

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 4**](#_Toc115077323)

[**Domain Model 4**](#_Toc115077324)

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 9**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 12/10/24 | Carlos Rodriguez | Changes made to the Recommendations section for picking Linux as the platform. |

**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 are looking to develop their web- based game, Draw It or Lose it, that can be used in multiple platforms. The purpose of the game is teams are competing by trying to guess what an image would represent with a timed response to answer it correctly, create unique names for each team, have one or more teams playing, assign multiple players on a team and have only one game running at a time. The issue that our client is facing is that they do not know how to set up the environment to make sure that the requirements I just listed are applied to the game. There will be key components that will be provided to the code framework to make sure that these requirements are being considered. The way we will cover this is by applying the design patterns of Singleton and Iterator Patterns. The Singleton Pattern will be used to make sure one game instance is running at any given time. The Iterator Pattern will be used to make sure that the team names are unique and are not being used. The design process will provide efficient search mechanics and help support any new updates for future use and help develop across multiple platforms.

## Requirements

The Game Room’s business requirement is to be able to provide the Draw It or Lose It game across multiple platforms and expand from their Android app. The technical requirements needed to develop the game is to have only one game running at any given time, have unique game and team names for the games played, a game will have one or more teams playing, and have multiple players assigned to each unique team.

## [Design Constraints](#_2et92p0)

1. A design constraint to implement is that it must manage multiple users to play the game simultaneously and interact concurrently. The implication for this is that its scalability must be able to support all the users that would be on the application and not cause it to slow down. The implementation of a concurrent mechanism like transactional operations would help stop the data corruption and stop the issues with the game play.
2. Another design constraint is to make sure that only one game should be running at a time and the gameplay names should be unique. The implication of this constraint is to check the applications consistency and implementing certain coding like, Singleton and Iterators, to check for uniqueness.
3. An additional design constraint is providing a similar user interface that would be able to be used across multiple platforms. This implication makes it easier to keep a consistent and manageable framework across the different platforms. Using Java, HTML or CSS would be a good programming language that is compatible across multiple platforms.
4. Another design constraint is the resource constraint that a mobile device has because of its limited battery life, processing power and memory. The implication for this constraint is that the application development must be optimal for the capabilities of a mobile device and should minimize the number of requests to the servers. It also needs to have graphics that does not go too big for the small screen of a phone.
5. A design constraint to implement is data encryption and data security to protect the information of the users and make sure that game play is not affected by outside sources that affect the outcomes. This implication for the game development is crucial to protect the integrity of the game and making sure that the user’s feel safe that their information will not be stolen or misused.

## [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 UML class diagram provided shows the 5 classes and the ProgramDriver and SingletonTester that is used the run the code for this application. The Entity class is built to have the attributes of id and name and have setters and getters for id and name. This is created as a base class where Game, Team ad Player class can share the code reuse and having it centralized into one spot to avoid repetition. The Player class is used to create the player for each team and generates their name. The Team class takes from the Player class to make the list of players on a team. The methods used is addPlayer and Team to create the team’s name for the game. The Game class then takes the Teams that will be playing to set up the game play. The methods are addTeam and Game to build the starting point and have the game run properly. The GameService class brings everything together to meet the requirements of our client. This is where we would apply the Singleton pattern to make sure that only one game is running at a time. It takes from the Game class to retrieve all the information from the players and the teams that were created. The ProgramDriver and SingletonTester are utility classes that are used to verify and test that the program itself runs correctly and to make sure that the Singleton pattern is following what it should be doing. The way these classes relate to each other is that Game, Team and Player all inherit from the Entity class. Team class contains zero or more Player objects, which allows for multiple players on a team. Game class contains zero or more Team objects, which allows for multiple teams per game. ProgramDriver is dependent on SingletonTester and it is used to verify that the Singleton pattern is correctly used for the GameService class. As mentioned, the object-oriented programming principle being used is inheritance, which occurs when Game, Team, and Player all inherit from the setters and getters of the Entity class. Encapsulation occurs in GameService and Team class because of its own attributes and methods that need to be set. We set private methods to make sure the code stays hidden and will not change and keep the other classes public so that way the user input can be applied to the program. Polymorphism is shown when Team gets the addPlayer function, when there already is a separate class of Player. It is able to be used in more than one location and the toString() method can be used to override information on each of the classes that has it included.

**"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** | Mac has stable and faster software when processing items, but it also does not have as much support as other servers like Linux and Windows. Mac is UNIX-based which helps create effective web-based applications. Their technical issues are catered more for in house fixes and licensing costs can be on the higher end. | Linux can process big loads for performance and scalability which is a strong reason many choose it for its reliability and performance. It is big on open-source and makes it cost less. A weakness is that the learning of this program can be a drawback in the process. | The advantage of Windows is that they have a hug market share in this sector and have many tools to help for development process. It is one of the most relied on programs, so it is easier to collaborate with many people. The weakness is the licensing fees that can prove to be costly. | The pros for it are that the processing does not to be that big for it, but this still shows that it does not have a lot of processing power to begin with. The limited data storage and servers proves to be a big drawback for the requirements needed for the programs. |
| **Client Side** | Mac users typically have a lot of Apple products already so the expertise level will translate fast when using the software’s. The common browsers used are Safari and Chrome. The cost is much because of the pricing Apple has for its products and not having many alternatives for third party’s with genuine Apple products. | Linux products prove to be cost effective for users and its ease to use development tools and test out their environments. Common browsers used are Firefox and Chromium. The downside is that the market share is smaller for them and the expertise level might not be there for many because of it. The standardization tends to overlook Linux and caters towards Apple and Windows. | Windows has the biggest user base and is used by many/ go to for many clients. The expertise level is the highest for them with product knowledge and usage. They dominate the market when it comes to desktop usage. The downside is that there are multiple Windows versions that might not be as easy to test or use for the software applications. | The convenience of using a mobile phone is something users like but it also has a lot of features that cannot be used that computers would be able to do. Battery life is something that gets taken into consideration because the processing power can drain the battery faster. The cost will depend on how it is allocated for both Android and Iphone users. |
| **Development Tools** | The relevant languages used are Java, Python, JavaScript and Swift are some of the relevant languages used for Mac. The tools needed for development are Xcode, IntelliJ IDEA, Visual Code Studio and React. The team would need the technical requirements of knowing the Apple-based tools like Xcode and Swift. They must make sure it runs properly on Safari and the teams need to separate to make sure the programs are accounted for. The licensing costs for these products does tend to add up and have a high bill because of the few options for third party Apple vendors. | The languages needed are Java, JavaScript Python, C++, PHP, and Ruby. The tools needed for development are Open- Source, Eclipse, IntelliJ IDEA, Visual Studio Code, Node.js, React and Docker. The development team would need to know how to use open-sources tools to be able to efficiently run this platform. Linux has little to no cost for the licensing and tends to be user friendly. | Python, Java, JavaScript, Power BI and C++ are relevant languages used by Windows for software development. The tools that are used are Visual Studio, Eclipse, React, Angular, NET Framework/ Core. The team must be familiar with Windows based- tools like .NET and Visual Studio. They should be able to manage multiple teams to test each tool, so it runs effectively on Edge/ Google Chrome. Their licensing costs is high but the products they produce are widely renowned and have a big market reach. | Mobile Devices use Dart, JavaScript, Java, Swift, and Kotlin for their programming languages. The tools that they use are Xcode, Android Studio, React Native, Flutter, and IntelliJ IDEA. The teams must have expertise on both Android and iOS functions to make sure the features are properly addressed for both. Multiple teams will be needed to develop both sides and make sure they run properly on any mobile device. These programs help with the reduced costs of licensing and reduces the need of complex programming. React Native and Flutter are both free, with the option for fees when you add more features to it. |

## 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 will allow The Gaming Room to expand Draw It or Lose it to the computing environment is Linux. The reasoning for this is because of the reliability and scalability that it has for its applications. Linux is compatible with the big development tools like Python and Java so it would be a great choice to have. Since it is open source, it is a lot more cost effective to have for its development to help the server-side. Linux is efficient when it comes to using server loads in multi-user environments, like the game we are trying to develop.
2. **Operating Systems Architectures**: The architecture of Linux is Bootloader, Kernel, Init System, Daemons, and more desktop environments. It can support various frameworks and servers that help with the performance and demand from the servers. Linux has its hardware layer, kernel, shell, and system/ application layers. The hardware layers includes the CPU and RAM components of the system. The Kernel architecture has important features like process management, memory management, and file system within this area. Linux does a good job managing the gameplay by allowing numerous players and teams to participate in the game, which is a key requirement for the game functionality. The shell would be the area where users send commands to the Kernel, for it to process. The system and application layers would be what makes up the application and game itself for the users to utilize and where it resides.
3. **Storage Management**: A storage management to use for Draw It or Lose It would be Google Cloud Storage. This is where we would keep the image files to store for the game and where we would need to load necessary data, in case any data were to be lost. The tool of Ext4 would be able to do this by using metadata journaling. What this does is it keeps the changes that are made for data consistency and in the case of the system crashing. Continuing on, the content delivery network feature would help load the assets and functionality to users all throughout the world and be able to integrate many users to the game. This is the area where the users will have the game data saved permanently, even when the system is shut off. To maximize the storage usage, we can try to decompress images to make them smaller but keep the high quality of it.
4. **Memory Management**: Linux uses memory management techniques for the Draw It or Lose It software by utilizing. Setting up the cache for the game will help load the images faster and utilize only what is being loaded to minimize overload on the server. A technique that they use is virtual memory, which divides physical memory into smaller fixed- size pages to help manage the systems memory process. To help with only loading what is necessary, Linux uses demand paging to retrieve the physical memory that is needed from the command of the users. To help with the memory allocation, Linux uses Slab allocator to help manage the memory of the kernel objects to a proper location for efficient management. Providing proper memory management helps with the overall performance, speed and efficiency of the game play by providing what is most necessary during gameplay.
5. **Distributed Systems and Networks**: In order to have various platforms and multiplayer gameplay, The Gaming Room should include a client-server model to help with the functionality across multiple platforms. The way that the players can communicate with each other would be to establish a Peer-to-Peer Network to make it easier for them to interact with their teammates. To address network connections, the TCP/IP is a protocol that helps with making sure there is an efficient network connection between the servers and the clients and that it helps with game data transmission. Another consideration for the networks is the emphasis on latency and bandwidth. To reduce latency, there must be a high-speed network. For bandwidth, there must be enough to make sure that the transfer of data is delivered and helps for the peak usage time of the gameplay. This will help with user experience and create a faster, more fun experience. The dependencies range from the server, network and client dependencies. The server dependency would be having clients rely on them to make the necessary updates, gameplay and security for them. If there was an error on the server side, the gameplay would be affected. The network dependency is crucial for the speed for the users and any issues can lead to disconnections and data not being properly saved. The client dependency would be for them to make the proper updates to make sure it is compatible with their device and that their security features are up to date to prevent any security breaches. The Gaming Room must make sure that the software and networks communicate effectively and are able to be compatible with each other. Linux has proper set ups to ensure that this is not an issue across multiple platforms.
6. **Security**: To protect user information on various platforms, Linux can implement various security features to make sure that unwanted attacks and issues do not affect The Gaming Room or its users. One of the security features to implement would be data encryption. This would be using a system like Transport Layer System to help send data encryption of important messages between the client and the server to protect unwanted data breaches. The team will also have a database management to help monitor any unusual activity and configure authorizations to certain users. For the users, The Gaming Room will only provide the necessary accessibility for the players to use so that the game runs properly and they do not make any adjustments to the overall game. This would be the user having a unique name and team name, to login into their account and provide/ load the information before the game starts. Another tool would be to implement a firewall, so the system can block and prevent attacks to the game. Also, making sure that all the software’s and tools are constantly being updated for their security and constantly monitored for any service error that might occur. The security should be a heavily emphasized part of the game to ensure the data integrity and safety of the user information, which Linux should be able to do a good job at enforcing.