

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 4**](#_Toc115077323)

[**Domain Model 4**](#_Toc115077324)

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

[**Recommendations 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0  2.0  3.0 | 03/19/2024  04/07/2024  04/19/2024 | Mitchel Dauk  Mitchel Dauk  Mitchel Dauk | Summary, requirements, constraints, model  Evaluation  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)

Draw It or Lose It is a game similar to Pictionary, in that there is an image that is drawn out for players to guess the image. Within 30 seconds, the team must guess the correct name of the drawing or lose the round. The environment for the game is the main concern and the team needs help setting it up. The program will be made with different classes, which contain important methods, that connect together to formulate the details in the game. The libraries will store the necessary information and images for the game, so they can be accessed. There are many design constraints that the company has requested to be in the program.

## Requirements

Business requirements:

* Web-based game
* Android app to multiple platforms

Technical requirements:

* One or more teams
* Multiple players per team
* Unique game names and team names
* Ability to check for game or team name already used
* One instance of the game exists at a time
* Unique identifiers for each instance of a game, team, or player

## [Design Constraints](#_2et92p0)

Business constraint:

* Multiple platform game made originally for Android

Limit constraint:

* Number of images limited in library

Technical constraint:

* One or more teams
* Multiple players per team
* Unique game and team names
* Ability to check for game or team name already used
* One instance of the game exists at once
* Unique identifiers for each instance of a game, team, or player
* 1 minute timer for each round
* 30 second timer for the first team to guess the image
* 15 second timer if the team fails to guess the correct answer

## [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 diagram provides the details for “The Gaming Room” UML design. The class “GameService” will flow into “Game”, “Team”, and “Player”, and uses their resources. The three classes work together to provide “GameService” with the necessary ids and lists. For example, the “GameService” class will get the game id from “Game”, which will get the team id from the “Team” class, which will get the player id from the “Player” class. The classes “Game”, “Team”, and “Player” will be inherited into the “Entity” base class and identified. “ProgramDriver” uses a “SingletonTester” and does not connect to the other classes. All of the code used in the diagram will be a stepping stone so all programmers stay on the same page.

**"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** | -Easy to use  -Closed source  -Community support  -High quality apps  -Slightly less customizability  -Easy to secure from viruses and malware  -Must have Apple product to use IOS | -Open source  -Less chances for security errors  -Complex and hard to use  -Less used than other OS  -Support is normally user-based  -Some programs aren’t compatible  -Customizable | -Easy to use interface  -Closed source  -Easy updates  -More common  -PC can be built or bought to specification  -Susceptible to viruses and malware  -Many programs are compatible | -Mobile devices used by every person  -Limited to certain capabilities  -Restricted function ability  -Must be cross platform for each OS  -High support  -Easy updates |
| **Client Side** | -Expensive  -Must have Apple brand equipment  -Easy to navigate  -High technical support  -Low expertise needed  -Better security means less compatibility  -Security costs money | -Free  -Complicated to navigate  -Takes more time to navigate  -Originally had no interface  -Complex updates  -High expertise needed  -Free security  -Free updates to newer versions | -High license costs  -Each user costs extra money  -Easy to use  -Time saving features  -Easy updates  -Quick and easy support  -Medium expertise needed  -Security costs money | -Devices are expensive, but everyone has one  -Easy to navigate  -Easy interface  -Simple updates  -Can work for anyone with an updated mobile device  -Must be made for multiple devices |
| **Development Tools** | -HTML  -JavaScript  -Python  -C++  -Ruby  -Objective-C  Swift  -Code::Blocks  -CLion  -NetBeans | -Python  -JavaScript  -Ruby  -Perl  -PHP  -MySQL  -Rust  -NetBeans  -Geany  -Qt Creator | -Eclipse  -Visual Studio  -Command Prompt  -HTML  -JavaScript  -Python  -PHP  -Ruby  -C++ | -Swift  -Kotlin  -Flutter/Dart  -React Native  -Objective-C  -Xcode  -Eclipse  -HTML  -JavaScript  -CSS |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: In order for the Gaming Room to expand, they should expand to all mobile devices. This means that The Gaming Room originally being only on Android, will be extended to iOS. iOS, should at minimum, double the user base. Since it is still a mobile platform, cross platform will be easier to develop and will save the most money. If the expansion to “The App Store”, through iOS, is successful, The Game Room can expand to PCs and maybe even consoles or virtual reality.

**Operating Systems Architectures**: The iOS platform consists of layers, which include core, media, and application (GfG, 2023). The core layer includes the frameworks of core data, cloud, address, location, social, and store. The core layer basically covers the necessary requirements in order for the application to run at its base. The media layer includes the frameworks of media, graphics, animation, interfaces, and rendering. This layer fine tunes the original plan, and adds details to the base. Finally, the application layer includes the frameworks of controllers, game center data, map kits, and support. This layer works to bring the mechanics together so that they intertwine with the base and its necessities.

**Storage Management**: In order to run the game smoothly, iOS offers their own cloud service called iCloud. The original development will need to be completed on a personal computer, so the hard drive space needed will have to exceed 256 GB (Gunduz, 2023). Xcode takes up roughly 80 GB, SDK version are 20 GB, build artifacts can be up to 30 GB, tools are 20 GB, and additional components can be 50 GB. There also needs to be space for upgrades and updates. Files will continuously get bigger and bigger, so it is wise to have enough space ahead of time.

**Memory Management**: Memory management for the iOS platforms uses ARC, or automatic reference counting, through Xcode. ARC keeps instances of classes and frees them up automatically so there is no need for the user to do it. There are also ways to manually manage the memory in the event that a class or sub-class is not released fully or properly.

**Distributed Systems and Networks**: The different platforms will work together with something called “cross platform”. Since both applications will be made on mobile platforms, it is an easy transition for the program. When the code is created, the code will be similar so that only a few parts are different between the platforms. Cross platform allows the game to be made on the new platform and exactly like the old one, with less modifications. This means less costs, less time, and better accessibility.

**Security**: Fortunately, Apple has an intense security set up that only allows users to download apps that have been thoroughly scanned and built with security. By doing this, no outside apps can be installed on their devices. This being said, to abide by Apple’s protocols, there must be secure code, communication, encryption, regular updates, and settings that invoke security practices for the user. If security is presented improperly, applications can work as a gateway into the platform device, and access confidential information.

References

Gunduz, C. (2023, August 2). *Hardware requirements for IOS development (May 2023)*. FlineDev. https://www.fline.dev/hardware-requirements-for-ios-development-may-2022/#:~:text=That%20adds%20up%20to%20240,at%20least%20512%20GB%20SSD.

GfG. (2023, January 23). *Architecture of IOS operating system*. GeeksforGeeks. https://www.geeksforgeeks.org/architecture-of-ios-operating-system/