

<Win, Lose or Draw>

# **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 | <05/24/2020> | <Eric Engman> | <Software Design Review> |

**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)

<The difficulties when developing a web application is starting with a good foundation to build on. That is what we’ve accomplished here setting up your team with a strong base to later build upon.>

## [Design Constraints](#_2et92p0)

<With developing a web-based environment you will be constraining your users through their internet connection. This essentially means that optimization is key when creating a web-based application.>

## [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 UML diagram below exemplifies the structure for the game’s environment. With the Superclass being Entity, you have the different subclasses that inherit from it being Game, Team, and Player. In those subclasses you have methods for creating different instances of those objects each with custom characteristics. Storing the teams and players in lists you are able to check for duplicate team or player names when creating a new team or player. >

****

## [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** | <Terminal commands allow for plenty of configuration to the server.> | < Similar to Mac but typically more cost friendly.> | <Most open source compared to other OS.> | <Not exactly the most optimal solution.> |
| **Client Side** | <Moderate expertise and time required, cost similar to windows.> | <Typically, the most expertise and time required, lowest cost.> | <Least expertise and time required; cost is similar to windows.> | <Slightly more difficult to implement than other devices, but it provides flexibility to both clients and developers to see updates at any place.> |
| **Development Tools** | <Most documentation would likely be an XCode and Swift or Objective C combo.> | <Typically programmed in C and can utilize various IDEs and other tools.> | <Plenty of choices in windows for your language, typically what you are most comfortable with. Likely using Visual Studio Code for your IDE, very well optimized environment.> | <Depends on the mobile device. iOS is very prominent, and similar to deploying on Mac, using XCode and Swift to develop this app would be a good idea.> |

## 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 recommend iOS due to the broad userbase, as well as the ease in development. Something to consider is using Flutter in development. Though Xcode is typically the ideal development environment and Swift is the ideal iOS language, Flutter is a growing platform that can support building both iOS, Android as well as (not refined, but there) web applications. This could speed up moving to different platforms if that is ideal.>
2. **Operating Systems Architectures**: <The iOS architecture is layered and contains an intermediate layer between the applications and the hardware, so they do not communicate directly to give a more secure environment.>
3. **Storage Management**: <For storing the 200+ 8mb photos I recommend utilizing Google’s Firebase Storage platform. Due to the ease of use and the up to date documentation I think that will be the fastest way to integrate an online storage management system. A close second is AWS S3 bucket system due to its lower cost. Though the drawbacks is the ease of integration with iOS due to the more outdated documentation that could prove a difficulty in getting off the ground with your online storage management system.>
4. **Memory Management**: <iOS memory management is now ARC (Automatic Reference Counting) so you don’t have to retain and release the objects, it is handled for you automatically.>
5. **Distributed Systems and Networks**: <Here is one point that I would recommend AWS over Google. AWS was essentially first to the cloud computing game and has plenty of advantages over Google and Microsoft in both capabilities like setting up virtual servers and EC2 instances as well as the cost is more favorable using AWS services.>
6. **Security**: <Here I would recommend Google for the quick integration and set up. AWS can be very complicated when it comes to security, but Google provides a very helpful backend user interface to manage its users and security levels.>