

<Name-of-Software-Application>

# **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 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 4/23/23 | Maneekarn Thangkum | Initial changes. |

**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 wants to expand their game Draw It or Lose It to other computing environments, and they want the game to communicate between various platforms. They also need a secure way to protect user information. We recommend using a cloud-based server platform that allows for scalability and can run multiple operating systems. The chosen storage management system should be highly available and fault-tolerant, and the memory management should be optimized for high performance. To ensure security, we recommend using a platform that has strong user protection and security capabilities.

## Requirements

The Gaming Room is looking to expand their game Draw It or Lose It to other computing environments. The client requires the software to be secure, with user information protected across various platforms. The software must also be designed to handle a large number of users and have the ability to communicate between various platforms.

## [Design Constraints](#_2et92p0)

Designing the game application in a web-based distributed environment comes with several design constraints. One of the major constraints is the need for scalability to handle a large number of users. This requires the use of distributed systems and network technologies, which can introduce complexities in terms of connectivity, outages, and dependencies between components.

Another constraint is security, as the game application will be accessed from various platforms, which increases the potential attack surface. The client requires the software to be secure, with user information protected across various platforms. This requires the use of secure authentication and authorization mechanisms, as well as data encryption to protect user information in transit and at rest.

The choice of operating platform and storage management system must also be carefully considered, as they can impact the performance, scalability, and reliability of the game application. The chosen operating platform and storage management system must be capable of handling large amounts of data and high traffic, while providing the necessary level of security and reliability. Additionally, the chosen platform must be able to support the desired features and functionality of the game application, including real-time communication and collaboration.

Overall, the design constraints for developing the game application in a web-based distributed environment require careful consideration of scalability, security, performance, and reliability, as well as the selection of appropriate technologies to support these requirements.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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** | Advantages: Mac has a user-friendly interface and a strong focus on design, making it an attractive option for creative professionals. It also offers tight integration with other Apple products, such as the iPhone and iPad, which may be beneficial for the client's target audience. Additionally, Mac has a good track record for security, which is critical for any web-based application.  Weaknesses: Mac is a closed ecosystem, which means that it may not be as flexible as other operating systems. It also has a smaller market share compared to Windows, which could potentially limit the client's reach. | Advantages: Linux is highly customizable, which makes it a great choice for developers who want to fine-tune their environment to their specific needs. It is also free and open-source, which means that there are no licensing fees or vendor lock-in. Linux also has a strong focus on security and is widely used for web hosting and server administration.  Weaknesses: Linux can be more challenging to use than other operating systems, especially for non-technical users. It also has a smaller market share compared to Windows, which could potentially limit the client's reach. | Advantages: Windows has a user-friendly interface and is widely used, which makes it a familiar choice for many users. It also has a wide range of software and hardware compatibility, which makes it easier to find solutions and troubleshoot issues. Additionally, Windows offers tight integration with other Microsoft products, such as Office and Azure, which could be beneficial for the client.  Weaknesses: Windows is more vulnerable to security threats than other operating systems, which could be a concern for a web-based application. It also has a reputation for being less stable than other operating systems, which could impact the reliability of the client's application. | Advantages: Mobile devices are highly portable and offer users the convenience of accessing applications on-the-go. They also have a range of sensors and features, such as GPS and cameras, which can enhance the functionality of the client's application.  Weaknesses: Mobile devices have limited processing power and screen real estate, which can impact the performance and user experience of the client's application. They also have a range of different hardware and software configurations, which can make it challenging to develop and test applications across multiple devices and platforms. Additionally, mobile devices are more vulnerable to security threats, which could be a concern for a web-based application that handles sensitive user data. |
| **Client Side** | For Mac, software development considerations include the cost of purchasing and maintaining Mac-specific hardware and software, as well as the need for developers with expertise in Mac-specific programming languages and development environments. The time required to develop and test the software on Macs may also be longer due to the smaller user base and potential hardware and software incompatibilities. | For Linux, software development considerations include the need for developers with expertise in Linux-specific programming languages and development environments, as well as the potential need to create multiple versions of the software for different Linux distributions. However, the open-source nature of Linux may allow for more cost-effective development and testing, as well as the potential for a larger user base. | For Windows, software development considerations include the need for developers with expertise in Windows-specific programming languages and development environments, as well as the need to ensure compatibility with multiple versions of Windows. The larger user base of Windows may make it a more lucrative platform for development, but the potential for hardware and software incompatibilities may increase development time and cost. | For mobile devices, software development considerations include the need to design for multiple screen sizes and resolutions, as well as the need to ensure compatibility with multiple operating systems and versions. The cost of developing for mobile devices may be higher due to the need for specialized development tools and expertise in mobile app development, but the potential for a large user base may make it a worthwhile investment. |
| **Development Tools** | Programming languages: Swift, Objective-C, JavaScript, HTML, CSS  IDEs: Xcode, Visual Studio Code, Atom  Other tools: GitHub, Homebrew | Programming languages: C, C++, Java, Python, JavaScript, HTML, CSS  IDEs: Eclipse, NetBeans, Visual Studio Code  Other tools: Git, Make, GCC, Bash | Programming languages: C#, C++, Java, Python, JavaScript, HTML, CSS  IDEs: Visual Studio, Visual Studio Code  Other tools: Git, NuGet, MSBuild | Programming languages: Swift, Objective-C (iOS), Java, Kotlin (Android)  IDEs: Xcode (iOS), Android Studio (Android)  Other tools: Git, Gradle, CocoaPods (iOS), Android SDK (Android) |

## 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**: We recommend using a cloud-based operating platform such as Amazon Web Services (AWS) or Microsoft Azure to expand Draw It or Lose It to other computing environments. Cloud-based platforms offer scalability, flexibility, and cost-effectiveness, allowing the game to run on multiple operating systems and devices. The cloud-based operating platform can be customized to support the required operating systems and software, making it easier to expand the game to different computing environments. Additionally, cloud-based platforms offer built-in security features, making it easier to secure the game data and user information.
2. **Operating Systems Architectures**: The chosen operating platform, such as AWS or Microsoft Azure, typically utilizes a distributed architecture model. This architecture distributes the game's resources across multiple servers, making it easier to handle increased traffic, scale the game up or down, and maintain high availability. The distributed architecture also offers redundancy, meaning that if one server fails, another server can take over without any downtime or data loss.

The cloud-based operating platform also offers a variety of different operating systems, such as Windows, Linux, and Unix. This allows the game to be run on different operating systems and devices, which is critical for expanding the game to various computing environments. Furthermore, the platform typically provides virtualization capabilities, allowing for multiple virtual machines to be created and managed on a single physical server. This virtualization feature can help reduce hardware costs and simplify system management.

1. **Storage Management**: We recommend using a cloud-based storage management system such as Amazon S3 or Microsoft Azure Blob Storage, which is integrated with the recommended cloud-based operating platform. These storage systems offer scalability, durability, and high availability, allowing the game to store and access large amounts of data, such as game assets and user information, from anywhere in the world.

These cloud-based storage systems also provide a simple and secure way to store and manage game data, with built-in security features such as encryption, access controls, and versioning. Additionally, they can be easily integrated with other cloud-based services such as databases and analytics tools to provide a comprehensive storage solution for the game.

1. **Memory Management**: The recommended cloud-based operating platform utilizes memory management techniques such as virtual memory and garbage collection to efficiently manage the memory used by the Draw It or Lose It software.

Virtual memory allows the operating system to use disk space as an extension of physical memory. This means that the game can use more memory than is physically available, improving performance and allowing the game to run on systems with limited memory. The operating system automatically manages the mapping of virtual memory to physical memory, making it easier for the game to run on different computing environments.

Garbage collection is a memory management technique that automatically frees up memory that is no longer being used by the game. This ensures that the game does not use more memory than necessary and reduces the risk of memory leaks or crashes. The garbage collection process runs automatically in the background, making it transparent to the game and minimizing the impact on performance.

Overall, these memory management techniques ensure that the Draw It or Lose It software runs smoothly and efficiently on the recommended operating platform.

1. **Distributed Systems and Networks**: To allow Draw It or Lose It to communicate between various platforms, we recommend implementing a distributed system architecture using a message-oriented middleware (MOM) such as Apache Kafka or RabbitMQ.

With a distributed system architecture, the game can be split into multiple smaller components or microservices that can be deployed and scaled independently. Each microservice can be responsible for a specific task or functionality, such as game logic or user authentication. These microservices can communicate with each other using the MOM, which acts as a messaging broker, sending messages between different components.

The MOM allows for asynchronous communication, which means that messages can be sent and received without the sender having to wait for a response. This ensures that the game can continue running even if one or more components are temporarily unavailable, such as during network outages. The MOM also provides fault tolerance and can be configured to automatically retry failed messages or redirect messages to alternative components in the event of an outage.

In terms of the network that connects the devices, we recommend using a cloud-based infrastructure such as Amazon Web Services (AWS) or Microsoft Azure. These cloud platforms provide reliable and scalable network infrastructure, including load balancing, auto-scaling, and other features that can help ensure high availability and performance for the game.

Overall, by implementing a distributed system architecture with a message-oriented middleware and cloud-based infrastructure, Draw It or Lose It can communicate between various platforms efficiently and with high availability and fault tolerance.

1. **Security**: To protect user information on and between various platforms, we recommend implementing a multi-layered security approach that includes both server-side and client-side security measures.

At the server-side, the recommended operating platform should provide secure user authentication and authorization mechanisms to ensure that only authorized users can access the game and their personal information. The platform should also provide secure storage and transmission of user data, such as passwords, personal information, and game data. This can be achieved by using secure data encryption techniques, such as Transport Layer Security (TLS) and Advanced Encryption Standard (AES).

In addition, we recommend implementing a security policy that includes regular security audits, vulnerability assessments, and penetration testing to identify and mitigate potential security risks.

At the client-side, the game should implement security measures such as secure password policies and user verification mechanisms. The game should also ensure that user data is not stored locally on the device in an unsecured manner.

The recommended operating platform should also provide security capabilities such as secure boot, secure firmware updates, and hardware-based security features such as Trusted Platform Modules (TPMs) to enhance the overall security of the game and user data.

Overall, by implementing a multi-layered security approach that includes server-side and client-side security measures, regular security audits, and utilizing the security capabilities of the recommended operating platform, The Gaming Room can ensure the security and protection of user information on and between various platforms.