

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[Software Design Template 1](#_Toc162794929)

[Table of Contents 2](#_Toc162794930)

[Document Revision History 2](#_Toc162794931)

[Executive Summary 3](#_Toc162794932)

[Requirements 3](#_Toc162794933)

[Design Constraints 3](#_Toc162794934)

[Rationale 3](#_Toc162794935)

Evaluation……………………………………………………………………………………………………………………………………….4

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/11/2024 | Christopher Taylor | Initial draft of software design document |
| 2.0 | 03/31/2024 | Christopher Taylor | Added Evaluation section to software design document |
| 3.0 | 04/08/2024 | Christopher Taylor | Final draft of software design document, added Recommendations section |

Instructions: Fill in all bracketed information on page one (the cover page), in the Document Revision History table, in the footer, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## Executive Summary

The objective of this project is to evaluate the characteristics, advantages, and weaknesses of various platforms for deploying the game application Draw It or Lose It. The game, inspired by the television game Win, Lose or Draw, challenges teams to guess drawings rendered from a stock library within a time limit. The client, The Gaming Room, seeks to expand the game onto multiple platforms using different software patterns in a distributed environment. To aid in their decision-making process, an evaluation of traditional operating platforms (Linux, Mac, and Windows) as well as mobile platforms has been conducted.

Each platform offers distinct advantages and weaknesses. Linux provides robust support for web-based applications and offers low licensing costs, making it an attractive option for hosting the game. Mac systems, while suitable for hosting web applications, come with moderate licensing costs. Windows platforms offer robust hosting capabilities but also incur moderate licensing costs. For mobile platforms, Android development caters to a diverse range of devices but requires optimization for different hardware configurations, while iOS development targets Apple's mobile devices with attention to compatibility and performance optimization.

Ultimately, the choice of platform will depend on factors such as scalability, compatibility, licensing costs, and development considerations. By carefully evaluating each platform's characteristics, The Gaming Room can make an informed decision that aligns with their goals for expanding Draw It or Lose It to multiple platforms within a distributed environment.

## Requirements

Broad Platform Support: It's imperative that Draw It or Lose It functions seamlessly across iOS and Android platforms to engage a wide range of smartphone users effectively.

Budget Compliance: Adhering to the allocated budget is essential for efficient planning and execution throughout the development process of Draw It or Lose It.

User Engagement: The design of Draw It or Lose It must be captivating and user-friendly, encouraging extended interaction with the game platform to enhance user engagement.

Consistent Functionality: Ensuring that the mobile app version of Draw It or Lose It mirrors the key features of the game's original concept to provide a cohesive and enjoyable user experience across different platforms.

## Design Constraints

Cross-Platform Framework Utilization: Leveraging a cross-platform framework that supports both iOS and Android from a single codebase is strategic for Draw It or Lose It, enabling streamlined development and cost-effective management.

Proper Design Patterns: Implementing appropriate design patterns for the game's entities, such as Drawings, Clues, and Game Rounds, ensures scalability and maintainability of Draw It or Lose It.

Entity Integration: Integrating essential entities like Players, Teams, and Game Sessions into Draw It or Lose It's software design pattern enhances the game's structure and functionality.

Strategic Feature Implementation: Draw It or Lose It's development will prioritize embedding essential features within budgetary constraints, laying the groundwork for future expansion based on user feedback and additional funding.

Performance and Usability Focus: Enhancing Draw It or Lose It's speed, responsiveness, and minimizing resource consumption are paramount to delivering a satisfying user experience across various devices.

## Rationale

Opting for a cross-platform development approach for Draw It or Lose It directly addresses the necessity for widespread device compatibility while considering the project's financial constraints. By developing the game to run seamlessly on both iOS and Android platforms, we maximize its reach to a larger audience without inflating development costs.

The decision to incorporate proper design patterns and integrate essential entities into Draw It or Lose It's architecture reflects a strategic initiative aimed at ensuring the game's scalability, maintainability, and overall functionality. By adhering to established design principles and integrating core elements like Drawings, Clues, and Game Rounds, we create a robust foundation for future enhancements and updates.

Prioritizing app performance and a user-friendly interface is paramount for Draw It or Lose It's success. By focusing on optimizing speed, responsiveness, and resource utilization, we aim to deliver a satisfying gaming experience across a variety of devices. Additionally, a streamlined user interface enhances user engagement and retention, which are crucial for driving the game's growth and revenue generation over time.

**Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Hosting** | **Licensing Cost** | **Development Considerations** | **Development Tools** |
| **Linux** | **Yes** | **Low** | Linux offers robust support for hosting web-based applications. It provides a stable environment for server-side deployment, often favored for its reliability and scalability. Development considerations include ensuring cross-platform compatibility and responsiveness, requiring thorough testing across various web browsers and devices. | JavaScript, HTML, CSS |
| **Mac** | **Yes** | **Moderate** | Mac systems are suitable for hosting web applications, offering reliable performance and scalability. However, licensing costs may be moderate. Development considerations include compatibility with various web browsers and mobile devices, emphasizing the need for responsive design and thorough testing to ensure optimal user experience across platforms. | Xcode, Swift, Objective-C |
| **Windows** | **Yes** | **Moderate** | Windows platforms provide robust hosting capabilities for web applications, with moderate licensing costs. Development considerations include ensuring compatibility with various web browsers and mobile devices, requiring thorough testing to guarantee seamless functionality across platforms. Emphasis should be placed on responsive design and performance optimization. | Visual Studio, C#, .NET |
| **Android** | **No** | **Low** | Android development focuses on creating applications for the Android operating system, catering to a diverse range of devices and screen sizes. Development considerations include optimizing user interfaces for various Android devices and versions, ensuring compatibility and consistent performance across different hardware configurations. Thorough testing is essential to address fragmentation issues. | Android Studio, Java, Kotlin |
| **iOS** | **No** | **Moderate** | iOS development targets Apple's mobile devices, requiring attention to compatibility and performance optimization for iPhones and iPads. Development considerations include ensuring compatibility with various iOS devices and versions, optimizing user interfaces for different screen sizes, and adhering to Apple's design guidelines for a seamless user experience. Thorough testing is necessary to identify and resolve platform-specific issues. | Xcode, Swift, Objective-C |

Evaluation Summary:

Linux, Mac, and Windows offer server-based deployment methods suitable for hosting web-based software applications. However, licensing costs may vary, with Linux generally offering lower costs compared to Mac and Windows.

Supporting multiple client types (iOS, Android, and desktop) requires careful consideration of compatibility, performance optimization, and responsive design. Development considerations include ensuring cross-platform compatibility, which may increase development time and expertise requirements.

Relevant programming languages and tools such as JavaScript, HTML, CSS, Xcode, Swift, Objective-C, Visual Studio, C#, .NET, Android Studio, Java, and Kotlin are essential for building software for various platforms. These technical requirements impact the development team by necessitating expertise in multiple languages and tools, potentially requiring collaboration among specialized teams. Additionally, licensing costs may apply to certain development tools, influencing budget considerations.

**Recommendations**

*Operating Platform:* Following a thorough assessment of Draw It or Lose It's requirements and the imperative for scalability and compatibility across diverse computing environments, my recommendation leans towards leveraging a cloud-based server platform. Specifically, Amazon Web Services (AWS) emerges as the optimal choice owing to its robust infrastructure and versatile suite of services. AWS stands out for its extensive toolkit and resources, poised to seamlessly facilitate the game's expansion across various operating systems and computing environments.

*Operating System Architectures:* In terms of the operating platform's architecture, I advocate for embracing a microservices approach. This architectural framework entails decomposing the application into smaller, independently deployable services, fostering easier development, deployment, and scalability. By embracing microservices and capitalizing on containerization technology like Docker, we can ensure a consistent and efficient performance across different environments.

*Storage Management:* To meet the storage management needs, I propose a hybrid approach utilizing a blend of AWS storage services. Amazon Simple Storage Service (S3) offers scalable and durable object storage for game assets, while Amazon DynamoDB provides low-latency performance and automatic scaling for storing game-related data.

*Memory Management:* The recommended operating platform integrates advanced memory management techniques such as garbage collection and memory pooling to optimize memory usage and enhance performance for Draw It or Lose It. Garbage collection effectively mitigates memory leaks, while memory pooling minimizes overhead and fragmentation, thus enhancing stability and resource utilization.

*Distributed Systems and Networks:* Enabling seamless communication between various platforms necessitates implementing a distributed software architecture with REST APIs for interaction between game clients and backend services. Incorporating messaging queues and WebSocket protocols will facilitate real-time communication, with fault-tolerant designs and redundancy mechanisms addressing network dependencies and outages.

*Security:* Emphasizing the paramount importance of security, my proposal includes the implementation of robust authentication, authorization, and encryption mechanisms to safeguard user information across platforms. Leveraging AWS Identity and Access Management (IAM) alongside data encryption through AWS Key Management Service (KMS) and Transport Layer Security (TLS) protocols will ensure secure access and transmission of sensitive data. Continuous monitoring and auditing via AWS CloudTrail and Amazon CloudWatch will afford visibility into security events, enabling timely response to threats.