

The Gaming Room

# **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 | 1/27/21 | Payton Mitchell | Designing the Gaming Room Software |

**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 software that we are designing here is a game that allows the users to guess what is being drawn. The game will pull from a database of stock drawings that will steadily become more and more visible. The users of the game will have 30 seconds to guess what the image is and if they are unable to guess it the opposing team has 15 seconds to try to guess what the image is. What we are trying to build here is a web-based version of this gaming application. This can be done on a multitude of web-based frameworks. The framework that we are choosing here is the unity game engine.

## [Design Constraints](#_2et92p0)

First off, the main requirement is that we are making a web-based game. Some other constraints include being able to have one or more teams involved. Each team must have multiple players, game and team names must be unique and only one instance of the game can exist in memory at a time. This means that we will need unique identifiers for each instance of a game, team or player. One thing that we might want to take into consideration is that the quality of our servers. Our servers need to be able to support all of the players so server traffic needs to be monitored.

## [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)

Here we can see that the singleton pattern is being used for the Game Service class. As we were told in the rubric, there can only be a single instance of the game in memory and this is how it is implemented. The game service class has a list of games, variables for next id of players and teams, methods to get all of these variables, a method to add a game and a variable for a game service. The game service method inherits from the game class. Within the game class there are a list of teams. The game class also has a constructor that has a string and id argument. This is inherited from the entity class. There is also an add team method and a to string method. Next, we see that the game class inherits from the team class. The team class has a list of players, a constructor, an add player method ad a to string method. The team class inherits from the player class. The player class is pretty small with only a constructor and to string method. The player class inherits from the entity class. So basically, the game, team and player classes all inherit from the entity class. The entity class has the id and name variables that show up in all of those other constructors. It also has a no argument constructor, a constructor that takes in two arguments, a to string method and getters. Here a big OOP principle that I see is inheritance and encapsulation. We see that the entity class is using encapsulation because of the fact that they have getters for the variables. Also, inheritance because so many classes inherit from the entity class.

****

## [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** | When using a Mac, we have a few options to deploy this web-based game which include Apache which is built I to the system and also AWS. Some potential costs are the AWS fees which charge based on traffic. I would like to add that the built in Apache server is free of cost. | Linux is actually known for its servers and are in fact built into the Linux operating system. Linux has many different types or “flavors” of the operating platform based on purpose. One of the big pluses when it comes to Linux is that it is free. Some companies that require premium support however can pay up to 4000 a year. | Microsoft Windows does have a server program available to users. One big advantage to Windows servers is that it can be integrated with Azure which is Microsoft’s cloud-based platform. Costs of Windows servers ranges depending on the program. The highest reported costs can be up to 6200. | In terms of mobile devices, we know that these will be hosted on the iPhone app and the Google Play stores. To be able to put an App on the iPhone store one must have a membership which costs 99$. For the Google Play store has a one tie fee of 25$. |
| **Client Side** | For our game to be compatible on the Mac we simply have to design our program in JavaScript. If we develop our program in JavaScript and deploy it on one of our server options, then it should work on Mac devices. In terms of cost the average cost of a JavaScript developer is 35 per hour. In terms of time I think that this should take about 4 to 6 months. | Again, when looking at how to make the game compatible on Linux we will be able to access it if it was developed using JavaScript. We could use other technologies but given the popularity of JavaScript this was my choice. In terms of time we simply have to develop that first web application that will take 4 to 6 months. For example, if we deploy our application on Mac servers, we will be able to access it on Linux devices. | When looking at time and cost of making the game it will take the 4 to 6 moths as I said before. The cost would be depending on the deployment method and the cost of the JavaScript developers which I think will be 3 developers or more. In terms of cost I think that we should deploy the game on Linux servers because of the low costs. | The average salary of an IOS developer is around 94k, which is a cost that may have to be multiplied depending on how many developers are required for the game which I estimate as no more than 2. To ensure that the game is compatible on all IOS devices we have to simply deploy it onto the App store. In terms of time developing the IOS application should not take more than 2 months because of the fact that we already have the Android template. |
| **Development Tools** | Some IDE’s for developing web applications using Mac include IntelliJ, Atom and Visual Studios. There are many supported languages of these technologies which include Java, Swift and Python. In terms of licensing costs of IDE’s there are yearly costs of Visual and IntelliJ. There will likely be a single development team for all pc platforms. | Developing web application on Linux can be done with a number of programming languages like PHP, C# and Ruby. Some of the IDE’s include Visual Studio, Atom and WebStorm. In terms of costs of IDE’s, the only that I see here is the yearly fee for Microsoft Visual studio. This will be a part of the pc development team. | Windows web applications can be built using one of the many java script frameworks available. Some IDE’s include eclipse and Atom. In terms of developing teams, I believe that for all pc platforms we will need a single development team. Fees associated with IDE’s include IntelliJ and Visual studio yearly costs. | When developing for mobile devices there are basically two options which are Android and Apple. Android is developed using Java and Android Studios. iPhone apps are developed using XCode and swift. This will be a separate development team than the pc development team. This team will most likely be a smaller team than the pc. In terms of cost of IDE’s both android studio and XCode are free on their platforms. |

## 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**: The server platform that I feel would best support the gaming room is AWS. Amazon Web Services scale with the application so if we were to get a sudden spike in users because of the release of the pc platform, we would be able to handle it. AWS is compatible across all platforms. This also charges as you use it so if we do not have a lot of user s initially we won’t have to worry about cost of using a not needed server.
2. **Operating Systems Architectures**: Using windows, the development platform that I recommend using is Microsoft Visual Studio. I chose this because of it’s support of JavaScript and popularity with web-based applications. Also as stated before this will be hosted on AWS.
3. **Storage Management**: The storage management system that I would recommend using for this game would be MYSQL. This is the case because of its popularity and speed. This database management system is compatible with Windows and all other operating platforms.
4. **Memory Management**: Our windows operating platform will pull whatever images from the SQL server and render them to the user at a steady rate.
5. **Distributed Systems and Networks**: We know that this program will communicate to multiple users or platforms at the same time. This will be accomplished through using Amazon servers. We will be using AWS to deploy our application which means we will not have to worry about things like outages.
6. **Security**: We will protect user information and the game as a whole first by using best programming practices. This means using encapsulation so that access to user information is limited. We will also implement JavaScript security APIs that are known to be very reliable. Also, we will require each user to make a password that has a very strong minimum strength.