

Draw It or Lose It

# **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 | 07/20/24 | Theresa Engstrand | Update to the Executive Summary, Design Constraints, and Domain Model. |
| 1.0 | 08/04/24 | Theresa Engstrand | Supply an Evaluation of server side, client side and development tools on different operating systems. |
| 1.0 | 08/17/24 | Theresa Engstrand | 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)

The Gaming Room is seeking to set up a gaming environment that is compatible on multiple platforms. Their game, Draw It or Lose It, is currently a web-based game available as an Andriod app only. The solution is for us at Creative Technology Solutions to prepare a software design document and begin developing the game application, ensuring that The Gaming Room’s software requirements are met.

## Requirements

The Gaming Room’s requirements include:

Having the ability to have one or more teams involved

Each team 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 by creating unique identifiers for each instance of a game, team, or player

## [Design Constraints](#_2et92p0)

There are a few different design constraints and related implications that go along with developing this gaming environment. For starters, the app is currently only available to Android users. Platform compatibility is one design constraint. To make this app compatible on other operating systems, certain designs can be limited, including input method and processing power. As users and teams evolve and more games are added, the program needs to be scalable to be able to handle the amount of people playing at once. Another design constraint related to the requirement of having unique game and team names would be to have unique identifiers in order to avoid a conflict of team name or game name.

## [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 Gaming Room UML diagram listed below depicts the relationship between all the classes required to make the game function. The Entity class is a base, or parent, class. It holds attributes and behaviors that all the child classes have in common. Id and name are private attributes within Entity that are encapsulated to prevent external code from altering their data. There is a private constructor Entity(), which holds default arguments. A public Entity() which take the arguments id and name can also be found as a method of Entity. Other public methods include the getter methods to access the id and name. They are getId() and getName(). There is also a public method toString(), which will mutate another data type to a String data type.

The child classes, Game, Team, and Player inherit the attributes and behaviors found in the Entity class. In addition to the attributes and behaviors of the Entity class, the Game class also has a private variable, teams, which is a list that stores the team names. The methods in the Game class include Game(); which takes id and name as arguments, the mutator method addTeam(); which takes name as an argument to add a team name, and toString(). The toString() method is an example of polymorphism because each child class is responsible for its own toString() code.

The Team class has a private players attribute, which is a list that holds all the player names. Other than the overloaded toString() method, Team also has Team(); which takes id and name as arguments, and addPlayer(); which takes name as an argument to set the name of the player.

The third child class is Player. It contains no attributes. There is a Player() method which takes id and name as arguments, and the toString() method.

The three child classes are associated with one another with a possibility of zero to many instances between them. The Game class has an associated class called GameService, which is where the flow of the code will take place. Private attributes, such as the list of games, nextGameId, nextPlayerId, nextTeamId, and the object service are included. The methods include a constructor method, GameService(), and accessor and mutator methods getInstance(), addGame(), getGameCount(), getNextPlayerId(), and getNextTeamId. There are also instances of method overloading with two different getGame() method options. One takes a name as an argument, and one takes the id as an argument.

Aside from the classes which stem from the Entity class, the ProgramDriver is where the code will interact in the main() method. The ProgramDriver class uses the functionality of the SingletonTester class. The SingletonTester class is used to test whether there is only one instance of a game stored in memory.

**"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** | **Characteristics**:  Operates using Unix, which excels at multi-tasking and handling multi-users.  Provides a stable and reliable environment for developers.  Comes with a powerful terminal and command-line interface.  **Advantages**:  Mac has seamless integration between Apple products, like iPhone.  Mac is known for development in creative projects and its stability and reliability.  **Weaknesses**:  Hardware is more expensive than competitors.  Limited to Apple’s hardware options. | **Characteristics**:  Open-source operating system (freely available and customizable).  Stable and able to handle heavy workloads.  Frequent security updates to keep Linux secure.  Customizable distributions depending on project.  Powerful command-line interface.  **Advantages**:  Free.  Compatible with various web technologies, like MySQL and Python.  **Weaknesses**:  Some applications or tools are Windows-specific and may not be natively compatible. | **Characteristics:**  Windows Server OS integrates well with other Windows products and services.  Internet Information Services handles HTTP requests and serves web pages.  Microsoft SQL Server, a database system for data storage and management.  ASP.NET allows interactive website development.  **Advantages:**  Compatible with other Windows products, like Microsoft 365 and Azure.  Scalability allows the growth of resources.  **Weaknesses:**  Licensing can be pricey.  Limited open-source support. | **Characteristics:**  User-friendly.  Flexible to allow developers to update from any location.  Relatively inexpensive.  **Advantages:**  Cloud hosting allows for ease of scalability.  Pay for what you use with cloud hosting.  Transfer speeds of cloud hosting are comparable to on-site hosting.  Reliable with built-in disaster recovery.  **Weaknesses:**  Limited hardware access restricts access to device hardware, like cameras.  Smaller screen size, platform specific requirements, and input method may involve time constraints. |
| **Client Side** | **Cost**:  Mac hardware and software licenses are more expensive.  **Time**:  Depends on the complexity of application.  Various tools and libraries that can help speed up development.  **Expertise**:  Experience with Swift and Xcode. | **Cost:**  No licensing fees.  Possible customizing fees, depending on project.  **Time:**  Compatibility testing could take time since each distribution may have unique limitations or features.  **Expertise:**  Knowledge in system administration and security practices and ensuring cross-platform compatibility. | **Cost:**  Licensing costs for tools, libraries, and frameworks.  Possible cost for development tools.  Windows-based testing infrastructure.  **Time:**  Learn integration with Windows features.  Regular updates take time.  **Expertise:**  Windows knowledge, such as .NET and Win32.  Knowledge of Windows design guidelines.  Knowledge with Windows security best practices. | **Cost:**  Depends on scope and complexity.  **Time:**  Cross-platform development to write code once and develop it for multiple platforms will help to save time.  **Expertise:**  Knowledge of UX/UI design.  Platform-specific requirements.  Knowledge of integrating with the device features. |
| **Development Tools** | **Language**:  Swift is the primary language.  **IDE**:  Xcode provides a comprehensive set of tools for writing, testing, and deploying applications across all Apple platforms. | **Language:**  Python  C++  Vim  **IDE:**  NetBeans, free cross-platform IDE that supports Java and other languages. | **Language:**  Python  Java  C#  **IDE:**  Windows Presentation Foundation for building desktop applications.  Visual studio. | **Language:**  Apple/iOS: Swift  Android: Java or Kotlin  React Native: JavaScript  Flutter: Dart  **IDE:**  Apple/iOS: Xcode  Android: Android Studio |

## 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**: The operating platform that I feel would be best to allow The Gaming Room to expand Draw It or Lose It to other computing environments is Windows 10 or 11. I chose Windows because of its flexibility and customization abilities. Windows is ideal for The Gaming Room to use because it can be easily integrated with various hardware setups, making it easier for Draw It or Lose It to evolve over time.
2. **Operating Systems Architectures**: Windows has optimal graphics performance and efficiency using features such as DirectX 12, a collection of Microsoft-developed application programming interfaces (APIs). APIs enable gaming software to communicate efficiently with the computers’ hardware components. Windows operating systems is also compatible with many games and gaming software already, making it an ideal operating system for expansion of Draw It or Lose It.
3. **Storage Management**: The storage management system that I recommend using with Windows includes disk management, virtual storage pools, automatic cleanup, and the use of some third-party tools. Disk Management is a built-in tool for Windows that enables the ability to manage hard drives, allowing for the creation, deletion, and formatting of the drives and paths. Virtual Storage Pools pull together multiple physical disks into a single pool of storage. Then, virtual disks, or storage spaces, can be created. These disks appear to Windows as regular disks. Cloud storage can also be useful to The Gaming Room to allow the game to have scalable storage as more users are added. Cloud storage is ideal for gaming collaboration to enable users to access their data from anywhere via internet connection. Storage Sense, an automatic cleanup, manages disk space by automatically deleting unnecessary files. Storage Sense can also be configured to run automatically or manually. There are many third-party tools available for storage management. One, for example, is AOMEI Partition Assistant, which can convert the operating system to a solid-state drive (SSD). Migrating to an SSD would allow for improved speed, performance, and reliability, to name a few benefits.
4. **Memory Management**: Windows uses dynamic memory management for allocating and deallocating memory. In the Draw It or Lose It software, dynamic memory management allows the game to handle the number of users, teams, and the state of the games efficiently. Windows uses a cache management system to keep frequently accessed data in memory which is essential for speeding up access times and for optimal performance. Virtual Memory is used by Windows to allocate the available physical memory using disk space, which allows for a greater ability to multitask. With virtual memory, each process has a unique virtual address and is mapped to physical memory as needed. Utilizing these memory management techniques, Draw It or Lose It will be able to handle multiple users and teams without running out of memory, and the game will run smoothly with quick access to necessary data and minimal lag.
5. **Distributed Systems and Networks**: Distributed software allows computers and devices, or nodes, to communicate over a network to share resources and coordinate actions through message passing. Using distributed systems for Draw It or Lose It would ensure efficient communication between various platforms. TCP/IP is a communication protocol used for internet communication. This ensures reliable communication between the devices, or nodes. HTTP/HTTPS is used for web-based communication, which enables cross-platform communication. Other tools, such as RabbitMQ, ensure messages are delivered even if some nodes are unavailable. Draw It or Lose It should be developed to use a combination of communication protocols to ensure reliable communication. Then there are middleware tools, like Apache Thrift, that standardize communication protocols and formats across different platforms. Network connectivity is an important aspect of ensuring effective communication. Local Area Network (LAN) is best for local geographical communication. Wide Area Network (WAN) is best for communicating over larger distances. WAN is essential for internet-based communication. The internet is the largest WAN, providing efficient communication globally between devices. Redundant nodes and network paths are used to keep the system running if some components are not functioning. Other techniques can be implemented to ensure fault tolerance, like replication. Monitoring the system and receiving automated health alerts can be beneficial in detecting issues.
6. **Security**: Protecting user information is of utmost importance. Windows has robust security features designed to protect user information, such as Microsoft Defender Antivirus, firewall & network protection, account protection, and device security. The antivirus protects against malware, viruses, and other threats in real-time. The firewall prevents unauthorized access. Windows Hello and Dynamic Lock protect your account at a device level while the user is away. Device security protects against firmware attacks. Windows 11 has enhanced security, such as virtualization-based security, hypervisor-protected code integrity, and Microsoft Pluton security processor. VBS helps to prevent malware attacks by isolating parts of the operating system. HVCI allows only trusted code to run within the operating system. The security processor isolates sensitive user data, such as login information and encryption keys. Encryption ensures secure communication and data protection across multiple platforms by encrypting data on the sender’s side that can only be decrypted by the intended party. Multi-Factor Authentication adds to the level of security by requiring more than just a password for user entry, such as a password and a push notification sent to the user’s phone, or a password and a biometric credential. Windows has an added open-source modification called AtlasOS. AtlasOS is specifically designed for optimal performance, privacy, and usability. AtlasOS provides privacy enhancements, such as telemetry reduction, which reduces data collection to improve privacy. It also eliminates pre-installed apps to ensure a cleaner experience. System monitoring is imperative as it allows for regular updates to ensure the latest version of security patches.