

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 6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/18/2023 | Alexis Del Rosario | Added executive summary, design constraints, domain model, evaluation, and recommendations. |

**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 contracted Creative Technology Solutions (CTS) to create a web-based version of their popular game Draw It or Lose It. The game is currently only available on Android devices. In the game, teams compete to guess what is being drawn. A picture is pulled from a library of images and one team guesses what the image is until time runs out.

The Gaming Room has requested that the application meets certain software requirements including being able to have one or more teams playing the game at once, each team having multiple players, and making sure that game and team names are different from one another. To meet said requirements, the development of a web-based application being built on top of the Android application would be the best way to go about it. This will ensure consistency and user experience across multiple platforms.

## 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](#_2et92p0)

Certain design constraints may present themselves when designing a web-based application. These design constraints must be addressed during the application development. These are some design constraints that have been identified:

1. Network latency: This causes delays in communication between clients and servers in a distributed environment. To fix this issue, we can optimize the game’s performance and use asynchronous communication protocols.
2. Security: it is imperative that the game application and user data is secure. Implementing secure communication protocols, user authentication and encryption shall help achieve this.
3. Cross-platform compatibility: To make sure that the game application is available on mobile devices, laptops and desktops, we should use responsive web design techniques as well as develop a platform-agnostic application.
4. Multiple users at once: The application must be able to handle a big number of concurrent users at once. To achieve this, we should use architecture patterns such as load balancing and caching.

## [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 Domain Model is represented by the UML class diagram for the Draw It or Lose It game. It describes relationships, objects and attributes. The base class, known as Entity.java, holds the shared attributes and behaviors for all other classes. Game.java represents a single instance of the game and can have one or more teams. Team.java represents a team and can have more than one player assigned to it. Player.java represents a player who belongs in a team. To ensure that only one instance of the game exists in memory at a certain time, each class has identifiers like gameID, teamID, and playerID. The UML class diagram also uses several features commonly used in object-oriented programming such as association, encapsulation and inheritance.

**"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** | Popular in web hosting. The terminal commands are flexible in configuring the server, accessing and making changes | Open-source OS that is free and can be modified. Highly secure. Many web development frameworks and tools available. May have limited commercial support options compared to other OS. | OS developed by Microsoft. Known for how easy it is to use and many users use it for personal use. Has a high compatibility with many software and hardware. Can be more expensive than other OS | Mobile devices don’t have as much processing power and memory compared to desktops and laptops. They can be used for edge computing, which allows fast response times. May be more susceptible to cyber attacks and have more security vulnerabilities |
| **Client Side** | Typically, more costly than other OS. May take longer to support Mac clients due to requiring additional development and testing for compatibility. Special expertise required due to unique development tools and programing languages. | May be more costly due to having a fragmented market. May take longer than other OS due to having many unique Linux distributions. Specialized expertise with knowledge of Linux libraries. | Not as much time or expertise required compared to other OS. Cost is moderate. | Cost, time and expertise will vary depending on application structure. Cost will increase with more clients supported. Time needed to make sure application can run on multiple types of devices. Will require expertise in mobile OS. |
| **Development Tools** | * Objective -C (Language) * Swift (Language) * Xcode (IDE) * Cocoa (Framework) | * C/C++ (Language) * Java (Language) * Python (Language) * Eclipse (IDE) | * C# (Language) * Visual Basic (Language) * Visual Studio (IDE) * .NET (Framework) | * Java (Language) * Swift (Language) * Objective-C (Language) * Android Studio (IDE) * Xcode (IDE) |

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**: Microsoft Windows would be the recommended operating platform. It is widely used and is a familiar environment for users. It also has a large development community, making it easy to create and maintain.
2. **Operating Systems Architectures**: Windows supports architectures such as x86 and x64. These are commonly used with desktops and laptops. For maximum compatibility I would recommend x86 and x64 but it would ultimately depend on the target device.
3. **Storage Management**: Microsoft Azure Storage would be an appropriate storage management as it is compatible with the selected operating platform (Windows). It provides flexible and scalable storage solutions and includes security measures such as encryption and access controls to protect data.
4. **Memory Management**: Windows uses management techniques such as virtual memory. This allows applications to use more memory than is available physically. The structure interchanges data between physical and virtual memory to make sure the application runs efficiently.
5. **Distributed Systems and Networks**: Microservices would be a good, distributed service architecture to enable communication between multiple platforms. The network that connects devices can be secured by secure communication protocols like Transport Layer Security (TLS) and encryption to ensure data is being protected during communication. Redundancy and failover mechanisms should be used to mitigate connectivity issues or outages.
6. **Security**: Windows offers good protection and security capabilities. Built-in security features that it uses include Windows Firewall, Windows Defender and BitLocker to protect user information. It is imperative that encryption and secure communication protocols are used to provide optimal security. Also, security audits and testing should be implemented regularly to ensure that there are no vulnerabilities.