

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 | 11/14/20 | Shawn Whittaker | Expanding on initial request to articulate each operating system’s requirements |

**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)

Chat Away is looking to expand their successful social media website to mobile applications. They would like to be in the Apple App Store, as well as the Android Marketplace, in order to expand their customer base and increase revenue.

## [Design Constraints](#_2et92p0)

1) Having to program the same application for two different operating systems.

2) The coordination of both teams to create the same front-end experience.

3) Double the time and resources to make both OS applications.

## [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)

ProgramDriver is the main function and is what calls all other functions to run the game properly. It uses, or calls, the SingletonTester function which supplies data members to be used while calling other parts of the program to ensure proper functioning.

The ProgramDriver will call the Entity function, which holds all attributes and methods for the game. Entity inherits attributes and methods from each of the Player, Team, and Game functions, which all hold their respective information. This is most efficient, since we can edit any attributes or methods used in different functions without affecting the overall function of the game.

The GameService function acts as the menu, allowing the user to start the game, select and edit teams, and edit players. This function shares all it’s attributes and methods with Player, Team, and Game functions, as they all can share information back and forth with each function called.

****

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Flexible terminal commands for configuration | Flexible terminal commands for configuration, but more cost effective than Mac | Quantity and quality of available software compared to Mac or Linux | Not ideal, better if server is immobile. |
| **Client Side** | Moderate amount of time and expertise required, most expensive | Most amount of time and expertise required, cheapest | Minimal expertise and time required, less expensive than Mac | Allows client flexibility, more difficult to implement, but most user friendly |
| **Development Tools** | Javascipt, HTML, CSS, Python.  Visual Studio, Eclipse | Java, Python, Ruby on Railes, HTML. Github, Repl.it, nodejs, Visual Studio | Python, C++, HTML. Eclipse, repl.it | HTML, php, C++, Python. Github, Visual Studio, repl.it |

## 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**: I would recommend using Windows OS, as it provides user ease-of-use, at a relatively moderate price, and a variety of software packages for developer abilities. Compared to Mac’s iOS, Window is more budget friendly and able to provide a wider range of software compatibility. Compared to Linux, Windows is less budget friendly, but the cost far outweighs the impracticality of labor needed to efficiently run Linux.
2. **Operating Systems Architectures**: Windows provides a plethora of software engineering abilities, Eclipse, Microsoft’s Visual Studio, allowing programming in C++, Java, and Python, which also allows the program to be run on MacOS as well as Linux. As we further explore store and memory management below, we will be able to examine the architecture we can provide to the Gaming Room. Our Window’s run architecture allows us to split the application between User-mode and Kernel-mode. At CTS, our server will run the kernel-mode, allow user’s easy access into the user-mode, more on this will be discussed under distributed systems.
3. **Storage Management**: I would recommend using a Window’s based server storage system, as although it is not quite as flexible as Linux servers, it is easier to maintain than Linux and more cost effective than Mac. Here at CTS, we can provide long-term storage of the program on our server. This provides kernel-mode support in terms of file systems and allows secure storage while simultaneously giving the most efficient use of storage resources.
4. **Memory Management**: Along with developing in Windows OS, we will be able to utilize C++, which is able to manage large amounts on memory usage efficiently. At CTS, we can provide effective memory management, as we are able to provide the user with a swift load time and running of the application smoothly and effectively. As we utilize our distributed system, we can split the memory resources being used between the user’s machine and our own servers.
5. **Distributed Systems and Networks**: Developing this game on Windows OS, and using a Windows OS server storage system, we would be able to utilize a client-server distributed system. This allows us to format the data, store it on the server, and display it back to the end-user seamlessly between MacOS, Linux, and Windows users. This would also allow changes to be made from the client and commit back to the server.
6. **Security**: Utilizing our client-server distributed system, we can provide encrypted exchange between server and client, which would optimize security. With the use of the distributed system, we can hold and application long term in our safe storage servers, as well as run them within our own memory. The user, once logged in, will be able to access this information, but will not be able to change any part of the application, or access any unnecessary data within the storage. This provides a far more secure method of running the application.