

Jantzen Springer

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 3/24/2024 | Jantzen Springer | * The game will have the ability to have one or more teams involved. * Each team within the game 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. |
| 2.0 | 4/04/2024 | Jantzen Springer | * Identify each of the operating platforms and what they offer a server-based deployment method. * What are the potential licensing costs to the client, The Gaming Room, for the server operating system. * Determine the software development considerations that are necessary for supporting multiple types of clients. * What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices? * What impact do these technical requirements have on a development team? Consider whether multiple development teams may be needed |
| 3.0 | 4/17/2024 | Jantzen Springer | * **Operating Platform:** Recommend an appropriate operating (server) platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments. * **Operating Systems Architectures:** Describe the details of the chosen operating platform architectures. * **Storage Management:** Identify an appropriate storage management system to be used with the recommended operating platform. * **Memory Management:** Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software. * **Distributed Systems and Networks:** Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on). * **Security:** Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform. |

**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)

Our client, Draw It or Lose It, is currently developing software designs for their game. This presents an opportunity to enhance operational efficiency, streamline processes, and achieve strategic objectives. Our solutions will provide Draw It or Lose It with a uniquely named, team-based gameplay experience that remains scalable and flexible. With efficient memory management and unique identifiers for each instance of the game, we will also include robust security measures, protecting user data and ensuring system integrity. This will all be done while ensuring resource management and regulatory compliance to meet industry standards.

User engagement is paramount for Draw It or Lose It. The software design will allow users to create their own unique teams and players. This provides the user with a personalized experience that greatly increases the player experience. The specialized experience can provide the user with a unique and competitive engagement during gameplay, increasing social interaction during game events and challenges.

With system design in mind, Draw It or Lose It will remain scalable. This will ensure that as Draw It or Lose It increases their competitiveness within the market and attracts an increasing number of users, the backend will meet industry trends and design solutions. Maintaining flexibility will ensure that improvements and interactions are ongoing and user feedback is prioritized in software design.

To meet the requirements of the Draw It or Lose It, the implementation of software design solutions will incorporate team-based gameplay, unique naming system, and memory management. These requirements will be met by optimizing resource utilization. We will implement unique identifiers for each game instance, team, and player. These keys will be readily available on memory, allowing the system to quickly access and retrieve information about the game instance, teams, and players. These faster response times will improve overall system performance, versus having to access databases or storage operations. This will reduce over latency and drastically improve real-time performance.

## Requirements

These listed requirements meet the clients overall end state for “Draw it or lose it”. There first needs to be team-based gameplay implemented in the game. The game is centered around teams, which means that the application should allow the user to create their own unique teams. With there being multiple players and teams, an implementation of a unique naming system will allow for unique instances for when teams are created, or players need unique identification.

Since there will be multiple players or teams that are playing at a single time, responsiveness needs to be managed to ensure that the user engagement is adequate for gameplay. To ensure that the responsiveness and overall gameplay is acceptable, feedback integration allows for the users to identify any bugs within the programming, which could be the user experience better.

Memory and storage management will assist in the responsiveness of the programming. Efficient memory allocation will minimize overall and improve overall system performance, equating to a better user experience.

Lastly, security measures need to be implemented to protect user data and privacy. Some examples of this could be encryption traffic that is being passed to the user and the server, as well as ensuring proper authentication.

## [Design Constraints](#_2et92p0)

The biggest constraint regarding game application in a web-based distributed environment is network access. Users must be able to access a stable network that supports bandwidth which meets the requirements of the game. Along with this, standard network protocols of latency, security, and cross-platform compatibility remain an issue. As previously mentioned, scalability and flexibility is priority for Draw It or Lose It and will be implemented throughout all aspects of the game.

To access resources for Draw It or Lose It, the user must have web-based, network access, which supports the bandwidth and data transfers of the game. Optimizing the game’s architecture and meeting industry standards will mitigate network latency and have less impact on the responsiveness of the game. An option that can also mitigate network issues is data prioritization. Prioritizing essential game data can minimize bandwidth usage.

As users increase, the game must remain scalable to meet user requirements. Maintaining and implementing the industries based reliable and fault tolerance systems, this will ensure that the user experience is maximized, even with growing player demand.

Security will remain paramount for Draw It and Lose It systems. Security threats such as unauthorized access, data breaches, denial-of-service attacks will consistently pose a threat in a web-based environment. Robust security measures such as encryption, authentication, and access control will ensure player data remains secure and the integrity of the game is safe.

## [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)

This UML class diagram discusses the structure and relationship between various classes within the software design for Draw It or Lose It. Following Object-oriented programming (OOP) design, this UML represents multiple principles, such as singleton pattern, inheritance, encapsulation, and association. These principles are essential in managing the game instances, players, and teams within a web-based application.

The program begins with the programDriver class. This is the entry point for the program and the main method is location to initiate the application. The programDriver class calls on a method provided by the Singleton class. This is primarily to ensure that only one instance of the Singleton class is created and returned.

The Entity class is the parent class of Game, Team, and Player class. Inheritance is a fundamental OOP concept where a child class, in our case Game, Team and Player class, can inherit attributes and behaviors from another class, Entity class. Entity class contains multiple attributes and behaviors which are reuse and extend functionality to subclasses without duplicating code. Implementing this centralized functionality can increase efficiency within the game processes and minimize the amount of code written by the development team.

gameService class is part of the singleton pattern. It ensures that only one instance of GameService exists throughout the application. These include functionalities for adding new games, retrieving existing ones, and accurately counting the total number of games stored. Moreover, the GameService class provides specialized functionality to generate and assign unique identification numbers for both players and teams, ensuring seamless tracking and organization of game-related entities.

The Game class represents a game instance. It contains a list of teams, as well as providing methods to add teams. Within this class, we can identify which teams are participating in the game, as well as add new teams to the game.

The Team class represents the teams within the game session. Like the Game class, the Team class maintains a list of players that are assigned to the team. This includes the methods for adding new players to the teams.

The Player class is the representation of each individual player. This class is directly linked to the team class and contains the attributes and behaviors that are associated with each player. The Player class is the foundational block to Team class, which in turn will build the Game class.

We briefly mentioned the OOP principle of inheritance. This UML diagram represents more than just one type of OOP foundation. As stated in the second class, this program represents Singleton pattern. This is managed by ensuring only one instance of the class exists in the system. Encapsulation is represented by developing multiple classes, which can restrict internal details and functionalities. The incorporation of encapsulation helps protect data and prevent unnecessary access and manipulation to classes. Association is represented throughout the software design, but primarily throughout gameService class to player class. These classes are developed so that objects of one class can connect with objects of another class, allowing for flexibility.

This UML design implements many fundamental principles of OOP, including Singleton pattern, inheritance, encapsulation, and association. With careful design, these principles enable efficient management of instances and objects within the web-based application of Draw It or Lose It. The integration of these OOP features highlights the importance of the design and increases the operation of the game application.

**"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** | Stability and reliability is a strong suite for Mac, but it may have limited software and application compatibility. | Highly favored for their web-based software applications due to wide-range, open-source software tools, which make it cost-effective to implement security and scalability | Wide-range of web-based software applications but may not be as cost effective as Linux systems. | Limited processing power and network connectivity. Dedicated servers will provide a much more reliable environment. |
| **Client Side** | Must have expertise in MacOS and apple hardware. Proprietary systems may increase cost-effectiveness, which could increase time constraints. | Robust distros list can increase necessary support for all the different types of clients, which may in turn increase cost-effectiveness. | Compatibility with multiple versions of Windows and hardware configurations, which can create considerations for costs, time, and expertise. | Expertise support for multiple clients. The mobile development field is very broad and includes multiple OS which can increase costs, time, and the necessary expertise. |
| **Development Tools** | MacOS and iOS Xcode IDE for development with the primary languages of Swift, Objective-C, and C/C++. | Endless support and applications make for robust distros in Linux. Visual Studio Code and languages like Python, Java, C/C++, and PHP are found within Linux environments. | Like Linux, Visual Studio Code is a very popular IDE which supports a plethora of languages such as C+, VB.NET, and JavaScript. | The (2) largest mobile OS are Android and iOS. Relevant languages are Java or Kotlin for Android devices within Android Studio. Xcode remains the IDE for iOS with Swift and Objective-C as the primary languages. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: Linux stands out as the optimal choice for the operating platform due to its cost-effectiveness and reliability. Being open-source, Linux eliminates the need for licensing fees, significantly reducing the financial burden on the client. This cost-effective approach ensures that the client can allocate resources efficiently, directing them towards other critical aspects of the project.

Moreover, Linux boasts a robust and active community that continuously updates and improves the system, enhancing its reliability and performance. With readily available support and updates, the client can rely on Linux to provide a stable foundation for deploying web-based applications, ensuring smooth operations, and minimizing downtime.

**Operating Systems Architectures**: Included in the architecture needs to be a language that supports client -side development of modern web applications and the ability to revolutionize front-end development. Due to this, I would choose JavaScript or React as the foundation for my client, Draw it Or Lose it. JavaScript is the primary language for web development and provides versatility and compatibility across different browsers and platforms. It allows for unique user interfaces facilitating seamless interactions and user experiences. There is also an extensive list of libraries and frameworks built around JavaScript, such as Node.js and Express.js, which can assist developers in creating complex applications.

A specific JavaScript library worth noting is React. React is a front-end development tool that allows developers to break down complex user interfaces into manageable, self-contained components. The JSX syntax allows developers to write HTML like code directly in JavaScript, which enhances readability and maintainability. The lifecycle of React products allow developers to work on various stages within the lifecycle from initialization, updating, and cleanup.

**Storage Management**: Storage options are widely accessible. Cloud-based storage would immediately allow for scalability, especially to a newly deployed application. Cloud-based storage also provides redundancy, whether the primary method of storage is onsite, the client could utilize a third-party cloud-based agency to support redundancy, which is paramount in the digital world. Per the 3-2-1 redundancy strategy, having an offsite backup would enhance the data resilience of the company and minimize the risk of permanent data loss in the event of a disaster.

**Memory Management**: Efficient memory allocation strategies would optimize resource usage by allocating memory dynamically. This would minimize overall water. Automated garbage collection processes reclaim memory occupied by unused objects or variables, preventing memory leaks, and improving performance. Memory profiling tools help identify memory-related issues such as leaks, excessive memory consumption, or inefficient memory usage patterns, facilitating optimization efforts. Virtual memory systems expand available memory by using disk space as an extension of physical memory, allowing for larger address spaces and improved multitasking capabilities. Memory protection mechanisms prevent unauthorized access to memory regions, enhancing system security and stability.

**Distributed Systems and Networks**: Implementing technologies such as web-based APIs or RESTful services would promote real-time communication over the network. This would eliminate the potential for platform conflict and manage network connectivity. Managing systems like web-based APIs are significant in that they run protocols that are common to nearly all systems on a network. RESTful services are distributed over HTTP-based communication supporting many of the users that the client would intend to have using their application.

**Security**: Web-based APIs would support a lot of necessary security features that would ensure that data is protected communication systems as well are secure. These features like HTTPS would assist in encrypting communications. Along with this, developers will utilize best practices while developing to ensure proper classification for user data. Implementation of secure authentication mechanisms, username and password, certificate-based authentication, and multi-factor authentication (MFA) would ensure security compliance. Being a web-based system security measures to ensure that SQL injection and cross-site scripting would be essential to maintain privacy and secure storage mechanisms for users.