

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_gjdgxs)

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/15/24 | Frankie Lawrence | Initial documenting of software design template. |

**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](#_35nkun2)

The purpose of this document is to outline the design of The Gaming Room’s Draw It or Lose It. This application draws loose inspiration from the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. In this application, images from a photo library will be rendered, and an active team will have the opportunity to score points by guessing the image correctly. If the active team fails to guess within their time allotment, the opposing team will have a brief opportunity to guess. After four rounds of play, the team with the most points will be declared the winner.

## Requirements

* **Puzzle/Photo Library Management**:
  + The game must access a large library of stock drawings to provide puzzle clues.
* **Platform Compatibility**:
  + The game must support **web-based functionality** and be accessible across **multiple platforms**, transitioning from the current Android-only app.
* **Round Timing and Gameplay**:
  + Each round must last **one minute**, with the drawing fully rendered by the **30-second mark**.
  + Teams have **15 seconds** to guess the puzzle if the primary team fails.
* **Environment Setup**:
  + The client requires guidance on setting up the environment for developing and running the application.

## [Design Constraints](#_1ksv4uv)

* **Business Constraints**
  + Platform Expansion - The application is moving from android-only to a web based solution. This will be a large step for the company to make.
  + User familiarity - Web based version must retain similar functionality and end user experience as the existing android version.
  + CTS is providing The Gaming Room (TGR) the initial environmental setup, as such, CTS must provide good documentation for future deployments and maintenance.
* **Techincal Constraints**
  + Parent company of Creative Technology Solutions (CTS), Southern New Hampshire University has asked all CTS applications to be designed in Java.
  + The web application requires an active internet connection. It will utilize web sockets to ensure multiple players remain connected to the game server and receive real-time updates concurrently.
  + The application will ideally require a database storage system to facilitate the efficient uploading, removal, and management of photos used as game hints.
  + The application must be compatible with multiple platforms, including web browsers on desktop and mobile devices, ensuring a seamless user experience regardless of the device.
  + As the application comes online, security systems should be in place to protect company and player information.
  + The application will utilize an iterator design pattern to parse through teams and players to ensure players and teams cannot have the same identity.
  + The application will utilize a singleton design pattern to ensure only one instance of a game can be active at a time.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

This UML diagram outlines multiple key relationships and object oriented principles that will help organize the application’s code base.

The primary entry point to the application is the **GameService class.** The **GameService class** implements the **Singleton** design pattern ensuring that only one instance of **GameService** exists at runtime to manage game logic and data. It hosts multiple instances of the Game classand holds many methods for retrieving game and player data.

The next primary class of the application is the **Entity class**. This class demonstrates inheritance and polymorphism.Game, Team and Player classes all inherit directly from the **Entity class.** **Entity class** provides these child classes with unique identifiers and accessors. The unique identifiers in the **Entity class** will help our Itertator pattern parse through each of the following to ensure each child class instance is unique.

The **Game class** as mentioned previously is hosted by the GameService class and hosts multiple teams playing the game.

The **Team class** has the ability to add multiple players to an instance.

The **Player class** will be the class that acts directly with the game functionality once that portion of the application gets built out.

During development the **SingletonTester class** hosts the majority of our unit tests.

Lastly, the **ProgramDriver class** acts as our main() function for the program.

**"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](#_z337ya)

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** | Higher hardware and licensing costs. MacOS is found to be very reliable and secure, but does not possess the influx of tools for enterprises and community support that Linux offers. | Open-source-support, highly scalable with very little to no licensing costs. Linux is widely adopted for hosting web applications so much community support will exist. | Windows' main selling point is ther .NET framework which offers great enterprise tools and support. However like MacOS there are significant licensing costs. | Mobile devices are not often found suitable for server-side operations. Heavily limited by their hardware resources which are generally insufficient for handling the typical demands of a web server. |
| **Client Side** | **Cost** Moderate;  developers versed in MacOS browsers (Safari, Chrome).  **Time** May require additional testing for Safari compatibility.  **Expertise** the team needs to ensure a consistent macOS experience across all devices and browser standards, HTML, Javascript, and CSS responsiveness. | **Cost** Low;  community support for browsers like firefox and chromium widely spread.  **Time** Least time consuming, since Linux browsers need to follow stricter web standards.  **Expertise** Should be the easiest expansion to achieve. Ensuring browsing standards of HTML, Javascript and CSS should work across all distributions. | **Cost** Moderate;  Will require robust testing on the existing versions of Windows that are still active.  **Time** Testing time can be lengthy due to the range of browsers supported.  **Expertise** Developers need to ensure a consistent experience across the different Window devices and browsers | **Cost** possibly High;  developing for Android only will require less than for both Android and iOS. iOS development may be locked behind specific development enviroments.  **Time** High;  ensure platform compatibility and CSS and touch screen optimizations for various screen sizes.  **Expertise** Developers need to understand features like touch interactions and adaptive layouts. |
| **Development Tools** | Existing tools such as Eclipse don’t have native support on macOS. toos like **Xcode** may be required for Java-based development. This may add to costs for proprietary software. | Very flexible with a variety of IDEs for Java based development. We can also continue using **Eclipse** if we so choose. | **Microsoft’s Visual Studio** is commonly used but may have some licencensing costs like MacOS. **Eclipse** also works great on Windows. | While not necessary, tools like **Android Studio** and **Xcode** can be immsenely helpful for mobile-development. However this requires a little bit extra familiarity with both environments and their SDK’s and APIs |

### Recommendations

To optimize the performance, scalability, and security of *Draw It or Lose It*, the following solutions are recommended:

*All Chapter references reference the following text:*

Silberschatz, A., Galvin, P. B., & Gagne, G. (2009). *Operating system concepts* (8th ed.). John Wiley & Sons.

**1. Operating Platforms**To support a scalable and resilient gaming experience, we recommend using a combination of stateless distributed file systems (DFS), serverless computing, and Linux-based operating systems.

* **Linux-Based Operating Systems**: Linux is ideal for serverless environments due to its lightweight, open-source architecture, and extensive compatibility with cloud platforms like AWS Lambda. Its scalability, robust security features, and cost-effectiveness make it well-suited for distributed and modular applications.
* **Distributed File Systems (DFS)**: A stateless DFS (e.g., NFS) avoids reliance on persistent server-side states, simplifying failure recovery and enabling many-to-many relationships between clients and servers. This architecture enhances resilience and scalability, as outlined in Chapter 10, Section 10.5.2.
* **Serverless Computing**: Serverless functions (e.g., AWS Lambda) complement DFS by offering modular, event-driven execution. These functions dynamically scale to workload demands and operate on a cost-efficient, pay-per-execution model.

**Key Benefits:**

* Simplifies recovery and reduces operational costs.
* Scales to meet dynamic user demands effectively.
* Leverages the flexibility, security, and efficiency of Linux systems.

**References:**

* Chapter 10, Section 10.5.2 (Remote File Systems).
* **Bashir, F. (2019, July 5).** What is serverless architecture? What are its pros and cons? *freeCodeCamp*.<https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/>

#### 2. Operating Systems Architectures

For structured data management, **SQL databases with lookup-based storage mechanisms** are recommended.

* **Efficiency**: SQL databases (e.g., MySQL, PostgreSQL) leverage indexing and logical block storage to minimize query execution times. This approach aligns with serverless architectures for rapid and efficient data handling.
* **Lookup Storage**: Hash-based indexing enables faster metadata retrieval and directory management.

**Key Benefits**:

* Supports high concurrency and scalability.
* Enables rapid, structured data retrieval.

**References**:

* Chapter 12, Section 12.1.1 (Magnetic Disks).
* Chapter 11, Section 11.3.2 (Hash Tables).

#### 3. Storage Management

To ensure efficient memory usage, **virtual memory with demand paging** is recommended.

* **Virtual Memory**: This approach separates logical and physical memory, allowing processes larger than physical memory to execute efficiently.
* **Demand Paging**: Only required pages are loaded into physical memory, reducing memory waste and optimizing I/O operations
* **Copy-on-Write (COW)**: COW minimizes memory duplication during process creation by sharing pages until a write operation occurs, enhancing dynamic environments.

**Key Benefits**:

* Supports multitasking and dynamic workloads effectively.
* Reduces memory overhead while optimizing resource utilization.

**References**:

* Chapter 9, Sections 9.1, 9.2, and 9.3.

#### 4. Distributed Systems and Networks

A robust **distributed system architecture** is essential for ensuring reliable communication across platforms.

* **Coordination Mechanisms**: Logical clocks and event ordering ensure causally consistent actions, critical for multiplayer scenarios.
* **Network Structure**: Using both LAN for low-latency communication and WAN for broader connectivity provides users with a consistent experience across devices.
  + LAN (Local Area Network)
    - Players on same network can create and join private lobbies without requiring internet
    - Approach reduces latency
  + WAN (Wide Area Network)
    - Ensures global connectivity, allows for people across the globe to interact
    - The central server still handles requests for images and other data in sync.

**Key Benefits**:

* Maintains reliable communication across platforms.
* Enhances scalability and platform consistency.

**References**:

* Chapter 16, Section 16.3 (Network Structure).
* Chapter 18, Section 18.1.2 (Event Ordering).

#### 5. Security

To protect user data and ensure secure communication, a **multi-layered security approach** is recommended.

* **Access Control**: Role-Based Access Control (RBAC) and the principle of least privilege minimize the risk of unauthorized access by limiting permissions.
* **Encryption**: End-to-end encryption (e.g., TLS) protects transmitted data, while AES secures stored data.
* **Authentication**: Multi-Factor Authentication (MFA) enhances security by combining passwords, devices, and biometrics.
* **Network Protection**: Measures like firewalls, intrusion detection systems, and cryptographic timestamps prevent threats such as replay and DoS attacks.

**Key Benefits**:

* Builds user trust through robust data protection.
* Mitigates a wide variety of potential threats.

**References**:

* Chapter 14, Section 14.2 (Access Control).
* Chapter 15, Sections 15.1, 15.3, and 15.4.