

The Gaming Room

Michael Savage

# **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)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <05/19/2022> | Michael Savage | Provided an analysis on the project of Draw It, or Lose It for the client, The Gaming Room. |

**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 Client, The Gaming Room has requested that Creative Technology Solutions develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. In Draw It, or Lose It teams compete to guess what is being drawn. Rather than a player drawing images on an easel to help team member guess the puzzle, the application will render images form a large library of stock drawings as clues. The game will consist of four rounds of play each lasting for one minute. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before the time runs out the remaining teams will offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

Some following design constraints that have already been identified are as follows:

* Needs one or more teams to be involved.
* Each team will have multiple people assigned to it.
* 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 in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* Must be able to run on multiple platforms

The client has requested that we meet the following above requirements while the game is developed and the software is written.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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** | Mac OS X servers are an inexpensive option as a server is only $20 to implement. However, it is a newer OS compared to Windows and Linux, so it is not as popular as a choice for preforming the programming tasks. | Linux OS offer free server capabilities depending on the size of the project. This project would likely exceed those requirements however the Linux OS servers are still low cost and may vary. It is an open-source platform and is not as popular as Windows or Apple products. | Windows servers can prove costly to implement and maintain. Windows is however one of the most used OS in the industry, so using windows servers would prove to easier to implement. | Mobile devices do not contain the same potential as other devices so hosting a server may prove difficult. However the cost of implantation is low. |
| **Client Side** | Mac OS are not an open-source system. Cost would remain minimal for the application that we will be developing. Time would vary depending on the skill level of the dev team and would require a team more experienced in Mac Application development. | Linux is a free OS, however the development on these platforms can prove to be time consuming. Most of the cost on these Linux OS will come from the development process of the application. The time of development would high as Linux is not a commonly used OS and can prove to be difficult to developers with minimal experience. Ideally this Development team would have prior experience in Linux OS. | Since Windows is not open source the cost for Widows OS is $199.99 annually for the size of the project we would need for this product. Time would vary depending on the level of experience provided by the developers. | Cost of development on iOS and Android would not be an issue as development on these platforms is not relatively time consuming. The most time-consuming part about mobile application would probably be the multiple operating systems that each application would need to be able of function on. |
| **Development Tools** | Swift would likely be the language used to write this application for Mac devices. | Eclipse and Atom are commonly used IDE’s on Linux. Eclipse excels in it use for Java applications however Eclipse can support languages such as C++, HTML, and C# | Windows would use the Eclipse and Visual Studio development tool for the software. Visual Studio can use several programming languages such as Java, HTML, C++, C, C#. All of which can be used for this application. | Android conducts most of their software development in the Java programming language, iPhones use swift |

## Recommendations

1. **Operating Platform**: The Gaming Room should use a Cloud-Based Operating Platform for the game “Draw-It or Lose-It”. A Cloud-Based Operating Platform will make the overall upkeep of the product easy to maintain as well as provide a wide range of client outreach. Operating on a cloud-based system will transfer the responsibility of server upkeep to the providing party. Using a cloud operating system will also be a cheaper alternative to the traditional method because it will only cost the client “The Gaming Room” the subscription like charges monthly/yearly rather than having to provide their own servers and staff to maintain and provide the upkeep of the servers.
2. **Operating Systems Architectures**: To develop Draw It or Lose It, to have cross compatibility across a wide range of different Operating Systems, we will first need to design each program through its own OS. Each computer functions on different operating systems therefore we will need to develop the software to work in Windows OS, Linux OS, macOS, and both Mobile OS, iOS/Android OS. Using the Cloud Based Servers we can implement cross compatibility with each of these OS so that players can play with each other regardless of the operating systems that they use.
3. **Storage Management**: With Cloud-Based Operating Platforms the storage will be left to the third-party service provider to be stores “in the cloud.” All the data from the game and its users will be stored on the service providers servers and will not be required to be upkept and maintained by the client “The Gaming Room.” Storage capacity limits may play a factor in this method but, the service providers usually have options to upgrade these limits with a higher subscription base.
4. **Memory Management**: Memory will be stored on the Cloud Servers provided by the service of Cloud-Based Operating Platforms. Once the game is developed, we can store the memory such as images on the servers so that way they are easily accessed by the application when needed.
5. **Distributed Systems and Networks**: The cloud-based platform will provide a single server application for the game Draw It or Lose It. Each client can access the network while the server is up and running and play the game with one another regardless of operating systems that they are using. This allows easy cross compatibility and a cheaper overall maintenance upkeep for the project. It also makes the application easier to pull for maintenance but may cause frustrations as the game will be inaccessible while maintenance is preformed or when outages occur.
6. **Security**: The security aspect of the Software and user data will be the responsibility of the service provider. This will be accomplished by using encryption communication with all data that is sent back and forth between the user, server, and other users. This encryption method will provide the users with the safest method of data communication and protect users’ information such as passwords and potential payment information.