

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View**](#_heading=h.4d34og8) **4**

[**Domain Model**](#_heading=h.2s8eyo1) **4**

[**Evaluation**](#_heading=h.17dp8vu) **5**

[**Recommendations**](#_heading=h.26in1rg) **7**

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/25/24 | Brady Weber | Update Executive Summary, Design Constraints, and Domain Model. |
| 1.1 | 06/09/24 | Brady Weber | Update the Evaluation section. |
| 1.2 | 06/23/24 | Brady Weber | Update the Recommendations section. |

## [Executive Summary](#_heading=h.35nkun2)

The Gaming Room has developed a game, Draw It or Lose It, that is currently available on an Android app only. They want to develop a web-based version that is available to multiple platforms. The current development team for The Gaming Room isn’t confident in setting up the environment for multiple platforms, so the company has hired us as a consultant to help streamline the process. We will help them achieve this by providing a team of developers who are proficient in web development languages that will transition the game from their current Android development language into a web-based development language. Based on the current frameworks and structures that the game is developed on, this process may be more resource consuming than the client is anticipating. We may have to recreate the entire game in a new language.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_heading=h.1ksv4uv)

Technical Constraints

* Platform Compatibility: The app must run on MacOS, Windows, Linux, Android, and iOS.
* Browser Compatibility: The app must run on multiple browsers i.e. Safari, Chrome, Firefox.
* Cross-Platform Integration: The app must be able to share user data across all platforms.
* Web-based Limitations: The web-based app must be able to run as efficiently on a browser as it does on a mobile app.
* Database: The app must utilize a database that integrates with all of the platforms.

Business Constraints

* Budget: The project must not exceed the client’s allotted budget.
* Team Compatibility: The client has an existing team for the project that we will have to work with in order to develop the software.
* Cross Training: This project will require cross-training for the existing team in order to effectively work with our development team.

Operational Constraints

* Data Retention: The app must retain all user data as long as the user account is active.
* User Environment: The app must be compatible with different screen sizes and user input methods.

Implication of Constraints

Due to the technical constraints, we must utilize a language, a database, and a hosting service that is able to be integrated with all of these different environments. The program must also run efficiently on a browser’s limited storage capabilities hence a heavier reliance on a database. The business constraints identify two conflicting constraints, budget and cross training. In order for the existing team to help develop the web application they may need to be cross trained in the language that we use. This will increase the project costs. For Operational Constraints, since the user environment will vary, the software will have to consider different types of user input and the user interface will also have to be reconfigured for web based versions.

## [System Architecture View](#_heading=h.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](#_heading=h.2jxsxqh)

This program utilizes inheritance by having the Game, Team, and Player classes inherit the Entity class. This ensures that all of these classes have at least the base member variables and functions of the Entity class. This helps to fulfill the software requirements of each Game, Team, and Player has a unique name and ID. By having all of the code grouped into different classes this program has good portability and maintainability. This is important for the program to be able to be run on multiple platforms since different elements of the code can be changed to adhere to certain platform requirements without having to edit the entire program. Encapsulation includes the use of private variables to protect the integrity of the program which can be important for the software’s security especially considering the application will be developed for the web which will require more security than a mobile app. The use of private variables is also a key component of the Singleton pattern which is how we will implement the software requirement of only having one instance of the game in memory at any one time. This program also uses the principle of polymorphism. The Game, Team, and Player class override the toString method in order to give specific class information when this method is used. This can help provide uniformity to the User Interface when displaying the given information.

**"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](#_heading=h.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** | MacOS Server has a small initial cost, but may not scale well with thousands of users.  Provides useful tools to developers and integrates well with other Apple devices. | Free and open source server options with plenty of scaling capability and paid options for better support.  Requires more technical expertise to set up and maintain.  May have hardware compatibility issues. | Familiar development environment with many tools and resources.  Very capable scaling potential.  High server costs. | Not designed for hosting a centralized game server.  May utilize a peer-to-peer network for certain features. |
| **Client Side** | High initial investment.  Moderate development and deployment time.  Moderate expertise needed, but prior Unix knowledge helps.  Extra testing necessary for Safari web browser. | Lower costs.  More time required for initial set up. Faster deployment time.  Higher level of expertise needed.  Extensive web browser testing needed due to the large quantity of browsers available. | Moderate costs.  Generally faster development and deployment times.  Lower to moderate expertise needed.  Extra testing needed for Edge web browser. | Moderate costs.  Longer development and deployment times.  High expertise needed for mobile development.  Extensive web browser testing to ensure quality across different screen sizes with different browsers. |
| **Development Tools** | Swift and Objective-c development languages used.  Xcode IDE used.  A mac-specific development team will likely be required. The Apple Developer Program and Xcode have licensing costs. | Capable of running multiple languages and IDEs.  An extra team would likely not be needed.  No extra costs would be incurred depending on the language and IDE chosen. | C# language.  .NET framework.  Visual Studio IDE.  An extra team may be needed depending on the current team’s comfortability with previously stated tools.  Visual Studio has a licensing fee. | Swift/Objective-c languages needed for iOS.  Java/Kotlin languages needed for Android.  Potential use of React Native to cut down on multiple languages.  Multiple teams would likely be required if developing iOS and Android separately.  If React Native is used, it may only require one additional team if the current team is unfamiliar.  React Native is open-source and free to use.  The Apple Developer Program has a licensing fee.  Both Android and iOS app stores may have distribution fees. |

## 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**:

I recommend the Windows operating platform for The Gaming Room to expand Draw It or Lose It to other computing environments because of its compatibility and scalable servers. Windows has powerful development tools and IDE’s like Azure and Visual Studio and easily supports game development engines like Unity, Unreal, and React. It also seamlessly provides an additional platform in Xbox through Universal Windows Platform if the client chooses to expand to consoles as well. Windows servers are outstanding for creating a scalable and secure game. The cost of Windows may be higher than other choices, but considering its capabilities, I believe it to be worth it for expanding the game.

1. **Operating Systems Architectures**:

The Microsoft operating platform architecture has a kernel mode and a user mode. Windows NT Kernel is used for the kernel architecture that focuses on low-level core services and implements microkernel principles. The user mode provides the environment that users interact with and run applications. Windows also utilizes processes and threads to improve performance. The New Technology File System is used as its primary file management system.

1. **Storage Management**:

The storage management system that I recommend using with Windows is Microsoft Azure Storage because of its scalability, cost, and integration capabilities. Azure Storage can scale with the size of the player base and its costs are reflected accordingly. Since it is a cloud-based system, no local servers have to be used which reduces the initial costs as well. It also integrates well with other Microsoft Azure tools and game engines like Unreal and Unity.

1. **Memory Management**:

Windows operating platform uses memory management techniques like garbage collection, prefetching, and virtual memory. Garbage collection helps developers to avoid memory leaks which can lead to the software crashing. Prefetching is when the program loads data before it is actually executed in order to reduce startup times. Virtual memory allows the system to exceed its physical memory capabilities. This can also prevent issues like overflow that result from trying to constrain more data to a space than that space allows.

1. **Distributed Systems and Networks**:

In order for Draw It or Lose It to communicate between various platforms, the program will implement a RESTful API that utilizes a client-server design pattern and basic HTTP methods such as GET, PUT, and DELETE. The client-server pattern allows for a separation of concerns in order to develop and maintain each independently which makes it easier to diagnose issues and perform maintenance. The HTTP methods provide simple and reliable methods for the client to access server resources. By utilizing a centralized database, the game has more consistency across all platforms. A principle of REST is ensuring the program is cacheable; caching resources can reduce the load on the server which can promote connectivity across the network and increase uptime. By utilizing a cloud-based server like Microsoft Azure, the network can scale during heavy traffic to help decrease load times. Also using load balancing can help to even the workload across the distributed system. Monitoring and logging can help to identify issues and troubleshoot bugs during outages to get the system back up.

1. **Security**:

In order to protect user information on and between various platforms, I recommend using a combination of encryption, authentication, authorization, input validation and sanitization, and firewalls. Encryption should be implemented at rest and for transmitted data. In order to encrypt transmitted data HTTPS should be used for the client-server communications; at rest, Azure Storage provides encryption for its database. Using a tool like OAuth 2.0 is a good way for users to be properly authenticated. Role-Based Access Control should be implemented to ensure users are authorized to execute certain tasks. Input validation and sanitization can be used to prevent injection attacks. Windows Firewall is a native resource that provides network security and controls traffic.