

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 5/21/2023 6/4/2023   6/18/2023 | NOYAL REJI JACOB   NOYAL REJI JACOB | Iterators for game methods have been completed, and team and player ID methods have been implemented in GameService.java.  Recommendations was filled based on updated pros and cons for each OS |

**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 project aims to develop a cross-platform web-based game inspired by the existing mobile game "Draw It or Lose It," which is currently available only on Android. The objective of the game is to provide an engaging experience for multiple teams, each composed of several players, competing across four rounds, with each round lasting one minute. During each round, a picture is randomly selected from a library of images, and one team attempts to guess the picture within the given time limit. If the team fails to provide the correct answer, each member of the opposing team gets a chance to answer within a 15-second timeframe. By expanding the game's accessibility to various platforms, such as web browsers, the Gaming Room project aims to offer a broader audience the opportunity to enjoy the exciting gameplay of "Draw It or Lose It" and promote multiplayer engagement and competition.

## Requirements

The client wants to make their game, Draw It or Lose It, available on different platforms like iOS, Android, and desktop computers. They also want the game to work smoothly on all devices, with a modern and responsive design. It's important for the game to handle many players, so it needs to scale up easily. The client wants to keep the costs of the server operating system low.

From a technical perspective, the game needs to be hosted on Linux, Mac, and Windows operating systems. It should be compatible with various web browsers and mobile devices. The development team needs to use the right programming languages and tools for the job. They also need to consider how the project will impact the team and whether they'll need multiple teams. Additionally, the team should evaluate any licensing costs associated with the development tools. Overall, these requirements will shape how the software application is designed and built to meet the client's goals and ensure a successful gaming experience across different platforms.

## [Design Constraints](#_2et92p0)

1. Storage Constraint: The server needs to accommodate sufficient storage capacity to store the client's collection of stock photos.

2. Player/Game/Team Management Constraint: A player management system must be developed to effectively oversee players, games, and teams.

3. Concurrent Requests Constraint: The server should be capable of handling a large number of concurrent requests from a multi-platform player base.

4. Security Constraint: A robust security system is required to ensure secure login and authentication for players.

5. Player Login Constraint: A player login system must be implemented to enable players to identify themselves within the game.

6. Programming Language Constraint: The existing code for the client's Android app is written in Java.

Implications on Application Development:

Developing the game application requires selecting compatible web technologies, implementing collaborative features, designing synchronization mechanisms, and incorporating validation checks. These design constraints influence architectural decisions, choice of frameworks, and data management strategies to create a robust and user-friendly web-based game application.

## [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)

In the UML class diagram, the Entity class serves as a superclass that establishes a relationship with the Game, Team, and Player classes. This relationship implies that Game, Team, and Player inherit or receive information from the Entity class. This inheritance allows them to share common attributes and behaviors defined in the Entity class. Additionally, the diagram demonstrates a "has a" relationship through aggregation. Team and Player have a "has a" relationship with Game, meaning they are instances of one class and have a reference to an instance of another class. In this case, Game has a reference to Team, and Team has a reference to Player.

Furthermore, the diagram shows that GameService has a reference to Games, indicating that GameService manages a collection of Game instances. Similarly, Games has a reference to Teams, indicating that a Game consists of multiple Team instances. Lastly, Team has a reference to Players, implying that a Team is composed of multiple Player instances. The UML class diagram effectively demonstrates the relationships between the classes and adheres to object-oriented programming principles, such as inheritance and aggregation. This allows for efficient fulfillment of the software requirements by promoting code reuse, modularity, and the organization of related objects into meaningful structures.

**"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:  Simple deployment on a server  Proven track record of security  Extensive documentation available for macOS software  Excellent technical support plans and products  Disadvantages:  High cost compared to alternatives  Limited hardware options to choose from  Requires a Mac system running macOS  Updates are solely provided by Apple  New software versions may require additional payment | Linux is a reliable choice for hosting web-based software applications. It offers flexibility, stability, and a wide range of options to meet different hosting requirements. However, it may require more technical expertise to set up and manage compared to other platforms. | Windows provides a user-friendly environment and compatibility with a wide range of software and hardware. It is advantageous for organizations invested in the Windows ecosystem, but licensing costs and resource requirements should be considered.  However, it may be more resource-intensive and susceptible to security vulnerabilities compared to other operating systems. | Hosting web-based applications on mobile devices offers mobility and accessibility. However, limited computing resources and compatibility considerations with different mobile platforms can impact performance and scalability. Thorough testing is necessary to ensure compatibility across devices. |
| **Client Side** | Advantages:  Wide range of well-supported web browsers with abundant developer tools.  Convenient cross-browser testing software.  Moderate development time and deployment.  Disadvantage:  Requires Apple product with macOS. | In the case of Linux, the advantage lies in its cost-effectiveness, with free development tools available. However, developers should be prepared to adapt the application to different Linux distributions and possess proficiency in languages such as C, C++, or Python. | When targeting Windows as a client platform, developers may encounter licensing costs and the necessity to adhere to Microsoft's guidelines. Proficiency in languages like C# or .NET is crucial for efficient Windows software development. | For mobile devices like Android and iOS, developers must consider platform-specific tools and adapt the application to various screen sizes and resolutions. Proficiency in languages such as Java/Kotlin for Android or Swift/Objective-C for iOS is essential to ensure compatibility and optimal performance on mobile platforms. |
| **Development Tools** | For Mac development, programming languages such as Objective-C and Swift are commonly used, along with development tools like Xcode, Apple's integrated development environment (IDE). | In the case of Linux, programming languages like C, C++, and Python are widely employed, with popular development tools including GCC (GNU Compiler Collection) and IDEs such as Eclipse or Visual Studio Code. | When targeting Windows, programming languages like C# and .NET framework are commonly utilized, along with development tools like Visual Studio, Microsoft's comprehensive IDE. | For mobile phones, Android applications are typically developed using Java or Kotlin programming languages, with development tools like Android Studio. iOS applications, on the other hand, are built using Swift or Objective-C with Xcode as the primary IDE. |

## 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 expanding Draw It or Lose It to other computing environments, I recommend utilizing a cloud-based operating platform. Specifically, I suggest adopting a Platform as a Service (PaaS) model, which offers a managed environment for application development and deployment. This choice provides the necessary scalability, flexibility, and ease of deployment across multiple platforms.
2. **Operating Systems Architectures**: Within the chosen cloud-based operating platform, the underlying operating system architectures will vary based on the specific cloud provider. However, most cloud providers utilize a combination of virtualization and containerization technologies to provide a scalable and isolated environment for applications. These architectures typically involve hypervisors or container runtimes that manage the virtual machines or containers, respectively, in which the applications run.
3. **Storage Management**: To ensure efficient storage management for Draw It or Lose It, I recommend utilizing a distributed file system. Distributed file systems distribute data across multiple storage nodes, providing scalability, fault tolerance, and high-performance access. One popular choice for distributed file systems is the Hadoop Distributed File System (HDFS), which is designed to handle large datasets across a cluster of machines
4. **Memory Management**: The recommended operating platform, especially in a cloud-based environment, typically abstracts the details of memory management from the application developers. The platform dynamically manages memory allocation and deallocation based on the application's resource requirements. It utilizes techniques such as virtual memory and memory isolation to ensure efficient utilization of resources. The application developers can focus on programming without explicitly managing memory.
5. **Distributed Systems and Networks**: To enable communication between various platforms and devices, a distributed system architecture can be employed. Draw It or Lose It can utilize a service-oriented architecture (SOA) or a microservices architecture, where different components of the game are encapsulated as services. These services can communicate with each other through APIs over a network, allowing interoperability across platforms.
6. **Security**: To protect user information on and between various platforms, it is crucial to implement robust security measures. The recommended operating platform should offer strong user protection and security capabilities. This includes enforcing secure authentication and authorization mechanisms, encrypting sensitive data in transit and at rest, and implementing secure communication protocols. Furthermore, additional security practices such as input validation, secure coding practices, regular security audits, and intrusion detection systems should be implemented. Employing secure coding frameworks and libraries and keeping up with security patches and updates are also essential to mitigate potential vulnerabilities.

In conclusion, by adopting a cloud-based operating platform with a distributed file system, leveraging memory management techniques, implementing a distributed system architecture, and ensuring robust security measures, The Gaming Room can effectively expand Draw It or Lose It to other computing environments while providing a secure and seamless gaming experience for users.