

Creative Technology Solutions

# **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)2

[**Design Constraints**](#_2et92p0)2

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)5-6

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/20/2020 | Garrett Stubblefield | Initial documentation |
| 2.0 | 06/7/2020 | Garrett Stubblefield | Evaluation |

**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)

Here at CTS we are thrilled to be working with the Gaming Room to set up the environment for the game Win, Lose, or Draw. Since the game is based off the tv show we know how the game will look and feel. Bringing that nostalgic feel to each player will be at the core of how we want the game to feel. The design of the software is where we will be doing most of our work setting up each game’s functionality. The game will consist of unique instances of the game, and each player and team will have unique attributes that cannot be duplicated.

## [Design Constraints](#_2et92p0)

For the software design constraints will be focused around having a single instance of the entire game service. The game service will hold the id numbers that will be unique for each game, team, and player. While the games, teams, and players will all have unique id numbers we will also make sure that when they are named that they are all unique names not to be duplicated. A game will be able to consist of any number of teams competing against each other. Each game can add more teams throughout the process. For each of those teams there will be multiple players with no limit, but again each player will have a unique id and name. To be able to put these across the different web-based environments we will need to work with an IDE that can accommodate Window, Linux, and the Mac OS. In the past we have worked with Xamarin which has been able to not only create the application for the web, but also gives us the power to put it onto the different mobile app stores. This program uses C# and will automatically format it into HTML5 so it will work in all the different web browsers.

## [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)

This software will be made up of 7 different classes each of which will have their own functionality. The Entity class is the parent class for Game, Team, and Player class of which they inherit all the Entity classes attributes. The Entity class has the attributes of a name which is held as a string and an id which is held as a long. This lets the other classes stated before inheriting the ability to use those attributes in each instance of the class. The Game class holds an instance of one game. This class has the inherited ability of the name and id from Entity. The unique attributes of the Game class are a List which holds all the teams that are associated with the Game instance. The Game class can add a team and make sure that the team and print out all the teams in the game. When we add a Team, it comes with the unique attributes inherited from the Entity class which are a name and id. For each team we can add unique players to the players List in the Team Class. Each player also come with a unique name and id. The only attributes the Player Class has is the name and id inherited from the Entity Class. The largest and most important class is the GameService Class. This class will hold each game object in a games list. The GameService Class does not inherit from the Entity class because we only want a single instance of a GameService object. This is done by using a private constructor, the service method, and the Singleton Pattern which is used in the TestSingleton Class. In the GameService class is where we hold the next id number for the Game, Team, and Player objects. This makes it easy to make sure that when any new object is created that its id number is unique and helps show how many objects of each type have been created. The single instance of the GameService class is first initiated in the service method and will always return that single instance. The entire program is ran in the ProgramDriver class which will drive the main method.

****

## [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** | Mac was built focusing on graphical user interface so whatever our graphics will be will look best on the mac. Mac is not as common as Windows and should not take up as much time. | Servers using Linux as its OS can be useful because Linux is a free to use OS, but Linux is typically in flux and you will want to be sure how the version you use will be supported and for how long. | Windows Server OS generally will cost more money, but they get better support depending on what language you use. | Mobile devices will need to receive different views because of the smaller screen but will also use touchscreen to interact so will give an easier user experience. |
| **Client Side** | Using a REST API all the computer web-based sites will be made as together and we will have to tweak how the mac OS sends out the requests to our servers. This should not take much time, but a Mac OS expert will be necessary on the team. | Linux web interface is extremely interchangeable and sending requests should be simple on a website using Linux. Time and cost should be low but would be nice to have a Linux expert on the team. | Windows initial setup with REST API is already complete. Since we are using rest the servers will be doing most of the work and each client will interact with each server. I feel it would be best to work closely with the design trying to create a good UI. | A mobile app will be the most time and cost dependent form of the web-based application as the phone will do most of the work and not the server. The app will still need to be getting the game instance with all the other players moves to ensure a good connection. |
| **Development Tools** | For Mac we will be using one IDE to make the REST API so that the browser for the web application can communicate with the server and display the correct information. The IDE NetBeans should work great but does cost money to use. The newest versions of Java and the JDK will need to be installed. The security features will be used by MySQL. Jersey and JSON will also be incorporated with how the data will be requested and received. | With Linux we will be using one IDE to make the REST API so that the browser for the web application can communicate with the server and display the correct information. The IDE NetBeans should work great but does cost money to use. The newest versions of Java and the JDK will need to be installed. The security features will be used by MySQL. Jersey and JSON will also be incorporated with how the data will be requested and received. | For Windows we will be using one IDE to make the REST API so that the browser for the web application can communicate with the server and display the correct information. The IDE NetBeans should work great but does cost money to use. The newest versions of Java and the JDK will need to be installed. The security features will be used by MySQL. Jersey and JSON will also be incorporated with how the data will be requested and received. | To create a mobile app, we will wait until the web-based side is done. By doing that we can use an IDE like Xamarin that will allow us to take the code we already have, and the IDE will do a lot of the work to set the foundation of the application. This will need some tweaking and changing of the security but should save some time since the IDE will be doing some of the work for us. |

## 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**: For the Operating Platform to work over various end user platforms I would recommend using AWS. AWS will give Draw it or Lose the flexibility to grow at any rate and keep the costs at the same rate for every user. This scalability will be vital for the overall success of the game.
2. **Operating Systems Architectures**: The Overall System Architecture for this game will use some RESTful API to request information from the servers which if we used AWS then we would be using a cloud-based server. The information the users would be getting back would be small computations of the score, team/player names and id’s, pictures, and the answer to the player.
3. **Storage Management**: The AWS servers will act as the main storage for the code base, but also will need to keep the game instances data in there. The data that AWS servers will store will be 200 images and the string answers, the unique game id’s, the team names and unique id’s, the players name’s and id’s, the score, and also a rendering image microservice.
4. **Memory Management**: When it comes to memory management, I believe we have created an extremely efficient application and our computation costs with AWS should be minimal. When a game instance is requested the user’s device will given 4 pictures for the 4 rounds of gameplay, the string answers to those pictures, the other team names, and a image rendering microservice. The actual methods of running the game will also be sent and cached onto their running memory. This will make it so while the games are running there is not a tremendous amount of data that is being sent back and forth running up our AWS server computations.
5. **Distributed Systems and Networks**: Since we will be using AWS servers the communications done between the various platforms will be done with REST API. The users of the varying devices will request to join a game by id or name or start a new game instance. This will mean that the send a request to the server and will get back the appropriate data discussed above. When a round is finished the users will request an updated score from the server and all the users will see this happen. By sending all the larger data at the start of the round it cuts down on the computation time of the servers and allows to free up space quicker. When you are only storing the strings, and long ints it keeps the data to a minimum.
6. **Security**: Security is a huge problem with many programs that contain sensitive information. As of now we do not ask for any sensitive information and the names that the players put in will not be kept after the game is over. We will not be having logins as of now so our security should not be that extensive. AWS has secure servers so we should not be too worried about someone having direct access to our code base. We will still be adding in cryptography to secure our users IP addresses and their interaction with our servers. By using encryption it will give our users a sense of security and allow them to just enjoy the game. As the game grows we may want to add logins and more security features.