

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 6**

[**Recommendations 8**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 04/07/24 | Dennis T Sherpa | Addition of information in the Evaluation and Recommendations sections.  Edited the Design Constraints section. |
| 1.0 | 04/19/24 | Dennis T Sherpa | Addition of information in the Recommendations section.  Edited the Evaluation 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)

1. **Client/Customer**: The Gaming Room.
2. **Name of the Game (Client’s Game):** Draw It or Lose It.
3. **Game Introduction**: Draw It or Lose It is loosely similar to the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. Rather than a player drawing images on an easel to help team members guess the puzzle (a phrase, title, or thing), the application will render images from a large library of stock drawings as clues.
4. **Game Instructions (How the Game Is Played):** A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.
5. **Software Design Problem**: The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only.
6. **Solution**: Develop a web-based game for The Gaming Room based on their current game, Draw It or Lose It, which can serve multiple platforms.

## 

## Requirements

The Gaming Room has requested that the following **requirements** be met for the game application “Draw It or Lose It”:

1. A game will have the ability to have one or more teams involved.
2. Each team will have multiple players assigned to it.
3. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
4. Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The **design constraints** for developing “Draw It or Lose It” in a web-based distributed environment are:

1. Programming Language
2. Following the budget:
   * Assemble the team of developers for this project.
   * Provide the hardware and software necessary for success.
3. The web-based game must be operatable on macOS, Windows, Linux, iOS, and Android.

The reasons why the above are **design constraints** and their **implications** on application development are:

1. **Programming Language**
   * Our development team must utilize a programming language bundle emphasizing web-based game development because the correct group of programming languages will fulfill the game’s ideals. Therefore, I highly recommend that the programming language bundle be JavaScript, HTML, and CSS for the following reasons:
     1. They are the most employed combination of programming languages in web development and are considered by many the building blocks of the web.
     2. Quick overview of why:
        1. Hypertext Markup Language, abbreviated as HTML, would create the basic structure of “Draw It or Lose It.”
        2. Cascading Style Sheet, abbreviated as CSS, would specify the various ways different parts of “Draw It or Lose It” appear to users.
        3. JavaScript would be the front-end programming language that builds on top of both HTML and CSS, takes the work done with them, and turns it into a web-based game that a user can interact with.
     3. Fulfills all the necessities to develop “Draw It or Lose It.”
2. **Following the budget:**
   * **Assemble the team of developers for this project.**
     1. This is required because the quality and quantity of developers who can be hired according to the budget will determine the project's outcome.
     2. This constraint is important for design and development because each developer’s preferences and skill level will determine the program's final outcome.
     3. This is the most critical constraint that must be executed with the utmost care because this team of developers will build and develop the program.
   * **Provide the hardware and software necessary for success.**
     1. This is essential because the quality and quantity of hardware and software that can be provided according to the budget will be important in determining the project’s outcome.
     2. This is a constraint for design and development because each piece of hardware and software provided will decide the program's outcome. It will also determine the development experience, impacting the developer’s workflow and thought process.
3. **The web-based game must be operatable on macOS, Windows, Linux, iOS, and Android.**
   * This key factor must be achieved at the end of developing “Draw It or Lose It.”
   * This will shape the whole development process and produce guidelines to be followed and goals to be achieved throughout the development of “Draw It or Lose It.”
   * Ultimately, the development team must produce a program that can be deployed as a web-based game that is operatable on macOS, Windows, Linux, iOS, and Android.

## [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 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.**

**Overview of the UML diagram provided above:**

* There is one package named “com.gamingroom”.
* In the package, there are seven classes. They are ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player.

**The relationships present amongst classes in the UML diagram are:**

* ProgramDriver is directly associated with SingletonTester, where ProgramDriver utilizes SingletonTester to function.
* GameService is associated with Game, where GameService can have zero to many instances of Game.
* Game is associated with Team, where Game can have zero to many instances of Team.
* Team is associated with Player, where Team can have zero to many instances of Player.
* Game, Team, and Player all inherit from Entity.

**The object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently are as follows:**

1. **Polymorphism:**
   * GameService incorporates method overloading by creating two methods named getGame(), where one has an argument named “id” of the data type “long,” and the other has an argument named “name” of the data type “String.”
2. **Abstraction:**
   * ProgramDriver can use the methods of the SingletonTester without having to know how the method operates in SingletonTester.
3. **Inheritance:**
   * Game, Team, and Player all inherit the public methods of Entity, which allows them to call the methods of Entity for their operations without any additional work.
4. **Encapsulation:**
   * The package bundles the classes together, and the classes bundle their respective variables and methods together.
   * Each class's private variables and methods are restricted to outside classes, so only the class containing the private variables and methods can change its state.
5. **Association:**
   * GameService can create zero to many instances of Game.
   * Game can create zero to many instances of Team.
   * Team can create zero to many instances of Player.

## [Evaluation](#_2o15spng8stw)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aspect** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| Server Side | - Mac offers robust server-side deployment methods through tools like Apache or Nginx.  - It provides reliable hosting for web-based applications with scalability options.  - Mac's server deployment may have higher initial costs due to hardware expenses. | - Linux is renowned for its server capabilities and is often used for web hosting due to its stability and security.  - It offers various server deployment methods such as Apache, Nginx, or even custom configurations.  - Linux has minimal licensing costs, making it cost-effective for The Gaming Room. | - Windows provides server deployment options like Internet Information Services (IIS) and Apache.  - Windows servers are known for their user-friendly interface and extensive support.  - It may have higher licensing costs compared to Linux, affecting the client's budget. | - Mobile devices can host web-based applications, but their capabilities depend on the device's hardware and operating system.  - Compatibility issues may arise due to different browsers and screen sizes.  - Mobile hosting may require additional optimization for performance and user experience.  - Client should be aware of potential licensing costs associated with proprietary mobile operating systems or development platforms. |
| Client Side | - Developing for Mac desktop clients involves ensuring compatibility with Safari and other browsers.  - The cost may be higher due to the need for Mac hardware for testing and development.  - Time and expertise required may be moderate, but Mac developers are typically required to be well-versed in web technologies. | - Linux desktop clients primarily use browsers like Firefox and Chrome, requiring compatibility testing across these platforms.  - Cost and time required may be lower due to the availability of open-source development tools.  - Expertise needed may vary, but Linux developers are often required to have strong technical skills. | - Windows desktop clients primarily use browsers like Edge and Chrome, necessitating compatibility checks across these platforms.  - Cost may vary depending on the licensing model chosen for development tools.  - Time and expertise needed may be moderate, with a wide pool of Windows developers available. | - Developing for mobile devices involves ensuring compatibility with various browsers and screen sizes.  - Cost and time may be higher due to extensive testing across different mobile platforms (iOS and Android).  - Expertise required may be significant, especially for optimizing performance on mobile devices. |
| Development Tools | - Development for Mac involves using programming languages like HTML, CSS, and JavaScript for front-end development.  - IDEs like Xcode are commonly used for Mac development.  - Additional tools like Safari Developer Tools aid in debugging and optimization.  - Mac development may require familiarity with Apple's ecosystem and development tools, potentially necessitating specialized expertise within the team.  - While Xcode itself is free, if the development team requires advanced features or support provided through the Apple Developer Program, there may be associated costs. | - Linux development primarily involves using open-source languages and tools like HTML, CSS, JavaScript, Python, or PHP.  - IDEs like Visual Studio Code, Atom, or Sublime Text are popular choices.  - Linux developers benefit from a wide range of free and open-source development tools, which may reduce the need for specialized expertise.  - Linux offers many open-source development tools, reducing the dependency on costly licenses. However, if the development team chooses to use certain proprietary IDEs or tools for Linux development, licensing costs could arise. | - Windows development often involves using languages like HTML, CSS, JavaScript, .NET, or Python.  - IDEs like Visual Studio are commonly used for Windows development.  - Windows development may require familiarity with Microsoft technologies, potentially influencing the composition of the development team.  - Licensing costs may be incurred for proprietary tools like Visual Studio, particularly if the project requires features available only in the paid editions. | - Mobile development requires knowledge of languages like HTML, CSS, and JavaScript for web-based applications.  - For native mobile apps, languages like Swift (iOS) or Kotlin (Android) are used.  - IDEs like Xcode (for iOS) and Android Studio (for Android) are essential for mobile development.  - Cross-platform development frameworks like React Native or Flutter may also be utilized.  - Mobile development may require expertise in both iOS and Android platforms, potentially leading to the need for specialized teams or individuals proficient in each.  - Licensing costs may be incurred for mobile development tools, such as the Apple Developer Program membership for iOS development or additional features in cross-platform frameworks like React Native or Flutter.  - Overall, the mobile platform is not suited to be the development environment of Draw It or Lose It. |

**Recommendations**

1. **Operating Platform:**
   1. Linux is recommended as the operating platform for hosting the Draw It or Lose It web-based application due to its robust server capabilities, minimal licensing costs, and extensive support for web hosting. This choice aligns with the client's requirement for a scalable solution accommodating thousands of players.
   2. Linux's stability and security make it ideal for hosting web applications, ensuring reliable performance and minimizing downtime.
   3. Utilizing Linux, the web-based game “Draw It or Lose It” can be developed to be accessed by all operating systems.
2. **Operating Systems Architectures:**
   1. Linux operating system architecture provides a stable and secure environment for hosting web applications. Its open-source nature allows for customization and optimization according to the specific requirements of the Draw It or Lose It software. Linux's scalability ensures that the application can handle increasing user traffic without compromising performance.
   2. The following is a brief overview of Linux’s architecture from the highest to lowest level:
      1. Application (User Interface)
         1. Applications are the programs that the user interacts with.
         2. User input is provided from the application and output is received on the application.
      2. Shell
         1. The Shell receives user input/commands from the Application and converts it into code that can be interpreted by the Kernel.
         2. The Shell can be categorized into two sections:
            1. Command Line Shell

The Command Line Shell interprets commands typed (in text form) in the terminal, converts it into code understood by the Kernel, and sends the converted input to the Kernel.

* + - * 1. Graphical User Shell

The Graphical User Shell receives input from the computer’s peripheral devices like a mouse or keyboard. Then, converts it into code that can be interpreted by the Kernel, and sends the converted input to the Kernel.

* + 1. Kernel
       1. The Kernel is the core of Linux and the foundation for software that work on top of it. It handles and hides the complexities required for Linux’s proper functioning on the computer.
       2. Linux generally embodies a Monolithic Kernel, which means that the kernel runs as a single program.
       3. Some of its functions are:
          1. Device management: Manages the device’s hardware, including peripheral devices.
          2. Memory Management: Utilizes various complicated memory management techniques to optimize memory.
          3. System calls: Interprets commands from the shell and executes requested service.
    2. Hardware
       1. Hardware is the physical components of a computer like CPU, RAM, and input and output devices, that the Linux Kernel interacts with.
       2. Some components of hardware are:
          1. CPU: The Linux Kernel mainly interacts with the CPU (Central Processing Unit) to execute and perform instructions.
          2. RAM: The Linux Kernel manages the allocation of memory in RAM (Random Access Memory) to ensure all processes are handled accordingly.
          3. Input and Output Devices: Linux receives commands from input devices (example, keyboard and mouse) and sends output to output devices (example, computer screen).
  1. Furthermore, I recommend the client-server architectural pattern to be implemented as the operating platform’s architectural pattern.
     1. In layman’s terms, the client-server pattern comprises of the server listening to requests from clients and providing the requested services.

1. **Storage Management:**
   1. I recommend cloud storage to be utilized as the storage type.
   2. Furthermore, I recommend Amazon Web Services as the cloud storage provider.
   3. Amazon Web Services can effectively handle large volumes of game data and offer efficient data storage and retrieval mechanisms, which ensures that the Draw It or Lose It application can handle large volumes of game data effectively.
   4. Amazon Web Services fulfills the Draw It or Lost It application’s storage requirements and addresses its following considerations and specific approaches:
      1. **Minimum storage required:**
         1. Since there are 200 image files, each approximately 8 megabytes (MB) in size, a minimum of 1.6 GB disk storage is required.
         2. PostgreSQL can store 1.6 GB and more!
      2. **Consider data compression techniques to reduce storage footprint:**
         1. Data compression techniques can help reduce the storage footprint of image files without significantly impacting image quality. By compressing image files, the overall storage requirements for the application can be reduced, freeing up more space for other game assets or data.
         2. PostgreSQL contains this important feature!
      3. **Implement efficient file storage mechanisms for each operating platform:**
         1. Amazon Web Services has an efficient file storage mechanism that can be tailored to each operating platform, ensuring compatibility and optimal performance across various devices and operating platforms
      4. **Design storage system to be scalable for future expansion or updates:**
         1. Amazon Web Services is designed to be scalable, which allows for easy addition of new assets or data without requiring significant redesign or reimplementation. This ensures Draw It or Lose It can grow and evolve without sacrificing performance or user experience.
2. **Memory Management:**
   1. Linux employs memory management techniques such as virtual memory and page caching to optimize performance for Draw It or Lose It. By efficiently managing system resources, Linux ensures that the application operates smoothly even under heavy load conditions. This enhances the overall user experience and prevents performance bottlenecks that may arise due to inadequate memory management.
   2. Linux memory management system is a complex system with various configurable settings.
   3. The following is a brief overview of some of the intricate memory management techniques Linux incorporates:
      1. Virtual Memory
         1. Linux’s virtual memory utilizes a disk as an extension of RAM (Random Access Memory) so that the effective size of usable memory grows correspondingly. Where, the kernel writes the contents of an unused block of memory to the hard disk, so that memory can be used for another purpose. When the original contents are required, they are read back into memory.
         2. Virtual memory provides the system with more memory than the physical memory alone could have.
      2. Caching
         1. It is a memory management technique that stores data in a place so that future requests for that data can be served faster.
         2. Linux uses several caching mechanisms to improve system performance. One of them is page cache.
         3. Page cache
            1. Linux’s page cache is a disk cache that stores page-sized chunk of files in memory to improve Linux’s overall performance by reducing disk reads.

Disk cache is the small amount of RAM embedded on a hard disk drive, used to store frequently accessed data.

* + 1. Memory allocation
       1. Linux contains a memory manager, which is responsible for allocating memory.
          1. When a process (action taking place in the computer) requests memory, the memory manager allocates a suitable block of memory to fulfill the request.
       2. The memory can be allocated in multiple ways.
          1. For example, it can take an available space of physical memory and store it with new data from the process.
    2. Huge Pages
       1. Linux’s huge pages utilizes a larger memory block than the default page size to improve the operating system’s performance and optimize memory.
       2. They will help reduce the pressure on the Translation Lookaside Buffer (TLB) and lower the overhead of managing memory in Draw It or Lose It’s RAM.
          1. The Translation Lookaside Buffer (TLB) is a memory cache that stores the latest movement from virtual memory to physical memory.
    3. Zones
       1. There are hardware limitations, which is why Linux cannot treat all page as identical. This is why the Linux kernel groups memory pages into zones according to their possible usage.
       2. Linux partitions the system’s pages into zones to have a pooling in place to satisfy allocations as needed.

1. **Distributed Systems and Networks:**
   1. I recommend implementing RESTful APIs to accomplish communication between various platforms.
      1. RESTful API is an interface that two computer systems utilize to exchange information securely over the internet.
         1. REST (Representational State Transfer) is a software architecture that enforces conditions on how an API should function.
         2. An API (Application Programming Interface) defines the rules that one must follow to communicate with other software systems.
   2. By implementing a distributed architecture, such as RESTful APIs, it will facilitate communication amongst various platforms. RESTful APIs accomplishes this as follows:
      1. The fundamental function of a RESTful API is similar to browsing the internet.
      2. The following are the general steps for any REST API call (a communication between a client and server that utilizes the HTTP protocol to exchange data):
         1. The client sends a request to the server in accordance with the API documentation.
            1. For example, a user types “a photo of a Tux the penguin” and presses enter on the search bar on Chrome.
         2. The server authenticates the client and confirms they are allowed to make that request.
            1. Continuation of the previous example, Chrome checks whether the user is a bot and allows them to proceed only if they are proven to be not a bot.
         3. The server receives the request and processes it internally.
            1. Continuation of the previous example, Chrome’s software searches for images of Tux and websites containing an image of Tux.
         4. The server returns a response to the client. If successful, the response will include the information the client requested.
            1. Continuation of the previous example, Chrome returns images and websites containing an image of Tux.
   3. RESTful APIs set standardized communication protocols, that will allow the Draw It or Lose It application to seamlessly interact with different client devices, ensuring consistent gameplay experiences across desktop and mobile platforms.
   4. This approach enhances scalability for the future of Draw It or Lose It.
2. **Security:**
   1. Linux can be considered the most secure operating system by design because it has built in security features that prove stronger in most categories than other proprietary operating systems, like Windows or Mac.
   2. Linux implements various user protection and security measures for its servers and clients. The following are methods that explain how Linux provides client and server security:
      1. Linux provides robust security features, include built-in firewalls and user permission controls, which provide a strong defense for the server and client against cyber threats.
      2. Linux leverages encryption and secure communication protocols, which allows the Draw It or Lose It application to protect client information across platforms.
      3. Linux being an open-source operating system has a peculiar advantage
         1. Linux’s source code undergoes constant, thorough review by its intellectual, global open-source security community.
         2. When it comes to discovering security bugs, there is no team of developers that can match the strength of the worldwide community of Linux security user-developers who are highly capable and deeply invested in maximizing Linux’s security.
         3. This community alone should provide server and clients relief by knowing that they have a security team (comprising the world’s top security developers) that can tackle various levels of security threats!
      4. Linux’s User Privilege Model
         1. Linux greatly restricts root access through a strict user privilege model.
         2. On Linux, a superuser owns all the privileges, and ordinary users are only granted the bare minimum permissions to accomplish their tasks.
         3. These inherent restrictions act as a key defense against server compromise and attacks on the Draw It or Lose It application’s client security.
      5. Linux is highly flexible and configurable, so Linux’s security can be configured to deal with any and all threats to the server and clients.
   3. Methods to enforce security:
      1. Authentication
         1. It is the process of verifying the identity of a user, process, or device before they are granted access to digital resources.
         2. By authenticating the client, the client can be authorized accordingly.
         3. This will protect the application from fraudulent clients who claim to be somebody they are not.
      2. Authorization
         1. It is the process of giving a user permission to access a resource or function.
         2. After authenticating the client, authorization will allow The Gaming Room to give the client permission to access resources and data at their authorized level.
         3. This will safeguard confidential information and the application’s safety.
   4. Security maintenance recommendation:
      1. Regular security updates and patches should be applied to mitigate potential vulnerabilities and ensure compliance with industry standards.
      2. Implement secure coding practices and conduct regular security audits because this will further enhance the application's resilience to security threats, which will instill confidence in its security measures.
      3. In accordance with future types of security threats, configure the Linux’s security environment to be protected from the security threats of that present.