

<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 | <09/22/2024> | <Drina Baptiste> | <New information on Executive Summary, Requirements, Design Constraints, Domain Model, Evaluation, and Recommendations> |
| 2.0 | <09/22/2024> | <Drina Baptiste> | <Revised information on Design Constraints, Domain Model, and Evaluation> |
| 3.0 | <10/20/2024> | <Drina Baptiste> | < Revised 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 Project has a game called Draw it or Lose it, which is only available on Android. Our client wants to develop a web-based game in which it can serve on multiple platforms. The objective of the game is to guess what images are being drawn. A typical game consists of four rounds of play that lasts one minute each.

## Requirements

Software Requirements:

* The 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
* Only one instance of the game can exist in memory at any given time. Utilize unique identifiers for each instance of a game, team, or player

## [Design Constraints](#_2et92p0)

* Application Architecture Transition
* App is currently an android application that needs new infrastructure. This means adapting to a client-server architecture, where the client communicated to manage state, logic, and data storage
* Game runs on multiple platforms
* Since the game is on Android. We must translate the current app to a webapp in which, it would be able to be accessible from other operating systems~~.~~ The client wishes to have a web-based game application. Therefore, a website application must be built, and a hardware server machine or cloud-based equivalent must be provided, along with an accompanying operating system that will host the website.
* Security
* The web-based application will require a robust authentication mechanism, as mobile-based security (such as device security or Google account authentication) will no longer suffice. This might include implementing OAuth, JWT tokens, or multi-factor authentication to ensure secure user access.
* Storage Capabilities
* The web-based version of the game will require more storage capacity to accommodate stock images used in the game. Additionally, memory management will be crucial to ensure images are rendered quickly enough to match or surpass the performance of the current Android application.

## [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 Unified Modeling Language (UML) diagram will serve as a visual representation of the game system's design, helping to outline how different components of the program interact. From the diagram, we can observe that the Game, Team, and Player classes inherit from the Entity class.. Each class includes its respective variables and methods, which will be implemented during the development phase. The ProgramDriver class, located at the top left of the diagram, points to the SingletonTester class, demonstrating that the ProgramDriver will use the SingletonTester to verify that only one instance of the game exists in memory at any given time. This is crucial for testing the Singleton pattern, a core design requirement. Additionally, the GameService class will contain the complex methods that form the backbone of the game's functionality. One of the key requirements of the program is that each game, team, and player must be unique. To address this, the diagram outlines specific classes for each entity, ensuring clear differentiation between games, teams, and players.

The lines connecting the various classes in the diagram represent their associations, and the numbers between these lines indicate the multiplicity of these relationships. For instance, the GameService class may be associated with zero or more instances of the Game class, and this pattern extends to other related classes. This UML diagram will be referenced throughout the development process to guide the implementation of the system, and adjustments can be made as necessary. It provides a clear and structured framework for understanding how the different components of the program are connected and function together.

**"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 provides server-based environments that offer significant advantages for Mac clients on a network. A macOS server fully supports Mac applications and provides simplified administration through an intuitive graphical interface. However, maintaining a Mac server can be costly, and it is not ideal for large companies or enterprises that require extensive third-party software and customization options. | Linux server offers an opensource operating system. They can operate on physical devices or be virtually developed. Linux is cost-effective since it is open source. It provides stability and reliability, since it is known to handle high workloads. | Windows has a user-friendly interface. Which allows for an ease of use. It supports a wide range of technologies and applications. Typically, they have great support with third party applications. Windows is consistently improving with providing hardware and patch updates. Windows is one of the most used systems. | Although using mobile devices as web servers is uncommon and likely impractical for most scenarios, it is possible to implement. Oracle is one company that provides mobile server-side solutions through its Oracle Database Mobile Server which can manage applications, users, devices, and data across large deployments of mobile or remote devices. The key advantages of Oracle's database structure include support for both iOS and Android development tools, as well as synchronization with Oracle NoSQL databases. |
| **Client Side** | MacOS is costly. This means that there is a possibility to go over the budget. Development on Mac is a seamless process with not too much of a learning curve. There is plenty of resources and customer support available. There is also the benefit of Safari, that is MacOS’s default browser. | Linux has a lack of application support, which can slow the development team down. There is also a steeper learning curve to Linux than their counterparts. There is not much technical assistance available for users | Windows is costly. Windows is compatible with a range of applications, making it easier for the development team. It is compatible to all hardware. Windows also provides software development support. The ease of use will allow developers to develop efficiently. Windows also utilizes Microsoft Edge, which is the primary browser. Edge is compatible with a variety of plugins and applications. | Mobile Devices is a steep learning curve. There is not much support for developers. It is difficult to implement web-based applications. |
| **Development Tools** | The programming languages used are swift, python c++, java, ruby. IDE’s: Visual Studio Code, Xcode, PyCharm are the most common. | The programming languages used are C++, python, Java, JavaScript, Ruby, PHP, GO, Bash. IDE’s: VS Code, Eclipse, PyCharm, Sublime Text. | The programming languages are C#, c/C++, Python, JavaScript, Java, .net, and PowerShell. IDE’s include: Visual Studio, Visual Studio Code, PyCharm, Eclipse. | The programming languages: Java, Kotlin, Swift. IDE’s: Android Studio, Xcode, Visual Studio Code |

## 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**: To expand Draw it or Lose it on other operating platforms. Based on our research, we believe that Windows would be the most appropriate operating platform.
2. **Operating Systems Architectures**: Windows operates on a hybrid architecture, combining features from both monolithic and microkernel designs to deliver performance, flexibility, and security. It has two main layers: User Mode and Kernel Mode. User Mode runs applications and services, ensuring user processes are isolated and secure, while Kernel Mode manages core OS functions like memory, process management, and device control. The Hardware Abstraction Layer (HAL) allows Windows to run on various hardware platforms by abstracting hardware specifics. With its NTFS file system, robust security features, and support for a wide range of applications, Windows offers a reliable, scalable, and secure environment for web applications, making it ideal for business use.
3. **Storage Management**: Since, Draw it or lose it will be a web application. We recommend that a cloud-based storage is used. Based on our research, we believe that Amazon Web Services (AWS) S3 will be the best fit for this project. S3 is scalable and charges only for the resources that are used.
4. **Memory Management**: Based on our research, Java can be effectively used on Windows for memory management. The JVM provides automated garbage collection and memory handling, while Windows manages the physical and virtual memory resources. For optimal performance, the JVM's memory settings can be fine-tuned based on the needs of our web-based application and the available resources in the Windows environment.
5. **Distributed Systems and Networks**: The game components, including the server, clients, and database, are distributed across multiple systems connected via a network. WebSockets can handle real-time communication, while RESTful APIs manage less time-sensitive interactions. Cloud platforms like AWS S3 provide scalability, ensuring the game operates efficiently with fault tolerance. Security is maintained through SSL/TLS encryption and strong authentication protocols, safeguarding the game environment and user data.
6. **Security**: Based on our research with AWS’s S3. We believe that the security of data is secure with them. They offer a range of encryption features and access management tools. This ensures that data cannot be easily accessible.