

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

Paloma Rodriguez SNHU CS 230

April 10th 2023

## Table of Contents

[CS 230 Project Software Design Template](#bookmark) 1

[Table of Contents](#bookmark1)**2**

[Document Revision History](#bookmark2)**2**

[Executive Summary](#bookmark3)**3**

[Requirements](#bookmark4)**3**

[Design Constraints](#bookmark5)**3**

[System Architecture View](#bookmark6)**3**

[Domain Model](#bookmark7)**3**

[Evaluation](#bookmark8)**4**

[Recommendations](#bookmark9)**5**

## [Document Revision History](#bookmark10)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/13/2023 | Paloma Rodriguez | Revisions made to cover page, executive summary and design constraints, filled out information about each operating system and evaluations with recommendations. |
| 2.0 | 03/22/2023 | Paloma rodriguez | Revisions made under evaluations, answered questions based on client side, server side and development tools |
| 3.0 | 04/10/2023 | Paloma Rodriguez | Revisions made to cover page and recommendations section |

**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](#bookmark11)

The Gaming Room has asked for the development of a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. The game is a team-based drawing game where players need to guess what is being drawn based on a library of stock drawings. The game consists of four rounds of play lasting one minute each. The client has requested that the game application meet specific software requirements, including the ability to have one or more teams involved, unique game and team names, and only one instance of the game can exist in memory at any given time. To meet these requirements, we propose developing a web-based application that will utilize unique identifiers to manage instances of games, teams, and players.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#bookmark12)

The client's specific requirements present some design constraints that must be addressed during the development process.

* Team and Player Management: The game application must be designed to support the creation and management of multiple teams, each with multiple players assigned to it. This requires the development of a team and player management system that allows players to join and leave teams and teams to be created and disbanded.
* Unique Game and Team Names: The game application must ensure that game and team names are unique to avoid confusion and allow users to check whether a name is already in use when choosing a team name. This requires implementing a naming system that verifies name availability before allowing users to use a particular name.
* Instance Management: The client has requested that only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player. The implications of this design constraint are that the game application must track and manage each instance of the game, team, and player, ensuring that only one instance exists at any given time.

Addressing these design constraints requires careful planning and implementation of a structure that can handle multiple users and data, as well as implementing secure data storage and transmission protocols to ensure user data privacy and prevent unauthorized access to the game. Additionally, the game application must be designed with the user experience in mind, ensuring that users can easily create and join teams, choose unique game and team names, and navigate the game's different rounds and gameplay mechanics.

## [System Architecture View](#bookmark13)

The Draw It or Lose It game application is designed to provide a fun and engaging gaming experience for multiple players across different platforms.

The system architecture will consist of three main components: the client-side interface, the application server, and the database server.

* Client-side Interface: The client-side interface is responsible for displaying the game's user interface to players and communicating user input and game events to the application server. It is designed to be responsive and accessible across different devices. The interface allows players to create and join teams, choose unique game and team names, and navigate the game's different rounds and gameplay mechanics.
* Application Server: The application server will be the core of the game application and is responsible for processing player requests, managing the game state, and sending updates to connected clients. The server is built to be scalable and reliable. It communicates with the database server to retrieve and store game data, including player and team information, game state, and game events.
* Database Server: The database server is responsible for storing and managing game data, including player and team information, game state, and game events. The database server is designed to be highly available and scalable. The server communicates with the application server to retrieve and store data as needed.

The communication topology is based on a client-server structure. Clients connect to the application and send and receive data in real-time. The storage topology is stored in tables, with each table representing a specific entity such as players, teams, games, and game events. The database is designed to be stored in consistent and efficient manner. Overall, the system structure will be designed to be scalable, reliable, and responsive, providing a fun and engaging gaming experience.

## [Domain Model](#bookmark14)

Based on the UML diagram below, the diagram consists of multiple classes, including Game, Team, GameService and Player. These classes would relate to each other in the following ways:

* Game: The Game class will contain information about the game being played and the team name.
* Team: The Team class would contain information about each team, such as the team name and the players on the team. It also would have info about adding players.
* Player: The Player class would contain information about each player, such as their name.
* GameService: The GameService class would contain the games listed, the next game available, the player and teams information including more information about the game and player in depth.

The UML class diagram can demonstrate several object-oriented programming principles, including encapsulation, inheritance, and polymorphism. Encapsulation would be demonstrated by the fact that each class would have its own set of properties and methods that are hidden from other classes, ensuring that each class can maintain its own state and behavior. Inheritance would be demonstrated by inheriting properties and methods from the other classes. Polymorphism would be demonstrated by the allowing the game to be played with a variety of different themes and categories if needed. The UML class diagram provides a clear and efficient way to represent the relationships between the different classes in the game application, helping to ensure that the software requirements are fulfilled in an effective and maintainable way.

**"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](#bookmark15)

| **Dev. Reqs:** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Characteristics:   * Has a stable and reliable platform for web application development * Very user-friendly interface, which makes it easy to use and navigate.   Advantages:   * Comes with built-in tools and applications that are useful for web dev, such as Xcode, Terminal, and Git. * Hass plenty of resources and support available for people who have issues while working on their applications.   Weaknesses:   * Has a closed system, so its not customizable as other systems. * Hardware is expensive which may affect developers or business budget. | Characteristics:   * Linux is known for its security, stability, and reliability. * It is highly scalable and can be used for large-scale web applications   Advantages:   * Linux is known for its stability and reliability, making it an ideal platform for hosting web-based applications. * It is ideal for hosting sensitive data and applications.   Weaknesses:   * Learning the operating systems can be hard, which can be a challenge. * Linux may require more technical knowledge to set up and maintain. | Characteristics:   * Windows is a commercial operating system developed by Microsoft. * It has a graphical user interface (GUI) and supports a wide range of applications. * Windows is widely used in the enterprise environment.   Advantages:   * Has great support for Microsoft-based technologies. * It has a large user base, which means there is plenty of support available online.   Weaknesses:   * Can be more vulnerable to security threats. * Windows updates can cause compatibility issues with third-party software. | Characteristics:   * Small screen size compared to desktops and laptops. * Limited storage capacity. * Touchscreen interface.   Advantages:   * Mobile devices are portable and can be used on-the-go. * They offer wireless connectivity, allowing users to access web-based software applications from anywhere.   Weaknesses:   * Limited screen size can make it hard to use some web-based software applications. * Limited storage capacity can make it challenging to store large amounts of data. * Touchscreen interface can be less precise than a mouse and keyboard. |
| **Client Side** | Cost:  Developing for Mac it can be more expensive than other platforms due to the specialized hardware and software. Additionally, if the application needs to be tested on multiple versions of the Mac operating system, there may be additional costs associated with setting up and maintaining a testing environment.  Time:  Developing for multiple types of clients can impact the time it takes to complete a project. Developing may require additional time if specialized hardware and software is used, as well as the need for any platform-specific differences in the development process. If the application needs to be tested on multiple versions of the operating system, this can add additional time to the development process.  Expertise:  Developers need to know of Mac-specific technologies to be able to develop for the platform effectively. This may need hiring developers with expertise or investing in training for existing team members. | Cost:  Linux is mostly free and open-source, so theres no cost associated with the operating system itself but there can be additional costs associated with software development tools and frameworks.  Time:  Developing software that can run on multiple types of Linux clients may take longer than developing software for a single client type. This is because there may be differences in the underlying hardware and software that need to be accounted for.  Expertise:  Developing software for Linux requires a certain level of expertise in the operating system and associated tools and frameworks. In order to support multiple types of clients, developers will need to have an understanding of the various Linux distributions and the nuances of each. | Cost:  It is essential to ensure that the application runs smoothly on all versions of Windows, including the latest ones. This may require purchasing additional licenses for operating system versions or tools to support different versions.  Time:  Supporting multiple types of clients on Windows may increase the development time due to the need for testing and bug fixing on different platforms. Additionally, for more variation in hardware specifications, this can require more time to ensure compatibility.  Expertise:  Developers must be familiar with the latest development tools to ensure that the application is optimized for the different versions of Windows. | Cost:  Cost can vary because there many types of variations of cellular devices that testing on application for friendly user experience can be costly to consider and we need to keep in mind the different operating systems between devices so the application can run seamlessly on all platforms.  Time:  Mobile devices have varying capabilities such as processing power, memory, and storage, which must be considered. This can include optimizing application performance and ensuring that the application does not consume excessive resources on lower-end devices.  Expertise:  Supporting multiple types of mobile devices may require specialized expertise in different programming languages, tools, and frameworks. |
| **Development Tools** | Git, Xcode, Swift, Objective C, Homebrew, Interface Builder | Python, Java, Git, Docker, Redit, MySQL, Eclipse, PyCharm, Visual Studios, | Terminal, Visual Studios, C#, C++, Java, Python, .NET, | Java, Kotlin, Android Studios, Swift, Objective-C, Xcode, C#, Visual Studios, Flutter, Xamarin |

Server Side:

Evaluate various platforms for their characteristics, advantages and weaknesses for hosting a web-based software application. Consider the following in your evaluation and articulate your findings :

Does each of the operating platforms offer a server-based deployment method where the website will be hosted?

Linux, Mac and Windows all offer server-based deployment methods, but Linux is specifically known for it stability and scalability. Making it one of the most popular choices for hosting web applications. I understand the client prefers Mac because of it’s user friendly and well developed deployment and Mac is also a good choice if not wanting to use Linux, but it may not be the best option for hosting high-traffic websites because it is limited in hardware scalability which could be an issue since one of the main goal is to increase the audience across multiple platforms. Windows is also able to host web applications but it does require additional resources to maintain and may not be as cost effective as Linux is.

What are the potential licensing costs to the client, The Gaming Room, for the server operating system?

Quality of services can be a contender when thinking of budget costs but when thinking of licensing, Linux is an open source operating system which in summary means it is free to use and distribute. Mac and Windows both require licenses which can be costly depending not the number of servers need to host a web application. It can be costly to our application especially since the expansion is one of our main goals and on top of that wanting to maintain it on Mac can add on to company budget spending which may not be ideal. We can have both, but it is something to consider on costs.

Client Side:

The client wishes to move beyond their current Android-only application to supporting players on iOS and Android mobile platforms, as well as traditional desktop-based operating systems. The application must be delivered as a modern, responsive HTML interface running inside the web browser for desktop clients (Linux, Mac, Windows), as well as on mobile platforms. Each will be capable of communicating with the back-end web application running on the server :

What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices?

Supporting clients across multiple web browser platforms and mobile devices and ensuring the application is compatible will need to consider the above as well including cost, time and expertise from experienced professionals. Developing an application that is so widely available can become complex that requires specialized advice which can of course become costly as well. To ensure compatibility, developers may need to use tools such as Apache Cordova or Xamarin which are known to be used to develop across platforms and these tools will allow developers to write code once and deploy it on these platforms but they do require additional developers who have the knowledge to do so and can take longer time to develop so if time is of the essence this should be taken into consideration.

Development Tools:

Identify the relevant programming languages and tools (IDEs and tools) that are used to build this type of software for deploying on each operating platform.

What impact do these technical requirements have on a development team? Consider whether multiple development teams are needed. Are there licensing costs related to the development tools?

Identifying the necessary programming languages and tools will be essential before building this web based software application and having it being deployed on each operating platform. The commonly used languages will include HTML, CSS and Javascript but for the server side of things that can be Python and Ruby. The choice of tools can make a major impact on the teams and who to pick for this project to get done effectively and efficiently. Using multiple teams may be necessary if we plan on needing advise on experts on different platforms. Additionally, licensing costs may arise as well when working with Windows or Mac which will increase the overall cost as well. Overall, when choosing and evaluating which operating platform will be critical and developers chosen, the licensing costs and tools will be needed to solidify before moving forward.

## 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**:

For the game to be accessed and played on multiple computing environments, including Mac, Linux, Windows, and mobile devices, web-based platforms will be best because they are scalable and easily deployable, which will be beneficial for The Gaming Room as they plan to expand the game to a larger user base. As a recommendation, Linux would be best to expand Draw It or Lose it to other computing environments because Linux is a very popular open-source operating systems that is widely used for servers due to its stability, scalability and also security. It is also know to have a large community of developers so finding a team for development will be able to provide ease for the process.

1. **Operating Systems Architectures**:

I recommend using a micro services architecture. This architecture allows for the development of smaller, independent components that can be easily managed and deployed. It also enables easier integration with other systems and components, making it easier to add new features and functionality to the game in the future. Each micro service is designed to perform a specific task or function. A micro service architecture is service oriented, and designed to be decentralized. This means that there’s no central point of control or coordination and each service is responsible for its own data and functionality. Since each micro service is developed and deployed independently, this allows for maintenance to be more easy and allows scalability also. Another great thing is that it is agile, it promotes developers to respond quickly to changing requirements as they come along and even customer needs. Overall, a micro service architecture offers many benefits such as scalability and agility but it is also good to keep in mind the new challenges one may face as well, like coordination and testing requirements!

1. **Storage Management**:

Using a cloud-based storage solution such as Amazon S3 or Microsoft Azure will be a great solution to pair with Linux operating systems. First off, with a cloud based storage management system, you can easily scale your storage infrastructure up or down as company needs change, this will be great to implement once the Gaming Room receives a spike of users when they launch to more platforms and needs to rapidly expand their capacity. Cloud storage can also be more cost effective than a traditional on-site server service and the company would only need to pay for what it’s using and not having to invest in the physical equipment. Compared to an on-site server, you can access the data you need from anywhere and all you need to be is authenticated and authorized to get what you need. It should be mentioned as well that Linux based tools can easily automate cloud based storage systems and can help even streamline tasks and reduce errors.

1. **Memory Management**:

# Linux has multiple ways of memory management techniques to make most of its available system resources and provide a stable and efficient operating environment. Some key techniques used are virtual memory which is when the operating system is able to allocate memory to processes on an as-needed basis, rather than allocating all available memory up front which leads to another feature of memory allocation, Linux uses various algorithms to allocate memory to processes, such as first-fit, best-fit, and worst-fit. These algorithms help to optimize memory utilization and reduce fragmentation. Linux also uses memory caching to speed up access to frequently used data. This involves keeping a copy of recently accessed data in memory, so that it can be quickly retrieved if needed again. Caching helps to reduce the number operations required, which can significantly improve system performance. Overall, Linux's memory management techniques help to provide a stable and efficient operating environment, even in systems with limited physical memory.

1. **Distributed Systems and Networks**:

# Using a RESTful API would provide a standardized way for different parts of the system to communicate with each other over the network. One benefit of using a distributed system with a RESTful API is that it can be easily meshed with other systems and components. For example, if Draw It or Lose It needed to program a way for users to purchase tokens or skins with a payment gateway, it could be accomplished using the same API. Connectivity is one of the most important things to consider because communication between different components relies on a network connection and in the event of a network outage, communication between components may be disrupted, causing data loss and many other issues. To lessen the risk of network outages, it's important to ensure that the network is reliable. Prevention measures like firewalls to prevent unauthorized access and encryption to protect data in transit can continue even in the event of a network outage.

1. **Security**:

Implementing a security strategy that includes encryption, access control, and data backup will be best. Platforms include security features like built-in firewalls and secure communication protocols but its best to add add-ons. User information should be protected using encryption techniques, and access control should be implemented to ensure that only authorized users can access sensitive data. To elaborate on some security methods that can be taken is encryption, which is the process of converting plain text into a secret code that can only be deciphered with the correct key. Linux has multiple encryption tools such as OpenSSL, and Cryptsetup, which can be used to encrypt sensitive data on various platforms. Access controls can also be used to restrict access to data and resources to only authorized users. Linux has built-in access control tools such as file permissions, user and group permissions, and role-based access control that can be used to manage user access. Security updates will be necessary to up-keeping the system and software with the latest security updates and patches to prevent security vulnerabilities. Linux has a solid and secure platform that can be used to protect user information on and between various platforms.