

<Name-of-Software-Application>

# **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 4**

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0  2.0 | <11/20/24>  <12/8/2024> | Zinedine De Leon  Zinedine De Leon | First part  Evaluation Added |

**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 *Draw It or Lose It* game will be a web-based, multiplayer drawing and guessing gamethat seeks to provide a fun group experience for its users. The game consists of four one-minute rounds with images incrementally rendered over thirty seconds. The system should support one or more teams with multiple players and only one game instance in memory. These teams and players must have unique identifiers for each instance of a game, team, or player. The requirement of having one game instance in memory will be solved by implementing a client-server architecture that will provide a backend to manage game logic, data storage, and real-time updates. The application will also feature an intuitive user interface, scalable and efficient real-time performance, and secure data handling to ensure that the user has an enjoyable experience while meeting system requirements.

## 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 clearly and concisely.>*

## [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.>

1. **Real-Time Performance:** The application must support real-time guess attempts, drawing, and rendering without considerate lag, even if multiple teams are participating. Choosing the right technology for the job will be crucial, affecting how efficiently the system can handle resources across different network speeds and user loads.
2. **Unique Identifiers:** All teams, players, and games must have a unique identifier to avoid unwanted errors and conflicts when setting up game environments. This can be achieved using a centralized database that maintains the game’s data state and verifies that the identifiers are unique.
3. **Cross-Platform Accessibility:** The game must be accessible to users on various devices and browsers. This can be achieved by following web standards, implementing responsive design, and testing the game on different platforms.
4. **Single Active Game Instance:** The game system must only have one active game instance in memory, and it should prevent multiple instances from being created. This can be accomplished by carefully designing the backend to manage game lifecycles. To supplement this, the database will handle the unique identifiers of each game.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: Nothing is 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 UML class diagram shows the Entity base class implements encapsulation with /private attributes and methods. Also, it displays the use of inheritance by being extended by the Game, Team, and Player classes by inheriting the id and name attributes. The GameService class demonstrates the Singleton pattern with the getInstance() method. It can act as the central manager and maintain games. The Game class holds multiple Team objects with a List (as shown by the 0..\*). The same relationship is present with the other two child classes, as the Team class also holds Player objects through with a List (still a 0..\* relationship). The ProgramDriver class executes the entire program and the SingletonTester tests the GameService implementation.

**"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** | MacOS is known for its performance, stability and security. This makes it a suitable choice for hosting web applications; however, it lacks many of the production-grade features that Linux and Windows offer. The cost of hardware may be a deal breaker when compared to Linux and Windows. | Linux is an open-source operating system that is known for its reliability, flexibility and security. These facts make it an excellent choice for a web server. Its open-source nature also allows it to be less expensive than competitors. A challenge may arise in the expertise level needed to effectively use Linux.  Linux is also readily available on all the major cloud providers. | Windows is an excellent choice as a server when integrated with other Microsoft technologies. It simplifies the process by providing users with a graphical user interface. The pricing for Windows Server license does tend to be much higher when compared to options like Linux.  Like Linux, Windows is also available on all the major cloud providers. | It is technically possible to use a mobile device as a server, but it is not recommended as hardware specifications will naturally be more limited than a desktop operating system. Cost is hard to predict, as it is not a common way to build a server. |
| **Client Side** | To develop applications for MacOS, a team will need developers who are skilled with the tools of the Apple ecosystem. This includes the Xcode IDE and programming languages such as Swift and Objective-C. These languages and platform may be viewed as more niche, thus meaning that  MacOS development is limited to only Apple desktops and laptops, thus meaning that acquiring the proper hardware and devices will be costly.  MacOS also has a much smaller market share at around 15% when compared to Windows’ 72%. | Linux can support a variety of languages. Including popular languages like Java, C++, and Python. By leveraging these popular languages, it will be less difficult to find developers with experience in these languages.  Although Linux supports a variety of programming languages and software, it still requires a high level of expertise to work with. Thus, putting a cap on the speed of development and ease in finding qualified developers.  Linux holds about 4.5% of the desktop operating system market share. | Development on windows is primarily done with the C# programming language and the .Net framework. The barrier into Windows development is low, as these tools are open source.  Finding developers with these skills will be possible, but it will be imperative that they specialize in Windows development.  Licensing for tools and servers are more expensive, so that is one thing to consider.  Windows holds most of the desktop operating market share at 15%. | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.>  Mobile devices are typically made for a single user, but it is possible to implement a multi-client approach.  Android applications are built in Java or Kotlin. Both, have transferrable skills when coming from a Linux or Windows background, as they are both object-oriented languages.  iOS apps are built in both Objective-C and Swift, so knowledge when developing a MacOS application is transferrable. |
| **Development Tools** | MacOS has a few technologies dedicated for development on their platforms. The first is the Xcode IDE. The two main programming languages for the platform are Swift and Objective-C  Mac also can run most IDEs. Some examples being Visual Studio Code, IntelliJ, and PyCharm.  An Apple Developer account will incur a fee of $99 per user. | Linux supports a plethora of programming languages. Java and Python stand out as options. Free IDE options exist for Java such as IntelliJ Community Edition and Eclipse. For Python PyCharm Community Edition and Visual Studio Code stand out as free options. | Windows’ parent company Microsoft produces many development tools for Windows development. For programming languages and frameworks, the company offers C# and the ASP.NET Core framework. For IDEs, the Visual studio IDE is the choice for windows development.  Using Visual Studio incurs one of two fees. They are $45 or $250 per year depending on the desired features. | For Android, Google’s Android SDK is what powers android apps. The programming languages used to develop android apps are Java and Kotlin. The IDE of choice is Android Studio, which is free to use.  iOS applications are developed with Swift or Objective-C. Popular frameworks are UI Kit and Swift UI. The Xcode IDE is the development environment of choice for iOS. The IDE has a $99 fee per user. |

## 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**: The operating platform of choice is Linux. The open-source nature of Linux allows for the server cost to be much lower than both Windows and MacOS and more readily available on cloud providers. The client/front end won’t be hampered by the server choice, as it will communicate via REST API. This means that it will be possible to have native user interface technology depending on the target device.
2. **Operating Systems Architectures**: The architecture will be a client/server architecture. This means that the client and server are loosely coupled. The client will make requests to the server. The server will return responses. This allows for a separation of concerns.
3. **Storage Management**: Using an external database system will be ideal. Having all system data centralized in one location will be best for long-term storage.
4. **Memory Management**: Each operation uses different memory management methodologies, so it will be variant across the board.
5. **Distributed Systems and Networks**: Using a distributed system will be crucial, this can be done through API gateways provided by cloud providers (AWS, Azure, GCP). It will allow a middleman between the client and the server. It will be a valuable tool for directing requests to the proper resources.
6. **Security**: There are various ways to secure the system. Using proxies and firewalls are some ways to protect the server itself. When it comes to protecting user resources, encryption methods, and user authentication will need to be implemented.