

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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 7/15/2023 | Dani Allen | Initial version |

**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 has an existing android app game called Draw It or Lose It that they wish to develop into a web-based game that can serve multiple platforms. In this multiplayer game, teams compete to guess what is being drawn by the application. It renders images from a large library of stock drawings as clues. Over a thirty second interval, the application renders a drawing at a steady rate. If the team cannot guess the puzzle before the 30 seconds are up, the remaining teams can offer one guess to solve the puzzle with a 15-second time limit. Creative Technology Solutions is contracted to assist with facilitating the development of the web-based version of their Android game app.

The client requests that the game be able to have one or more teams involved, with multiple players able to be assigned to each team. The game and team names must be unique to allow users to check whether a name is already used when choosing their team’s name. Only one instance of the game can exist in memory at any given time. This can be accomplished by using unique identifiers for each game, team, or player instance.

## Requirements

* Web-based version of the Android app Draw It or Lose It
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* 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)

* **Compatibility** – The game should be compatible with different web browsers and OS platforms to ensure a consistent user experience across devices.
* **Visual Design** – Web-based version should retain a similar look and feel to the Android app version of the game to ensure a consistent user experience across devices.
* **Performance** – The application should be able to generate images steadily.
* **Unique Instances** – Game, Team, and Player names must be unique.

## [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.*

* Front End – interface
* Backend – game logic
* Database – information storage such as games, teams, players

## [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.**

The ProgramDriver class contains the main method to initiate the code and is associated with the SingletonTester class. This test class validates that the instance names created by the code are unique; this class exists for testing purposes. The Entity superclass is a base class from which Game, Team, and Player inherit common attributes and functions. An association is shown between GameService and Game, Game and Team, and Team and Player. For an instance of GameService, zero to many Games can exist. The Game class is an instance that can have multiple team instances associated with it, and those teams can have multiple instances of players.

## [Evaluation](#_2o15spng8stw)

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** | User-Friendly interface with high performance and reliability for hosting web-based applications. Limited options for server hosting and hardware options may be more expensive than Linux and Windows. | Open-sourced OS that offers stability and efficiency and can be installed on a wide range of hardware. Cost-effective and has a vast selection of server-related software and tools with excellent community support. | Widely used OS with extensive development tools and resources for building web applications. Licensing costs are higher than Linux and servers may require more resources for increased performance levels. | Reaches a vast audience for iOS and Android devices. Mobile devices have limited processing power and memory compared to desktop servers. |
| **Client Side** | Popular among developers and designers for its user-friendly interface and design-oriented applications. Supports a wide range of web browsers. Hardware and software can be expensive, and the user base is smaller than Windows and Mobile devices. | Cost-effective and open-sourced making it free to use and stay on budget. Diversity of versions can lead to compatibility issues. User experience and visual design might not be as refined as Mac or Windows. | Dominates the desktop market, user-friendly, and familiar to a large audience. Extensive support for web tools and IDEs. Offers compatibility with various web browsers. Security should be considered due to the widespread use of Windows, making it a target for cyberattacks. | Widely used for internet browsing and app usage. Varying screen sizes and capabilities require adaptive web design. Can reach a large and diverse audience. Compatibility across different devices can be challenging, requiring in-depth testing. |
| **Development Tools** | Xcode – Apple’s integrated IDE used for macOS and iOS. Supports various languages like Swift, HMTL and C++.  Visual Studio Code – Lightweight, highly customizable, and available for macOS, Windows, and Linux. | Visual Studio Code – Lightweight, highly customizable, and available for macOS, Windows, and Linux.  Sublime Text – popular for Linux known for its speed an extensibility through plugins | Visual Studio – comprehensive tools for web app developing using .NET, ASP.NET, and C#. Robust development for windows-based web applications.  Visual Studio Code – Lightweight, highly customizable, and available for macOS, Windows, and Linux. | Android Studio – IDE for Android app development that can also be used for web app development.  Xcode – IDE for iOS development that can also support web app development.  Visual Studio Code – Lightweight, highly customizable, and available for macOS, Windows, and Linux. |

## 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 would recommend using Windows due to its ability to expand to other computing environments through its ability for a cross-platform approach.
2. **Operating Systems Architectures**: Windows provides a familiar environment for users and has a firm foothold in the desktop application market. It is designed to support various hardware configurations making it quite versatile.
3. **Storage Management**: The Gaming Room can utilize Microsoft SQL Server as a storage management system. Microsoft offers free in depth documentation on their website for training and support.
4. **Memory Management**: Windows handles memory management through multiple features such as paging, virtual memory, and memory caching.
5. **Distributed Systems and Networks**: To enable communication between platforms, APIs can facilitate communication between the front and back end components of the game. WebSocket technology can enable real-time communication between clients and the server.
6. **Security**: Windows offers robust security features such as user access control, firewalls, and security updates. The Gaming Room can implement additional security through encrypted communications (HTTPS) and secure data storage practice to prevent data breaches.