

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | <08/10/2024> | Nicholas Wyrwas | Updated main design constraints, added executive summary as well as a domain model |
| 1.1 | <08/18/2024 | Nicholas Wyrwas | Updated Recommendations Sections and reviewed rest of content to ensure everything is up to date. |

**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)

Our new client, The Gaming Room, have decided to create a web-based game based on Draw It or Lose It. This current free game is available on Android only. They would like to expand the compatibility of the game and make it available to users who use other operating systems such as macOS, Linux, and Windows.

In the game, Draw It or Lose It, two teams compete to answer questions about images that are drawn. A user takes turns with this drawing. In each round of 60 seconds, there are 4 rounds.

## Requirements

Web-Based Access: The game should be playable in a web platform ensuring playable compatibility in all Web browsers and across operating systems.

Gameplay Rules and Game Modes: The game needs to include at least one team as a playable option, with the possibility of assigning multiple players to a single team for a memorable multiplayer experience.

Unique Game and Team Names: All games/teams need to have unique names or else there will be collisions. Users should be able to first see whether the name is already taken before they create a team.

Single Instance: The application must restrict itself to one and only one active instance in memory at a time.It should use unique identifiers for: The Game, The Team, The Player

Round Structure: Each game round can have five or six seconds of silence, followed by one minute of play time to allow drawings to be revealed, one segment at a time. The entire drawing would be revealed by the 30-second mark to provide a timed and exciting game experience.

Opportunity for Post-Time Guessing: in case a team is unable to solve the puzzle given the allotted time, the rest of the teams should get a chance to make one guess each in a 15-second window hence adding an element of fair and strategy to the game.

## [Design Constraints](#_2et92p0)

Cross-Platform Compatibility: The game’s an Android app now. To make it cross-platform compatible, we can transform its code into a web app, accessible from any OS. Since a universal language required for all platforms would prolong the process of choosing the programming language, using a REST API to communicate through HTTP (which is independently supported in every platform) will help in this regard.

User interface: Game is an Android application that has already been deployed (brought to market). Create a user interface matching the current design or, alternatively, implement a new design and deploy it over a new game.

Support for Multiple Teams and Players: It needs to have multiple teams, each team with multiple players. We need to handle the client-server architecture so that the server can handle multiple players at the same time.

Unique IDs: Each game name must be unique, as must each team name and each new game to be created. So, we need unique identifiers for the games, the teams, the players (which in this case are named human beings), etc. The application will run on different platforms, so we need to handle memory management, etc. Each player has a platform ID.

Images and Copyrights: The Gaming Room will use images in the Draw It or Lose It game. We will be using the images that are already in the Android version, they are built for all other platforms. We will also obtain licenses for images that might need to be added and/or copyright permissions as we are designing the game.

## [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)

Since we have the UML diagram, we will depict a visual representation of the design feature of the game system. Now let us look at the development of the program based on this diagram.

This UML diagram outlined the three core classes such as entity, game, team, and player with their relationship. This is with a start with the entity class contains the relationships with other three classes. The blue rectangle with white line arrow indicates that the game, team, and player classes develop with import another super class. The UML classes, variables and methods will be created and developed.

In the top left, the Program Driver class – which runs the program – points to the Singleton Tester class, telling it that the ProgramDriver will invoke the SingletonTester to ensure that the code is correct. This is where we can test the assertion that this specific implementation ensures that there’s only ever a single instance of the game in memory at any point in time.

All the complex methods needed to make the game work will be handled by the GameService class. Uniqueness – the program’s most important requirement – of games, teams and players is the main reason behind designing dedicated classes for each in the diagram below.

The classes are connected by associating links and the numbers between them are the number of associations for each class. For example, the GameService class could have 0 to many of the Game class. You will follow this diagram to build your program. However, you might need to change these lines.

**"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** | In particular, the macOS company provides server-based environments which greatly enhance the capabilities of clients of Macs operating together on a network.  One of the main benefits of having a macOS server is that it has full support of capability for Mac programs. Additionally, general administration is easy because of a well-designed graphical interface.  But there are also cons to using a Mac server. One of them is that it is quite expensive to maintain. The other is that it may not be very useful for enterprise-level companies and businesses which have a lot of third-party tools and customized programs. | Linux is free and open-source software, which means that tools can be deployed and used at very low cost or at no cost at all. Secondly, it is very secure because Linux is a customizable operating system that allows organizations to control and tailor their security settings. Thirdly, when using Linux web hosting, you can use many different programming languages such as Python, PHP, Perl and Ruby.  On the other hand, there is a learning curve to using a Linux server. If you have no experience with Linux, you might not like the learning curve. Some applications you might want to use are not available. And, migrating from Windows to Linux is not so easy. | Windows is a solid operating system. If you decide to go that route, be aware that we are talking about proprietary software, and licensing costs money. However, using a Windows web host server is a sound choice - here are some of the reasons why. Firstly, the support for a diverse array of applications and third-party software is excellent. Secondly, Windows patch updates and hardware upgrades are easy to access. Then, there is the fact that developers who are comfortable with the platform already have access to scripting languages like ASP.NET and databases like MySQL. | Though using my mobile device as a web server is not common, it is just a question of putting the right building blocks into the equation. Some companies such as Oracle provide mobile server-side solutions: Oracle Database Mobile Server enables the management of applications, users, devices, and data across large deployments of mobile or remote devices.  The advantages of it are, Oracle support iOS development tools, Android development tools and NoSQL databases synchronization. |
| **Client Side** | Although it involves higher initial investment, it usually has a longer lifespan. Its primary benefit is easy to use since it has excellent user interface. Secondly, it has strong security due to using the app store and it is stronger. | You need to learn it, and the interface is a bit more inviting, admittedly. But then again, being open source helps to secure it: numerous programmers are constantly improving and strengthening the OS and helping to develop it. | Extremely user-friendly Windows is a system most people are familiar with if they have ever been on a computer, and the price options are diverse. It is imperative that you properly set up the security on your system. | Android and iOS have advantages and disadvantages. Google's Android is highly customizable, and you're spoilt for choice in terms of price point, as well as the breadth of devices available. However, if the operating system is fragmented, this could affect the quality of your experience. You also must contend with not getting regular updates made available. In contrast, although Apple's iOS offers a more consistent experience, has regular updates, has a better reputation when it comes to security and opens to you a huge selection of well-designed apps, this comes with a higher price tag and very little customization ability. |
| **Development Tools** | Swift is the most popular language as software development has evolved despite that Java, JavaScript and Python remain the most powerful for a foreseeable time.  Generally, there is a lot of Integrated Development Environments (IDEs) supporting these languages, like Visual Studio family, Android Studio and PyCharm etc. | The most used languages are Java, Python, C, and C++. There aren’t many ‘mainstreams’ IDEs on Linux (Eclipse is one, followed by a bunch more that some of your existing users will immediately recognize). | Windows primarily uses C and C++. It gives you a large choice of IDEs such as Visual Studio or Android Studio, and PyCharm, and Eclipse, to enable efficient and flexible development. | There are languages in the frontend such as HTML, CSS, JavaScript while to those is supporting libraries. Developer tools is also a case in point. We have IDE tools from Eclipse to CP and PyCharm. HTML, PHP, C++ and Python are among the languages supported by developer tools. |

## 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**: Our preference is Linux. It’s an open-source operating system. Aside from the cost reduction through no licenses, availability of development tools is not an issue. Customizability and solid security are some of its strong points, which are important for web application development and hosting.
2. **Operating Systems Architectures**: Generally speaking, all operating systems consist of three layers: hardware, kernel, and shell. In more technical terms, hardware consists of the memory, the CPU, I/O devices, disks, and other physical parts of the system. The kernel is the entity that deals with interaction with applications, namely information processing; in other words, everything that happens between an application and the device. Shell is the interface that allows the user to interact with the operating system. Knowing these layers is essential for the design of applications that efficiently use system resources.
3. **Storage Management**: Since Draw It or Lose It will be a web application, we suggest using cloud storage. With cloud storage, we have the flexibility to scale storage capacity as the game's popularity grows without having to invest massive amounts in physical hardware. Google Cloud is a great option to power this project as it is highly scalable and comes with a full tool set of features.
4. **Memory Management**: Linux is a good system for this project because it can be so customized; we can use Java which does backend development for us and has its own memory manager in the form of a garbage collector, meaning memory management will be automatic and we can focus on development instead of code where we would have to manually manage memory.
5. **Distributed Systems and Networks**: Being web-based means that the game is available from any web browser that is compliant with established web standards. With Google Chrome available on all major platforms, users will not be restricted by the operating system they use.
6. **Security**: In terms of security, as the hardware layer is supplied by Google Cloud, we get all the benefits of Google’s infrastructure. The cloud provider is responsible for data center security, and we also have several security standards in place within Google. For software security, the automated updates should be included in the game and we need to make sure that an automated patching mechanism is present as well. Testing within the QA phase will help us to identify and manage security vulnerabilities.