`

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.i7c1x2pcgojq)

[**Table of Contents 2**](#_heading=h.c1dvj9xqf8c1)

[**Document Revision History 2**](#_heading=h.g1jdqytioof6)

[**Executive Summary 3**](#_heading=h.ob4im9lj2ez)

[**Requirements 3**](#_heading=h.p6uztyr3z2k3)

[**Design Constraints 3**](#_heading=h.8dbbta3yeyt4)

[**System Architecture View 3**](#_heading=h.e7ng2ar49e7k)

[**Domain Model 3**](#_heading=h.1p5okj9fgkph)

[**Evaluation 4**](#_heading=h.8t04kslmd5zr)

[**Recommendations 5**](#_heading=h.yxbqowg88lon)

## [Document Revision History](#_heading=h.21cqdqyjjt59)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/17/25 | Brandon Landrum | Initial draft. Added Executive Summary, Design Constraints and Domain Model description. |

**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](#_heading=h.ytonspbmz6h7)

The client, The Gaming Room, seeks to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It. This can be achieved by creating a browser-based web application. This will allow the game to be accessed and played from any platform with access to a web browser.

## Requirements

Technical Requirements:

* must be a web-based game
* must be able to access game on multiple platforms
* must be able to access game from a browser
* A 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 to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_heading=h.ut7gjzqpmgsz)

* App must function across different web browsers and devices.
* App only allows for one instance in memory at any given time by using unique identifiers for each instance.
* App must allow for some form of communication between players.

## [System Architecture View](#_heading=h.urhpxx62a484)

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](#_heading=h.gsuk30698f0a)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

The Entity class is a superclass that provides common functionality for its child classes. Player, Team, and Game all inherit from Entity. There can be many Players per Team, many Teams per game, and many Games per GameService. GameService is a Singleton class and there can only be one instance of GameService. The ProgramDriver class uses the SingletonTester class to ensure that singletons only have one instance. The ProgramDriver class also serves as the starting point for executing the main program.

**"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](#_heading=h.tpgvww8nrt6p)

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** | Server configuration and GUI are user-friendly. Not optimized for hosting servers. Expensive to set up/maintain. | Command-line driven. Generally secure because of its permission model and being open source. Inexpensive to maintain. High learning curve. | Robust application support. Integration with Microsoft Ecosystem. Expensive to license. | Limited hardware specs. Limited scalability. Limited web server software availability. |
| **Client Side** | Minimal cost. Minimal time to test the application in the browser. Expertise with Safari developer tools required. | Some expertise with Linux is required to test the web app in the browser. | Minimal expertise required. | Significant development cost to implement Responsive Web Design. Expertise in RWD and mobile web performance optimizations. |
| **Development Tools** | Languages: HTML, CSS, Javascript.  Tools: Visual Studio, Github, Pycharm. | Languages: HTML, CSS, Javascript.  Tools: Vim, Visual Studio, Github, Pycharm. | Languages: HTML, CSS, Javascript, C#.  Tools: Vim, Visual Studio, Github, Pycharm. | Languages: HTML, CSS, Javascript, .  There is a lack of tools such as IDEs to develop web applications on mobile devices. |

## 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**: Recommendation: Linux.

Linux is a widely used and free operating system that is renowned for its stability, scalability, and performance.

1. **Operating Systems Architectures**: Linux utilizes a monolithic kernel with loadable modules. This allows OS services to run in a single kernel space. The Linux kernel’s scheduler allocates CPU time slices to different processes through preemptive multitasking. This improves responsiveness even when the CPU is heavily utilized. Linux also is an inherently multi user system.
2. **Storage Management**: The Linux server system can use an ext4 file system to store the operating system itself, the Draw It or Lose It game application, logs, and configuration files. A relational database can be utilized to store structured data, such as user accounts.
3. **Memory Management**: If there is sufficient free physical RAM available, Linux can use that free RAM to cache data read from the disk. When this data is needed, it can be read directly from RAM, which is significantly faster than reading from disk. If there is not much available ram, Linux can use swap space. This allows Linux to use more RAM than is physically available by utilizing disk space.
4. **Distributed Systems and Networks**: The Draw It or Lose It application can use a client-server architecture to enable communication between users on various platforms. The clients can run an instance of the game that will be responsible for UI and user input. The server can handle the game logic. The clients would connect to the server through a central API Gateway. Once connected the client sends requests via HTTPS. The API Gateway validates and forwards requests to the appropriate service to be handled.
5. **Security**: All data should be encrypted between the client and server. This can be done through HTTPS and WSS. User passwords should be hashed and never stored as plain text. Users should only be allowed to perform actions that are permitted for their role. All input received from clients should be validated and sanitized.