across different devices, The Gaming Room should use a web-based platform. This means the game can run in any modern web browser, whether on a computer, tablet, or smartphone—without needing to install anything. A web-based version makes it easier to update the game and ensures all players get the same experience, no matter what device they use. While Windows is great for building software, sticking to just one platform limits the audience. By going web-based, the game becomes more flexible and accessible. For the server side, using a Linux-based system is a smart choice. It’s reliable, secure, and works well with cloud services, making it easy to grow as more people play the game. This setup gives Draw It or Lose It the tools it needs to expand and succeed.

1. **Operating Systems Architectures**: To help Draw It or Lose It grow and reach more users, the game should use a web-based client-server architecture. The development can be done on Microsoft Windows, which is a great platform for building software thanks to tools like Visual Studio and VS Code. Developers can create the game using web technologies such as HTML, CSS, and JavaScript for the front end, and use languages like Python for the back end. Once the game is built, it should be deployed on a Linux-based server. Linux is a smart choice for hosting because it is reliable, secure, and works well with cloud services like AWS or Google Cloud. The architecture will follow a simple structure: the client side runs in a web browser on any device and handles the game interface, while the server side processes game logic and communicates with a database. The database stores player information, game progress, and scores. This setup makes the game easy to access, update, and scale, ensuring a smooth experience for users on any device.
2. **Storage Management**: An appropriate storage management system for Draw It or Lose It, based on the recommended web-based architecture and operating platforms (Windows for development and Ubuntu Server for deployment), would combine a relational database management system (RDBMS) with cloud storage. For structured data such as user profiles, login details, and game statistics, PostgreSQL is a strong choice. It is an open-source, reliable, and scalable database that integrates well with Ubuntu Server and supports complex queries, making it ideal for managing user and game-related data. For unstructured content like stock images, graphics, and other media assets, using a cloud storage solution such as Amazon S3 is recommended. Amazon S3 offers secure, scalable, and fast access to media files, allowing the server to offload large files and improve performance. During development on Windows 10, developers can also make use of built-in tools like Storage Sense to manage disk space and temporary files. However, in a live environment, the combination of PostgreSQL for structured data and Amazon S3 for media storage provides a flexible and efficient storage solution that supports both scalability and accessibility.
3. **Memory Management**: The recommended operating platform for Draw It or Lose It—Windows 10 for development and Ubuntu Server for deployment—uses effective memory management techniques to ensure smooth performance. On Windows 10, developers can use Storage Sense to organize and manage game assets like images and player data, while the operating system itself manages memory through virtual memory and paging to prevent crashes during development. In a web-based environment, modern browsers automatically handle memory using garbage collection, which clears unused data and reduces the risk of memory leaks as players interact with the game. On the server side, Ubuntu Server supports efficient memory management through built-in tools like system monitors, swap space, and containerization (e.g., Docker), helping to isolate processes and prevent resource overuse. Combined, these memory management features support reliable, scalable, and efficient operation of the game across different platforms and devices.
4. **Distributed Systems and Networks**: To enable Draw It or Lose It to communicate across various platforms, the game should use distributed software connected through a centralized server or cloud infrastructure, which manages real-time communication, game logic, and shared data access. Players on different devices—such as computers, tablets, or smartphones—connect to the server using internet protocols like HTTP/HTTPS for general, allowing instant updates and synchronized gameplay. A shared database stores user profiles, game sessions, and scores, ensuring that all clients have access to the same game state. To handle issues like connectivity loss, latency, or server outages, the system should include error-handling features such as auto-reconnection, state caching, and recovery procedures. Cloud services can further enhance reliability through load balancing, geographic redundancy, and auto-scaling. Together, these components allow the game to run smoothly across platforms while managing the dependencies and challenges of a distributed network environment.
5. **Security**: To protect user information across various platforms for Draw It or Lose It, a combination of system-level and web-based security measures should be implemented. On the development side, Windows provides built-in protections such as Windows Defender, firewalls, and secure user account controls, but additional steps like using encrypted drives, secure code repositories, and strong development practices further enhance safety. For the deployed web-based version, all data transferred between clients, and the server should be encrypted using SSL/TLS (HTTPS), while sensitive data such as passwords must be securely hashed and stored. User authentication should be handled through secure login systems, with optional two-factor authentication, and authorization must ensure users only access the features they’re permitted to use. The server environment should include firewalls, secure API access with authentication tokens, and protection against common threats like brute-force attacks and DDoS. Using cloud infrastructure adds further layers of security, including monitoring, access controls, and automated updates. Together, these practices ensure strong protection of user data both in storage and during transmission, keeping the game secure and reliable across all platforms.