

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 4.0 | 06/14/2024 | Seth Richey | Edited Recommendations. |

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

The gaming room desires to expand the reach of their popular android game, “Draw It or Lose It”. They want a web based version of the game to increase accessibility. They also want to increase the number of players by adding teams to the game.

To this end the proposed idea is to develop a scalable framework for the game focused on user engagement and compatibility across platforms and devices. The game will focus on intuitive design, and a singleton instance of the game in memory.

## 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)

The program needs to allow real time interaction between multiple users across multiple platforms; this requires, Servers to handle multiple game instances simultaneously, low latency transferring of data between players, and cross platform compatibility.

This requires an efficient backend as well as an easily scalable UI.

## [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)

Classes:

Game inherits from Entity and aggregates Team, each game can have multiple teams.

Team inherits Entity and aggregates Player, allowing each team to consist of multiple players.

This ensures that games, teams, and players are all unique.

Object-Oriented:

Inheritance allows programmers to save time by not having to retype code multiple times, allowing Game, Team, and Player to share common properties through Entity.

Encapsulation ensures that game state is managed internally, exposing only necessary methods for interaction.

Aggregation represents the "has-a" relationship between games and teams, and teams and players, organizing the model into a coherent structure.

**"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** | Flexible terminal commands for server configuration, access, and modifications  Characteristics:  Widely used in web hosting  Advantages:  Upgradable  Offers diverse options for various web hosting needs  Disadvantages:  Less favored for web hosting services | Similar to Mac but more budget-friendly  Characteristics:  Secure and highly favored  Advantages:  Security vulnerabilities are identified early, making it the top choice for web hosting  Disadvantages:  Challenges in finding applications that meet specific web hosting needs | Abundance of software compared to other operating systems  Characteristics:  Dominates over other platforms  Closed platform  Advantages:  Requires high resources  Faster loading times  Offers high comfort  Disadvantages:  Vulnerable to viruses  Limited technical support | Preferable for the server to be stationary and trackable in a single location. Other devices have superior specifications.  Characteristics:  Widely favored and highly portable  Advantages:  Greater reach  Improved compatibility  Cost-effective  Disadvantages:  Limited compatibility with different smart mobile devices  Weak security measures |
| **Client Side** | To ensure the application is compatible with all web browser platforms and mobile devices, a moderate level of expertise and time commitment is necessary. The costs are comparable to those of Windows. | To ensure the application is compatible with all web browser platforms and mobile devices, the development process demands a high level of expertise and time investment while keeping costs to a minimum. | To ensure the application is compatible with all web browser platforms and mobile devices, the development process demands minimal expertise and time, with costs comparable to those of Mac. | Allows clients and developers the flexibility to view updates from any location. Slightly more challenging to implement compared to other devices. |
| **Development Tools** | When using Macs for programming, Swift is a popular choice. Macs support all programming languages, including HTML/CSS/JavaScript, along with libraries for frontend development and general-purpose languages like Java, Python, PHP, and Ruby. Additional tools like Notepad++ can also be integrated for a seamless coding experience. | Linux is compatible with Visual Studio, Eclipse, and Notepad++ among other user-friendly tools. It supports a wide range of languages and tools, including HTML/CSS/JavaScript, along with libraries for frontend and general-purpose programming languages such as Java, Python, PHP, and Ruby. | Linux is less complex to use compared to other operating systems but offers compatibility with a wide range of programming languages and tools. Popular development environments like Visual Studio and Eclipse are available, along with user-friendly tools like Notepad++. Supported languages include HTML, CSS, JavaScript, as well as libraries for frontend and general-purpose programming such as Java, Python, PHP, and Ruby. | You have the ability to develop numerous applications using Android and Swift. These languages and software are compatible with all three types of machines. Supported languages include HTML, CSS, JavaScript, as well as libraries for frontend and general-purpose programming such as Java, Python, PHP, and Ruby. |

## 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**: Operating Platform: The recommended operating platform for The Gaming Room to expand Draw it or Lose it to other computing environments is a cloud-based server platform. A cloud-based server platform provides scalability, flexibility, and cost-effectiveness that meets the client's requirements. Cloud-based platforms like Amazon Web Services, Microsoft Azure, and Google Cloud Platform offer a range of services that can be used for storage, computing, and networking.
2. **Operating Systems Architectures**: Windows offers services utilized by all Windows-based applications that allow applications to display a Graphical User Interface (GUI) while accessing system resources and beyond. These applications also encompass graphics and multimedia, messaging, and web services. These services can be accessed either through a user account or a dedicated server.
3. **Storage Management**: The recommended storage management system for the Draw it or Lose it software is a relational database management system (RDBMS). RDBMSs provide the ability to store and retrieve data in a structured and efficient way. They also provide features like data backup, recovery, and security.
4. **Memory Management**: When developing this game, you will have to establish a database or library containing numerous images. Memory allocation facilitates the convenient storage of images outside the standard picture directory. This enables you to maintain your entire project in a more secure location on your computer, even when interacting with your Integrated Development Environment (IDE) and accessing files from it to build the game.
5. **Distributed Systems and Networks**: Due to the variations in each operating system, I explored methods to release the game for compatibility across all devices. I discovered Develop 4, a tool that facilitates cross-platform game development. It serves as an Integrated Development Environment (IDE) compatible with various devices. After creating the game, you can easily export the game file to web, iOS, Android, and other platforms to enable cross-play, which aids in managing dependencies. To mitigate issues such as downtime or connectivity problems, the company must ensure that their servers are robust enough to handle high player volumes and have backup power in place for instances of power outages.
6. **Security**: Windows includes built-in security protection software, but it is advisable to use additional security measures to safeguard user data and information. While Windows provides protection on the machine, it scans for malware, viruses, and security threats in real-time. The system automatically updates to address evolving threats and ensure the safety of the system and user information.