

<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 | 1/22/25 | Megan Tallmadge | Added information for executive summary, requirements, design constraints, and system architecture view |
| 1.1 | 2/9/25 | Megan Tallmadge | Edited the evaluation table |

**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 looking forward to developing a web-based version of the game “Draw It or Lose It” that can run on other operating systems. It is currently only available on android devices. This game has multiple players and teams. It has four, one-minute rounds, in which the team in play must guess the picture from a group of images until they guess correctly or run out of time.

## Requirements

* Must run on multiple operating systems
* Each team had multiple players
* One instance of the game can exist at any time
* Team and game names must be unique

## [Design Constraints](#_2et92p0)

* run on multiple operating systems
* create a system for each platform
* teams include multiple players
* include a number cap to make sure that the teams have enough players
* include database to save unique team and player names and have a cross reference so that the names are not used more than once

## [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 diagram provides a visual design of the game “Draw It or Lose It”. In the UML diagram, there is a relationship between the Game, Team and Player classes with the Entity superclass. The arrows in this diagram show that these classes will inherit from Entity. This diagram shows which classes, methods and variables will be used in the program. The ProgramDriver class is pointing to the SingletonTester, which shows that the ProgramDriver will use the SingletonTester to test its code. This helps to test the restriction of having one instance that the game can exist in the memory. The class that contains all the complex methods that are in the skeleton of the game and its functionality. This diagram shows that the Team, Player and Game classes should all be unique. The lines in the connecting the classes show which class they are associated with, and the numbers show how many. GameService could have zero or more games associated with it.

**"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 offers many server-based environments, offering many advantages for clients on a network. It has a stable and secure foundation for hosting web applications. One disadvantage of Mac is that it could be expensive to maintain. It is not recommended for big companies or enterprises that heavily rely on third-party programs and customizations. It also has limited scalability compared to Linux and Windows. | Linux is an open-sourced operating system that allows a lot of flexibility and customization, with a wide range of software and tools. It is highly scalable and is well known for its security and stability. Its server is very popular because it is free. Their web hosting supports Python, PHP, Perl and Ruby. Its weaknesses would be its GUI limitations and hardware compatibility. It can take a long time to learn its software also. There may be some migration issues from Linus to Windows. | Windows has a strong developer ecosystem and a wide range of software compatibility. It is more beneficial for corporate users. It also has a lot of easy patch updates and hardware updates. Scripting languages like ASP.Net and MySQL are fully supported. One disadvantage would be that it is lacking protocols that help prevent malware, spyware, and ransomware. | Even though it is not common to use mobile devices for web servers, it is still possible to implement. Oracle offers mobile server-side implementation. Their server can manage applications, users, devices, and data on a large variety of mobile devices. There are advantages such as IOS dev tools support, android dev tools support. The disadvantage is that it has limited screen size and varied hardware capabilities. Mobile devices are also usually more vulnerable to security risks. |
| **Client Side** | Mac’s user-friendly interface makes it easier to learn. When developing and maintaining multiple clients, the development costs and time increases and it may require a more diverse expertise. It is inconvenient that it's programs are not available on other devices besides MAC. | Linux is free to use and distribute. It is more complicated to learn, which would take more time. It would need diverse expertise for different clients. Linux data is required to use its’ OS. Because it is open-sourced and user-controlled, it is more vulnerable to security issues. | Windows is easy to understand and learn to support. This means it would take less time to learn. The licensing costs could be higher compared to other open-source alternatives. | With mobile devices, it is important to consider things such as responsive design and connectivity limitations. There are features such as the camera, GPS, and push notifications. Mobile devices usually lack features that are provided on a PC. It is difficult to make programs operable on other systems. |
| **Development Tools** | Mac uses the Swift programming language. IDEs such as Xcode, Xcode Cloud and VSCode are used frequently. This allows a team to build, test and deploy apps more efficiently. | Linux includes a rich ecosystem that is compatible with IDEs such as VSCode. Sublime, and Atom. It has a robust command line interface and package management systems such as yum or apt. Docker and Docker Hub are often used easily build and deploy software applications. | Frameworks such as C# and .NET are popular for building Window-based web applications. Other languages such as HTML, CSS and JavaScript are also used. Popular IDEs that are used would be Eclipse, VSCode and JetBrains. | Languages such as Kotlin, Swift and Objective-C are commonly used along with Java and JavaScript. Mobile devices need libraries to support languages. Android Studio and XCode are commonly used. There are also many device emulators and simulators. |

## 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**: Adopting a web-based operating platform is advised in order to enable the Gaming Room to extend Draw It or Lose It to more computing environments. This would allow the game to be accessible and played on a variety of web-browsing devices, including smartphones, tablets, laptops, and desktop computers. The game may reach a wider audience and offer consistent experience across platforms by utilizing online technology.
2. **Operating Systems Architectures**: The architecture for the selected web-based operating system will mostly focus on web-based and client-server technologies. The client-side architecture will render the game interface and manage user interactions using HTML, CSS, and JavaScript. A multi-tier architecture with presentation, application, and data tiers can be used on the server side.
3. **Storage Management**: A relational database management system (RDBMS) and cloud storage services together would make an ideal storage management solution. User profiles, game statistics, and structured data pertaining to game progress can all be handled by the RDBMS. Media assets, including stock photos, can be stored on cloud storage systems, which guarantee scalability and cross-platform accessibility.
4. **Memory Management**: The suggested web-based operating system makes use of the automatic memory management features offered by contemporary web browsers. Developers no longer need to manually manage memory because browsers use garbage collection mechanisms to control memory allocation and deallocation. This stops memory leaks and ensures effective memory usage.
5. **Distributed Systems and Networks**: The game can make use of network connectivity and distributed software design to facilitate collaboration across multiple platforms. A centralized server or cloud architecture that acts as a communication center for game clients can be put in place to do this. The server may manage player messaging across many devices, real-time updates, and game synchronization. The system should have a suitable error handling and synchronization mechanism to account for network connectivity problems, such as sporadic outages or low bandwidth.
6. **Security**: A number of security measures can be put in place to safeguard user data both within and between platforms. Secure communication protocols ought to be supported by the suggested web-based operating systems. It is possible to restrict access to user profiles and game features by implementing user authentication and authorization procedures, such as login and password. Sensitive user data that is sent across a network or stored in databases can also be protected with data encryption techniques.