

Draw It or Lose IT

# **CS 230 Project Software Design Template**

Version 3.0

## 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)4

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/29/20 | Ryan Denoncourt | Initial draft of design template |
| 2.0 | 12/10/20 | Ryan Denoncourt | Made minor adjustments |
| 3.0 | 12/18/20 | Ryan Denoncourt | Made minor adjustments to recommendations |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

We will be translating the game *Draw It or Lose It* from an Android game to a web based game. This was originally developed by our client The Gaming Room. This will require a different skill set than what was used for Android app. It will also need to be supported on Mac, Linux, Windows, and mobile web browsers and will need to be hosted on one of those types of servers.

## [Design Constraints](#_2et92p0)

* Need a hosting server
* Needs to support browsers across multiple platforms
* Requires different skill sets and development tools

## [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)

Following along with the UML diagram below, the program will start with the GameService class which can have multiple Game classes associated with it. Each Game class can have multiple Team classes and each team class can have multiple Player classes. The Entity class will hold common attributes between the Game, Team and Player classes. The ProgramDriver and SingletonTester classes will be used for testing. The properties and methods associated with each class are clearly outlined with each class below.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac’s tend to be more of a popular option for the developing side of these programs and not for web hosting since they can be fairly locked down. | These servers tend to be very customizable and lightweight making it a decent choice for Webhosting. | Though maybe not as lightweight as Linux, Windows does offer its own dedicated server operating system that has very good support. | Being a mobile device, these will not end up being a feasible option for hosting the server. We’d want something with more reliability. |
| **Client Side** | Since this is going to be a web based game, there won’t be much catering needed across multiple operating platforms though we will want to test on the most commonly used browsers on each platform like Safari for Mac. | We’ll want to make sure everything works fine on the Opera browser on Linux. | We’ll want to make sure it works fine on Chrome, Firefox and Edge with Windows. | We will have already tested Chrome and Safari which are the most popular mobile browsers but since they have a smaller form factor, we’ll want to make sure it scales down to fit the screens. |
| **Development Tools** | For Mac’s we could use Visual Studio Code which supports the common languages used for web development such as HTML, Javascript, Python, etc. | Visual Studio Code is also supported on Linux but we could also user PHPStorm which supports PHP, Javascript and HTML. | Windows has the option of using the full Visual Studio suite which will allow for a more robust environment for developing and testing. | We will not want to be developing on mobile devices but we will want to test on these thoroughly to ensure a good user experience. |

## 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**: Since there is already a mobile app, I’d recommend focusing on using the Windows platform to reach out to the most non-mobile users.
2. **Operating Systems Architectures**: Though since it is a web-based game, it should expand to the mac and Linux platforms without much of a hitch.
3. **Storage Management**: We can use a cloud based SQL data base in order to store the data. We know that we will also need at least 1.6 gigabytes of storage for the high-definition pictures. Other data should not take up as much space.
4. **Memory Management**: Modern Windows does a good job of dealing with memory allocation for applications. This should not be an issue for Draw It or Lose It. That being said, we will want to sift through the code to ensure we aren’t making unnecessary calls to memory by creating unneeded variables. This will ensure the smoothest possible experience to the end user.
5. **Distributed Systems and Networks**: Using the cloud based data base, we can build different front ends on multiple devices that talk to the same database which will allow us to easily interact with users on different platforms.
6. **Security**: We can setup an https domain which will ensure a secure, encrypted connection. We can also enforce strict passwords on users’ account to make them less likely to get compromised.