

Draw it or Lose it

# **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 | 04/23/2023 | Euntak Jang | We suggested adopting React Native or Flutter for cross-platform development, allowing Draw It or Lose It to work on numerous platforms. In addition, we covered cloud-based storage options, distributed systems, and security measures to provide a scalable, safe, and accessible gaming experience. |

**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 software design task at hand entails the creation of a gaming application that organizes numerous games, teams, and players. The suggested solution is a Java-based application that uses object-oriented programming and singleton design techniques to effectively handle game data. The application supports the insertion of games, teams, and players and guarantees that there is only one instance of the game service in the whole application. Clients must be aware of the usage of the singleton pattern and the modularity of the code, which will permit future upgrades and changes.

## Requirements

The following are the client's needs for the game application:

1. The ability to add and manage several games.
2. Capability to add and manage teams within each game.
3. The ability to add and manage members within each team.
4. Memory utilization and application performance are optimized.
5. Scalability and flexibility for future enhancements.

## [Design Constraints](#_2et92p0)

Design restrictions for designing the gaming application in a web-based distributed system include:

1. Limited resources: The web-based environment may have memory and processing power restrictions, which might affect program performance.
2. Network delay: Due to the dispersed nature of the web-based environment, network latency might influence the responsiveness of the application.
3. Concerns about security: Data confidentiality, integrity, and availability must be protected in a web-based system.
4. Compatibility issues: The program must be compatible with a variety of web browsers, operating systems, and devices.

These design limitations have ramifications for application development, such as the necessity for effective data management, application performance optimization, and the usage of secure coding methods.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The presented UML class diagram has five classes: Entity, Game, Team, Player, and GameService. Entity is the basic class for Game, Team, and Player, and it has common properties such as 'id' and 'name.' The Game, Team, and Player classes all derive from the Entity class, and each has its own set of capabilities. The Game class handles teams, the Team class maintains players, and the Player class represents individual players.

The GameService class uses the singleton design pattern to ensure that there is only one instance of the service across the application. This class is in charge of handling the games, teams, and players as well as maintaining the general state of the application.

The graphic illustrates object-oriented programming principles such inheritance, encapsulation, and polymorphism. Inheritance is utilized to eliminate code duplication by having Game, Team, and Player classes inherit from Entity. Encapsulation is used to limit access to certain class characteristics and functions, maintaining data integrity. Polymorphism allows multiple classes to be handled as objects of a single superclass (Entity), which simplifies code and allows for more efficient handling of varied object kinds. These concepts lead to the successful and efficient fulfillment of software requirements.

**"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 servers provide a dependable, user-friendly environment for hosting web-based applications, and they integrate seamlessly with other Apple devices. They can, however, be costly.  Advantage: Simple installation and management  Disadvantage: More expensive than Linux and Windows, but with flawless connection with other Apple products. They can, however, be costly.  Advantage: Simple installation and management  Disadvantage: More expensive than Linux and Windows. | Linux offers a highly configurable, secure, and cost-effective platform for hosting web-based applications that can scale to thousands of participants. It does, however, need additional technical knowledge.  Advantage: High customizability and low cost  Disadvantage: Setup and administration require higher technical expertise. | Windows servers provide a familiar environment for web-based applications, particularly those that use Microsoft apps and services, although they can be more resource-intensive.  Advantage: Seamless connection with Microsoft products.  Disadvantage: More resource-intensive than Linux servers. | Due to their low processing power and storage capacity, mobile platforms are often unsuitable for server-side hosting of web-based applications.  Advantage: This does not apply to server-side hosting.  Disadvantage: Server-side operations have limited processing power and storage. |
| **Client Side** | Developing for Mac customers entails taking into account the expense of Apple hardware and software, as well as knowing macOS-specific frameworks and languages such as Swift and Objective-C. Using Xcode speeds up the process, but developers must assure cross-platform compatibility for the web-based application. | Because of its open-source nature, developing for Linux clients allows for flexibility and freedom, yet supporting several Linux versions and desktop environments can be difficult. Developers must guarantee that the web-based application operates properly across all distributions and browsers. | Developing for Windows clients gives developers access to the greatest desktop user market share, but developers must consider the expense of Microsoft development tools and understanding Windows-specific frameworks. Cross-browser compatibility is required for the web-based application to run on Windows systems. | Developing for both the Android and iOS platforms necessitates specialized knowledge and expertise with platform-specific languages and frameworks (Java/Kotlin for Android and Swift/Objective-C for iOS). Developers must build a mobile-friendly HTML interface that is compatible with varied screen sizes and device capabilities. |
| **Development Tools** | Swift, Objective-C, and C++ are programming languages that may be used to create applications for Mac operating systems. Xcode and AppCode are two popular Integrated programming Environments (IDEs) for Mac programming. Homebrew, CocoaPods, and Git are more tools that may be utilized in Mac programming. The technological constraints for Mac development might have a variety of effects on a development team. Team members, for example, may require particular skills and understanding of Mac development tools and languages. If the product is to be released on several versions of macOS, multiple development teams may be required. There may also be license fees connected with some programming tools, like Apple Xcode. | Programming languages such as C, C++, and Python are commonly used in Linux development. Eclipse, IntelliJ IDEA, and Visual Studio Code are popular IDEs for Linux programming. Git, Docker, and Jenkins are some more tools that may be utilized in Linux development. The technological requirements for Linux development can have a variety of effects on a development team. Team members, for example, may need to be familiar with Linux operating systems and the command line interface. If the product is to be distributed on several versions of Linux, multiple development teams may be required. Some development tools may be open source and free, while others may have license fees. | Programming languages such as C, C++, and Python are commonly used in Linux development. Eclipse, IntelliJ IDEA, and Visual Studio Code are popular IDEs for Linux programming. Git, Docker, and Jenkins are some more tools that may be utilized in Linux development. The technological requirements for Linux development can have a variety of effects on a development team. Team members, for example, may need to be familiar with Linux operating systems and the command line interface. If the product is to be distributed on several versions of Linux, multiple development teams may be required. Some development tools may be open source and free, while others may have license fees. | Relevant programming languages for developing applications for distribution on mobile devices such as Android and iOS include Java, Kotlin, Swift, and Objective-C. Android Studio, Xcode, and Visual Studio are popular IDEs for mobile programming. Git, Gradle, and CocoaPods are among more technologies that may be utilized in mobile development. The technological requirements for mobile development can have a variety of effects on a development team. Team members, for example, may need to be familiar with the mobile platform and its special needs. If the program is to be distributed on many mobile platforms, multiple development teams may be required. Some development tools are open source and free, while others, such as the Apple Developer Program for iOS programming, may have license fees. |

## Recommendations

1. **Operating Platform**: A cloud-based server platform, such as Amazon Web Services (AWS) or Microsoft Azure, is recommended. These platforms provide scalable, dependable, and cost-effective alternatives for bringing Draw It or Lose It into new computer settings. They also offer outstanding compatibility for a wide range of operating systems, making it easier to distribute the game across several platforms.
2. **Operating Systems Architectures**: The cloud platform of choice supports a variety of operating system architectures, including Windows, Linux, and macOS. Because of this adaptability, Draw It or Lose It may operate on the most popular operating systems and reach a wider audience. Furthermore, the cloud platform offers a variety of services, such as virtual machines, containers, and serverless computing, to meet the needs of various applications and optimize resource utilization.
3. **Storage Management** We recommend utilizing a managed storage solution such as Amazon S3 or Azure Blob Storage. These services enable long-term, scalable, and secure storage for game assets, user data, and other game-related information. They also include built-in redundancy and data backup features, guaranteeing that the game data is always secure and accessible.
4. **Memory Management**: Memory management for Draw It or Lose It is handled by the selected cross-platform framework (React Native or Flutter), which isolates memory allocation and deallocation. These frameworks employ garbage collection algorithms to manage memory automatically, releasing resources when they are no longer required. Furthermore, the framework's memory management algorithms aid in the prevention of memory leaks and the overall efficiency of the program.
5. **Distributed Systems and Networks**: Draw It or Lose It: Distributed Systems and Networks It can use distributed systems and networks to allow communication between platforms. This may be accomplished by employing a client-server architecture, with game clients (running on many platforms) communicating to a central server. The server can manage game state synchronization, matchmaking, and other client interactions. To create a flawless gaming experience, network connectivity, latency, and gracefully resolving outages should all be taken into account
6. **Security**: Encrypt data storage and transfer to safeguard user information and ensure platform security. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols can be used to provide secure communication between clients and servers. Furthermore, user authentication should be performed utilizing secure mechanisms such as OAuth2 or OpenID Connect to ensure that user data is only accessible by authorized parties. To assist protect user data, the cloud storage provider (AWS or GCP) should have built-in security mechanisms and adhere to industry-standard security procedures.