

# **Draw It or Lose It**

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 2/8/2022 | John Sullivan |  |
| 1.1 | 2/18/2022 | John Sullivan | Module 5 update. Evaluation added. |
| 1.1.1 | 2/20/2022 | John Sullivan | Module 5 corrected |
| 1.2 | 2/23/2022 | John Sullivan | Module 7 |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming room is seeking to develop a web app for their popular game “Draw It or Lose It”, The game is currently only available as an application for Android. Implementation of the web application format will expand its market to a significantly larger market. The game will allows team based play with the possibility of multiple teams to compete at once.

## [Design Constraints](#_2et92p0)

The game will be reliant on a server platform hosting the web-based application. This will allow the players to have a stable a secure connection to the game. The application must only allow one instance of the game be loaded into memory via the singleton method. The player must be able to load the game via a unique Player, ID and/or Team. The game name and the team names must be unique, the application will need to have a system in place to prevent the creation of duplicates.

## [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 Game, Team, and Player classes are all inherited by the entity class. The Game service class holds the data used by all the other classes, its associates first to game class, game associates to team, team associates to player. Games list, the teams list and the players list are held in private functions allowing only the appropriate data to be passed along between the classes via the encapsulation of the data. The passing of the array list and overloading are examples of the polymorphism that will allow the singleton method to remain true.

**"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)

+ means the attribute is a positive

- Means the attribute is a negative

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | -Limited Hardware Options.  -Limited scalability  -Expensive  +Uniformed software and hardware  -Potential ARM architecture with limited support. | +Open source  +Highly optimized operating systems available.  -Limited third party applications to assist.  -Generally more intrusive and time consuming set-up and maintenance. | -Potentially expensive licensing costs  + Wide support for third party software.  +Regular security updates backed by a reliable entity.  -Significant overhead usage on hardware. | -Bandwidth  -Hardware limitations  -connection stability  -cost  -lack of support |
| **Client Side** | +Web browser is able to function as client.  +No special requirements  +HTML client will work. | +Web browser is able to function as client.  +No special requirements  +HTML client will work | +Web browser is able to function as client.  +No special requirements  +HTML client will work | -dev time for mobile app  +expanded user base  +quick ease of access app made with java for android  +quick ease of access app made with swift for IOS  +The tools to develop can be at low cost to license. |
| **Development Tools** | +Full support for free development software.  +Simple and user friendly installation and use.  - ARM processor hardware may not be compatible.  +unix  +Xcode support  +can VM windows | -Less user friendly  -Issues are timely to repair  +large community of support.  -lack of updated hardware support.  + Most secure  +unix  +can VM windows | +wide gambit support for applications  +hardware acceleration  -potential third-party software costs.  +Hardware driver’s native support.  -Least secure  +C# .NET support  +can VM linux  -cannot VM mac | - Limited to online tools. |

Recommendations

1. **Operating Platform**:

For a web-application, the users that are on PC or Mac and others a basic web browser should suffice for the client. For Android applications the Game shall be written in Java, allowing for a wide range of use within the android ecosystem. For Apple IOS devices the application shall be written in Swift, this will allow the application to work across all updated IOS devices. The Servers that are hosting the game services preferably will be running Windows Server, it will provide simplicity and reliability for the system admins. Windows Server will also provide a strong support system via Microsoft and a huge community.

**2. Operating Systems Architectures**:

The client architectures are dependent on what device the user is using. There should be no requirement differences for the web browsers. While for the mobile devices, if they are written in the appropriate languages (Java/Swift), the difference in architecture should not be a concern.

The server architecture shall be platformed on a cloud service, ideally Azure, for the cost, analytics, and additional tools that assist in providing web-application back ends. With proper implementation the cloud architecture will allow “Draw it or Lost it” to grow without limitation. The cloud platform is assumed to be x86 unless stated otherwise.

**3. Storage Management**:

Clients using the web browsers will have no storage requirements excluding potentially

some cookies. The applications on IOS and Android will require the storage of the interface to the web application as well as potentially the image files. Storing the image files locally on the mobile devices will reduce the mobile data usage and reduce load time while the user has lower connection speeds. On the Server side, the storage and concerns about RAID options are not necessary with cloud options. Storage for the images and the user information should be stored into “micro services” as they will allow for more security and reduced duplication of data reducing overall storage requirements.

**4. Memory Management**:

To reduce the overall memory requirements the game application should only have the required images and services active in the application at any given time. For the clients the memory requirements will be small enough that it is not a concern for modern day devices. On the server side, experimentation and analytics will be required to make sure that the allocated memory per module is kept to the minimum to increase the number of instances that can operate on the equipment and reduce costs. Ideally with low operating system overhead each of the virtualized instances could be down to below 1 Gb of RAM required. To Reduce the memory usage for user and game information populating, a singleton pattern in the code will cause only a single instance to be loaded into the RAM, and non-active instances to remain in storage.

**5. Distributed Systems and Networks**:

By utilizing RESTful API, we will have a uniformed input to the hosting services. This will allow for all the clients to update the server-side application accurately, as well as interoperate the information accurately. Because the game is going to require timed input the time should be shared by the host service, this is critical due to the difference in time between iPhone and android, as there is about 15 seconds different in their clocks. The clients will have no direct connection to each other, all connections will be made through the host server. This will increase user security, while also keeping the game active for momentary data disconnects that will inevitably occur with mobile users.

**6. Security**:

The web site hosting the game must have SSL certification to provide encrypted communications between the clients and the hosts. SSL certification will also increase SEO. The SSL will allow for the user information such as username and password to be sent to the host securely. On the host side the passwords will be stored hashed, when the password is received by the client, the password is immediately passed through the same hash algorithm then compared to the newly hashed password with the existing hashed passwords. To increase the security of the user information, the server storing the user information can be hosted on a private local IP that will only allow specific requests to the information from registered equipment. The username and password entry boxes will need protections to prevent bad actors from injecting malicious code. The REST API will also provide a layer of protection as any inputs received by the clients that are not expected will be ignored.