

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

**[CS 230 Project Software Design Template](#_Toc115077317)** [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 | 07/15/2023 | Brittney Jackson | Updated Executive Summary and Requirements |
| 1.0 | 07/16/2023 | Brittney Jackson | Updated Design Constraints and Domain Model |
| 1.0 | 07/29/2023 | Brittney Jackson | Updated Evaluation |
| 1.0 | 08/7/2023 | Brittney Jackson | Updated Recommendations |

**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)

Creative Technology Solutions has been approached by the Gaming Room to develop a web-based version of their existing game, Draw It or Lose It. This game is currently only available on Android, and the Gaming Room aims to expand across multiple platforms. Draw It or Lose It is a team-based game where players guess a puzzle based on rendered images. CTS will provide the necessary expertise to set up the development environment and streamline the development process.

## Requirements

* *Develop a web-based version of the game to expand its availability beyond Android devices*
* *Enhance user engagement by enabling team-based gameplay*
* *Support multiple teams and players within a game instance*

## [Design Constraints](#_2et92p0)

* Game application must be built using web-based technologies such as HTML, CSS, and JavaScript to ensure compatibility across different web browsers and platforms
* Application must be designed to handle concurrent requests efficiently.
* Needs to handle sensitive user information, such as team and player details.
* Design needs to consider the varying screen sizes, input methods, and capabilities of different platforms.

## [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)

The diagram include classes like Program Driver, SingletonTester, Entity, GameService, Game, Team, and Player. The ProgramDriver class serves as the entry point while the SingletonTester class tests the Singleton pattern implementation. The Entity class showcases encapsulation by encapsulating state and providing access through getter methods. The GameService class demonstrates the Singleton pattern by ensuring a single instance throughout the system and providing centralized access to game-related operations. Composition relationships exist between Game-Team and Team-Player, indicating that a Game can have multiple teams and Team can have multiple players. In addition, classes Game, Team, and Player inherit or associate with the Entity class by sharing common attributes or behaviors.

**"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** | The Mac is an operating system with a Unix foundation that provides high-quality performance, reliability, and security, making it appropriate for hosting web-based software applications. With its many server software options, it includes built-in capability for running web servers. Simply said, compared to other platforms, Mac hardware is more expensive. | Linux is a popular operating system for hosting web applications because it is very adaptable and versatile. It’s widely recognized for being capable of managing huge workloads. With the numerous web server software options available Linux is frequently chosen due to its affordability and open-source status. However, setting up and configuring Linux may call for greater technical knowledge. | Windows offers a user-friendly environment. It supports major programming languages and has a wide range of compatibility with different web server technologies. It is accessible and delivers good performance, making it ideal foe developers at various levels. Licensing costs for Windows server operating system may add to the client's expenses. | Web-based apps are frequently accessed using portable devices like smartphones and tablets. Users are able to play games on the go due to its portability. Mobile devices can run a variety of operating systems, including IOS and Android, each of which has strengths and limitations. However, they have a large users base, mobile devices might not have the processing power to screen space for demanding gaming apps. |
| **Client Side** | Mac offers a user interface that is simple and an agile environment for software development. Depending on the complexity of the application and the level of experience of the development team, the price of the development of the Mac software can change. | Popular web browsers are supported, and they work well with web standards. Obtaining cross-browser compatibility and maximizing performance across various distributions are major development considerations. With Linux being a open-source, the cost and time involved in supporting multiple clients can be moderate. | It is well-documented and supports a wide range of programming languages and frameworks. Due to the widespread use of Windows , there is a sizeable development community and an abundance of information. To ensure compatibility across various versions and setups, developing for Windows could need more time and effort. | In order to support several mobile devices, it is necessary to take various screen sizes, touch-based interactions, and device-specific features into account. Depending on the complexity of the program development costs and time may differ. |
| **Development Tools** | Mac programmers often utilize Swift or Objective-C. The official IDE from Apple, Xcode, offers a complete collection of tools for creating, testing and debugging apps. These tools are generally free for macOS users. An interface builder for creating user interfaces and an IOS Simulator for testing apps without actual IOS devices are both included. | Programming languages like C++, Python, or JavaScript may be useful for creating a game app on Linux. For game development, IDEs like Visual Studio Code, Eclipse, or PyCharm from JetBrains can be used along with frameworks like Qt or SDL. These tools often open-source and have no licensing costs. | C#, C++, or JavaScript are suitable programming languages for creating the gaming application on Windows. Comprehensive development environments are offered for Windows app development by IDEs like Visual Code or Visual Studio Code. Popular options for creating games on Windows include game development tools like Unity or Unreal Engine. These tools may have licensing costs depending on the specific editions used. | Swift for IOS development, Java for Android development, or Kotlin for both platforms, are suitable programming languages for game application for mobile devices. Mobile app development is supported by IDEs like Xcode for IOS and Android Studio for Android are usually free. Cross-platform mobile game creation is possible using frameworks like SpriteKit or Unity can have licensing costs for commercial use. |

## 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**: Linux is the recommended operating system. Linux is a great option for hosting web-based software applications like CentOs or Ubuntu Server. Linux offers several benefits like reliability, security, flexibility, and cost-effectiveness. Wide-ranging web server software is supported with several customization options. It’s highly versatile and capable of handling demanding workloads, making it appropriate for facilitating future growth and expansion.
2. **Operating Systems Architectures**: Linux, with its Unix-like architecture and monolithic design, provides a strong foundation for the operating system. This approach allows the kernel to directly access hardware resources, resulting in better performance and efficiency. The kernel efficiently handles critical tasks, including process scheduling, memory management, and device drivers. Moreover, Linux supports various types of dynamically loadable kernel modules, such as device drivers that enable communication with hardware devices and file system modules that facilitate interaction with different storage formats. These modules enhance the kernel's functionality without requiring a full system restart, contributing to Linux's flexibility and adaptability to various hardware configurations and user needs.
3. **Storage Management**: An ideal storage management system for Draw It or Lose It involves a combination of efficient file systems and a reliable database management system. With various file systems such as ext4 and XFS, which offer great performance and data integrity, for storing various game assets such as graphics and audio files. Implementing a reliable database management system like MySQL or PostgreSQL can be used to handled structure data, including player profiles, game progress and achievements providing advanced querying capabilities to enhance data organization and retrieval.
4. **Memory Management**: For a program like Draw It or Lose It, the use of Linux's virtual memory management strategies is effective for efficient memory allocation and management. Linux employs demand paging, which means data is loaded into memory only when needed, maximizing resource usage. Virtual addresses are translated into physical addresses in Linux using page tables and memory mapping algorithms. This translation process allows the operating system to efficiently manage memory, ensuring that each process has its own isolated virtual address space while utilizing physical memory resources optimally. Additionally, the Memory Management Unit (MMU) ensures protection and isolation of processes' memory spaces, preventing unauthorized access and maintaining memory integrity. Furthermore, using memory deallocation and allocation methods efficiently manages memory, minimizing the risk of memory leaks and ensuring optimal performance. Automatic garbage collection periodically identifies and releases memory that is no longer reachable by the program, preventing memory leaks caused by forgetting to deallocate memory.
5. **Distributed Systems and Networks**: To enable communication between various platforms, Draw It or Lose It can implement real-time communication APIs and protocols such as WebSocket and RESTful APIs. A client-server architecture can be utilized, with the server managing game logic and client-to-client communication. Standard networking protocols like TCP/IP or UDP can facilitate reliable and secure data exchange between connected devices. Employing redundancy and load balancing techniques further enhances availability and mitigates outages by distributing network traffic among multiple servers and replicating critical components or data across different locations.
6. **Security**: We will implement robust security measures, including access control mechanisms such as file permissions and user management, along with secure remote access protocols like SSH (Secure Shell) for server management. To ensure secure data transmission between platforms, we will use encryption techniques like Transport Layer Security (TLS). User authentication will be implemented to control access to sensitive features or data. Regular security updates will be applied to address vulnerabilities, and security audits will be conducted to strengthen the system's overall security posture.