

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 | 07/17/2021 | Michael Moreno | Creation of document |
| 1.1 | 08/01/2021 | Michael Moreno | Updated Server side, Client Side, and development tools |
| 1.2 | 08/15/2021 | Michael Moreno | Revision of Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions has requested a game called Draw it or Lose It that is based on the 1980s television game Win, Lose, or Draw. Teams will compete to guess a picture from the library stock and the first team to guess wins. This is a web based game that does not have defined hardware requirements at this time. These will be based on the software decisions.

## [Design Constraints](#_2et92p0)

Each game will have one or more teams involved and each team will have multiple players. Game and team names are unique and will need to be checked. Only one instance of a given game can exist.

## [System Architecture View](#_ilbxbyevv6b6)

Hardware requirements will come after software application decisions are made.

## [Domain Model](#_8h2ehzxfam4o)

ProgramDriver is the main function that runs the application. SingletonTester is a test instance of the program used in testing the overall application.

GameService relates to the Game class in a 0 to many relationship. Therefore a game service can contain many Games. Game relates to Teams which relate to Players in the safe 0 to many relationship. Therefore each Game can contain many teams and each team can contain many players. Each of these Game-Team-Players instances comprise an Entity for the game to utilize with a unique ID to identify it.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Simple server setup with the IOS server and mac servers. Integrates and scales using Xsan platform. Connection for mac clients will be simple and secret keys will need to be used to scale out to others. | Excellent customization and freedom to design as needed with Linux. Cost is lower since there are no licensing feeds. Weaknesses are still that not all developers are able to utilize Linux. Linux tends to be more secure as well. | Cost for windows servers is higher due to licensing but compatibility and integration with other windows devices is simpler. Cost will still be a factor and windows often slows down under multi-database tasking. | Online options are the easiest to use in this situation and many tools within each ecosystem can simplify this process. This will be difficult to implement when it comes to integrating with windows and mac devices as requested by the client. |
| **Client Side** | Mac integration will be doable but less secure when factoring in the safari browser. Security is not the strongest when using a web based application in macOS. | Integration when running inside a web browser won’t be difficult with java being the primary language for the application. API connections could the difficult part depending on the users distribution of Linux | No compatibility opportunities will arise with similar factors that Linux will run into. Testing will need to be done for Chromium browser to ensure defaults browsers but suggested Chrome or Firefox to users can be done if needed to ensure best platform. | Application is currently compatible with android and the porting of it to the IOS system can be streamlined by reusing textures and processes. Development with Swift will be where the process will slow. |
| **Development Tools** | JavaScript, C, C++ ,Objective-C Swift are the primary languages. Development must be done on a Mac unit in order to test. Any IDE compatible with Mac is acceptable. | Python, C++, C, JavaScript, Java. A shorter list would be to name languages that aren’t compatible. Eclipse, Visual Studio, Newbeans for a free solution for IDEs. | Python, C++, C, JavaScript, Java. A shorter list would be to name languages that aren’t compatible. Eclipse, Visual Studio, Newbeans for a free solution for IDEs. | Java for Android App development has already been put in place. Swift for iPhone users will have to be done using a mac device which is not currently available to developers. Swift language will also be an area that has to be learned. |

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**: Given the simplicity of the game a Linux platform can be used to allow for customization across multiple platforms.
2. **Operating Systems Architectures**: Linux is a portable and open source operating system that allows for heavy customization. It is multiuser which will be integral to the roll out of the application. It is compatible with multiple programming languages which will allow for each of compatibility if other languages are needed in the development process. It is also free which will save on up front cost of the operating platform.
3. **Storage Management**: Mobile phones utilize flash storage which will need to be considered when looking at the size of the application. Size constraints will be a concern. Compression will have to be relied upon heavily for these mediums. Storage provisioning across the distributed systems will allow for space management across the servers and machines that are utilizing the application.
4. **Memory Management**: Flash memory is very responsive but application usage will need to ensure that memory is released after each image is shown. This will reduce the need for memory and ensure speed in the program.
5. **Distributed Systems and Networks**: Since the application will be communicated between multiple platforms then it will need to have its services exposed via an API to allow the applications various functions to be called and managed by any service. Java can be managed through each operating system natively so there are minimal concerns for compatibility.
6. **Security**: All connections will need to be sanitized before each entity is written to ensure that any database inputs and calls are clean. This will minimize security concerns and protect the back-end data. Obfuscation of the code will be done to ensure that any attempts to reverse engineer the code to allow for hacking or cheating can limit access to the API calls.