

Draw It or Lose It

# **CS 230 Project Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design** 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 | 03/18/2023 | Tam Huynh | The document is a CS 230 Project Software Design for the development of a web-based version of Draw It or Lose It, an Android app, for The Gaming Room. The new game should support multiple platforms, teams, and unique games and team names. The design constraints include developing the application for a web-based distributed environment, ensuring unique game and team names, supporting multiple teams with multiple players assigned to each, and only one instance of the game should exist in memory at any given time. The Domain Model consists of four classes: Entity, Game, Team, and Player. The UML class diagram efficiently meets the software requirements and ensures the unique identifier and name constraints. The document evaluates the characteristics, advantages, and weaknesses of each operating platform, including Linux, Mac, and Windows, and mobile devices to consider the requirements outlined in the document. |

**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 requested Creative Technology Solutions to develop a web-based version of their Draw It or Lose It Android app. The new game should serve multiple platforms and allow for one or more teams with multiple players assigned to each. Unique games and team names must be supported, and only one instance of the game should exist in memory at any given time. The project involves software design and development with software design templates and patterns.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The design of the web-based game application must consider the following constraints:

The application must be developed for a web-based distributed environment.

The application must support multiple teams with multiple players assigned to each.

The application must ensure unique game and team names.

The application must support only one instance of the game at any given time.

The above constraints have several implications for application development. The application must be designed to be scalable and fault-tolerant, and the implementation should follow best practices to ensure security and performance.

## [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 for the game application represents the Domain Model. The diagram consists of four classes: Entity, Game, Team, and Player. The Entity class is a base class introduced to hold common attributes and behaviors.

The Game class represents a game instance and has a list of Team objects associated with it. The Team class represents a team in the game and has a list of Player objects associated with it. The Player class represents a player in the game.

The UML class diagram demonstrates the following object-oriented programming principles:

Inheritance: The Game, Team, and Player classes inherit from the Entity class.

Aggregation: The Game class has a list of Team objects, and the Team class has a list of Player objects.

Encapsulation: The classes have private attributes and public methods to access and modify those attributes.

The UML class diagram meets the software requirements efficiently. The classes' relationships and the Entity class's implementation ensure the unique identifier and name constraints. The implementation is scalable and allows for adding more features to the game application in the future.

**"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 OS is a popular choice for hosting web-based software applications. Mac OS is known for its reliability and security, making it a great choice for hosting a web-based software application. However, it may not be the most cost-effective option as Mac hardware is typically more expensive than other options. | Linux is a popular choice for hosting web-based software applications. It is known for its stability and security, and many web servers run on Linux. It is also an open-source platform, which means that there is a lot of support available for developers. However, it can be more challenging for beginners to use. | Windows is a popular choice for hosting web-based software applications, especially for those using Microsoft tools. Windows has a user-friendly interface, and many developers are already familiar with it. However, it is more susceptible to viruses and malware, making it less secure than Mac OS and Linux. | Mobile devices can be used for hosting web-based software applications, but they are not as powerful as desktop platforms. They may not be suitable for hosting complex applications, and they may not have enough storage or processing power to handle large amounts of data. |
| **Client Side** | Supporting multiple types of clients on Mac requires careful consideration of the software development tools and technologies used. Mac OS supports a wide range of programming languages and development tools, including Xcode, Objective-C, and Swift. However, there may be a higher cost associated with developing software for Mac OS as Apple's development tools and hardware can be more expensive than alternatives. | Supporting multiple types of clients on Linux requires knowledge of the different Linux distributions and the software development tools and technologies used on each one. Linux supports a wide range of programming languages and development tools, including Python, Ruby, and C++. However, it may be more challenging to find developers with expertise in developing software for Linux. | Supporting multiple types of clients on Windows requires knowledge of the different Windows versions and the software development tools and technologies used on each one. Windows supports a wide range of programming languages and development tools, including Visual Studio, C#, and .NET. However, there may be a higher cost associated with developing software for Windows due to licensing fees for some development tools. | Supporting multiple types of clients on mobile devices requires careful consideration of the user experience and the limitations of the devices. Mobile devices support a wide range of programming languages and development tools, including Swift and Java. However, developers need to take into account the smaller screen size and the reduced processing power of mobile devices. |
| **Development Tools** | The relevant programming languages and tools for developing software for deployment on Mac include Xcode, Objective-C, Swift, and Cocoa. Xcode is Apple's integrated development environment (IDE) for macOS and iOS development, and Cocoa is Apple's application development framework. | The relevant programming languages and tools for developing software for deployment on Linux include Python, Ruby, C++, and GCC. GCC is a popular compiler used for C and C++ programming on Linux. | The relevant programming languages and tools for developing software for deployment on Windows include Visual Studio, C#, .NET, and Microsoft SQL Server. Visual Studio is Microsoft's IDE for Windows development, and .NET is a popular development framework. | The relevant programming languages and tools for developing software for deployment on mobile devices include Swift, Java, and Kotlin. Xcode and Android Studio are the most popular IDEs for mobile app development on iOS and Android, respectively. |

## 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 recommended as the operating (server) platform for hosting the web-based software application. Linux is known for its stability, security, and scalability, making it suitable for hosting high-performance and mission-critical applications like Draw It or Lose It. It also offers flexibility in terms of customization and configuration, which can be advantageous for the specific needs of the game.
2. **Operating Systems Architectures**: The recommended operating platform architecture for Draw It or Lose It on Linux would be a client-server architecture. This architecture involves a centralized server that manages the game logic and data, and multiple client devices that connect to the server to interact with the game. The server would handle the processing and storage of game data, while the clients would be responsible for rendering the game graphics and sending user inputs to the server.
3. **Storage Management**: An appropriate storage management system for the recommended operating platform would be a distributed file system, such as Hadoop HDFS or GlusterFS. These distributed file systems provide fault tolerance, scalability, and high performance, which are important considerations for a game with a large amount of data that needs to be stored and processed.
4. **Memory Management**: The recommended operating platform would use various memory management techniques to efficiently allocate and manage memory resources for the Draw It or Lose It software. This would include techniques such as virtual memory, caching, and garbage collection, which can help optimize the performance of the game and ensure smooth gameplay experience for the users.
5. **Distributed Systems and Networks**: To enable communication between various platforms for Draw It or Lose It, a distributed software architecture would be implemented. This could involve using technologies such as RESTful APIs or message queues for communication between the game clients and the server. The network that connects the devices would need to be reliable and secure, with appropriate measures in place for handling connectivity issues and outages, such as redundant connections and failover mechanisms.
6. **Security**: Protecting user information is a critical requirement for the Draw It or Lose It game. The recommended operating platform, Linux, has strong built-in security capabilities, such as user authentication, access controls, and encryption. Additionally, measures such as SSL/TLS for secure communication and encryption of sensitive data, regular security patches and updates, and intrusion detection systems should be implemented to ensure the security of user information on and between various platforms.

In conclusion, based on the evaluation of different operating platforms and their characteristics, Linux is recommended as the operating (server) platform for hosting the Draw It or Lose It game. A client-server architecture with distributed file system storage management, memory management techniques, and distributed software architecture for communication between platforms can be implemented to ensure efficient and secure operation of the game. Strong security measures should also be in place to protect user information.