

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/16/2023 | Tyler Pimental | Initial Update |
| 1.1 | 07/29/2023 | Tyler Pimental | Updated Server/Client & Dev. Tools |
| 1.2 | 08/13/2023 | Tyler Pimental | Updated 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)

CTS would like a web based game to be developed. This game include a game where images teams take turns guessing what image is being slowly revealed before a timer expires. Each team will have more than 1 player assigned to them. Game and team names will be completely different. Players will be able to check if the requested name is in use. With this, only instance of the game will exist.

## Requirements

* Games will have one or more teams playing
* More than 1 player per team
* Game names will be different
* Players can check if team name is taken
* One instance of the game will exist in memory

## [Design Constraints](#_2et92p0)

This game will be developed for a web environment. Due to this nature, the game will be played on devices that have an internet connection and must be optimized accordingly for iPhone, Android, Mac and Windows devices. There is no defined budget or timeframe for this project, however these will still be kept in mind.

## [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 program will utilize OOP with its GameService and Game classes. The relationship here is a 0 to many, with only at most one GameService existing. From here, any game can have 0 or many teams. Any team can then have 0 or many players. All three of these classes will extend Entity and inherit its parameters. Our ProgramDriver keeps the entire application alive as well."The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.

## [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** | Advantages – Strong hardware and provides excellent security.  Disadvantages – Ecosystem is closed and can be quite expensive | Advantages – Open source and very configurable  Disadvantages – Strong learning curve and possible compatibility issues. | Advantages – User friendly and extremely compatible. Disadvantages – Well known exploits can exist as well as performance may take a hit | Advantages – Widely popular  Disadvantages – Minimal hardware power and battery capacity. |
| **Client Side** | Advantages – User friendly UIs and trusted environment/ecosystem  Disadvantages – Requires knowledge, licensing for application development | Open source and widely used. Strong expertise is required in order to develop reliable and secure environments. | Windows environment is widely known and has many tools to ease cost and time. May need extra time and cost to ensure proper security is put in place | Minimal cost and effort required for mobile environment. Depending on the vendor, may need additional time/effort |
| **Development Tools** | Utilizing popular products, JavaScript and visual Atom IDE would be best pics. Adobe products can be used for graphics as well | If possible, utilize Bash and built in CLI editors such as VI or nano. | Utilize the .NET framework and visual studio suite | Options here are somewhat limited. HTML is a realistic language which can be developed in any cloud accessible editor |

## 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**: Based on the criteria given for this project, I believe utilizing a UNIX based server would be great. Linux flavors give you the option to fully automate deployments and environments with ease. Since there will be no real domain in this scenario, I don’t see a full-on windows server suite being entirely needed. This will save on licensing costs too. There will be a slight learning curve to learn the needed syntax and operating environment, but once configurated correctly, most tasks will need very little administrator involvement.
2. **Operating Systems Architectures**: We are told the staff at the Gaming Room does not know how to set up the environment. By utilizing a cloud-based environment, this will take the burden of physical hardware off the team. Additionally, this will support the team's future growth potential by easily scaling out servers if needed. By using cloud architecture, we can also leverage the ease of networking aspect as well. Deploying firewalls and routes can be very complex for a team. Providers such as Microsoft Azure make it painless to click and deploy these resources and use recommended configurations. This will play a significant role with this project, considering clients from different devices will be connecting to our main game server.
3. **Storage Management**: Since we have opted for a cloud environment, it would be efficient to also host our storage solution in the cloud. The storage requirements for this project are not too intense and we could probably get away with a mechanical drive solution. This will save on costs since we are only using a few Gigabytes to store our images. Additionally, we will need to implement an efficient storage method on end user’s devices for once images are no longer required. For our files that will reside in the cloud, we can utilize our Cloud Provider’s built-in recovery services to handle any event of data loss. Also, we can implement a hot tier for our storage, since most of our storage will need to be readily accessed.
4. **Memory Management**: We already are at an advantage by using a UNIX based system in our design to host our main processing server. We can opt for a headless deployment, meaning there will be on graphical user interfaces. This will conserve on memory usage and allow us to allocate more memory for the actual workload. Additionally, by using a cloud environment, we can easily scale up and down based on the needs of the player or the time of day. This will ensure we stay cost optimized and performance optimized.
5. **Distributed Systems and Networks**: Using a cloud environment will allow us to create seamless connections between our clients. Whether we choose to keep those clients on android only devices or move to a more cross-platform approach and include iOS or even desktop operating systems. Will need to ensure the code can be cross platform and will interface find across all layers of the OSI Model. This could include a standard HTML and JavaScript domain where it is known to be very reliable.
6. **Security**: While hosting our game, our largest concern would be keeping our source code safe along with our player’s data. We would need to utilize all industry standard protocols and procedures. This will include any front facing web addresses to use HTTPS vs HTTP to ensure a secure connection. For our administrators, they will need to be using SSH over telnet to log into any servers, and ensure proper AAA is handled to make sure the right personnel with the right permissions are accessing the correct resources. Unused or dangerous ports need to close and monitored by firewalls and antivirus software. Additionally, all data at rest should be encrypted to ensure no data is stolen.