

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

Brandon Ellis 2-25-24

## 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 |
| --- | --- | --- | --- |
| 1.5 | 02-25-24 | Brandon Ellis | Updated for recommendation section |

**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 a new online game that works on different devices. They want it to be like the current game "Draw It or Lose It," which is only available on Android right now. The game should show drawings from a collection at a consistent speed for players and support several teams, but only one group can play at a time. Players need to guess the puzzle based on the displayed image at various points in time.

## Requirements

Business:

* *Acquiring a database of images without copyright problems and in a cost-effective manner.*
* *A set budget with room for some flexibility.*
* *Getting the necessary licenses for the operating systems we'll be using.*

Technical:

* Games lasting 5 minutes, with each round lasting 45 seconds to a maximum of 1 minute.
* Compatibility with multiple operating systems.
* Ensuring security.
* Displaying full images every 30 seconds.
* Only one game can be played at a time.

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

## [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)

Below, there are multiple classes that are connected and utilize inheritance and serving as subclasses. The class named Entity functions as the parent class for the Game, Team, and Player classes. Additionally, the GameService class is inherited which lets information pass effeciently. The ProgramDriver serves as the main component and functions as a singleton to instantiate a single instance of the game.

**"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** | Mac is good for hosting web-based software with its sleek design and user-friendly interface. However, it might not be as customizable as some alternatives, and it could be pricier. | Linux is solid for hosting web-based software due to its stability and security features. It's also open-source and free, but it might not be as user-friendly for those less familiar with technical aspects. | Windows is versatile and widely used for hosting web-based software, offering compatibility with many applications. However, it may be more susceptible to security issues compared to some other operating systems. | Mobile devices are convenient for accessing web-based software on the go, but they might face limitations in terms of processing power and screen size. Security can also be a concern, and they may not offer the same level of control as traditional operating systems for hosting complex applications. |
| **Client Side** | Developing software for Mac requires considering the cost, which might be higher due to the specific tools and hardware involved. The time needed might be influenced by the platform's unique features. Expertise in Mac development is essential, and hiring skilled Mac developers might be a bit more specialized. | Developing software for Linux involves considering costs, which can be lower since many tools are open-source. Time might vary, but expertise in Linux development is crucial for efficient coding. The skillset required may be more technical and specific. | Creating software for Windows involves costs that can be influenced by licensing and development tools. Time may vary depending on the complexity of Windows-specific features. Expertise in Windows development is important, and it's generally a more widely available skill compared to some other platforms. | Developing software for mobile devices involves considering costs related to app development tools and testing on different devices. Time can be affected by the need for compatibility across various mobile platforms. Expertise in mobile app development is crucial, and it may require knowledge of specific programming languages for each platform. |
| **Development Tools** | To build software for Mac, developers often use programming languages like Swift or Objective-C. Common tools include Xcode as the Integrated Development Environment (IDE), providing a platform for coding and testing. Additionally, developers might use tools like GitHub for version control. | Building software for Linux involves using programming languages like C, C++, or Python. Developers might work with IDEs such as Visual Studio Code or Eclipse, along with tools like Git for version control. Linux being open-source, various command-line tools are also commonly utilized. | For building software on Windows, developers often use programming languages like C#, C++, or Java. Popular IDEs include Visual Studio, providing a comprehensive environment for coding, testing, and debugging. Version control tools like Git or Azure DevOps may also be employed. | Developing software for mobile devices involves platform-specific languages such as Swift for iOS, Kotlin or Java for Android. IDEs like Xcode for iOS and Android Studio for Android are commonly used. Additionally, tools like Flutter or React Native allow for cross-platform development, using languages like Dart or JavaScript. Version control tools like Git are also essential for mobile app development. |

## 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**: Windows provides a versatile operating platform widely used in the industry, offering compatibility with various applications. Choosing Windows aligns with The Gaming Room's goal of expanding Draw It or Lose It to other computing environments.
2. **Operating Systems Architectures**: Windows operates on the x86 and x64 architectures. These architectures support both 32-bit and 64-bit applications, providing flexibility for running software on different computing environments.
3. **Storage Management**: Windows Storage Spaces offers a scalable and resilient storage management system. It allows for efficient utilization of storage resources and supports various RAID configurations, ensuring data integrity and availability for Draw It or Lose It.
4. **Memory Management**: Windows uses a virtual memory management system, allowing Draw It or Lose It to utilize more memory than physically available. The operating system employs techniques like demand paging and page swapping to efficiently manage memory resources, ensuring optimal performance for the software.
5. **Distributed Systems and Networks**: Multiple users will access server systems requiring a shared database among players. This database, distributed over the network, allows players to interact and share information.
6. Security: Windows provides user account controls, enabling granular control over user permissions. Utilizing strong authentication mechanisms, such as multi-factor authentication, adds an extra layer of user protection.