

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 09/17/23 | Sean Flynn | Updated to include Executive Summary, Requirements, Constraints and Model |
| 2.0 | 10/2/23 | Sean Flynn | Finished recommendations as per project 2. |
| 3.0 | 10/17/23 | Sean Flynn | Finished as per project 3 |

**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 Game Room is looking to set up an environment for their web based app Draw It or Lose It. The game will render images over the course of 30 seconds that will be used as clues while teams of players guess what the image is. Games will last for 4 rounds each being a minute long. The game is currently on Andriod only but The Game Room would like to develop a web based version to serve multiple platforms.

## Requirements

* The game will allow for 1 or more teams.
* Each team can have multiple players.
* Game and team names must be enforced to be unique.
* Only one instance of the game service can exist in memory at a time.

## [Design Constraints](#_2et92p0)

The game service will only ever have a single instance, this means that a singleton design pattern should be used to enforce this. Also it is important that both teams and games are given unique names. For this we should use an iterator pattern to review the list of both when a nee game or team is made. The game service is what will hold both game and team information.

## [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)

Entity will be the super class from which Game, Team and Player will all inherit from. As each of those three objects are entities that share overlapping information such as name and id. The GameService will then be created to manage the entities. There will only be one GameService which will hold a list of the created games. The players will be added to teams which will be added to the games. From there the GameService will be responsible for managing the games.

"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** | Built on Unix so it is powerful and stable. Easy to use and supports other Apple products. Not great for large or high performance needs. Expensive. | Open source, has many different distribution options. Powerful command line functions. Very scale-able. Has a steep learning curve. | Windows serves can have high licensing costs as well as require a lot of resources. They are very compatible for web based applications and work well with other Microsoft products. | Mobile devices are not suited to this task. |
| **Client Side** | Macs are more expensive than most of the other options and a smaller share of the market when it comes to gaming. | Linux has a steep learning curve and does not support as many libraries as Windows or Mac when it comes to game development. | Windows computers are more expensive than Linux. Windows would have the most access to tools geared towards game creation.  This would be the best platform to create a game for. | Mobile devices would be able to reach a vary wide market. They have plenty of tools available for development and have easy access to a market for distribution. This would be a good choice to develop for. |
| **Development Tools** | Many web based tools available on Mac will also be available on Windows when it comes to development. Including Java, Python, JavaScript, and IDEs such as Eclipse. | Linux has access to many of the same languages and IDEs as Windows and Mac though some people find programming on Linux computers more user friendly than Windows due to it’s robust command line features. | Many web based tools available on Windows will also be available on Mac and Linux when it comes to development. Including Java, Python, JavaScript, and IDEs such as Eclipse. | Although mobile devices are good for clients they would not be used to develop. There are many tools to develop software that runs on mobile apps though such as: Swift for iOS, Flutter by Google, and Ionic which is a frame work for using web tech like HTML, CSS and JavaScript. |

## 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 believe the best operating platform for The Game Room to use would be a cloud based service, in this case I would recommend Google Cloud Platform. This would allow us to use other Google services to manage things like storage in the same place.
2. **Operating Systems Architectures**: Google Compute Engine uses virtual machines that are well suited to running web based applications. They work well for inter platform communication, and are designed to work with many different APIs. In our case we would use a x86-64 system architecture.
3. **Storage Management**: I think the best storage option for the web based client would be a cloud storage option. I would recommend Google cloud storage because it is highly scale-able and would save on costs. In addition it would mesh well with the choices made above. Because we are planning to use a Microsoft system with x86-64 architecture we will need to make sure that our files are compatible with that.
4. **Memory Management**: Google Cloud Platform uses several different tools for memory management. They offer powerful optimization tools as well as monitoring. There are also load balancing features that are well suited to handling various levels are traffic. Because we are using a cloud based system this problem should be mostly solved for us.
5. **Distributed Systems and Networks**: The GCP is a very reliable service meaning we can expect little trouble with downtime or outages. Google already has a global network infrastructure tied to data centers meaning our product would always be online and usable.
6. **Security**: Because we would be using a VM provided by GCP we would also be taking advantage of the security features that come built in. All data is encrypted and the virtual machines can be individually scanned. Also individual roles can be set up allowing users to create profiles through the IAM interface easily restricting who has access to data. The Security scanner is designed to look at web based applications and assess any security flaws before the product is launched and after any significant changes. I addition on our client side we will be using user authorization through a user name/password setup.