

Draw it or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## 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 | 03/25/2024 | Alexander Ouellet | Initial Submission |
| 2.0 | 04/07/2024 | Alexander Ouellet | Project 2 Update |
| 3.0 | 04/21/2024 | Alexander Ouellet | Project 3 Update |

## [Executive Summary](#_sbfa50wo7nsh)

The project for The Gaming Room involves the expansion of their current Android-based game, "Draw It or Lose It," into a web-based application that can serve multiple platforms. The game, inspired by the 1980s television game "Win, Lose or Draw," revolves around teams guessing puzzles based on images rendered from a stock drawing library. The web-based version aims to maintain the engaging essence of the game while making it accessible across various devices and platforms.

The transition to a web-based platform presents an excellent opportunity to leverage modern web technologies to create a responsive, scalable, and accessible game application. Utilizing software design templates and patterns, such as Singleton and Iterator patterns, will ensure that the application meets the critical requirements of having a single instance of the game in memory and maintaining unique identifiers for games, teams, and players. The design will also address the necessity of unique game and team names to avoid conflicts during the selection process.

## Requirements

**Summary:**

The Gaming Room’s expansion of "Draw It or Lose It" into a web-based application introduces both business and technical requirements crucial for success:

**Business Requirements:**

1. **Multi-Platform Accessibility**: The game must be available on various operating systems, including desktop and mobile devices, to increase market reach and user engagement.
2. **Cost-Effectiveness**: The deployment and maintenance costs should be minimized to ensure the project remains within budget.
3. **Scalability**: The ability to handle an increasing number of concurrent users without performance loss is essential to support the game’s growth.
4. **Brand Consistency**: The look and feel of the game should be consistent across all platforms to maintain brand identity.

**Technical Requirements:**

1. **Cross-Platform Compatibility**: The game should provide a seamless experience on Linux, Mac, Windows, and mobile devices.
2. **Responsive Design**: The user interface must adapt to different screen sizes and resolutions, ensuring usability and accessibility.
3. **Performance Optimization**: The game must perform well on all targeted platforms, with quick load times and smooth gameplay, even on lower-end devices.
4. **Network Efficiency**: Since the game relies on network connectivity, efficient use of bandwidth and robust handling of network variability are required.
5. **State Management**: Real-time interactions necessitate reliable state management across server and client connections.

## [Design Constraints](#_2et92p0)

Developing "Draw It or Lose It" as a web-based application entails several design constraints, including:

* **Cross-Platform Compatibility:** The application must function seamlessly across different devices and browsers, necessitating a responsive design that adapts to varying screen sizes and resolutions.
* **Network Dependence:** As a web-based application, its performance is contingent on network speed and reliability. Optimizing for low bandwidth and handling disconnections gracefully are crucial.
* **Scalability:** The game should support an increasing number of users without degradation in performance, requiring efficient resource management and server architecture.
* **State Management:** Ensuring that the game state is consistently managed across multiple clients and servers presents a challenge, especially with real-time interactions and updates.

These constraints influence the application's development by necessitating choices that balance performance, accessibility, and user experience. Adopting a microservices architecture could mitigate some scalability and state management concerns, while choosing web technologies that are widely supported ensures cross-platform compatibility.

## [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 for "Draw It or Lose It" outlines the primary classes and relationships essential to the game's functionality:

* **Entity Class:** Serves as the base class for Game, Team, and Player, encapsulating common attributes like **id** and **name**. This demonstrates the object-oriented principle of inheritance, promoting code reuse and modularity.
* **GameService Class:** Implements the Singleton pattern to ensure a single instance manages game operations. It handles game creation, retrieval, and maintains unique identifiers for games, teams, and players, showcasing encapsulation and the management of global state.
* **Game, Team, Player Classes:** Each derived from Entity, these classes represent the game's core components. The relationships between them (Game contains Teams, Team contains Players) illustrate the object-oriented principle of composition, reflecting real-world relationships and promoting a clear hierarchical structure.

The design efficiently fulfills the software requirements by employing object-oriented programming principles such as inheritance, encapsulation, and composition. The use of the Singleton pattern for the GameService class ensures that only one instance manages the game's state, aligning with the requirement for a single game instance in memory. The Iterator pattern, although not explicitly mentioned, will be crucial in iterating over collections of games, teams, and players, especially for enforcing unique names and managing additions.

A diagram of a computer

Description automatically generated

## [Evaluation](#_2o15spng8stw)

**Server Side Evaluation for "Draw It or Lose It" Web Application**

**Server-Based Deployment Assessment**

* **Linux**: The choice of Linux as a server platform is rooted in its unparalleled customization capabilities and flexibility, which are crucial for managing the intensive workloads of a multiplayer online game.

Distinct flavors of Linux, such as Ubuntu Server or CentOS, offer different advantages in terms of package management and system administration, with both providing excellent performance and security features. The availability of open-source tools like Apache and Nginx further enhances the Linux proposition, allowing for a tailored and cost-effective server-side infrastructure.

* **Mac**: macOS Server, with its refined user experience and integration with Apple services, might be considered for niche deployment scenarios. However, in the context of "Draw It or Lose It", where broad accessibility and cost management are paramount, the Mac platform is less compelling despite its merits in design and user experience. There are also arbitrary constraints placed on how one can develop for MacOS server that by definition make it a difficulty if not impossible platform for true multi-platform development.
* **Windows**: Microsoft’s Windows Server is known for its enterprise-level management features and support for .NET applications. The decision to include or exclude this platform will hinge on the game's backend technology choices and the need for integration with Windows-based enterprise systems. While its licensing costs are on the higher end, Windows Server offers a familiar environment for developers and administrators, which may accelerate deployment and reduce training needs.

**Server Side Conclusion**

* The recommendation to deploy "Draw It or Lose It" on a Linux-based server environment is a strategic one, considering both immediate and long-term benefits. Linux’s flexibility in handling high-traffic situations, its security track record, and the potential for cost savings offer a compelling case for a game with the expected reach and engagement levels of "Draw It or Lose It".

**Client Side Evaluation for Multiple Platforms**

**Expanded Cross-Platform Compatibility**

* The web-based "Draw It or Lose It" will be crafted using the latest web standards, ensuring compatibility with a wide array of browsers, including but not limited to, Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Responsive design will be employed through frameworks like Bootstrap or Foundation, which provide a grid system and flexible templates that adapt to different screen sizes—from desktop monitors to tablets and smartphones.
* For mobile platforms, particular attention will be paid to touch interactions and performance optimizations that consider the varied computing power and screen resolutions of devices. Testing on real devices, as well as emulators, will ensure that "Draw It or Lose It" delivers a consistent experience across all platforms. Frameworks like Ionic can be used here to make sure that the game can be deployed in a true cross-platform mobile optimized format.

**Client Side Conclusion**

* By adopting a responsive design approach and targeting a broad spectrum of browsers and devices, "Draw It or Lose It" will offer an engaging and inclusive gaming experience. This strategy embraces the diversity of the gaming community and ensures that players can access the game whenever and wherever they choose.

**Development Tools Evaluation for "Draw It or Lose It"**

**Comprehensive Programming Languages and Tools Overview**

* The application’s user interface will be developed in HTML5, leveraging its multimedia capabilities to render game drawings and animations smoothly. CSS3 will style the interface, with SASS or LESS used to manage complex stylesheets efficiently. JavaScript, with its vast ecosystem, will be the cornerstone for interactivity, with frameworks such as React or Vue.js chosen for their reactive data binding and component-based architecture, which facilitates maintainable and scalable codebases. At this point, there is an early preference for React with Tailwind as the team has worked with these frameworks in the past and appreciate their flexibility and speed to develop.
* On the backend, Node.js presents itself as an excellent choice for its non-blocking I/O model, which can handle numerous simultaneous connections with low overhead - a critical feature for online multiplayer games. Additionally, Python's simplicity and vast library ecosystem can be utilized for scripting and automation within the game’s server-side logic. We can further specify that we can use a framework for Node such as Express specifically built for mobile web apps to provide further value.
* The development environment will be standardized using Docker, ensuring that all team members work within consistent, containerized setups that mirror the production environment. Visual Studio Code, augmented with extensions for linting and debugging, will streamline development workflows. Git, hosted on platforms like GitHub or GitLab, will facilitate version control and collaborative development practices, allowing for continuous integration and deployment pipelines to be established. The preference here is to use GitHub as it is the industry standard and offers tools like GitHub CoPilot AI for developer assistance in tasks like writing test coverage.

**Development Tools Conclusion**

* The comprehensive suite of development tools and languages selected for "Draw It or Lose It" has been strategically chosen to harmonize with the overall goals of the project. These tools not only provide the necessary capabilities to build a responsive, cross-platform web application but also foster a collaborative and efficient development environment that can scale as the project grows.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The macOS Server may present an elegant interface and user experience; however, it is not traditionally chosen for web hosting services, especially for applications requiring extensive cross-platform compatibility. It imposes certain development constraints and carries higher costs, which may not align with the goal of cost-effective deployment. | Linux stands out for its robust performance, security, and scalability—all without significant licensing costs. It’s an open-source platform favored for web applications and provides extensive customization options, making it a suitable choice for hosting "Draw It or Lose It". | Windows Server is recognized for its enterprise-level features and extensive .NET support, which could be advantageous if integration with other Windows-based systems is necessary. However, the licensing costs are higher compared to Linux, which may affect the project's budget. | While not traditional server platforms, the mobile platforms Android and iOS require consideration for server interaction. The backend must be optimized for mobile API calls and potentially leverage cloud services to handle scalability and performance demands effectively. |
| **Client Side** | Developing for macOS demands attention to detail to match the platform’s design standards. The application must be optimized for Safari and other macOS-compatible browsers, ensuring a smooth and native feel. | The client-side on Linux must be browser-agnostic and optimized for various desktop environments. Performance and compatibility testing across popular Linux distributions are vital. | Client-side development for Windows should ensure full functionality in browsers like Chrome, Firefox, Edge, and Internet Explorer, considering the broad user base and the diversity of Windows versions. | iOS and Android development will require a responsive HTML interface, with touch optimization and consideration of mobile browser capabilities. Testing on a variety of screen sizes and resolutions is essential to ensure universal usability. |
| **Development Tools** | Development tools like Xcode can be used for macOS, providing a comprehensive suite for building responsive web applications. Additional tools such as Safari's Web Inspector can be essential for client-side optimization. | ools like Visual Studio Code, JetBrains, and Eclipse are popular on Linux and provide the necessary functionality for developing web applications. Open-source tools align well with the Linux philosophy, often reducing licensing costs. | Microsoft's Visual Studio provides a robust environment for developing web applications on Windows, with extensive support for backend and frontend technologies. | Mobile development can leverage cross-platform tools like React Native or Flutter for app development, and browser developer tools for optimizing the mobile web experience. |

## Recommendations

**1. Operating Platform**

* **Details:** Amazon Web Services (AWS) is recommended as the primary operating platform for the expansion of "Draw It or Lose It." AWS offers a robust, scalable, and efficient cloud infrastructure that supports a wide array of operating systems including Linux and Windows. This flexibility allows for the use of a variety of development environments and tools, accommodating the diverse needs of the game's development and deployment.
* **Details and Benefits:** AWS's global infrastructure ensures reduced latency and increased scalability through its extensive network of Availability Zones. These zones allow for the deployment of applications close to the user base, enhancing performance and user experience. Moreover, AWS's scalability is a key advantage; services like Amazon EC2 and Auto Scaling enable the game to handle an increase in load (e.g., user traffic) by adjusting the compute capacity automatically, thus maintaining smooth gameplay during peak times.
* **Pros and Cons:** The primary advantage of AWS is its comprehensive service offerings that support all aspects of the game's lifecycle, from development to deployment and maintenance. However, the complexity of AWS pricing and the potential for unexpected costs if not managed properly can be seen as drawbacks. The decision to use AWS is based on its ability to provide a high degree of flexibility, scalability, and reliability, crucial for a multi-platform, high-traffic online game.

**2. Operating Systems Architectures**

* **Details:** Within AWS, Amazon EC2 provides the virtual computing environment. This service allows for the selection of different instance types which are optimized for various applications, including compute optimized, memory optimized, and more. For "Draw It or Lose It," choosing compute optimized instances can be beneficial due to their ability to handle heavy computational tasks like real-time data processing during multiplayer gaming sessions.
* **Benefits and Integration:** EC2 instances can be easily configured and scaled to meet the demands of "Draw It or Lose It," ensuring that the game's performance remains stable. Furthermore, the ability to choose between Linux and Windows operating systems on EC2 allows developers to use the OS that best fits their expertise and the game’s technical requirements, providing flexibility in terms of software environment and toolsets.
* **Pros and Cons:** A significant advantage of using EC2 is the control and flexibility it offers over the virtual computing environment. However, managing EC2 instances requires a deep understanding of AWS services and proper configuration to optimize costs and performance. The chosen architecture with EC2 at its core is justified by its ability to scale and adapt to different loads, a critical requirement for a game expected to grow rapidly in user base.

**3. Storage Management**

* **Details:** For storage management, Amazon S3 is identified as the ideal service for storing static assets like game graphics and animations, whereas Amazon DynamoDB is selected for dynamic data such as player scores and game state. S3 provides high durability and availability, which ensures that the game assets are reliably stored and are accessible at any time from any location.
* **Explanation and Benefits:** S3 is known for its robustness and ease of use, offering simple storage solutions that can scale automatically. It integrates well with other AWS services, providing a comprehensive data storage solution for web applications like "Draw It or Lose It." DynamoDB offers quick read and write capabilities ideal for applications requiring real-time access to data, supporting the game's need for fast and efficient state management.
* **Pros and Cons:** The primary advantage of using S3 is its scalability and reliability for asset storage, but it may incur higher costs at scale due to data transfer rates and storage pricing. DynamoDB provides high-performance data storage solutions but requires careful design to avoid potential scaling issues and to manage costs effectively. These services are chosen for their proven capability to handle large-scale web applications, aligning with the game’s requirements.

**4. Memory Management**

* **Details:** Amazon ElastiCache, particularly with Redis, is proposed for memory management in "Draw It or Lose It." ElastiCache enhances web application performance by allowing data to be stored in-memory, reducing the time needed to access frequently requested data. Redis supports data structures such as strings, hashes, lists, and sets, which are useful for various gaming functionalities like leaderboards and session management.
* **Benefits and Integration:** Redis on ElastiCache is highly effective for scenarios requiring quick data access and minimal latency, crucial for multiplayer and real-time aspects of the game. It supports atomic operations to manage complex game states, providing a robust solution for high-performance back-end services.
* **Pros and Cons:** While Redis offers fast data access and excellent support for complex data types, managing Redis instances and ensuring data persistence can be challenging and might require additional configuration and maintenance efforts. However, its integration into the game’s architecture is justified by the need for real-time data processing and quick access times, essential for maintaining a seamless gaming experience.

**5. Distributed Systems and Networks**

* **Details:** To support distributed systems, AWS offers Amazon VPC, which allows the creation of a segregated virtual network within the cloud. This network can be finely tuned to secure and manage traffic between the different components of the game’s architecture, such as the application servers, database servers, and client connections.
* **Benefits and Integration:** VPC provides advanced security features, including security groups and network ACLs, which can be used to create a secure environment for the game data and operations. Additionally, AWS Direct Connect can be utilized to establish a dedicated network connection, enhancing the reliability and speed of data transfers between on-premise systems and AWS.
* **Pros and Cons:** The advantage of using Amazon VPC lies in its flexibility and security, allowing for customized network configurations that align with specific application needs. However, setting up and managing a VPC and Direct Connect can be complex and requires networking expertise. The decision to use these services supports the need for a secure, reliable, and efficient network architecture, crucial for the game’s distributed nature.

**6. Security**

* **Details:** For security, AWS provides several tools and services to safeguard applications. AWS IAM enables detailed access control policies, ensuring that only authorized personnel have access to specific AWS resources. AWS KMS allows for the encryption of data in transit and at rest, which is vital for protecting sensitive player information.
* **Explanation and Benefits:** IAM policies can be tailored to the specific roles within the game’s development and operations teams, providing necessary access without compromising security. KMS integrates seamlessly with other AWS services like S3 and DynamoDB to encrypt sensitive data, ensuring that player data remains secure from unauthorized access.
* **Pros and Cons:** The primary benefit of using AWS security services is their integration with the overall AWS ecosystem, which simplifies management and enforcement of security policies. However, the complexity of configuring and maintaining these security measures can be a drawback, requiring specialized knowledge. The use of these security services is essential for compliance with data protection regulations and for maintaining the trust of the game's users.