

Draw it or Lose it!

# **CS 230 Project Software Design Template**

Version 3.0

## 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)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 15 May 2022 | Sergio Mateos | The documentation of Creative Technology Solutions will provide an idea of how the game would be developed, specifications, and trouble that the time might face. |
| 2.0 | 31 May 2022 | Sergio Mateos | Evaluate the characteristics, advantages, and weaknesses of the operating platforms |
| 3.0 | 12 June 2022 | Sergio Mateos | Analyze the characteristics of techniques specific to various system architectures |

## [Executive Summary](#_sbfa50wo7nsh)

The Game Room wants to develop a web-based game that will serve multiple platforms on their current game, the game is currently available on Android only. The “Draw It or Lose It” it’s a team game in which the opponents of rival teams must guess what image has been pulled from the library until the time runs out. The team will have four rounds of one minute each, and if not answered, the opposite team will have 15 seconds to try to solve the puzzle.

## [Design Constraints](#_2et92p0)

1. The game will allow access to multiple teams.
2. Each team will be integrated with multiple team members.
3. Game and team names must be unique to verify whether the name is being used.
4. The instance of the game can exist in memory at any time which will be accomplished by creating a unique identifier for each instance of a game, team, or player.

## [System Architecture View](#_ilbxbyevv6b6)

The Game Room specifies these requirements to be met while Creative Technology Solutions write the code and develop the app and website. The developing team has the Android platform, since The Game Room must be on multiple platforms, the development team needs to make sure the game can be playable on iPhone, Mac, and Linux (for the computer website) as well.

## [Domain Model](#_8h2ehzxfam4o)

The Gaming Room UML makes **Entity** a superclass which means that subclasses like **Game**, **Team**, and **Player** will inherit the attributes like id and name, and the methods like getId, getName, to String, etc. The subclasses will have their unique methods and attributes. **Game**, **Team**, and **Player** have an association, **Game** has an association with **Team, Team** has an association with **Player,** and **Game** has a unique association with **GameService.** All these subclasses have a multiplicity association from 0 to many. **SingletonTester** has an association with **ProgramDriver**, this association is unique since **SingletonTester** uses **ProgramDriver.**

**"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** | Configuration and change are adaptive on the servers but with a high cost of server use. GUI is easy to understand and use. The security is high/moderate which helps to secure the application and the users. | Linux has a problem finding apps to support the web hosting. The navigation is more difficult requiring a high level of expertise.  Very secure and excellent server implementation. The inexpensive scaling tools make change reachable. | Larger software available and community. Windows (Azure) have a high price. GUI is easier to understand and provides plenty of options for developers.  Windows tend to be less secure. | It is easier to reach a larger community, in this case, there are more Android users than IOS worldwide. The problem would be implementing the game across all the different platforms.  The security would be poor. |
| **Client Side** | Cost: High cost.  Time: Moderate  Expertise: Moderate knowledge. | Cost: Low cost.  Time: High.  Expertise: High knowledge. | Cost: Moderate cost.  Time: Low.  Expertise: Low/Moderate Knowledge. | IOS and Android  Cost:  Time: Both can be updated at any time.  Expertise: Usually IOS is more difficult to implement compared to Android since IOS is not set a stone. |
| **Development Tools** | Programming Language: Swift and Objective C  Development tool/IDE: Visual Studio (Free and Open-Source), XCode (Free), CLion (Free and Paid), AppCode (Free and Paid), PhpStorm (Paid), Atom (Paid), Brackets (Free and Open-Source), Sublime Text (Free and Paid), Espresso (Paid), TextMate (Free and Paid). | Programming Languages:  C, C++, and Bash.  Development tool/IDE: CLion (Free and Paid), Eclipse (Free and Open-Source), Code::Blocks (Free and Open-Source), Visual Studio (Free and Open-Source), Brackets (Free and Open-Source), EMACS (Free), Visual Studio (Free and Open-Source). | Programming Languages:  Visual Basic, C#, and JavaScript.  Development tool/IDE: Visual Studio (Free and Open-Source), Eclipse (Free and Open-Source), Atom (Paid), Azure (Free and Paid), Cloud 9 (Free and Paid), Axure (Paid). | Programming Languages:  IOS: Swift.  Android: Kotlin and Java.  Development tool/IDE:  IOS: XCode (Free)  Android: Android Studio (Free) |

## Recommendations

1. **Operating Platform**: Based on the requirements of The Game Room, the best operating platform would be Linux, since is open source and the licenses are free which will keep the project within the budget. Even if require a high level of expertise, the security implemented from Linux is extraordinary.
2. **Operating Systems Architectures**: The Linux operating system architecture is composed of kernel, shell, and software. The Linux kernel has a popular open source to the servers which provides a tree structure for a better administration and organization of data and files. The advantage of Linux is that can operate across the board with a different browser like Google Chrome, Safari, Microsoft Edge, etc.
3. **Storage Management**: The Game Room will need the proper store management for the files, images, and any updates in the future. Also, the importance of security, reliability, scalability, and money are important factors to consider. The best solution for their storage management would be cloud storage since they will satisfy The Game Room criteria. Providers like Amazon Web Services, Microsoft Azure, Google Cloud, and Salesforce are the best option for companies who are seeking to maximize their services or products.
4. **Memory Management**: The development of a fast application is possible by using a built-in memory of Java. The compression of images, network configuration, and delivery of images properly will limit the CPU overload and the users will have a more enjoyable experience. Consider the different parameters of the browsers for memory usage for a seamless experience and, one of the most popular is Goggle Chrome.
5. **Distributed Systems and Networks**: Draw it or Lose it! It’s going to be multiplatform, and it needs to create a successful communication between different platforms. The server network will be required to be strong enough to be available to connect the different platforms on the game. The expansion on different servers across the country will reach many different users and consider a backup server in case of any problem. The .NET Standard Library is available to share across different systems.
6. **Security**: Security on the application is something that will take as a priority to prevent any leak of information or possible hack to users. Different types of security checks are going to be implemented.
   1. Linux is a safer operating platform that provides HTTPS support and encryption.
   2. Strong password and backup email if needed and required a password update at least once every year.
   3. If the hardware allows, before finishing any transaction the app will require any attribute identification (fingerprint, face identification, etc.) to verify the authenticity of the user.
   4. Game updates a certain amount of time (within 2 months preferable) to improve any security weakness.
   5. On The Game Room side, masking any debit/credit card add as a type of payment.
   6. Cloud servers offer arrays of policies, technologies, and control to protect the app and data storage.