

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 | 01/27/2024 | Serenity Rogers | This revision implements an Entity class that holds inheritable common attributes and behaviors. |

**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 wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently only available in an Android App. In order for the game to serve multiple platforms, we will utilize UI adaptability by designing a responsive user interface that adjusts to difference screen sized and orientations.

## Requirements

* *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.*
* *Cross-Platform development framework for efficient development across iOS and Android.*
* *Scalability design for the mobile app architecture to handle a growing user base.*
* *API Integration to ensure data consistency.*
* *Performance optimization for various devices considering different screen sizes and resolutions as well as implementing caching and data compression strategies for faster loading times.*
* *Push notifications to engage users and keep them informed.*
* *Thorough testing on different devices and operating system versions.*
* *Analytics tools to track user behavior and gather insights for continuous improvement.*

*Business Requirements:*

* *Business strategy to define monetization for the mobile app and establish a plan for customer acquisition and retention.*
* *Brand consistency across the website and mobile apps to ensure the app reflects the company’s identity and values.*
* *Marketing and Promotion.*
* *Multiplayer functionality allowing players to compete or collaborate.*
* *Regular updates to introduce new features, fix bugs, and improve overall game performance.*
* *Accessibility to ensure the game is accessible to users with disabilities.*
* *Legal and Licensing to ensure the development adheres to relevant copyright and intellectual property laws.*
* *Support and Customer service to address user inquiries, acquire feedback, and to fix technical issues as well as provide tutorials to help users understand the game.*

## [Design Constraints](#_2et92p0)

* Platform-specific Guidelines
* Device Fragmentation
* Data Privacy and Security
* Resource Constraints
* Budget and Time Constraints
* Legal and Compliance Constraints

## [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 Entity class is the base class where we will implement the Singleton method. The Player, Team, and Game class will inherit the ID and Name from the Entity class. The Player class will store the player’s name and ID. The Team class will keep a list of player names and id’s and create its own name and ID. Game will be able to store a list of teams as well as create its own name and ID. Game Service class will store a list game and add players and teams to the game dependent on whether they are stored already or not. No team or players repeat. The Program Driver will start the Game Service. The Singleton Tester class will test that the Singleton Method works as designed within the Entity class.

**"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](#_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** | Advantages:  Developer Friendly. Graphic Capabilities. Stability. Security Features.  Weaknesses:  Cost. Customization. Market Share. Compatibility. | Advantages:  Cost-Effective. Security Features. Community Support. Scalability.  Weaknesses:  Hardware may be unavailable. Learning curve. Less polished desktop experience. | Advantages:  Ease of use. Comprehensive dev tools. Commercial Support. Remote Desktop services.  Weaknesses:  Licensing cost. Resource consumption. Server core complexity. | Advantages:  Cross-Platform compatibility. No Installation. Instant updates. Portability. Touch Input.  Weaknesses:  Limited offline functionality. Browser limitations. Network Conditions. |
| **Client Side** | Cost: Ongoing updates may incur costs as well as deployment to different app stores.  Time: Updates may require additional dev time. Designing multiple platforms may extend design phase.  Expertise: UI/UX designer must have expertise in designing for diverse platforms ensuring consistency. | Cost: Integrating the application. Ongoing maintenance. Cross-platform  Time: Ensuring seamless integration may extend development time.  Expertise: Developers need expertise in Linuz system programming. | Cost: Cross Platform initial costs. Integration with Windows-specific features and components.  Time: Updates  and seamless integration may increase development time.  Expertise: Developers must be skilled in C#, C++, .NET, and other commonly used windows dev languages. | Cost: Cross-Platforms may involve higher cost. Separate cost for iOS and Android.  Time: Seamless integration extends dev time. Submitting to app stores may take time. Updates.  Expertise: Devs need expertise in platform-specific APIs and services. Skilled in languages such as Swift, Objective-C, Kotlin, and/or Java. |
| **Development Tools** | Languages: Swift, Objective-C.  IDE: Xcode  Tools: Sketch, Adobe Illustrator | Languages: C/C++, Python  IDE: Visual Studio Code, QT Creator,  Tools: Inkscape, GIMP, | Languages: C#(.NET), C++  IDE: Visual Studio  Tools: Adobe XD or Figma, Microsoft Blend | Languages: Swift, Objective-C, Kotlin, Java  IDE: Xcode, Android Studio  Tools: Sketch or Adobe XD, Figma or Adobe XD |

## 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**: Flutter would be the best recommendation due to its ability to have developers write code only once, and then deploy it on multiple platforms including iOS, Android, Windows, macOS, and Linux, which significantly reduces development time and effort.
2. **Operating Systems Architectures**: Flutter provides a vast set of pre-designed widgets allowing the developer to create a consistent visually appealing user interface across multiple platforms.
3. **Storage Management**: Hive would be the recommended storage management as it is a lightweight and fast NoSQL database specifically for Flutter. Hive also involves minimal overhead and doesn’t require complex configuration or SQL knowledge, making it developer friendly.
4. **Memory Management**: Flutter utilizes Dart, which is a Flutter programming language that employs automatic garbage collection so that developers don’t need to manually allocate and deallocate memory. Flutter also provides tools and extensions for memory profiling which are tools that help developers identify memory usage patterns and potential leaks.
5. **Distributed Systems and Networks**: A client-server architecture can be utilized where the game logic and data are on a centralized server. For real-time communication and instant updates, WebSockets will be enabled to drive communication between the clients and the server. Monitoring and logging tools will be utilized to trach the performance and health of the system using an alerting system and regular audits.
6. **Security**: Data encryption will be implemented for sensitive user data. User authentication will be utilized to ensure only authorized users have access to sensitive information. Security headers will enhance security by preventing cyber-attacks. Security audits and testing will be conducted to identify potential vulnerabilities. Privacy regulations will be adhered to protect user privacy and maintain legal compliance.