

Draw It or Lost It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 1/27/2023 | Jason Robbins | Initial Design and Documentation |
| 1.1 | 2/12/2023 | Jason Robbins | Re-assessed Evaluation of Operating Platforms |
| 1.2 | 2/24/2023 | Jason Robbins | Updated 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)

CTS has consulted with The Gaming Room(Hereafter referred to as the Client) in producing a new web-based game that will be based on one of their Properties called “Draw It or Lose It.” CTS will be in charge of providing a solution to introduce this game to a web environment and optimizing the game to run on other operating systems outside of Android. This will be done by creating a web-based runtime where the different operating systems will connect to and control with a player client.

## Requirements

The Requirements include:

* Emulating the gameplay of the original Android game
* Securing connection with the webserver that will host the game
* Providing a stable and responsive connection between the server and the client
* Optimizing the individual player client for each operating system to make the most of each OS

## [Design Constraints](#_2et92p0)

Client has not specified a deadline or a budget for this project. Internal limitations are still provided at 2 months of development time for the completion of the project. Teams will be split between the server side developers and operating system developers. Look and feel of the program should be non-obtrusive to the original design of the Android game and be quick to function.

## [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)

Inside of the main Driver program will be the GameService which will be running the main bulk of the program. Within the Game Service will be several nested classes that all take their inheritance from an Entity class that will host all of the classes names and Identification numbers. Each instance of Teams will have a list of players, each instance of a Game will have a list of Teams, and Each Game Server will have a list of Games. Each id will be unique and will pull its id from the main Game Service entity.

**"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 OSX has a higher price due to licensing, has a higher stability, and is easier to interface with while running thanks to its GUI while running.  With Mac OSX Formally discontinued, cannot guarantee successful future integration with updating software | Linux Server offers a low price due to being an open-source software with high stability thanks to its BSD System but is difficult to use while running because of its command line interface. | Windows Server has a higher base price due to licensing, has lower stability due to blue screens, system crashes and bugs, but is easier to use thanks to GUI while running. | Mobile Devices should never be considered to run Server based operating systems. While it may be possible, it by no means that it will be beneficial when trying to run a web-based server due to limitations in the mobile devices hardware. |
| **Client Side** | Mac is generally run on higher end hardware with excellent resource management. There will still need to be a team devoted to testing on each system and with different browsers regardless of this. | Linux is the least resource intensive operating system meaning that many lower end computers will run Linux. Having an expert in Linux systems and increased development time to try and reduce resource hogging will ensure that most Linux users will be able to play the game. | Windows is generally the most resource intensive of the big 3 operating systems so increased development time to ensure that Memory and CPU resources are minimized when playing on a browser through a windows computer would be recommended. | Assuming that the program is not running the browser for the mobile platforms, we will need two separate teams that will develop for mobile iOS and for retrofitting the previous Android app into one that will interface with the new Server Side HTML Game. This may be where most of the development cost will go towards. |
| **Development Tools** | Client has requested to develop with a browser-based HTML Which you usually done with a text Editor like Komodo on Mac.  If the browser client was to replace it with a standalone Mac app, a developer can choose to develop with an IDE like Visual Studio Code and Xcode, and use a programming language like Swift or Java. | Client has requested to develop with a browser-based HTML Which you usually done with a text Editor like Sublime Text on Linux.  If the browser client was to be replaced with a standalone Linux app, you could develop it using Visual Studio Code and using a language like C++, but development for Linux is very versatile and can be done with a plethora of Programming Languages. | Client has requested to develop with a browser based HTML Which you usually done with a text Editor like Notepad++ on Windows  If the browser client was to be replaced with an Windows app, a developer can choose an IDE like Visual Studio and use a programming language like Java, C++, or 34 other languages. | Apple and Android both do not require Licenses to develop applications for their software, they do require licenses to post the developed apps to their stores.  Swift is the official programming language for the development of iOS apps alongside the Xcode IDE, however Xcode supports but languages for development.  Android Development requires the Android SDK and comes integrated with Android Studio and supports the Java, Kotlin, and many more languages. |

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**: For the Operating Platforms we recommend AWS Lambda for it’s scalability across multiple platforms and it’s development languages include Python, Java and Node.js.
2. **Operating Systems Architectures**: AWS Lambda supports a wide range of Server operating systems to employ, for this instance CTS recommends the use of the Windows Server architecture, specifically the 2022 version. CTS has chosen particularly because of it’s GUI and the fact that this will be TGR’s first web application on a serverless architecture, despite the increased costs due to licensing. If the cost of Licensing is an Issue then CTS recommends a Linux Server Kernel, for example, Ubuntu Server.
3. **Storage Management**: For Storage Services, CTS Recommends the use of 2 services, S3 (Simple Storage Solution) and DynamoDB. S3 will be used to hold the bulk of the static images used for the game files and will interface with the HTTP in order to send the images directly to the client software without having to download through the Lambda functions this happening after the user is verified and that a call to download the images goes through Lambda. DynamoDB will be used to host the Game, Team and Player information where their edits will go through Lambda. CTS believes that having these 2 separate services will cut down on the overall memory traffic and function runtimes through Lambda, cutting down on operational cost.
4. **Memory Management**: AWS Lambda allows the developer to configure the amount of memory to each lambda function. Which means for each function you can choose how much memory to allocate to its execution. An example would be that for the larger functions to import images from S3 to the client function you could increase the memory allocation while for updating the Games/Teams/Players you could keep the minimum memory allocated. Since Lambda charges by the amount of time consumed by a function, increasing the memory allocation on the more taxing functions will decrease the runtime, in the long run saving TGR money.
5. **Distributed Systems and Networks**: Our main clients will be accessing the game through the web browser and the mobile apps across all of the Systems. Both of the Apps and the web Browser will interact with Cognito for User Verification and send back a security token. Once they have the security token they interact with an HTTP and verify the security token, and from there can access the game functions through the lambda server. These means that if the client can create an app that can interact with an HTTP and login through Cognito, they can effectively create a connection to the game at the backend through the Lambda Server and expand their game to new platforms.
6. **Security**: For Security we would recommend a Security token service and a user authentication service. For the User Authentication Service, we recommend Amazon Cognito for it’s ease of use and its quick set up without being too intrusive to the user. Cognito can authenticate using social identity providers like Facebook, making it easier on the user to login. Cognito is mostly used for clients that register and have login credentials, working by giving the security token The Security Token Service generates temporary credentials to the client to make sure that they