

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 5**

**Recommendations 8**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/26/2024 | Cody VanGosen | Implemented singleton pattern to assist in single instance of games running. Created Entity class and made the Player and Team classes derive from it. Refactored Game Class to inherit from the new Entity class. |
| 1.1 | 06/09/2024 | Cody VanGosen | Expanded upon the Evaluation section. Dove into further detail to provide extensive analysis of costs, time, and development requirements. Included analysis of capabilities and challenges associated with cross-platform implementation of proposed server offerings. Corrected errors associated with Table of Contents reference pages. |
| 1.2 | 06/19/2024 | Cody VanGosen | Updated provided information in the Recommendations section. Provided further insight and discrete recommendations to fulfill customer needs. Expanded upon security, network, and software architecture in more detail to provide a better foundation of understanding for product delivery. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room seeks to expand their Android-only game, Draw It or Lose It, to a web-based cross platform experience available across a multitude of devices. This document will outline the software design implementation for this process. The game is a guessing game that pits competing teams against one another while they work together to identify images from a library of stock images. Integration to a web-based platform will facilitate the use of a distributed system to support unique team and game identifiers, thereby ensuring only a single instance of a game is stored in memory at a time, while also allowing real-time gameplay across different devices. The offered solution seeks utilizing web-based technologies to meet the requirements of while facilitating ease in scalability, performance, and a seamless user experience.

## Requirements

Game Structure: Each game must have the ability to host one or more teams, each with one or more players.

Uniqueness: Team and game names must be unique. In addition, ID values will be employed to account for duplicate player names.

Instance Management: Only one instance of a game can exist in memory at a time, with unique identifiers for games, teams, and players.

## [Design Constraints](#_2et92p0)

Scalability: The application must manage multiple concurrent users and games. While ensuring only one unique instance of each game is running at one time and stored in memory.

Performance: Real-time updates are critical, necessitating efficient rendering and data handling. This will facilitate access across multiple different devices and integrate turns and play of each game instance.

Security: Protect user data and ensure secure communication between clients and servers. Data storage and transmission must ensure that private user information is kept secure and confidential.

Cross-Platform Compatibility: The application must be accessible from various devices and operating systems. The company is seeking to offer real-time play and expand from the Android platform. Cross-platform capabilities will facilitate the success of this web-based solution.

## [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 class diagram below represents a domain model of the game application that consists of the following classes, GameService, Game, Team, Player, Entity, ProgramDriver, and SingletonTester. The Entity class serves as the base class of all entities within the application and provides the shared attributes of ID and name. From the pattern of inheritance, the other classes can inherit these attributes and associated methods. The Entity class inheritance pattern reduced code redundancy and provided consistency across the program. The Game class represents a single game running within the application. This includes a list of teams and players. This also includes methods to add players and access a string representation of each team. The Player class provides storage for an individual player and similarly access to a string representation of each. These classes follow the principle of composition, where larger or more complex objects are built upon through the combination of smaller and simpler objects. For example, the Game class consists of various teams, each team is filled with multiple players. This provides a hierarchical structure to manage, access, update, and delete games, teams, and players in a convenient and efficient manner.

**"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** | Mac servers offer a suite of their industry-leading security and stability. This leads to a significant increase in cost resulting in their scarcity of web hosting.  Offers familiar environment for developers experienced with Mac hardware. Prohibitive costs, limited server=specific optimizations, and less support compared to Linux servers. Offers lightweight HTTPS-based protocol named MDM, very suitable choice for cloud hosting. | Linux servers provide a lightweight and open-source solution. It provides flexibility in scaling, stability, and security while being cost-effective in nature. Very cost-effective while offering a large amount of customization and scalability. Community support is large with a plethora of forums and documentation. Requires expertise in command-line driven OS to effectively manage. Apache is an open-source HTTP server often used with Linux. In addition, common hosts such as AWS offer very affordable and scalable external hosting. | Windows provides a large commercial solution for enterprise support. The popularity and widespread use coupled with its integration with Microsoft services can result in higher overhead costs. Very user-friendly with excellent support for .NET applications resulting in widespread use in many enterprises. This results in higher licensing costs as well as a risk of security vulnerabilities that come bundled with a large OS like Windows. Makes use of an Internet Information Services web server to manage web publishing and deployment through use of the IIS Web Deploy Handler. | Mobile devices are atypical solutions for server-side solutions. This is due to the large amount of variance in available hardware as well as the limitations of mobile hardware in terms of form factor and structure. They are more effective from a client-side solution. Offers many benefits to client-side applications that might run through the mobile server. Offers seamless integration with many different cloud service offerings and hosts. Device hardware and capability are the two main bottlenecks for this server solution. Storage and processing power are much more limited than traditional computer OS servers. Often rely on cellular or dedicated internet connectivity. |
| **Client Side** | Mac client development often requires expensive upfront costs due to the closed nature of Mac and iOS software platforms. This can make cross-platform capability more difficult or costly to integrate. The initial overhead is large due to the dependence upon integrating into Apple’s device ecosystem. Development time would be moderate due to the similarities with the Unix-based architecture. Requires expertise in Mac and iOS environments to understand user experience and expectations. Likely require a deep understanding of APIs and importance of considering programming within a container to foster portability and promote cross-platform access. | The open-source community offers dedicated support for Linux clients, but in terms of general consumer use it is far less common. This facilitates more time, cost, and effort to ensure widespread compatibility. Offers a very low-cost solution due to many open-source client offerings being completely free or donation driven. Requires a moderate time commitment due to the diverse nature and amount of available Linux distributions. Careful consideration must be given to using common command line-driven Linux distros or the growing consumer-friendly GUI offerings. Will require expertise in traditional command-line driven OS and programming as well as experience with multiple Linux distributions. This can present challenges with cross-platform capability between distros. | Windows clients are plentiful and operate off common integrated environments. Its widespread popularity facilitates ease of development while offering copious amounts of online support and available tools. This offers a moderate to prohibitive cost due to the high licensing fees associated with the Windows OS. Time requirements for development are moderate and like Mac development time. This is due to the broad support of the platform, familiarity of the platform, and the extensive frameworks and included software/security packages. This will require expertise in navigating and working with the traditional UX/UI of Windows. It will be important to understand several different Windows versions. Since many users may still use devices with older versions. | Mobile client development is extremely important and a common user access point. Development requires careful consideration of varying factors and differing OS offerings as well as different OS versions. The costs are high compared to the other offerings. This is due to the significant difference in architecture between popular Android and iOS devices. As well as the above another issue stems from the large variety of available hardware offerings when compared to the relative similarity of OS versions seen with the other client options. Will require a large amount of time for testing and optimization to ensure broad support for available hardware and software versions. Wil necessitate expertise in mobile development frameworks. Knowledge of Java, Kotlin, Swift, C++, React Native, and Flutter will all be useful to effectively work with various devices. |
| **Development Tools** | Xcode and Swift are two of the largest standard languages for Mac and iOS development. Cross platform frameworks such as React Native are also often used. Requires experienced developers that are familiar with the Apple ecosystem. Licensing costs are high due to the requirement of the Apple Developer Program fees. VS Code (or the full version of VS), while IntelliJ provides a common Java-based IDE when incorporating JVM or container-based solutions. | IDE offerings are plentiful with Eclipse and IntelliJ IDEA being two common sources. Languages such as Java, JavaScript, and Python are often used depending on requirements and front or back-end solutions. In addition, other common languages include PHP, Ruby, MySQL. Several IDEs are popular for Linux development such as Eclipse and IntelliJ IDEA. Inherently promotes use of open-source tools, fosters cross-compatibility, but will require a unified team to meet the development needs of the desired amount of Linux distros. Low cost and often free due to the availability of open-source tools and documentation. | Visual Studio and .NET are extremely robust offerings for Windows. They work very within the Windows inter-connected services and systems. C++ and C# are two common language offerings. SQL Server and JavaScript are often used for web-development with Windows solutions in mind. Will necessitate familiarity with Microsoft and Windows architecture. Prohibitive costs due to licensing fees. In addition, costs are further inflated due to license fees associated with Visual Studio and Windows Server commercial/enterprise licenses. Best standards of HTML/CSS must be implemented to facilitate cross-platform functionality and display of web-development. | Android Studio is a common IDE offering a multitude of emulators for testing various hardware and Android OS versions. Java and Kotlin are two of the most commonly language implementations used. For iOS mobile development requires knowledge of Swift and C++. Oftentimes a general understanding of React native can also be beneficial. Will require knowledge of mobile frameworks, API, and several language libraries to facilitate ease of development and lower costs and time investment. Licensing costs are variable according to the chosen framework. With solutions such as Android Studio and the associated fees being lower than the comparable Xcode and Apple Developer Program licensing fees. |

## Recommendations

1. **Operating Platform**: Linux provides an impressive set of compromises for the server-side operations due to the cost-effective nature of the open-source platform as well as the stability and customizable security it offers. This makes Linux a robust server support offering due to its adaptability to meet the business demands securely and efficiently. Furthermore, the open-source nature and community provided support offer many benefits to stay up-to-date with latest emerging advancements in technology and computing. Client-side support would facilitate the need to cater to Windows, Mac, and other mobile OS offerings to reach a broad audience. Tailoring the development to work in this cross-platform environment will ensure the highest levels of user accessibility. This will lead to an increase in cost but is necessary to ensure the highest level of cross-availability.
2. **Operating Systems Architectures**:

Linux-based server architecture offers a scalable and cost-effective approach. Containerization technologies such as Docker, allow for deployment in fixed settings while conferring the benefits of server scalability according to business needs. These Docker containers could be used to encapsulate the game server environment allowing for on-demand upwards or downwards scaling. This sense of provided flexibility will ensure a robust server offering to effectively handle the varying size of server loads more efficiently. The wide availability of open-source resources and APIs such as RESTful can allow for efficient communication between clients and servers. Such an API helps to facilitate interoperability while maintaining consistent and responsive gaming experiences in cross-platform environments.

1. **Storage Management**: Cloud-based storage solutions will best serve the business needs in this case. The importance of centralized storage and access is paramount for cross-platform and device communication in real-time. Solution suites such as AWS offer unprecedented reliability and scalability. Such a selection for cloud storage offers a centralizes location that is openly accessible in real-time from any geographical location. AWS offers a wide selection of support and features including concepts such as using S3 for object storage and pairing it with a relation database management offering. A relational database such as PostgreSQL would offer a structured solution for data management.
2. **Memory Management**: Linux operating systems employ efficient memory management techniques such as paging and swapping to offer efficient use of physical memory. This more easily manages handling multiple instances of games and user data. Paging facilitates the use of disk space as an extension of RAM, thus ensuring the server can handle larger workloads without issue due to running out of memory. Swapping further enhances on this practice by moving inactive pages from active memory over to disk space instead, thereby freeing up RAM to instead be used for active processes. These techniques will all help ensure “Draw It or Lose It” is capable of running smoothly despite the load it might be under.
3. **Distributed Systems and Networks**: The need for real-time communication and updates between clients and server could easily be met through implementation of WebSocket. This service offers a full-duplex communication channel through a single persistent connection. The interactive and persistent game requirements for “Draw It or Lose It”, makes this a great solution. The use of WebSocket helps to ensure updates are pushed out to clients in real-time, thus ensuring a seamless gaming experience for all users. Complementing this with robust error handling and fallback systems can help mitigate damage or downtime due to network outages further facilitating the persistent game world.
4. **Security**: HTTPS would be an important asset for offering secure communication due to its wide use cases and substantial number of industry adoptions. This service is nearly essential for ensuring secure communication due to the encryption it provides to data sent between clients and servers, which helps to prevent tampering and eavesdropping along the transmission route. Services such as OAuth can facilitate user authentication while encrypting sensitive data in storage and during transit. This will help to ensure user data remains protected as well providing a standardized method for user authentication and authorization. The use of structured and scheduled security audits will manage updates to potential vulnerabilities that may present themselves at later times. It is crucial these audits are routine as they will provide insight into mitigating potential security vulnerabilities.