

Draw It or Lose It

# **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)3

[**Design Constraints**](#_2et92p0)3

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

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

[**Evaluation**](#_2o15spng8stw)4

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/20/2022 | Momtahin Chowdhury | Updating based on The Game Room requested specifications. |
| 2.0 | 06/04/2022 | Momtahin Chowdhury | Updating the Evaluation table |

.

## [Executive Summary](#_sbfa50wo7nsh)

The Game Room wants to develop a web-based game that serves on multiple platforms. The game is called “Draw It or Lost It” and it is currently available in an Android app only. The purpose of the game is multiple teams competing to guess what is being drawn, which is come from large library of stock drawings as clues. The teams will have 1 minute each round to guess, if unable to do so, the opposite team will have 15 seconds to answer.

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

The Gaming Room specifically requested the following software requirements for the game application:

* The game will have one or more teams involved.
* Each team will have multiple players.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist.
* Must be able to operate on multiple platforms.

While these requirements are for the game aspect, we will have to consider the application development. The Game Room requested to run the application on different platforms, such as Apple for mobile devices. Along with machines like Windows, Linux, and Apple. For the game to run on Apple devices, the code has to be written in swift or use existing code to run on other devices by inheriting other languages.

## [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 class. Allowing them to inherit or get information from each other. The UML class diagram shows this inheritance. Each class share common references like “name” and “id”, making Entity a superclass. In the diagram below, it shows Team and Player is a aggregation (has a) type, while Game has a Team and GameService has Games. The GameService has a reference of Games, Games has a reference of Team, and Team has a reference of 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Characteristics:  It’s popular in web hosting  Advantages:  It has various options for different web hosting requirements and it's upgradable  Weaknesses:  It depends on two servers  It has flexible terminal commands to configure the server, access, or make changes  Offers a server-based deployment | Characteristics:  It’s secured and most preferred  Advantages:  Security flaws are caught before they become an issue.  Open source and less support  Disadvantages:  It’s more difficult to find applications to support the web hosting required needs.  More cost-friendly  Offers a server-based deployment | Characteristics:  It’s dominant to the other platforms and private.  Advantages:  High resource requirements, less loading time, high comfortability  Paid, expensive, and more support  Disadvantages:  Easy to upload viruses  More software available compared to the other OS  Offers a server-based deployment | Characteristics:  High portability  Advantages:  Have a wider reach, better compatibility, cost-effective  Disadvantages:  Highly selective to various smart mobile devices. Poor security.  Offers a server-based deployment |
| **Client Side** | The cost will depend on the time it’ll take and how experienced the coder is. It’ll most likely take 3 to 6 months, to program the app and adjusting it on multiple platforms. | Similar to Mac, but will require maximum expertise, and time. Most of the time will probably be spent on working on running the app on Linux | Moderate expertise required with cost like Mac. It will also require same amount of time as Mac. | Slightly more difficult to implement than other devices. Most flexible to clients or even developers to see updates at any place. |
| **Development Tools** | We can run swift language on Mac and mix it with notepad++. We can also run HTML/CSS/JavaScript for front-end and Java, Python, PHP, and Ruby for back-end.  Development tool: XCode  Programming language: Swift or C | We can use Visual Studio, eclipse, along with notepad++ for Linux. we can use HTML/CSS/JavaScript for front-end and Java, Python, PHP, and Ruby for back-end.  Development tool: Eclipse  Programming Language: C | Like Linux but easier to use. HTML/CSS/JavaScript for front-end and Java, Python, PHP, and Ruby for back-end.  Development tool: Visual Studio  Programming language: Visual basic (Default) | We can use HTML/CSS/JavaScript for front-end and Java, Python, PHP, and Ruby for back-end to create an app. The only difference will be CSS, and how it depicts the code. |

## 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 would recommend The Gaming Room to use the Windows devices, it offers various software along with minimum expertise and cost to get the project started. There are also multiple IDEs available to work with.
2. **Operating Systems Architectures**: Windows provides services used by all Windows-based applications that enable applications to show s Graphical User Interface (GUI). These applications can be used using a user account or a server.
3. **Storage Management**: Windows 10 comes with a feature called storage sense. It allows manage files on the hard drive and how much space it takes up. There’s also clouded to save data. The built-in storage system allows files and projects to be saved quickly.
4. **Memory Management**: This project will require a large number of pictures. The memory allocation allows us to store pictures outside of the default picture folder. It keeps all the picture together in a more secure area on the computer.
5. **Distributed Systems and Networks**: I found a game engine that allows creation of cross-platform games, Unity. It requires a simple setup and work with a version of the code, and we choose the targeted platforms. The required SDKs are automatically downloaded.
6. **Security**: Windows comes with built-in security protection software, but to be extra caution it is recommended to get another layer of security to secure user data and information. The built-in protection software system scans for malware, viruses, and security threats.