

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/13/22 | Monticia Dunn | The client would like help developing a web-based version of their game that serves multiple platforms |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, would like to expand the reach of their Android-based game, Draw It or Lose It, by developing a web-based version of the game that serves multiple platforms. The client would like for the app to allow for multiple players on multiple teams and requested assistance in getting an environment setup and streamlining the development. They have also required that the games and team names be unique and only one instance of the game can exist in memory at a time. To meet these requirements, the team will utilize certain design patterns as well as evaluate the various operating platforms to determine if multiple different programs will be required to accommodate each platform.

## [Design Constraints](#_2et92p0)

There are a few constraints for the game aspect of the design including allowing for multiple players on multiple teams all with unique names. As well as ensuring only one instance of a game exists in memory at a time. As mentioned before, this will be addressed with specific design patterns to make sure each object instance follows the appropriate flow and may take some extra design/planning time before development can begin. The main design constraint, as well as the most complex, is making sure the app is accommodating of multiple platforms. This will also require extra design/planning time since the team will likely need to make certain adaptations to ensure that the application runs properly on different systems. Since the game is multiplayer, security and network concerns will also need to be considered since users will likely be connected to each other through the Internet and the traffic from that will need to be monitored and managed.

## [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)

We use the inheritance OOP principle in the Entity class as it creates a relationship between the Game, Team, and Player classes. This allows for reusability of parent fields and methods so we don’t repeat the same code in every child class when they all will need to utilize certain things from the parent class.

We use a singleton because we will only ever need one instance of the GameService object at a time. We don't need new GameServices objects we just need access to the one already created. This also meets one of the requirements from the clients to only have one instance of the game in memory at a time.

We use the iterator pattern, so we don't have to worry about the underlying data structure we use to hold all the Game objects, we just need an easy way to traverse through that data structure. This also meets the requirement ensuring team names be unique, as we will need to iterate through a data structure containing all the Team objects to check if the name has been used.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Advantages:   * High stability * high level of security * easy to install/setup   Disadvantages:   * high cost for web hosting | Advantages:   * high stability * very affordable * easily customizable   Disadvantages:   * high learning curve to use effectively | Advantages:   * best for hosting .NET software * many options available   Disadvantages:   * medium cost for web hosting * history of being less secure than Mac and Linux | Advantages:   * very affordable   Disadvantages:   * hard to account for different mobile OS’s * less secure |
| **Client Side** | * May be costly if team does not have Mac computers * Does not require high expertise to use * Takes average amount of time to adjust to if unfamiliar with Mac | * Less costly than Mac or Windows computers * Requires more expertise to use than more popular OS’s like Mac or Windows * Takes average amount of time to adjust to if unfamiliar with Linux | * May be costly if team does not have Windows computers * Does not require high expertise to use * Takes average amount of time to adjust to if unfamiliar with Windows | * May be costly for team since they will need access to iOS and Android * Does not require high expertise to use * Takes average amount of time to adjust to since both devices vary in different ways |
| **Development Tools** | Popular languages for Mac include:   * Swift and Objective-C for mobile development * HTML, CSS, JavaScript for frontend development * Java, Python and JavaScript for backend development   IDEs include:   * VS Code * Eclipse * JetBrains IDEs | Popular languages for Linux include:   * HTML, CSS, JavaScript for frontend development * Java, Python and JavaScript for backend development   IDEs include:   * VS Code * Atom * JetBrains IDEs | Popular languages for Windows include:   * HTML, CSS, JavaScript for frontend development * C#, Java, Python and JavaScript for backend development   IDEs include:   * Visual Studio * VS Code * Eclipse * JetBrains IDEs | Popular languages for mobile development include:   * Swift for iOS * Java for Android   IDEs include:   * VS Code * Android Studio * JetBrains IDEs * Atom |

## Recommendations

1. **Operating Platform**: In my opinion, the preferred operating platform for The Gaming Room is Linux. MacOS doesn’t really offer a server OS, so it’s non-applicable. Linux is free and open source, so there are no licensing fees like there are with Windows. It also enables seamless use and integration with other open-source software, unlike Windows, which is more focused on Windows applications. It may not necessarily be the easiest use since it will require more terminal use, unlike Windows which primarily utilizes a GUI.
2. **Operating Systems Architectures**: The Linux operating system is basically composed of four layers. Hardware layer, kernel layer, shell layer, application layer. The hardware layer consists of all physical devices attached to the system. System processes that interact with hardware run at the kernel layer. The shell layer collects input from the user and allows it to leverage system processes without directly affecting them. Finally, the application layer contains all utilities that run on top of the shell.
3. **Storage Management**: The application will need to store several pictures and be able to render and display those pictures at a fixed rapid rate, using a cloud-based storage system would make this the easiest, as it’s easily scalable, cheaper, and efficient to use as opposed to hardware storage.
4. **Memory Management**: The overall goal for memory management is to create a database to store the files (such as images) needed by the application, route requests for information from the database through an API and provide a single source of truth for all clients since they are all routed through the same middleware. The client can then fetch the images as needed and delete them when ready, avoiding large amounts of images taking up space and memory on the user's device.
5. **Distributed Systems and Networks**: Since Draw It or Lose It application would like to communicate between various platforms a client-server distributed system will be used. This is because each client application depends on the game's individual server application. Therefore, each client application can be developed according to the strengths of the client system. A strong server network is also required, again because this game depends on multiple clients connecting to a single server to play the entire game. I would recommend a utilizing REST style for the API as it’s become standard in the industry and is well-known for being an efficient way to design scalable networked applications and web services.
6. **Security**: Linux has long been used as the basis for web servers. It doesn't leak memory like Windows Server and only needs to be rebooted on kernel updates. Because it's open source, you will have to rely on the technical community for help, unlike Windows, which has its own technical support team. Plus, of course, there are many layers of security you can add on top of the default security that comes with your server. Install only required packages, backup SSH keys, install firewalls, etc.