

Draw It or Lose It

# **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 | 8/9/2023 | Johnny Segura | Adding recommendations to development process |

**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 Gaming Room is a company that has a game called “Draw It or Lose It” that is only able to run on Android platform. Multiple teams with multiple people on each team go through 4 rounds of play lasting 1 minute per round. An image is selected from a library of stock images that provides hints to a person place or thing. 15 seconds are provided to the remaining teams in the event the person place or thing is not guessed in the allotted time.

## Requirements

*We must create a way for the program to run across multiple platforms. This can be accomplished by writing the program in different languages that are compatible with other systems such as Apple, Windows, or Linux.*

## [Design Constraints](#_2et92p0)

* Needs 1 or more teams with multiple people on each team.
* Since game and team name must be unique, create alert if name is already in use.
* Only 1 instance of a game can exist at a time.
* Must run on Android as well as other platforms.

## [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 creates a relationship between Game, Team, and Player classes. Meaning they all inherit information from Entity. Each class will share “name” and “id” references. Entity therefore is a superclass. Team and Player is a “has a” type. Game has a Team and GameService has Games. This is called aggregation in UML. Based on the diagram GameService references Games, Games references Team and Team references Player.

**"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** | Easier to make changes or access the server with a more flexible terminal. Popular option with web hosting.  Various options for web hosting. But not the preferred option. | Linux has very similar benefits to Mac but less expensive. Preferred option among developers.  Security risks are spotted and addressed more efficiently. Most popular option with web hosting services.  Disadvantages are that it is more difficult to find apps to support web host requirements. | The most software options compared to other operating systems. The most versatile of all platforms.  High resource requirements, less load time and most user friendly.  Susceptible to viruses and weak tech support. | Best when the server can be tracked in a single location. Specs not as good as in other devices.  More popular and portable.  The advantages are cost-effective and have more compatibility.  Selective to several smartphones and mobile devices. Poor security. |
| **Client Side** | Moderate understanding and time are required.  What must be included to ensure the application can run across multiple platforms and mobile devices. | Most expertise and time are required of all the options.  What must be included to ensure the application can run across multiple platforms and mobile devices. | Least amount of expertise is required. The cost is similar to Mac.  What must be included to ensure the application can run across multiple platforms and mobile devices. | Flexible option for clients and developers. Not as easy to implement as other devices. |
| **Development Tools** | Swift can be used to run programs on Mac. Some language examples that can be used are HTML/CSS/JavaScript. Front-end processes can be created using Python, Java, or Ruby. | Linux is compatible with Visual Studio, Eclipse along with many other tools and languages. Some examples of languages are HTML, CSS, and JavaScript. The front end can be created using Python, Java, or Ruby and more. | More user-friendly than Linux but has similar functionality. Can support Eclipse and Visual Studio.  Some examples of languages are HTML, CSS, and JavaScript. The front end can be created using Python, Java, PHP or Ruby and more. | Android and Swift have a limitless creativity options. These can also run on any of the other platforms.  Some examples of languages are HTML, CSS, and JavaScript. The front end can be created using Python, Java, PHP or Ruby and more. |

## 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 Gaming Room should start with Windows since it is the easiest to use and most versatile. This option only has the lowest cost with several options for development environments. Coding languages that can be used across multiple operating platforms like HTML5, CSS, and JavaScript is also a great approach to take to support cross platform development. Using platforms with larger development communities is a great way to promote collaboration.
2. **Operating Systems Architectures**: Code that can be used across multiple platforms is the most efficient approach to minimize time and resources for the development process. Client-server architecture is the best approach for this game since it is a great option for games that require real-time interaction. Proper testing can also ensure code runs smoothly across different platforms.
3. **Storage Management**: Several storage options and features such as the Cloud and folder menu making it easier to keep track of the location of files. The cloud would be the best storage option since this makes it easy for The Gaming Room to add storage as needed. This option can also help to prevent performance issues.
4. **Memory Management**: Storing files such as the pictures for the photo library can be easily created and accessed. This makes it simpler to use them in different development environments. Caching frequently used data can also reduce load time and improve execution.
5. **Distributed Systems and Networks**: Develop 4 enables cross-platform game creation. This development environment can run on any platform to allow for play on multiple devices. Strong servers will help to handle a high volume of users at one time. Using load balancing can help to improve performance.
6. **Security**: Windows has several options both pre-installed as well as purchasable virus and malware protection and other security threats. Frequent updates to this security software help to keep information safe. Encryption can also help to prevent unauthorized access and modification of data on the server. Frequent vulnerability testing can also add security to the program.