

# 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 | 11/13/2020 | Rose E. Weber | Added the initial information based on the information given by the client |
| 1.1 | 12/3/2020 | Rose E. Weber | Added information relating to the evaluation section. |
| 1.2 | 12/8/2020 | Rose E. Weber | Added information relating to the Recommendations section. |

## [Executive Summary](#_sbfa50wo7nsh)

The game *Draw It or Lose It* needs to be moved from it’s current system as an android only application to a online web-hosted multiplayer game which will incrementally show an incomplete picture being completed with the players being able to guess what the picture contains and keep score for each team in the game during each match. Each game, team, and player should have a unique ID in order to prevent duplicates from being created. The things we need to take into consideration with this move are the security involved, processing power of the server and the necessary hosting abilities which will need to grow as more people play the game.

## [Design Constraints](#_2et92p0)

Memory Management: Since the application will be web-based, there won’t be as much processing power available as there would be in a native application.

Central hosting server: There will need to be a central hosting server to run the application and manage any user information.

Network communications: there will need to be redundancies in place with the hosting server so that any failures or issues on our end don’t impact the end user. Also, we will need to keep network traffic to a minimum to help mitigate any potential sluggishness from occurring.

Security: Since the program will send information to and from the main server, security will need to be addressed to prevent user information from being intercepted in transit.

User Interface: With this program being available on traditional screens as well as mobile devices there will need to be standards in the UI to provide the end user consistency across the different devices.

## [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)

Based on the UML diagram shown below, we can see that the Game, Team, and Player classes all use the programing principle of inheritance to inherit attributes from the base Entity class which will allow us to easily add features to all three classes and streamline our code. Moving on from there we can see that any one game can have zero to more teams and any one team can have zero or more players. All of which have unique names and ids associated with them. The program can only have one instance of the GameService class in order to make sure that each of the other elements (Games, Teams, and Players) have unique IDs and to assure that no two of the same game are created. The GameService class contains and controls the management of the Game instances. Currently, the ProgramDriver class contains the Main method which runs the program. The Singleton tester serves to verify that only one instance of the GameService class exists at the same time.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac OS is not widely used for server-side hosting. MacOS server from Apple does provide the basics for hosting at a fairly low price. Apple advertises it at $19.99 on their website. | Linux is generally accepted as good option for server-side hosting. It has good security and flexibility in it’s setup because it’s a open source platform. There are no licensing costs since it’s open source. | Windows is widely used as a server-side hosting platform. It has good support if issues arise and is easy to setup compared to a Linux based server. Looking at Windows Server 2019 the cost could be between $972 and $6155 according to Microsoft’s website. | Since Mobile Devices don’t have the hardware or software capabilities for hosting a server they are not generally used for server-side hosting. |
| **Client Side**  <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to each platform.> | Chrome, Safari, and Firefox are available as commonly used browsers on Mac.  JavaScript is a language that is available to use with multiple browsers so it would be important to have someone well versed in JavaScript in order to support multiple types of clients.  Cost and time should be similar for each platform since the game is web based. | Chrome and Firefox are available as commonly used browsers on Linux.  JavaScript is a language that is available to use with multiple browsers so it would be important to have someone well versed in JavaScript in order to support multiple types of clients.  Cost and time should be similar for each platform since the game is web based. | Chrome, Edge, and Firefox are available as commonly used browsers on Windows.  JavaScript is a language that is available to use with multiple browsers so it would be important to have someone well versed in JavaScript in order to support multiple types of clients.  Cost and time should be similar for each platform since the game is web based. | Safari, Chrome, and Edge are available on Mobile Devices.  JavaScript is a language that is available to use with multiple browsers so it would be important to have someone well versed in JavaScript in order to support multiple types of clients.  Cost and time should be similar for each platform since the game is web based. |
| **Development**  **Tools** | Languages: JavaScript, Java, C++, Python  IDE: Visual Studio  Other Tools: Chrome, Safari, and Firefox should be included for testing.  Mac also has the ability to run a Windows VM to test on Edge and Internet Explorer.  Cost for Visual Studio IDE depends on the edition. Free for personal, $45 per month for professional, and $250 per month for the enterprise edition. | Languages: JavaScript, Java, C++, Python  IDE: PhpStorm ($199 /year per user), Visual Studio($45 /month per user or $250 /month per user)  Other Tools: Chrome and Firefox.  Safari is not supported. | Languages: JavaScript, Java, C++, Python  IDE: PhpStorm ($199 /year per user), Visual Studio ($45 /month per user or $250 /month per user)  Other Tools: Chrome and Firefox.  Safari is no longer supported since ver. 5.1.7 | Languages: JavaScript, Java, Python  IDE: No full IDE’s are available for mobile devices. Coda is a code editor for iPad which costs ($30 /month per user for a team)  Other Tools: Chrome, Firefox, Safari (if Apple 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**: Based on the requirements of the application *Draw It or Lose It*, I would recommend using a Linux OS system to host the application. Linux has the benefit of being a low cost option as well as serving as a stabile environment for the application.
2. **Operating Systems Architectures**: Linux is an open source operating system that is generally accepted as standard as a server platform. I feel that this is an appropriate choice for an operating platform due to the limited cost as well as the flexibility and customization of the system.
3. **Storage Management**: Onsite overflow servers would be a good option to make sure that if the main onsite storage runs out that there is a secondary support system ready to help out. In addition, multiple backup servers indirectly connected to the main networked servers would be useful in case of a network wide issue.
4. **Memory Management**: Linux uses virtual memory in order to increase it’s memory capabilities. Virtual memory is the concept by which the storage disk temporarily acts as memory in order to prevent issues due to lack of memory. In addition, I would recommend not sending all the necessary images at once but instead sending them in groups at the beginning of each round as to not require lots of memory all at once.
5. **Distributed Systems and Networks**: In order to prevent any downtime as a result of a down server, I would recommend having a series of servers in different locations all of which play host to the application to form a mesh network. This would also help mitigate any connection issues from clients by allowing for them to connect to whichever server is closest.
6. **Security**: Having a good encryption method on any data being sent to and from the game server would be a good first step toward a good security system. In terms of basics, using the concept of the principle of least privileges to prevent external security issues. For user safety, only collecting specific information about the user will make it so if any security leaks occur only a limited amount of user data is available.