

# **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 | <08/18/24> | <Jose Munoz> | <project 3 submission> |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

## [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** | <Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Linux for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

## 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 Draw It or Lose It, Linux is the recommended operating platform. Linux is known for its cost-effectiveness, scalability, security, and performance. As an open-source platform, Linux provides a flexible and stable environment that can easily scale to accommodate increased user demands and expanding functionality. It also offers cross-platform compatibility, allowing the game to be expanded and integrated into multiple computing environments such as Windows, macOS, and mobile systems. The lightweight and high-performance nature of Linux makes it ideal for gaming environments that require efficient resource management and support for distributed systems.
2. **Operating Systems Architectures**: The Linux operating system is based on a modular architecture, which includes a monolithic kernel that manages core system functions like memory management, process management, and file systems. The architecture of Linux supports multithreading, multiprocessing, and the use of microservices, making it suitable for scaling the game across different servers and devices. Linux offers a monolithic and microservices approach, where specific services or features of the game can be isolated and run independently for better fault tolerance and maintenance. The platform also supports containerization technologies such as Docker, which enables the deployment of the game in isolated environments, reducing system conflicts and improving scalability.
3. **Storage Management**: For Draw It or Lose It, Linux can efficiently manage storage through various file systems like ext4 and XFS, which provide stability, performance, and large storage capacity. For distributed storage management, Linux can utilize Network File System (NFS) or Ceph, a distributed object store and file system designed to provide excellent performance, reliability, and scalability. Cloud storage services like Amazon S3 or Google Cloud Storage can also be integrated for redundancy and backup. These storage systems allow for scalable and reliable data management, ensuring that game assets and user data are always accessible and securely stored.
4. **Memory Management**: Linux offers advanced memory management techniques such as paging, swapping, and virtual memory. These techniques allow the operating system to efficiently manage memory resources by allocating the required memory to each process and using swap space to handle larger workloads when necessary. For a game like Draw It or Lose It, which may involve significant multitasking and processing, Linux's memory management will ensure that the system runs efficiently by dynamically adjusting memory allocation based on real-time demands. Additionally, Linux's kernel supports memory overcommitment, which allows the system to allocate more memory to processes than is physically available, optimizing the performance of the game during peak usage periods.
5. **Distributed Systems and Networks:** To enable Draw It or Lose It to communicate between various platforms, Linux supports distributed systems through the use of distributed computing protocols such as Remote Procedure Calls (RPC), gRPC, and RESTful APIs. The game can be structured to run on a distributed architecture, using multiple servers and services that communicate over a network. Linux supports load balancing to ensure that no single server is overwhelmed with requests, as well as clustering technologies like Kubernetes to manage and orchestrate multiple containers across different platforms. The use of distributed databases such as Cassandra or MongoDB will allow user data to be synchronized and shared across various platforms. Linux also supports robust networking tools to handle connectivity issues, ensuring that even in cases of outages or disruptions, the game remains functional and user data is preserved.
6. **Security**: Security is a critical concern for The Gaming Room, and Linux offers several mechanisms to protect user information. Firstly, Linux is known for its strong security model, which includes role-based access controls (RBAC), firewalls (iptables, nftables), and mandatory access control systems such as SELinux or AppArmor. To protect data during transmission between platforms, Linux supports SSL/TLS encryption. Additionally, Linux supports public-key infrastructure (PKI) for secure authentication. For data at rest, Linux can utilize encryption techniques such as LUKS (Linux Unified Key Setup) to encrypt hard drives and partitions. The platform also integrates with multi-factor authentication (MFA) solutions to ensure that only authorized users can access sensitive data. Finally, regular security patches and updates are provided by the vast Linux community, ensuring that the system remains resilient against new and emerging threats.