

# Draw It or Lose It

# **CS 230 Project Software Design**

Version 3.0

## 

## Table of Contents

[**CS 230 Project Software Design**  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 4**](#_Toc115077324)

**Evaluation 5**

**Recommendations 6-7**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 04/16/25 | Shane Colledge | Recommendations |

[**Executive Summary**](#_sbfa50wo7nsh)

The Gaming Room has requested a transition of their game, “Draw It or Lose It”, from an Android-only platform to a web-based version that supports multiple computing environments. The game requires a **robust, scalable, and efficient design** to accommodate multiple teams and players while maintaining a **single instance** of the game in memory. The **Singleton design pattern** will be used to ensure that only one **GameService** instance exists at any given time. Additionally, **unique name constraints** for games, teams, and players will be enforced using the **Iterator pattern**.

## Requirements

1. The game must support multiple teams and players per team.

2. Only one instance of the game can exist at any time (Singleton pattern in `GameService`).

3. Game, team, and player names must be unique to prevent duplication.

4. The system must be scalable and support multiple platforms.

5. The application will use object-oriented principles for maintainability and efficiency.

## [Design Constraints](#_2et92p0)

1. Web-Based Deployment: The game must function across multiple operating systems, requiring server-client architecture considerations.

2. Memory Management: A Singleton pattern must be used to limit memory consumption and prevent redundant instances.

3. Concurrency and Uniqueness: Names for teams and games must be unique, requiring efficient searching mechanisms (Iterator pattern).

4. Scalability: The system must be designed to support future expansions (e.g., cloud deployment, additional game features).

## [System Architecture View](#_ilbxbyevv6b6)

The application follows a client-server architecture, where:

- The server hosts the game logic, manages data, and ensures game integrity.

- Clients (users) interact with the application through web interfaces or mobile devices.

- APIs facilitate communication between clients and servers to handle game state, user authentication, and updates.

## 

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram illustrates the structure of the application. The key relationships are:

- `Entity` (Base Class) -- `Game`, `Team`, and `Player` inherit from it.

- `GameService` uses the Singleton pattern to manage `Game` instances.

- `Game` contains a list of `Team` instances.

- `Team` contains a list of `Player` instances.

- Associations:

- `GameService` -- `Game` (0..\*)

- `Game` -- `Team` (0..\*)

- `Team` - `Player` (0..\*)

The object-oriented principles applied include:

- Inheritance: `Entity` serves as a base class for `Game`, `Team`, and `Player`.

- Encapsulation: Data fields are private, with public getters.

- Polymorphism: Overriding the `toString()` method for clear object representation.

**"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)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| **Server Side** | Limited use as a production server. Not commonly used for large-scale web hosting, though technically possible (e.g., with MAMP). | Excellent for server-side hosting. Supports Apache, Nginx, Docker, and cloud environments. Stable, scalable, and widely used in industry. | Common in enterprise environments. Supports IIS for hosting, but less preferred for large-scale open-source web apps. | Not used for hosting but communicates with backend servers via web APIs. |
| **Client Side** | High compatibility with modern browsers (Safari, Chrome, Firefox). Excellent UX. | Fully compatible with all major browsers (Chrome, Firefox). | Fully compatible with major browsers (Edge, Chrome, Firefox). Most common desktop OS. | Required for mobile app support. Browsers on iOS and Android can run the HTML5 web interface. |
| **Development Tools** | Xcode (free with macOS), IntelliJ (paid), and VS Code (free). Costs depend on tool choice. | VS Code, Eclipse, and most CLI tools are free. No licensing costs for common development tools. | Visual Studio (free Community edition available; full version requires license). Some tools may have additional licensing fees. | Android Studio (free), Xcode (free on macOS), React Native tools (free). App Store/Play Store publishing requires annual fees. |
| **Licensing Costs** | macOS Server ($20) but requires Apple hardware ($999+ total) | Free and open source (e.g., Ubuntu Server, CentOS). $0 | Windows Server Standard license ($1,069) + CALs ($40/user) | Apple Developer Program ($99/year), Google Play ($25 one-time) |
| **Development Considerations** | macOS development requires specialized knowledge. Often used for mobile (iOS) and frontend development. | Developers need Linux familiarity; cross-platform tools like Node.js, React, and Java work well. | Development teams may need to manage .NET and Windows-specific issues. Cross-platform compatibility can be a concern. | Cross-platform development (e.g., React Native, Flutter) can reduce time/cost. Must test across devices and browsers. |

Recommendations

1. **Operating Platform**: For the expansion of *Draw It or Lose It*, a cloud-based operating platform is recommended. Specifically, Amazon Web Services (AWS) or Microsoft Azure would provide the scalability, high availability, and multi-platform support necessary to deploy, test, and continuously operate the game across a wide range of devices. These platforms offer Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) models, allowing developers to deploy applications using containers, virtual machines, or serverless architectures while minimizing infrastructure overhead.
2. **Operating Systems Architectures**: The recommended cloud platforms support modern modular operating system architectures such as Linux-based (e.g., Ubuntu Server) or Windows Server. These systems are designed with a layered architecture that separates kernel-level functions from user applications, enhancing security and stability. In a distributed game environment, these operating systems allow for efficient task scheduling, process management, and inter-process communication. Cloud-native OS deployments also support container orchestration systems like Kubernetes, allowing multiple game instances to be managed efficiently in real-time.
3. **Storage Management**: Given the high volume of user interactions and media content within *Draw It or Lose It*, object-based cloud storage such as Amazon S3 or Azure Blob Storage is ideal. These storage systems are designed to handle large datasets with high durability and redundancy. They allow for easy scaling, fast retrieval, and secure access to game assets, user data, and backups. Additionally, cloud database services (like Amazon RDS or Azure SQL) can be integrated for structured data, ensuring quick access to user scores, session logs, and game metadata.
4. **Memory Management**: The cloud operating environments recommended make use of virtual memory, paging, and dynamic memory allocation to optimize resource usage. Services like AWS EC2 or Azure Virtual Machines allow for configurable memory sizes, autoscaling policies, and container-based memory limits. These features ensure that each game instance of *Draw It or Lose It* runs efficiently without memory leaks or resource starvation, especially under fluctuating loads. Cloud monitoring tools also enable real-time tracking of memory usage, enabling proactive scaling and performance tuning.
5. **Distributed Systems and Networks**: To support cross-platform gameplay and real-time communication, *Draw It or Lose It* will rely on distributed microservices connected through RESTful APIs and WebSockets for synchronous interactions. A content delivery network (CDN) like Amazon CloudFront can be used to serve assets globally with low latency. Backend components (authentication, score tracking, etc.) should be distributed across availability zones to ensure fault tolerance. Load balancers and API gateways will help route traffic efficiently while minimizing downtime due to server outages or connectivity issues.
6. **Security**: Security is paramount. The recommended platforms offer end-to-end encryption, firewalls, and identity management systems such as AWS IAM or Azure Active Directory to control access to resources. User data will be protected in transit using TLS/SSL protocols and at rest through AES-256 encryption. Authentication mechanisms like OAuth 2.0, MFA (Multi-Factor Authentication), and role-based access control (RBAC) will be used to protect user accounts. Regular audits, automated security patches, and cloud-native threat detection tools (e.g., AWS GuardDuty, Azure Defender) will ensure the environment remains secure.