

Draw it or Lose it.

Version 3.0

# Table of Contents

**CS 230 Project Software Design Template** 1

**Table of Contents 2**

**Document Revision History 2**

**Executive Summary 3**

**Requirements 3**

**Design Constraints 3**

**System Architecture View 3**

**Domain Model 3**

**Evaluation 4**

**Recommendations 7**

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 3.0 | 04/17/24 | Marcus Moore | Added recommendations section going over some of the best technologies for this app. |

# Executive Summary

The software design problem is the development of a web application named “Draw It or Lose It” based on the 80’s TV game Win, Lose or Draw. The objective is to create an interactive platform where teams compete to guess what is being drawn, with images rendered from a vast stock library. To address the client's needs, our solution will accommodate multiple teams, each with several players. Unique identifiers will be assigned to games, teams, and players to ensure data integrity and efficient memory management. By leveraging object-oriented programming principles and web technologies, we aim to deliver a scalable and user-friendly gaming experience. Critical information includes the importance of unique naming for games and teams to prevent conflicts, and the necessity of a web-based distributed environment for seamless gameplay across different devices.

# Requirements

* The ability to accommodate one or more teams, each with multiple players.
* Unique naming for games and teams to avoid conflicts and confusion among users.
* Implementation of a web-based distributed environment to ensure accessibility and seamless gameplay across various devices.
* Utilization of unique identifiers for games, teams, and players to maintain data integrity and manage memory efficiently.
* Integration of drawing functionalities with a large library of stock images to provide diverse and engaging gameplay experiences.

# Design Constraints

Some design constraints include communication protocols such as HTTP/HTTPS must be utilized for client-server interactions to ensure data exchange security, scalability is crucial to accommodate increasing user traffic. Additionally, security measures such as encryption and authentication must be implemented to protect user data and prevent unauthorized access. Furthermore, the uniqueness requirement for game and team names imposes constraints on data validation and storage mechanisms, requiring efficient algorithms to check for name availability and prevent duplicates.

# Domain Model

The classes included are:

* **Entity:** Serves as the base class holding common attributes and behaviors shared by other classes. It includes attributes such as 'id' and 'name'.
* **Game:** Represents a single instance of the game, containing attributes such as 'id', 'name', and a collection of teams.
* **Team:** Represents a team within a game, with attributes including 'id', 'name', and a collection of players.
* **Player:** Represents an individual player associated with a team, identified by attributes such as 'id' and 'name'.

The relationships between these classes include many object-oriented programming principles:

* **Inheritance:** The Game, Team, and Player classes inherit common attributes and behaviors from the Entity class. This promotes code reuse and maintainability by allowing shared functionality to be implemented in the base class.
* **Encapsulation:** Each class encapsulates its attributes and behaviors, allowing for modularity and abstraction. For example, the Game class encapsulates the details of a game instance, including its name and associated teams.
* **Association:** The relationships between Game, Team, and Player classes demonstrate association, indicating how objects are connected and interact with each other within the game application. For instance, a Game has a collection of Team objects, and each Team has a collection of Player objects associated with it.

"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

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 systems are known for their reliability and stability, making them suitable for hosting web-based software applications. MacOS, based on Unix, provides a robust environment with built-in features such as Apache for web server capabilities. However, the availability of serverside software and tools may be slightly limited compared to other platforms. Advantages include excellent security features and seamless integration with other Apple products. Weaknesses may include higher upfront costs for hardware and potential compatibility issues with certain serverside software. | Linux is widely favored for hosting web-based software applications due to its open-source nature, flexibility, and scalability. Linux distributions offer a wide range of server-side software options, including Apache, NGINX, and  various databases like MySQL and PostgreSQL.  Linux's stability, security, and costeffectiveness make it a popular choice for server deployment. However, managing Linux servers may require more technical expertise compared to other platforms. | Windows servers are commonly used for hosting webbased software applications, especially in enterprise environments. Windows Server provides comprehensive features and tools for web server functionality, including Internet Information  Services (IIS) and  Microsoft SQL  Server. Windowsbased servers offer seamless integration with other Microsoft products and services,  simplifying management and administration tasks. However, licensing costs and potential vulnerabilities to security threats are notable weaknesses | Mobile devices, such  as smartphones and tablets, are increasingly used for accessing web-based software applications. Their characteristics include portability, diverse hardware configurations, and varying screen sizes. Developing software for mobile devices requires consideration of multiple platforms (iOS, Android) and adapting the user interface and functionality accordingly. Mobile devices offer advantages in terms of accessibility and user engagement but may present challenges related to performance optimization and compatibility across different devices and operating system versions. |
| **Client Side** | Supporting multiple types of clients on Mac may require considerations regarding software development costs, time, and expertise. Developing software for MacOS-based clients typically involves using Xcode as the primary integrated development environment (IDE)  along with programming languages such as Swift or Objective-C. The cost of Mac hardware and the expertise required in MacOS development  are factors to consider when targeting Mac clients. | Supporting multiple types of clients on Linux requires considerations related to software development costs, time, and expertise. Development tools commonly used for Linux clients include popular IDEs like Visual  Studio Code or  JetBrains IntelliJ IDEA, along with programming languages such as Python, Java, or C++. The opensource nature of Linux makes it costeffective, but expertise in Linux development and compatibility testing across distributions are essential. | Supporting multiple types of clients on Windows involves considerations regarding software development costs, time, and expertise.  Windows development often utilizes Visual Studio as the primary IDE, supporting programming languages like C#, C++, and .NET.  Licensing costs for development tools and compatibility testing across different Windows versions may impact development expenses. | Developing software for mobile devices requires considerations regarding software development costs, time, and expertise. Development tools for mobile platforms include Android Studio for Android development and Xcode for iOS development. Programming languages such as Java, Kotlin, Swift, and Objective-C are commonly used. Supporting multiple types of mobile clients requires expertise in platform-specific development and UI/UX design. |
| **Development**  **Tools** | Relevant programming languages and tools for deploying webbased software applications on Mac include languages such as Swift, Objective-C, and JavaScript. Xcode is the primary IDE used  for MacOS development, offering features such as code editing, debugging, and testing. | Linux offers a wide range of programming languages and tools for building webbased software applications, including Python, Java, PHP, and JavaScript.  Common IDEs used for Linux development include Visual Studio Code and Eclipse. Version control systems like  Git are used as well. | Windows development for web-based software  applications involves programming languages like C#, JavaScript, and  TypeScript. Visual Studio is the primary IDE for Windows development, providing features for code editing, debugging, and deployment. Additional tools such Azure for cloud deployment may also be utilized. | Platform-specific languages and tools. For Android development, languages like Java or Kotlin and IDEs such as Android Studio are commonly used. For iOS development, languages like Swift or Objective-C and IDEs like Xcode are used. Additional tools like Flutter or React Native may be utilized for cross-platform development. Testing frameworks such as Espresso for Android and XCTest for iOS are used for testing mobile applications. |

# 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**: Cloud platforms such as Amazon Web Services, Microsoft Azure, or Google Cloud Platform offer scalability, flexibility, and accessibility across different operating systems and devices.

1. **Operating Systems Architectures**: These platforms employ a distributed architecture, consisting of multiple servers and data centers distributed geographically. These architectures are designed to ensure high availability, fault tolerance, and scalability. Components such as load balancers, virtual machines, containers, and serverless computing services are utilized to distribute workload efficiently and handle increase in the volume of user.

1. **Storage Management**: We recommend a cloud storage service provided by the respective cloud service provider. These include Amazon S3, Google Cloud Storage or Azure Blob Storage

1. **Memory Management**: Cloud platforms offer virtual memory and garbage collection to efficiently utilize system resources.

1. **Distributed Systems and Networks**: Protocols like HTTP/HTTPS or WebSocket. The Draw It or Lose It software can utilize APIs or message queues provided by the cloud platform to facilitate communication between components.

1. **Security**: Cloud-based operating platforms offer many security features, such as encryption, compliance certifications, network security, and identity and access management.

## Client-Side for Cross Platform Web Application

**Requirements:**

Modern and responsive interface.

Compatible across all browsers and devices.

**Considerations:**

Responsive design principles.

Cross-browser testing tools

Updates

Front-end frameworks like React, Angular, etc.

## Server-side Evaluation

**Linux -**

* **Characteristics:** Widely supported, Open Source. Offers flexibility and is secure and reliable.
* **Advantages:** Customizable, variety of free, open-source software available. Scalable and efficient in handling multiple connections. Ideal for web apps.
* **Weaknesses:** Requires expertise to properly secure and manage. Compatibility with proprietary software.

**Mac -**

* **Characteristics:** Known for security and stability, based on Unix, offers a unique user interface.
* **Advantages:** Security and stability, seamless integration with other apple services.
* **Weaknesses:** Generally more expensive, limited customization, server deployment is less common.

**Windows -**

* **Characteristics:** Widely used and supported on a large variety of hardware, offers extensive documentation and support.
* **Advantages:** Easy to manage, easy to set up, broad compatibility with a wide range of services and software.
* **Weaknesses:** Potentially higher vulnerability to security threats compared to Unix and Linux systems, licensing costs for Windows Server editions.

## Development Tools

Common Languages like HTML, CSS, and JavaScript for the front end. And Python, Ruby, PHP for Server-side. An IDE would be needed, such as VS Code, Sublime Text or Eclipse. Git is recommended for version control and a testing framework is recommended as well.

The development team would need a diverse skillset to manage the full stack effectively. The mastery of front-end and back-end technology is crucial and may require multiple teams. This application would also need continuous integration and deployment processes need to be established to manage the code across a variety of environments.

Many open-source tools such as Linux, Python, Git, and Docker can reduce costs. There are also many IDEs and other helpful tools that offer free versions but may require a paid license for enterprises. Note that Windows Server licenses will incur costs over time.