

Draw It or Lose It game 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 | 09/18/2025 | Jacquelin Chang | Add executive summary, requirements, design constraints, and domain model |
| 2.0 | 10/03/2025 | Jacquelin Chang | Add evaluation |
| 3.0 | 10/15/2025 | Jacquelin Chang | Add recommendations |

**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)

Draw It or Lose It is a four-round, web-based multi-team party game where a sequence of stock drawings is rendered as clues for the teams to guess within a set time limit. The Gaming Room would like to expand the current Android-only version into a web application that supports multiple platforms. To ensure a consistent users experience, several specific requirements must be addressed in the design of the new application.

## Requirements

* Support multiple platforms
* Allow one or more teams per game
* Each team contains multiple players
* Game and team names must be unique
* Users must be able to check if a name is already in use when choosing team name
* Only one instance of the game can exist in memory at a time

## [Design Constraints](#_2et92p0)

* Cross-platform Compatibility: the application must run smoothly on multiple platforms, including Windows, macOS, Linux, and mobile devices. This requires using standard web technologies that are portable and responsive.
* Singleton Pattern: Only one instance of the GameService should exist in memory at any given time to ensure consistent game state management.
* Unique Identifiers and Unique Names: Each game, team, and players must have a unique identifier and name. This prevent conflicts and ensures that users can reliably check the availability of names.

## [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 diagram shows the relationship between classes in the application.

The Entity class serves as the base class, centralizing common attributes and methods, such as Id, Name, and corresponding getters and setters. All the domain objects inherit from Entity class.

The Game class, extends from Entity class, contains collections of Team objects as well as attributes and that represent the game state.

The Team class, extends from Entity class, contains collections of Player objects along with team-specific attributes.

The Player class, extends from Entity, stores player-specific data.

The GameService class acts as controller of the game, ensuring that only one instance exists per server. It is responsible for creating unique IDs for games, teams, and players, and for enforcing uniqueness of names.

**"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** | One advantage of macOS is its consistency, the system has maintained a stable design over time, making it easy to operate. However, macOS is not commonly used for production servers. Licensing costs and limited hosting options make it less practical for large-scale deployments compared to Linux or Windows. | Linux is the most widely used platform for hosting web-based applications. Its open-source nature and strict permission-based access controls make it cost-effective, secure, and highly reliable. It is also highly compatible with popular web technologies, making it an excellent choice for scalability. However, some disadvantages include fewer options for pre-built server hardware and occasional file format compatibility issues. | Windows servers are user-friendly and widely supported, making them attractive for organizations already invested in Microsoft products. A major advantage is that Windows often works “out of the box” without requiring many add-ons, and it integrates well with enterprise tools. On the downside, licensing costs are higher than Linux, and Windows servers are often targeted more frequently for security attacks, which requires stronger monitoring and protection. | Mobile devices could theoretically serve as small-scale hosts when user numbers are limited and applications are simple. However, their limited processing power, memory, and storage make them unsuitable for hosting a web-based application at scale. Additionally, because most mobile apps rely on cloud storage, they carry a higher risk of exposure to security threats compared to traditional server platforms. |
| **Client Side** | Web applications run well on macOS browsers like Safari and Chrome. Development costs are moderate, with testing needed to ensure compatibility. | Linux clients are less common, but web applications run smoothly on browsers like Firefox and Chrome. Minimal additional cost or time is required. | Windows is the most widely used client platform, so ensuring compatibility across major browsers (Edge, Chrome, Firefox) is critical. Testing effort is higher here. | Mobile devices require responsive design and extensive testing across different operating systems and screen sizes, which increases cost, time, and expertise needed. |
| **Development Tools** | macOS supports IDEs like Eclipse, PyCharm, and Xcode. Languages such as Java, Python, and JavaScript are well supported. | Linux provides the most natural development environment for web servers, with wide availability of tools like Eclipse, IntelliJ, VS Code, Git, and Docker | Windows supports powerful IDEs such as Visual Studio, Eclipse, and IntelliJ. Java, Python, and JavaScript are fully supported. | Mobile devices are not used for direct development, but tools like Android Studio and Xcode enable building and testing responsive or native mobile applications. |

## 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**: Windows Server is the recommended operating platform for The Gaming Room. It offers a familiar, user friendly interface for administrators, strong vendor support, and seamless integration with widely used development tolls and frameworks such as .NET, JAVA, and Python. Windows is also well documented and widely supported by cloud providers, making deployments and maintenance efficient. Its compatibility with web standards ensures that *Draw It or Lose It* can easily expand to other computing environments through web browser.
2. **Operating Systems Architectures**: Windows uses a hybrid operating architecture that combines graphical user interfaces with efficient multitasking and process scheduling. The structure supports efficient multitasking, process scheduling, and memory isolation while maintaining a rich graphical interface for administration. The architecture allows for robust driver and hardware compatibility, ensuring smooth performance across a variety of server configurations. Its modular design also supports scalability, meaning the system can handle multiple concurrent game sessions and user connections without compromising stability or responsiveness.
3. **Storage Management**: The NTFS (New Technology File System) is the recommended storage management system for Windows. NTFS supports advanced features such as file-level encryption, compression, access control lists (ACLs), and journaling to protect data integrity. These features make it ideal for managing large volumes of media files, including the 200 high-definition image assets required for *Draw It or Lose It*. NTFS also supports large storage volumes and efficient file indexing, allowing quick retrieval of images and minimizing delays during gameplay. For scalability, cloud integration with Azure Storage or AWS S3 can be used to host image libraries and backups, reducing local storage needs while ensuring reliability.
4. **Memory Management**: Windows uses a virtual memory management system that dynamically allocates and reclaims memory based on process demands. This system isolates processes to prevent conflicts and ensures that the application runs smoothly even under heavy load. The *Draw It or Lose It* game can benefit from features like paging and caching, which enable rapid access to frequently used assets without consuming excessive physical memory. Windows also includes built-in memory optimization and garbage collection mechanisms that improve performance when rendering large, high-resolution images in real time.
5. **Distributed Systems and Networks**: To enable cross-platform play, *Draw It or Lose It* should use a client-server architecture. The central game logic and data (managed by the GameService) will reside on the Windows-based server, while clients connect from any platform via standard web browsers or mobile apps. Secure communication can be established through HTTPS or WebSocket protocols, which allow real-time interaction between players and teams. The distributed system design should include load balancing and redundant servers to handle high traffic and prevent downtime during network outages. Data synchronization and session management across clients will ensure that all users experience consistent gameplay.
6. **Security**: Security is a key requirement for protecting user information and ensuring fair gameplay. Windows provides multiple built-in security mechanisms, including NTFS permissions, BitLocker encryption, and Windows Defender Firewall. Data transmitted between clients and the server should use TLS (Transport Layer Security) encryption to prevent unauthorized access or tampering. Authentication systems can use secure tokens or OAuth to manage user sessions safely. Regular security updates and server monitoring will further strengthen protection against malware and intrusion attempts. Additionally, role-based access control (RBAC) can be implemented to limit administrative privileges and maintain data confidentiality across the system.