

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 | 03/21/25 | Hailey Bondra | Changes were made to the cover page, the document revision history table, executive summary, requirements, 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 client, The Gaming Room, has asked me to develop a web-based version of their mobile game, *Draw it or Lose It*,that is only available to download as an Android app. The game is designed to have multiple teams compete to guess a drawing corresponding to a specific puzzle. Each game consists of multiple rounds as each team member takes a turn to guess. The application needs to be platform-independent where it is easy and accessible across different environments, and it needs to be scalable where it can be adaptable to changing demands. The goal of this project is to create a seamless transition between the existing mobile app to a web-based platform. The solution needs to support multiple teams consisting of multiple players on each team and a single instance of the game is retained in the memory.

## Requirements

The requirements for the *Draw It or Lose It* game application is one or more teams are involved in the game, multiple players are in each team, game and team names are unique to track each game, the singleton pattern is used in order to ensure that only one instance of the game can exist in memory at any given time, and the application must be accessible across mobile and web-based platforms.

## [Design Constraints](#_2et92p0)

There are three constraints to be considered in the development of a web-based game application. The scalability of the game needs to be considered to support multiple teams, players, and the game. The application needs to handle concurrent access to the game as multiple teams consisting of multiple players will be playing at once and any actions or guesses do not interfere with each other. The application should be compatible and seamless across multiple mobile and web-based platforms.

## [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 UML class diagram consists of seven classes. The ProgramDriver class, SingletonTester class, Entity class, GameService class, Game class, Team class, and Player class.

How the classes relate to each other:

-ProgramDriver uses the SingletonTester class

-Game, Team, and Player all inherit attributes and behaviors from Entity class

-GameService contains multiple Game instances

-Game contains multiple Team instances

-Team contains multiple Player instances

The project utilizes object-oriented principles such as the singleton pattern, encapsulation, inheritance, and aggregation. The singleton pattern is used in the GameService class to ensure that only one instance of the game is managed throughout the lifecycle. The getInstance() method in GameService class prevents multiple instances and ensures global availability. Encapsulation allows controlled access and modification and helps data hiding. The GameService, Game, Team, and Player classes all utilize this principle using methods that get/return attributes such as getId() in the Entity class. The Game, Team, and Player classes all inherit from the Entity class which allows the Entity class to share common attributes and methods without needing to repeatedly redefine them in each class. The GameService, Game, and Team classes aggregate other objects. So GameService contain Game objects, Game contains Team objects, and Team contains Player objects.

**"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 can be used for server hosting, but it is not commonly deployed for large scale web applications. Licensing costs are expensive for server OS. An advantage is Mac is known for its stability and reliability. The disadvantages are limited scalability and compatibility issues with non-Apple clients. | Linux is a popular server OS for web hosting. Linux web hosting supports languages like Python and Perl. There are no licensing costs as Linux is open sourced. An advantage is that it is scalable and is secure from regular updates and security tools implemented. A disadvantage is you need technical knowledge to be able to understand and manage it. | Windows is also another popular choice for web hosting and supports a range of technologies such as SQL Server, ASP, and ASP.NET. The licensing cost can be expensive. The advantage is it works well with other Microsoft products, simplifying network management and application deployment. A disadvantage is more resources are required, like paying a license compared to an open-sourced server for the same performance. | IOS and Android offer server-based deployment methods such as cloud hosting through Google Cloud Platform or Microsoft Azure. The advantage is that mobile devices allow portability and more accessibility. The disadvantage is that there are limitations such as the device could struggle to run web applications smoothly and it could not be as optimized for mobile devices compared to other servers. |
| **Client Side** | Supported web and mobile browsers such as safari will handle HTML content. It requires expertise in web design and development specifically for IOS. | Requires web development and Linux server knowledge for integration. For mobile devices and web browser compatibility, Chrome would be best. | Compatible with all browsers such as Chrome, Firefox, and Edge. Requires a responsive design and strong compatibility with web standards. | Chrome or Safari browsers will be the interface. Requires knowledge in front end development. |
| **Development Tools** | Programming languages such as HTML, CSS, JavaScript for web and Swift for IOS specific features can be used. Xcode and VS Code can be used for development tools. Requires IOS development knowledge and web development knowledge. Development tools are free. | Programming languages used are HTML, CSS, JavaScript for web and Python, Node.js for backend. Development tools consist of VS Code and Sublime Text. Requires general web development expertise. OS is open sourced, and tools are free. | Programming languages used are HTML, CSS, JavaScript for web development and C#, ASP.NET for backend. Visual Studio and VS Code are development tools used. Backend development knowledge is required and general knowledge of web technologies. VS Code and Visual Studio Community is free and full Visual Studio requires licensing cost. | Swift for IOS and Kotlin for Android, HTML, CSS, and JavaScript for web apps. Xcode, Android Studio, and VS Code are development tools used. No licensing costs. Requires expertise in IOS specific and Android specific development knowledge. |

## 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**: Using a Linux based cloud server platform such as Ubuntu hosted on a scalable cloud platform such as Amazon Web Services. This will provide the gaming application to be scalable, secure, and produce high performance.
2. **Operating Systems Architectures**: Ubuntu uses monolithic kernel architecture, so the operating system core runs all essential services in the same address space. This will improve the performance of the gaming application.
3. **Storage Management**: Amazon S3 for object storage will support game assets and user data. Amazon EBS for block storage will support encrypted storage for game logs and databases.
4. **Memory Management**: Ubuntu Linux utilizes virtual memory, paging, and swapping. These techniques support memory isolation through containers as this helps manage different services and optimizes the use of RAM.
5. **Distributed Systems and Networks**: Microservices architecture with APIS and message queues will manage communications between game components.
6. **Security**: Data can be protected with Hypertext Transfer Protocol Secure (HTTPS) when in transit and when at rest Amazon Web Services Key Management Services can be utilized. Multi-factor authentication such as OAuth 2.0 will use role-based access control and input validation to prevent violations.