

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc909082014)

[Table of Contents 1](#_Toc246757203)

[Document Revision History 2](#_Toc393905730)

[Executive Summary 2](#_Toc1084228174)

[Requirements 3](#_Toc461394103)

[Design Constraints 3](#_Toc783793223)

[System Architecture View 3](#_Toc723195021)

[Domain Model 3](#_Toc721019771)

[Evaluation 5](#_Toc1215371583)

[Recommendations 7](#_Toc564605837)

## 

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | 08/18/2024 | Kibret A. Tecle | Analysis of various systems architectures and provided recommendations for the Gaming Room. |

**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**

Creative Technology Solutions (CTS) has taken on a new client, The Gaming Room. The Gaming Room company wants to develop a web-based game that serves multiple platforms based on their existing game, Draw It or Lose It. Draw It or Lose it is currently available on an Android app only. A game in Draw It or Lose It consists of four rounds of play lasting one minute each with images as cues rendered steadily until fully completed at the 30-second mark. If a team fails to guess the puzzle, the opponents get 15-secods to guess to solve the puzzle.

The Gaming Room has requested specific software requirements be met while designing and implementing the game application. As part of these requirements, The Gaming Room wants each game to have one or more teams, and each team to have multiple players assigned to it. Only a single instance of the game can exist in memory at any given time. To meet these criteria, CTS plans to adopt a Singleton design pattern. CTS will also implement an Iterator pattern to check the availability of game and team names before adding a new one. Unique identifiers will be created to uniquely identify each instance of a game, team, or player.

## Requirements

* A game can 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. To accomplish this a singleton pattern will be adopted.

## [Design Constraints](#_2et92p0)

* Each game must have the ability to have one or more teams involved.
* Each team must have multiple players assigned to it.
* Game and team names must be unique, and the system must check the availability of a name when users choose.
* One and only one instance of the game can exist in memory at any given time.
* The Gaming Room application must be able to run on multiple 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.

## 

The UML class diagram provided below shows the classes that makeup the Draw It or Lose it game application. Each class in the UML diagram is represented by a rectangular diagram that consists of three sections. These three sections enable us to describe the class name, its attributes, and its methods. The –sign in front of the attributes and methods indicates private access modifier, while the +sign indicates that the attributes and methods are publicly accessible.

The UML class diagram also shows the various relationships between the different classes.

The Entity class serves as the base superclass for Game, Team, and Player subclasses. The Entity class holds the common attributes (id and name) and behaviors (Constructor, getters, and setters) shared by the subclasses. All the subclasses of the Entity class (Game, Team, and Player) extend the Entity class and inherit the common attributes and behaviors of the Entity class. This relationship is called Inheritance. Inheritance is one of the pillars of Object-Oriented-Programming paradigm. Inheritance enables subclasses to inherit properties(fields) and behaviors(methods) from the parent class. Inheritance enables code reuse, thereby minimizing code duplication and maximizing code maintainability.

Moreover, the UML diagram depicts an association between GameService and Game, between Game and Team as well as between Team and Player classes. From the UML diagram we can also see the multiplicity relationship between the classes. The GameService class can have zero or many games. On the other hand, the Game class can have zero or many teams involved in it. Similarly, the Team class can have zero or many players who play in a team.

The ProgramDriver class is the entry point of the application and holds the main function. The ProgramDriver class invokes the getInstance method of the GameService to obtain the only instance of the game. ProgramDriver class does not know the internal implementation details of the getInstance method. These details are abstracted in the GameService class, and the ProgramDriver class uses the public getInstance method to obtain the instance of the game. This is another important principle of the core OOP principles. This principle is called Abstraction. Similarly, we can observe that Game, Team, and Player subclasses invoke the constructor of the superclass Entity to instantiate an object. The internal detail of the constructor is abstracted in the Entity class. Besides, the ProgramDriver class uses the SingletonTester class to test the application.

**"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.**

## 

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Pros  -Macs have robust security features and are considered very Secure and Stable for development.  -Macs also have robust performance and can seamlessly be integrated with other Apple devices.  Cons  -OS X server is available for Mac, but the licensing can be expensive.  -They are exclusive to Apple Hardwares, hence  You need Mac computers for development works which can create an overhead in initial investment. Besides, only a few programs are compatible with Mac. Macs also have higher pricing compared to Windows and Linux | Pros  -Available as an Open Source.  -Its robust open-source nature makes Linux highly customizable.  -Linux servers are highly flexible and scalable.  -Linux also offers good stability and security.  -Linux is a popular web hosting OS and is very widely used.  -Unlike the licensing costs to use Mac and Windows, Linux is free, and cost efficient  Cons  -Hardware Compatibility limitation with some proprietary software.  -There could also be a significant learning curve for the command line interface.  -Linux also requires more technical expertise. | Pros  -Highly compatible.  -As a result of their Extensive software compatibility, windows can accommodate different drivers, applications, and games.  -Windows is popular and widely used and has a broad developer community.  -Technical and maintenance assistance is readily available online or offline.  -Besides, Windows have rich documentation and resources  Cons  -Windows has higher vulnerability issues compared to Mac and Linux.  -Windows is less expensive than Mac but is more expensive than Linux. You need to pay for licensing to use Windows | Pros  -Mobile devices are easily Portable.  -User friendly interface, with touch screen and other intuitive features.  -They are also widely used  -Hosting web applications on mobile devices can be optimal if the applications are light weight.  Cons  -Limited screen size. The screen size is minor compared to Mac and Windows  -Limited hardware capacity.  -Not so scalable. Good for personal use.  -Can be more vulnerable to security threats. |
| **Client Side** | Pros  -Macs are more secure and easy to use once understanding of the UI is gained. -Possesses a sleek and visually appealing interface design.  -Macs have robust performance, and they are regarded for their stability and security.  Cons  -Mac are more expensive than Windows and Linux.  -Macs are also less compatible outside Apple OS. They can run only on Apple devices. This can make it harder to find skilled developers for MacOS programs. --MacOS SDK is built in Objective-C or Swift which are less used than java, C# or python which are used in windows and Linux. This can also create difficulty in finding expertise. | Pros  -Linux is an open-source operating system with wide distributions choices.  -Linux distribution offers various desktop environments such as GNOME.  -Offers both Command line interface and graphical user interface catering to users’ preference.  -Thanks to its robust permissions system Linux is less susceptive to security threats.  -Popular and widely used programming languages are used in Linux development such as Java and Python.  -Big pool of expertise of developers is available  Cons  -May require steep learning curve for command line interface. | Pros  -Windows possesses user-friendly interface with familiar features.  -Windows also has extensive software compatibility with strong gaming support.  -The programming languages used for Windows development such as C# (.NET) and C++ are very widely used among developer communities.  -There is a large development community with extensive expertise. This can be an advantage in finding resources.  Cons  -because of the vast number of windows suppliers such as Microsoft, Dell, Lenovo etc., product quality can be inconsistent  -Windows are potential targets of ransomware and cyber-attacks. Malware, spyware, ransomware are more prevalent in Windows. | -Mobile devices can serve as well as personal computers or webservers.  -They are not optimal to be used for multi-user ecosystem.  -A wide range of applications are available for mobile devices, and hosting a web app on mobile devices can help reach a vast pool of users.  Cons  Requires compatible OS. iPhone and apple mobile devices operate on iOS and are inoperable with other OS. While Android smartphones run on |
| **Development Tools** | Front End tech stacks such as Node.js and JavaScript are widely used. VSCode and XCode are some of the common Integrated Development Environment used.  Eclipse is available for Mac | Well-developed and rich development ecosystem. Various IDEAs are available such as VSCode, Intellij, Eclipse. Besides the command line interface is robust, and efficient. Python also comes installed in most Linux systems. | C# and .Net frameworks are used to develop Windows.  Microsoft Visual Studio is an extremely popular IDE used in Windows development. Visual Studio offers great integration options with third party libraries. In this age of Artificial Intelligence, Microsoft Visual Studio offers an excellent integration with Copilot, which assists developers in refactoring existing code, identifying bugs and null pointers, and generating test cases. Other IDEs available for Windows include Eclipse and Intellij | Android smartphones Software Development Kit (SDK) is developed on java and is universally used. Android Studio is the official Integrated Development Environment (IDE) for Android app development.  On the other hand, XCode is one of the main IDEs used to develop on all Apple platforms. |

## Recommendations

1. **Operating Platform**: I recommend using Linux servers. Linux servers are more cost efficient compared to Mac and Windows servers. Due to its open-source nature, Linux offers high customization and flexibility. Developers have complete access to the code base; hence they can modify the code to meet their specific customization requirements. Moreover, Linux operating platforms have robust security features and are less susceptible to malware and viruses when compared to other operating systems such as Windows. In contrast to proprietary software, which is developed by a single company and usually requires a license fee, Linux is open source. This makes Linux servers cost effective and can minimize the cost overhead such as the licensing costs of Windows, and MAC, as well as the costs attributed to the specific requirements of MAC for macOS. On the other hand, the Front End of the application can be developed language-agnostic and can be implemented on the preferred languages of each platform, such as SWIFT for iOS and .NET for Windows. API calls can be made to connect the frontend with the backend.
2. **Operating Systems Architectures**: I recommend a backend architecture to manage Draw It or Lose It game environment, and a front-end for rendering data to the user. API calls can be made from the frontend to the backend to connect the system. Moreover, cloud native microservices architecture can enhance the system architecture while embracing SOLID (Single Responsibility, Open-Closed principle, Liskov substitution principle, Interface segregation and Dependency Inversion) software development principles. Building cloud native apps can also enhance scalability and availability. On the other hand, by using front-end rendering, we could offload the memory-intensive tasks from the backend.
3. **Storage Management**: For the Draw It or Lose It gaming application, various factors that impact the storage requirement such as the total number of images that need to be stored and the size of each image along with the programs and files needed to run the application and important user profile information need to be carefully studied to determine the storage management. When deciding about storage management, it is important to think about performance, reliability, recoverability, and capacity of the storage systems. Cloud storage is an optimal solution for Draw It or Lost It gaming application. Cloud storage providers like AWS store the data in a distributed multi-AZ architecture to provide redundancy, which in turn helps with data recoverability and availability. Apart from cloud storage options, physical servers with HDD or SDD might be considered for the Gaming Room, though careful analysis needs to be made to determine scalability as the user base increases and new features which require more images get added to the application. When used efficiently, cloud computing, like in AWS, can be cost effective. Proper management of resources, however, needs to be followed, such as downsizing resource purchase when there is less usage and increasing the capacity on-demand when pick usage is expected.
4. **Memory Management**: Memory management is of high importance in application design and architecture. Memory management plays a key role in determining the performance of applications. Since the physical memory in a system is a limited resource, Virtual memory, a memory management system that can be implemented in Linux systems, can be used to abstract the physical memory details of an application software while allowing only needed information in the physical memory (demand paging). When a user requests data from the hard disk, the data is copied to RAM. While the data is being copied, Linux systems place the data in page cache. Hence page cache stores recently requested data and provides faster and efficient access if the same data is requested again. In iOS memory management is handled primarily through Automatic Reference Counting (ARC). ARC keeps track of and manages the memory used by the objects in memory. On the other hand, the 64-bit Windows operating system allocated up to 16 TB of accessible memory to the kernel. Even though each operating system has its own mechanisms and advantages, I recommend Linux for the Gaming Room use case.
5. **Distributed Systems and Networks**: In the client-server system of the Gaming Room, downtime and outages can be prevented by building the cloud native applications and deploying the app instances in redundant multi-AZ cloud architecture, such as the services offered by AWS. Not only will this ensure availability, but it can also improve scalability. Besides communications between front-end and back-end can be achieved by implementing RESTful Application Programming Interface. We also need to ensure that connectivity and outage issues are prevented by ensuring servers and applications have enough capacity to meet the forecast client demands.
6. **Security**: Security is another essential criterion in software design and architecture of paramount importance. Linux operating system is known for its robust security features. Linux allows granular level of control over the permissions that determine which users can read, write, or execute the files and directory on a Linux system. Linux supports a security model called MAC which provides a higher level of access control. MAC in turn allows administrators to set up strict rules governing access to resources. Besides the open-source nature of Linux contributes to better security, as a vast community of developers can review and contribute to the security requirements of the software. To further enhance security, multi-factor authentication (MFA) should be implemented to authenticate users. For authorization needs, role-based authorization can be applied by following the least-privilege principle. Additionally, proper encryption should be implemented to protect APIs, for instance SHA 256. Personal identifiable information should also be masked using proper encryption mechanisms.