

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  2.0 | 03/20/2024  04/07/2024 | Jax Francis  Jax Francis | Information was placed in the executive Summary.  Create evaluation table. |

**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)

“Draw It or Lose It”, is an Android game that consists of teams competing to guess what images are being drawn. The game consists of four rounds of play. There is a guessing time limit, and all rounds last a minute each. This product is created “The Gaming Room”, which is a new client of ours. Their desire is to make this web-based application accessible across multiple platforms. Hoping to allow accessibility for all game users to compete amongst each other simultaneously. For Example, Operating Systems that our client would the product to be compatible are MacOS, Linux, Windows, etc.

## Requirements

* The ability to have one or more teams interacting.
* The ability for each team to have multiple players assigned.
* Unique name for players and team are allowed only if the system deems the name unused.
* At any given time only one game can exist in memory. This can only be accomplished by creating unique identifier for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

## Storage and Cross-Platforms – Each platform must be able to store the team and players. Plus hold them information for each team and players Cross-platforms.

* **User Interfaces for the Games, Players, and Teams** – They are already set for android application. So, the same type of interface must be created to handle multiple platforms. Along with that a management system should be created for the games, players, and teams.
* **Cross-Platform Support** – Android application must be converted into and/or compatible with other operating systems, such as MacOS, Linus, Windows etc.
* **Login IDs and Usernames** – Both IDs and usernames should be uniquely created. They will exist in memory after being verified through a non-duplication verification process. Which means no one user will have the same ID or usernames as another.
* **Security, Copyrights, and Languages** – Platforms must be secure for all user insuring informational privacy. All images must be original in the games or has been given permission from other companies. The language that the game is written in will be Java.

## [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)

The Unified Modeling Language Diagram of the “The Gaming Room” below shows the Entity, which is the base building block of the game. Everything from the players, teams, and games starts at the Entity. The Game class is the setup, which includes everything needed to play the game, such as which teams are playing. The team class makes sure each team has players, it also keeps track of them, from team names to players name on each team. The player class is designed to hold each player to their team. Everything is connected like a reversed nesting doll. From the Entity everything is placed into each other until there is one whole game. The entity and GameService holds much of the game information. The GameService class holds everything that makes up the game. It is the top of the tree. The next and the game class holds all the game information. The Team class holds all the players information, and the player class holds all the player information. The programDrivers class works with the SingletonTester, so this means that the development of these classes restriction of having one instance where the game can exist in memory. So during the development process the diagram below will be the final product blueprint.

**"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** | **Pros:**   * Easy to use server. * Strong Security proven * MacOS software is extensive documentation. * The products are next level with and excellent technical support plans.   **Cons:**   * Hardware is limited options. * MacOS can only be used by mac systems. * Apple are sole updates through Apple. * Payments are required for new software. * Expensive products. | **Pros:**   * The software is open source available. * The cost of products and software is most effective. * Strong stability proven. * Constantly updates for software and security.   **Cons:**   * Compared to other platform options there is not much software. * Too cheap in product quality. | **Pros:**   * Widely supported software range. * Hardware is moderate priced and easy to obtain. * Userbase is large than most platforms. * The server base is simple.   **Cons:**   * Less secure as its competitors. * Microsoft solely updates. * Mainly android based applications. | **Pros:**   * Code from these devices is not seen by user. * Data is persistent by stores. * Database queries are optimized for managing call to server side.   **Cons:**   * Physical servers and cloud services are needed for sever side. |
| **Client Side** | **Pros:**   * Web browsers are widely supported features for many tools and device. * Software testing cross-browsers are easy. * Deployment and Development time is Medium.   **Cons:**   * MacOS is required for Apple produces. | **Pros:**   * Web browsers are widely supported features for many tools and device. * Structure accommodation file tree. * All web browsers compatible because Plethora of open-source software. * Deployment and Development time is quick.   **Cons:**   * Hard to learn with no experience. * Most favored application may not be supported. * **Converting from one platform to the next may cause issues.** | **Pro:**   * Web browsers are widely supported features for many tools and device. * Cross-platforms are easily tested by sides MacOS. * Deployment and Development time is quick.   **Cons:**   * Has difficulties testing MacOS browser. | **Pros:**   * Android application development expertise.   **Cons:**   * Browsers are difficult to test. * Development time is lengthy. |
| **Development Tools** | Pros:   * MacOS deploys the documentation. * Windows and Linus can be easily runs thru local virtual machines.   **Cons:**   * All application must be approved by apple. * Eclipse must be used for to run java. * Apple Dev program is $99/year. | **Pros:**   * Can be deployed at any time. * Windows and Linus can be easily runs thru local virtual machines. * The community is an open source. * The licenses have not cost.   **Cons:**   * Eclipse must be used for to run java. | **Pros:**   * Can be deployed at any time. * Through VM Linux can run.   **Cons:**   * Eclipse must be used for to run java. * To run java/HTML, visual studio code is needed. | **Pros:**   * IOS is deployed thru XCode 12.   **Cons:**   * Apple Dev program is $99/year. |

## 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 operating platform that I would consider would be Windows. I believe this will expand “Draw It or Lose It”, because it is a strong in game development, widely used by developers, and is one of the most popular platforms to become familiar with. This would make getting the game set up easy for most players.

1. **Operating Systems Architectures**:

User mode and Kernel mode are the operating systems that Windows is separated into. User facing and the amount user interface with is the User mode processes. Networking, memory management, inputs and outputs are all involved in the kernel mode processes. Holding data is the directory structure that Windows utilizes. Customization of the systems are Windows supported through multiprocessing and modularity of hardware.

1. **Storage Management**:

The Cloud storage unit is recommended because the product uses a web-based application. By using the cloud, the application can easily run, and software update can be stored. The storage space can also be upgraded and expended if the product becomes more popular. Using google cloud and iCloud seemed to be the best fits for store the games data.

1. **Memory Management**:

Java will be used to as the main language used for backend development. Linux is the best memory management, because of its customization, versatility, and easy operating systems. Java has a memory management system within itself. “Garbage collector” is the name of the Java memory management system.

1. **Distributed Systems and Networks**:

Google chrome browser is available to all existing platforms. The game is web-based so it can be running OS systems. But because of google the OS system will be unnecessary.

1. **Security**:

Using google cloud services allow the insularity of data center security. Hardware and infrastructure are also provided by google. Time to time patches and updates are being made by using google apps. Before deployment a QA testing is ran.