Link to Github Repository: <https://github.com/cheyennenave/CS-230/tree/main>

The Software Design document is loaded into my repository, however, I have included it here as well for proper submission. The README has been updated to include all Module 8 questions.



Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.3

## 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 | <01/25/2024> | Cheyenne Nave | Initial version of Design Template for Draw It or Lose It |
| 1.2 | 02/05/2024 | Cheyenne Nave | Addressing strengths, weaknesses, characteristics, and security for Draw It or Lose It |
| 1.3 | 02/23/2024 | Cheyenne Nave | Analyzing Characteristics and Techniques specific to various system architectures for recommendation |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room needs help creating and launching Draw It or Lose It, a timed puzzle game loosely based on the 1980s game Win, Lose, or Draw. The game will be web-based and work across multiple operating systems which is currently only available on Android. In the game, multiple-player teams need to be an option and each game needs to be its own unique entity. The game itself comes with a set of data which is translated to a library of media. To achieve this, the budget needs to be reviewed as well as any time constraints due to the differing levels of experience our developers currently have.

## Requirements

* The game must have the ability to have one or more teams playing.
* Each team must have multiple players.
* Game names and team names must be unique. A function must be included to allow the player to check and see if the name they have chosen is already in use.
* Only one instance of the game can exist at any given time. Unique identifiers need to be created to support this.
* A 30-second timer per round must be included during rendering.
* A 15-second timer per round IF the opposing team fails to use their turn.
* Use of Singleton Design Pattern as well as Iterators to complete the app efficiently.

## [Design Constraints](#_2et92p0)

Budget needs to be evaluated to cover the complete cost of development across multiple platforms. Schedule needs to be reviewed as well to make sure complete development of the app is produced on time and published by the deadline. Because each operating system uses different programming languages and kits, our developer experience may vary.

## [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)

<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 ProgramDriver class holds the Main method that will be used to start and run the program. The Main method also uses the SingletonTester class which will check and make sure the Singleton Design Pattern has correctly been coded and applied in the program. The Entity class is a parent class to Game class, Team class, and Player class, which each inherit the Entity attributes, but also has their own specific to their class. Game Class adds teams and has an a unique id. Team Class adds players and has a name. Player class has the individual player information. The GameService class will be our Singleton that creates each unique game.

**"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 is expensive and licensing must be purchased. MacOS needs other Macs for development  **V1.2 update:**  Project cost can be minimal if an HTML is used to connect the browser to a back-end server. Authentication will be needed to only allow the websocket. However, cloud support is an option as well. | Licensing is free as well as open-source.  **V1.2 update:**  Linux has great security measures which can include multi-factor authentication. Linux is also involved in cloud support to help everything run smooth. Linux is easier to be tailored compared to Mac. | Windows is expensive and licensing must be purchased. High security.  **V1.2 update:**  As far as security, Windows uses permissions which can be set up by the main user. It is most common, so most users understand how to launch and use in general. Cloud support is optional through Azure as well. | Potentially used for development, however highly discouraged to use as a server.  **V1.2 update:**  Mobile devices are still highly discouraged to use as a server. A small amount of applications could only run at a time compared to Windows, Linux, or Mac. |
| **Client Side** | MacOS needs other Macs to be developed, so hardware could become expensive  **V1.2 update:**  The browser for Mac is Safari, so if our game uses JavaScript, we need to be sure that the JavaScript toggle is switched on Mac products. | Longest development time. User needs to be fluent in Python.  **V1.2 update:**  The browser of choice for Linux is FireFox. While Firefox is popular, it does have extensive pop-up blockers for ads and such that may be an issue if multiple windows are needed. | Windows certification highly recommended.  Efficient and Secure.  **V1.2 update:**  Internet Explorer and Microsoft Edge are the main browsers for Windows. Unfortunately, Explorer has rapidly declined in recent years as competitor browsers have come about. It is the least popular browser which could lead to it dying out completely. | Developers must have experience with mobile apps to understand display  **V1.2 update:**  For mobile devices, a universal browser like Chrome could be used, however many design elements must be kept in mind such as screen-rotation, accessibility across multiple brands, and various intricacies that may affect the app (like the JavaScript toggle for MacOS) |
| **Development Tools** | MacBook with Xcode. Other plugins may be possible.  **V1.2 update**  Eclipse IDE can be used even though Xcode is the main tool used for MacOS. Xcode works specifically for Apple, so Eclipse would be a better choice since it can be used across multiple platforms. | Python is the main programming language used on Linux.  **V1.2 update**  Eclipse or Visual Studio would be the choice IDEs to use. There are multiple choices for Linux, so developers could have their pick at what they’re most comfortable with, as each IDE is multi-platform accessible. | Visual Studio with C++ programming language.  **V1.2 update**  Again, Eclipse or Visual Studio would be the top choice IDEs for Windows. Both are versatile and free. | Unity using C++ which then can be converted to a mobile app depending on the OS  **V1.2 update**  It is not recommended to develop on a mobile device. |

## 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**: Linux is recommended due to its cost-efficiency, flexibility, and security measures. It can be tailored easily to the user and the client more efficiently. Even though it is the least familiar among users, the positive attributes that Linux provides outweighs those of the costly MacOS and Windows.
2. **Operating Systems Architectures**: Linux is recommended once again due to its structure. It is more flexible compared to the other operating systems which allows for future development as the game grows and expands. Essentially, Linux is modular which means that it can extend its capabilities whereas other operating systems currently cannot.
3. **Storage Management**: SSD storage will allow a faster and more efficient loading of the game and its assets. Also, Linux supports in-cloud storage to reduce costs if necessary.
4. **Memory Management**: Because Linux uses Paging to help Memory Management, it is recommended. Paging helps software run and perform better by only loading memory that’s currently being used. Also, to ensure that everything runs smoothly, all images in library stock need to be saved to the minimum required size as well as correct format so not to take up space.
5. **Distributed Systems and Networks**: Linux is recommended once more not only because it will be the Operating Platform and Operating Systems Architecture, but also because if the game is to expand across multiple platforms and continually grow, Linux is best at communicating through those various platforms. It is also still recommended to balance your network with rolling updates across the system so it will regulate processing time as well as help identify and fix bugs.
6. **Security**: Linux is recommended. Linux boasts efficient security, including encryption and multi-factor authentication. It also has SELinux; a security tool that when toggled on, can set access controls across the board so that all files are safe and accessibility is limited to access protocol.