

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.3

## 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 |
| --- | --- | --- | --- |
| 1.3 | 08/07/2023 | Scott Martel | Analyzes the characteristics of and techniques specific to various systems architectures and makes a recommendation to The Gaming Room. |

**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 Gaming Room has approached Creative Technology Solutions (CTS) with a request to develop a web-based version of their Android game, Draw It or Lose It. The game is based on the concept of Win, Lose or Draw, where teams compete to guess a puzzle being drawn. CTS will address the software requirements by creating a web application that allows multiple teams with multiple players to participate. To ensure uniqueness, game and team names will be checked for availability upon creation. Additionally, the application will use unique identifiers for each instance of a game, team, or player, ensuring only one instance exists in memory at any given time. By meeting these requirements, CTS will streamline the development process and enable the client to reach a wider audience across various platforms.

## Requirements

**Client's Business Requirements:**

1. Multi-team Gameplay: The game application should support multiple teams' participation, allowing them to compete in guessing the puzzles.
2. Unique Game and Team Names: The application should enforce uniqueness for game and team names, allowing users to check if a name is already in use before choosing one.
3. Platform Expansion: The client wants to expand the game's availability beyond the current Android app and develop a web-based version that can be accessed on different platforms.

**Client's Technical Requirements:**

1. Unique Identifiers: Each instance of a game, team, and player should have a unique identifier to ensure only one instance exists in memory at any given time.
2. Real-time Updates: The game application should provide real-time updates to all teams, such as rendering drawings and notifying teams of puzzle solutions.
3. Cross-platform Compatibility: The game application should be compatible with various platforms, including desktops, laptops, tablets, and mobile devices, to reach a wider audience.

## [Design Constraints](#_2et92p0)

1. Cross-platform Compatibility: Ensure compatibility across different platforms and screen sizes, utilizing responsive design techniques.
2. Security Considerations: Implement robust security measures to protect user information and prevent unauthorized access.
3. Scalability and Performance: Design for high scalability and optimize server handle concurrent users without performance degradation.
4. Real-time Communication: Incorporate real-time communication technologies for instant updates without page refreshes.
5. User Interface Design: Create an intuitive and visually appealing user interface that works seamlessly across devices.

## [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)

**Game Service:** This class is a Singleton ensuring only one instance of the game service exists at any given time. It contains the collection of games and methods for managing them and keeps track of the next IDs assigned to games, players, and teams.

**Entity:** The Game, Team, and Player classes are all inherited from this Entity class. It provides a common base of the properties name and id to the inheriting classes.

**Game:** The Game class is connected to the GameService class with a zero-to-many relationship, meaning a GameService can have zero or multiple games. Each game has a list of Teams and methods for adding teams.

**Team:** The Team class is associated with the Game class with a zero-to-many relationship, meaning a Game can have zero or multiple teams. Each team contains a List of Players and methods for adding players.

**Player:** The Player class is connected to the Team class a zero-to-many relationship, meaning a Team can have zero or multiple players. It has a constructor to create a new player.

The object-oriented programming principles demonstrated in this diagram are:

**Inheritance:** Game, Team, and Player classes all inherit from the Entity class, promoting reusability.

**Encapsulation:** Each class encapsulates specific attributes and behaviors. For instance, GameService manages the game operations, Game manages teams, Team manages players.

**Polymorphism:** The overridden function toString() method is an example of polymorphism. It is present in each class, but it will produce different outputs based on the class it is called on.

**"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** | Mac provides a stable and secure environment for hosting web-based applications. It comes with a user-friendly interface and is based on Unix architecture, so there is robust built-in command line tool. However, it comes at a significantly higher cost compared to Linux or Windows. | Linux is open-source and is extremely stable and secure. It is also very cost effective as it is free and offers flexibility with its highly customizable features and extensive programming language support. However, it requires more technical expertise than Mac or Windows. | Windows is widely used, and even offers versions specifically for running servers. Windows has a very compatible development environment that includes strong integration capability with Microsoft technologies. However, it does come with security vulnerabilities and is not as stable as Mac or Linux. | Mobile devices are highly portable and include wide user accessibility, and unique capabilities. Some disadvantages are platform fragmentation and device compatibility testing. Mobile devices also have significant limitations in terms of battery life and computational power compared to laptop and desktop computers. |
| **Client Side** | Developing for Mac clients may involve higher costs due to the need for Mac hardware and software development tools. Developers will need expertise in Mac-specific technologies and the Mac ecosystem. | Developing for Linux clients can be cost-effective as Linux is open-source and freely available to anyone. Linux development also requires experienced developers who are comfortable navigating the open-sourced Linux ecosystem. | Development for Windows clients may involve costs related to licensing and proprietary software tools. However, Windows provides a wide range of development resources and frameworks, offering flexibility in terms of cost options. | Development for mobile clients may require investment in platform-specific development tools, licenses, and device testing. Additional time and effort are also necessary to ensure compatibility across different platforms and optimize the user interface for all users. |
| **Development Tools** | Building software for mac requires Mac-specific technologies, such as using Apple’s Xcode IDE, and programming languages used for IOS development such as Swift and Objective-C. | To build software for Linux developers need to be familiar with the various Linux distributions, libraries, tools, and possess a knowledge of command-line interfaces as well as open-source technologies. | The Windows Visual Studio IDE is commonly used to build software for Windows. C#, C++, and the .NET framework are all popular and well supported programming languages used for developing for Windows. | Java and Kotlin are commonly used for Android app development along with the Android Studio IDE. Swift and Objective-C are used for iOS app development used along with Apple’s Xcode IDE. |

## 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**: Linux is my top recommendation for the operating platform due to its unmatched cost-effectiveness, stability, and security. With its inherent flexibility and vast library of compatible software, Linux not only meets current needs but also provides a robust foundation for "Draw It or Lose It" to seamlessly scale and integrate into diverse computing environments in the future.
2. **Operating Systems Architectures**: Linux gets its robustness from a Unix-like architecture, emphasizing modularity and isolation of processes. This ensures high stability, as a failure in one component often doesn’t compromise the entire system. Its open-source nature invites a global community of developers to contribute, fostering continuous improvements and quick responses to vulnerabilities. This collective effort enhances its security, making it particularly resilient against threats. Moreover, being freely available, Linux presents a cost-effective solution, reducing overhead costs for hosting web applications without compromising on performance or safety.
3. **Storage Management**: I suggest leveraging Amazon Web Services' cloud storage, complemented by a relational database like MySQL. This combination harmonizes perfectly with Linux, offering scalable storage solutions. The flexibility of cloud storage allows for effortless expansion, ensuring data availability from any location.
4. **Memory Management**: Software deployed on Linux servers, utilizes memory management techniques provided by the Linux kernel. The kernel manages memory allocation, deallocation, and optimization, ensuring efficient usage of system resources for running the “Draw It or Lose It” software.
5. **Distributed Systems and Networks**: To enable communication between various platforms, a distributed software architecture can be implemented. The “Draw It or Lose It” game application can use APIs and protocols to facilitate communication and data exchange between different devices and platforms. Implementing fault-tolerant mechanisms, such as redundant servers and load balancing can ensure uninterrupted gameplay even in the event of network disruptions or outages.
6. **Security**: Linux is open source so people are always looking out for vulnerabilities and can patch them very quickly making Linux incredibly secure. However, to maximize security, it's recommended to employ additional measures. This includes setting up firewalls and periodic security audits. Regular updates to the system and software packages will also ensure we're protected against known vulnerabilities. Amazon Web Services also offers secure storage that includes encryption of data at rest and in transit as well as multi factor authentication.