

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 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/10/2023 | Samantha Dur | Initial Documentation |
| 2.0 | 07/29/2023 | Samantha Durr | Addition to Evaluation Section |
| 3.0 | 08/10/2023 | Samantha Durr | Additions to Recommendation section |

**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 currently has an Android version of their game Draw It or Lose It. The company wishes to expand to a web-based version of the game.

## Requirements

The base requirements for the game is to work in a web browser with the ability to include multiple teams within an active game. The teams need to be assigned from the list of players in the game. The players, themselves, there needs to only be one session of the game running in memory at a time. All team names, game names, and usernames must be unique. For the drawing aspect of the game, there needs to be a library of images to draw the clues from. All rounds in the game need to be under a time limit. Game needs to check if the team has guessed correctly or let the other team guess.

## [Design Constraints](#_2et92p0)

1. The application must work on in a web browser.
2. Games need to be able to hold multiple teams.
3. Teams must be assigned from a list of players in the game.
4. There must only be one game session running in memory at a time for the players.
5. All team names, game names, and usernames must be unique.
6. Game must have a library of images to draw the clues from.
7. All rounds must be restricted to a time limit.
8. Check if the team has guessed correctly or the other team gets to guess.

## [System Architecture View](#_ilbxbyevv6b6)

Some of the system architecture should already be deployed, as the game has been released to Android devices. In terms of servers, Linux servers would be the best for keeping everything up to date for deployment of initial release and updates. The internal structure of the company is likely to use mainly Windows machines because Windows contains one of the largest selection of integrated development environments and IT is more familiar with it for necessary fixes.

## [Domain Model](#_8h2ehzxfam4o)

A provided UML shows the base domain model for the Draw It or Lose It game. This consists of classes such as Drawing, Evaluation, Game, and User. The Drawing class is for the drawings and the properties of the image data and the metadata. The Evaluation and Game classes work together. These classes inherit from the other methods of their parent classes. These two classes establish relationship and data exchange for the various objects used in the others. The user class is for the application users and has their username and password properties.

This is done using object-oriented principles, such as encapsulation and inheritance. The classes are the example of encapsulation with the class specific data and behavior being found within those classes. Inheritance appears directly in that nearly all of the classes inherit from the Entity class. These principles allow for the application to handle requests efficiently and for an easier time in updating the program as needed.

**"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)

Please see the table below for the advantages of using the different operating systems for server, client, and development.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | \* Offers server-based deployment from built-in Apache or MAMP.  \* Requires licensing costs to run the system.  \* Not commonly used for web hosting.  \* Was “discontinued as of April 21, 2022” (Clover, 2022) | \* Popular choice for hosting web applications.  \* Offers server-based deployment like Apache and Nginx.  \* Open-source with no licensing costs.  \* Works with Amazon Web Services.  Cost of Amazon Web Services is around free to what is “measured in $/hour” (Amazon, 2023) | \* Popular for hosting web applications.  \* Ease of use for familiarity.  \* Offers deployment servers via IIS or Apache.  \* Requires licensing costs to run the system. Cost around “$1069” for a standard edition (Microsoft, 2023). | \* Framework for native development exists on iOS and Android.  \* Requires knowledge of Swift/Objective-C, Java/Kotlin, and development tools.  \* Can communicate with back-end applications on servers using web API.  \* Most are free, but will cost later. (Cooper, 2023). |
| **Client Side** | Moderate expertise. Cost is a little less as long as the user already has a Mac, as operating system updates are free. Otherwise, a little steep when starting from scratch.  Cost variable, upgrades free, but initial investment can be more. Around a basic of “$5999” (Apple, 2023) to “1999” (Apple, 2023) | Expertise and time required for a Linux machine is higher than the other systems, but the cost is minimum.  As long as computers are already used, operating system is “zero cost of entry” (Linux, 2023) | Doesn’t require as much expertise and more people use Windows than other operating systems on. Cost is a little less than Mac when starting out.  Standard costing for a normal PC, Dell more frequently used from my experience. | Has flexibility to the clients and developers for updates on the main operating systems. A little more difficult to implement because more of it is through apps than run through web browsers.  \* Generally, the cost of the device. Use Apple as an example for higher end. |
| **Development Tools** | Can use many common tools, but does require certain tools, such as X-Code, to run certain languages, such as C++. Can commonly use HTML/CSS/JavaScript, and Java. This does require that the supporting libraries are installed.  Cost of tools range from free to around “$5999” (Microsoft, 2023) based on what your level needs. | Has some tools that are not as commonly used, but also has Visual Studio Code, which is commonly used and has a large number of extensions for other languages. Can also use Eclipse for Java.  Mostly free use for some of the majors, such as Eclipse and Visual Studio code. | Most commonly used for development and has the most options for code development. Can use Eclipse for Java, Visual Studio, and Visual Studio Code for development.  Ranges from Free for some to the same as mac up to “$5999” (Microsoft, 2023) depending on the level you need, mostly the higher costs are Visual Studio. | Not as easily used for development, mostly has Monitoring and Logging, Continuous Delivery, and Continuous Integration uses. Apps that do work are: Bitrise, AppCircle, CircleCI, CodeMagic, BuildPan, Kobiton, Buddy, and Esper.  Variable costing from free to around “$699 a month” (Hamilton, 2023) |

## Recommendations

## Recommendations

Please see below for the recommended actions for the architecture.

1. **Operating Platform**: The platform recommended for the server’s operating system is Linux. One of the reasons for this is that Linux “dominates the web hosting market with a whopping **80.9% of operating systems** worldwide” (Escobedo, 2023). This is due to variety of Linux distributions available with different features to enhance the experience. It is also free to low-cost and capable of being run on many cloud services, AWS and Azure being two examples.
2. **Operating Systems Architectures**: Since the recommended platform for the server is Linux, the architecture would best be to follow the system for Linux. For one thing, it “is a modular architecture” (Mitra, 2023). In this architecture, we have the kernel, device drivers, System calls, Libraries, shell, and applications. Below is the breakdown of each layer:
   1. Kernel: This layer is one of the most important layers of the architecture for the operating system. This is where the management of device drivers between both the hardware and the kernel takes place. It also is “responsible for the creation and management of threads of execution” (Wilson, 2023). Since code cannot run without a thread, the kernel ensures that these threads are executed.
   2. Device Drivers: The device drivers exist to allow “the operating system to communicate with various hardware components and provide support for a wide range of devices” (Mitra, 2023). Since we are ensuring that the kernel can manage these, the device drivers in the architecture allows for that hardware to be able to communicate with the kernel and vise-versa.
   3. System Calls: This relates to the application layer of the architecture itself. Without the system calls, the applications would have difficulty “interact[ing] with the kernel and manage system resources like memory, file systems, and device” (Mitra, 2023). This layer helps to manage resources that an application needs and uses to run.
   4. Libraries: Libraries, in this instance, are the system libraries that are the code for common functions. These are already pre-programmed into the system and are not something that the users, whether an IT person, a programmer working with a server, or even a client, would have to do themselves to make something work. Linux’s standard libraries are C, OpenSSL, and GNU C. These help with security, the graphical user interface, and even audio.
   5. Shell: This is most commonly considered the command line console, or the terminal if you’re using a Unix system, which Linux is. This is what allows users to interact with the system using pre-programmed in commands. Example: cd command to change directories.
   6. Application: This layer of the architecture is where the software installed on the system runs and can interact with the other components. The web browsers would be found here and the majority of browsers run on Linux.
3. **Storage Management**: For the storage management, there should be a system that is relational and uses NoSQL. One such NoSQL database is MongoDB and this also runs on Linux. This will handle the data and the scalability aspect of the servers, while handling any unstructured data as well.
4. **Memory Management**: Memory Management is important and is best to utilize the technique of garbage collection. Garbage collection helps to optimize the usage of resources and also increases the performance of the game.
5. **Distributed Systems and Networks**: Since Draw It or Lose it needs to be accessible across various platforms, there needs to be a distributed system and network. The following are recommended methods:
   1. *API Integration*. API Integration is important to allow communication amongst various devices to happen. This is because the API contains the specifications of the different components and how they need to exchange data securely while interacting with each other in a secure manner. As a part of this, we have to include REST.
      1. *REST*: Since there is an API call made by the client through their web browser to the server, there needs to be validation before sending anything back. REST is used as “a standardized way for two applications to communicate” (Juviler, 2023). This means that both the client side’s web browser and the server can understand each other through a set of specific rules that are understood by both sides.
   2. *Message Queuing* is another important element. This is needed for asynchronous communication of the components in the environment. Since we cannot guarantee that all components are available all of the time, this feature allows for the system to retain the messages for the components to receive when available. This helps to keep the game’s functionality running. Using these two elements allows for communication between all platforms and devices intact.
6. **Security**: There are 4 main ways to ensure security:
   1. *Encryption*: The use of Secure Socket Layer/Transport Layer Security keeps communications encrypted across the different platforms to reduce security holes that an attacker can exploit. Typical Advanced Encryption Standards are employed to keep the data at rest and in transit safe and unreadable if the encryption key is not included.
   2. *Security Audits*: Doing security audits and vulnerability assessments helps to find any weaknesses within the system that can be fixed before they become a problem as well as making sure that there are none. This also means having all the security updates installed and using security best practices.
   3. *User Authentication and Authorization*: This is used to verify the identity of the people using the app, as well as the added benefit of ensuring a user name is only used once. This can be done with username/password verification, multi-factor authentication, and even integrating third-party providers for authentication.
   4. *Secure APIs*: as listed in the previous version, using the APIs securely will require authentication. This can be done through access control and using the standard protocol of OAuth 2.0 to make sure that the requests are only allowed through authorized platforms.
   5. *An Added note here that is not a part of the main Security*: there does need to be an understanding that there is a chance people may try to cheat in the game. Detection for cheating may need to be implemented as people can sometimes exploit holes in code to cheat in the game.

**Citations**

Microsoft. (2023). *Microsoft*. Microsoft Store. <https://www.microsoft.com/en-us/store/collections/visualstudio?atc=true&icid=SMB_CP1_visualstudio&rtc=2>

Girdhar, A. (2021, February 23). *What is an Xcode? top features of Xcode: How much does xcode cost*. Appy Pie. <https://www.appypie.com/faqs/how-much-does-xcode-cost#:~:text=Top%20Features%20of%20XCode%7C%20How%20Much%20Does%20XCode%20Cost%3F&text=XCode%20by%20itself%20is%20available,store%20costs%20%2499%20per%20year>.

Microsoft. (2023). *Visual Studio Code Frequently FAQ.* Visual Studio Code. <https://code.visualstudio.com/docs/supporting/faq>

Eclipse. (2023). *Eclipse Public License – V2.0*. Eclipse Foundation. <https://www.eclipse.org/legal/epl-2.0/>

Hamilton, T. (2023). *21 Best Mobile App Development Tools & Software (2023).* GURU99.

<https://www.guru99.com/mobile-app-development-tools.html>

Apple. (2023). *Compare Mac Models*. Apple.

<https://www.apple.com/imac-24/compare/?modelList=iMac-24_2,Mac-studio-2023,Mac-mini-M2>

Linux. (2023). *What is Linux?*. Linux.

<https://www.linux.com/what-is-linux/#:~:text=That's%20right%2C%20zero%20cost%20of,for%20software%20or%20server%20licensing>.

Microsoft. (2023). *Pricing and Licensing for Windows Server 2022.* Microsoft.

<https://www.microsoft.com/en-us/windows-server/pricing>

Clover, Juli. (2022). *Apple Discontinues macOS Server*. MacRumors.

<https://www.macrumors.com/2022/04/21/apple-discontinues-macos-server/>

AWS. (2023). *AWS Lambda Pricing*. Amazon. <https://aws.amazon.com/lambda/pricing/>

Cooper, S. (2023). *The Best Mobile Device Management (MDM) Solutions – Updated 2023.* Comparitech.

<https://www.comparitech.com/net-admin/mobile-device-management-software/>

Apple. (2023). *iPad Air*. Apple. <https://www.apple.com/ipad-air/specs/>

Escobedo, J. (2023, April 20). *Best Server OS for your Website: Linux or Window?*. Liquid Web.

<https://www.liquidweb.com/blog/best-operating-system-for-web-hosting/#:~:text=Ergo%2C%20Linux%20is%20the%20most,than%20on%20a%20Windows%20server>

pp\_pankaj. (2023). *Kernal in Operating* System. Geeks for Geeks.

<https://www.geeksforgeeks.org/kernel-in-operating-system/>

Mitra, N. (2023, May 4). *Linux Architecture*. Scaler Topics.

<https://www.scaler.com/topics/linux-architecture/>

Wilson, K. (2023, August 8). *Interview with Kelly Wilson, Operating Systems Programmer* [Skype].

Gitlin, J. (2023, April 11). *A complete guide to rest api integration*. Workato. <https://www.workato.com/the-connector/rest-api-integration/>

Clement, N. (2023, February 16). *How Message Queues Work in Distributed Systems*. FreeCodeCamp. <https://www.freecodecamp.org/news/message-queues-in-distributed-systesms/>

Juvile, J. (2023, August 9). *REST APIs: How They Work and What You Need to Know*. Hubspot. <https://blog.hubspot.com/website/what-is-rest-api>

NA, N. (2023). *What is Encryption?*. Google Cloud. <https://cloud.google.com/learn/what-is-encryption>

Malik, K. (2022, November 21). *Application Security Audit: AN In-Depth Guide*. StackHawk. <https://www.stackhawk.com/blog/application-security-audit-an-in-depth-guide/>

Moningi, L. (2018, November 23). *Authentication and Authorization in ASP.NET*. C#Corner. <https://www.c-sharpcorner.com/article/authentication-and-authorization-in-Asp-Net/>

NA, N. (2023). *Understanding Authentication, Authorization, and Encryption*. BU TechWeb. <https://www.bu.edu/tech/about/security-resources/bestpractice/auth/>

NA, N. (2019, January 8). *API security*. RedHat.

<https://www.redhat.com/en/topics/security/api-security>

Kirmse, C. (1997, July 07). *Security in Online Games*. Game Developer. <https://www.gamedeveloper.com/disciplines/security-in-online-games>