

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 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

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

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/20/2023 | Matt Knutson | The Android app “Draw It or Lose It” will be moving into a web-based environment, where the game can be played on multiple platforms (Windows, Linux, IOS, etc…). |

## [Executive Summary](#_sbfa50wo7nsh)

We will be moving your Android app “Draw It or Lose it” to a multi-platform web-based application. In order to accomplish this, we will need to establish a structure that will have web servers, database servers, and will require software developers, that are fluent in several programming languages, to write the code for the game. We have an excellent team here at Creative Technology Solutions and we are more than qualified to handle all of the elements involved in this project. The approximate time that will be needed to finish the project, will be 9 months and the approximate cost will be $100,000.00.

## Requirements

Business Requirements – (1. A Roadmap Document to specify the applications objectives and establish timelines for the project. (2. Research and define your audience for analytical and marketing purposes. (3. Develop an Application Functionality Specifications Document.

Technical Requirements – (1. SSL certificate for security. (2. A Managed Server provider. (3. A Network (4. A working application. 5) A Firewall.

## [Design Constraints](#_2et92p0)

The web-based application will need to be set up with a web page, web servers, a database, middleware, and we will need to be able to ensure that the client and server are able to communicate continuously. We will also need to establish the programming languages and frameworks used for the project.

## [System Architecture View](#_ilbxbyevv6b6)

We will develop a webpage for the game to establish a link between the game players and our game web servers. Once they communicate through our servers to confirm their game Ids, they are directed to the game version that is written in the native language of their device. The game is then played, and images are rendered from the game database for the players to see. All information sent back to the user will be in the standard CSS, HTML, and JavaScript languages.

## [Domain Model](#_8h2ehzxfam4o)

**The function of the class Entity is to establish the name and Id from of another object and then print it. The classes Game, Team, and Player all inherits from the Entity class because they all share its properties. GameService is associated to Game with the properties of none to many. Game is associated to Team and Team is associated to Player with the properties of none to many. ProgramDriver inherits from SingletonTester.**

**"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)

Since the client’s game is already an Android application, it might seem like it would make sense to use the mobile-first design type when implementing responsive-web design. But this would cause a choppy transition from smaller devices to larger devices. The desktop-style first approach is the way to go concerning this project due to its scalability. Although this will mean that they game will need to be completely redesigned for PC, it will allow for a smooth transition to any device, of any size. A mobile subdomains design template would also work, but you would need to rebuild the application for PC as well. This design would also cause a headache for customers, as they would need to log in to a different domain when transitioning from a mobile device to a larger device.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Advantages:   * User-friendly. * Lots of “Mac Tools”.   Weaknesses:   * Closed platform. * Webhost must pay for new version every release. * Must use expensive “Mac” hardware exclusively.   macOS Server:  Pros:   * Unlimited License.   Cons:   * Only runs on Apple. * Discontinued as of 2022. | Advantages:   * Free software. * Open source. * Can be altered by anyone. * Cheaper server hardware. * Can run servers that are older or not up to date.   Weaknesses:   * Not user-friendly. * Heavy use of the command line.   Ubuntu Server:  Debian Server:  SUSE Linux Enterprise Server:  Pros:   * Open source. * Free & stable. * Exclusively customizable for business needs.   Cons:   * Requires extensive knowledge. * Will most likely need a systems admin. | Advantages:   * User-friendly. * Extremely compatible. * Lots of tools and ability to customize.   Weaknesses:   * Closed platform. * Server equipment is expensive. * Won’t run without certain qualifications such as the amount of RAM or CPU.   Windows Server 2022:  Pros:   * Lots of licensing options. * Unlimited VMs. * Third Party app support. * $2,999.99 annual licensing fee for entire server package.   Cons:   * Requires a minimum number of cores. * Requires CALs. * Requires additional virus protection. | Android Server: N/A  IOS Server:  N/A  Alternative Servers:  Red Hat Enterprise Linux:  Pros:   * Paid Linux server distribution. * Yearly license is $1,299.00. * Smart Management = $1,649.00. * High Availability = $399.00. * Resilient Storage = $799.00.   Cons:   * Licensing fees.   FreeBSD:  Pros:   * Free & fast. * Good Security. * Lots of tools.   Cons:   * Hard to learn. * Lacks support. |
| **Client Side** | * Programmers will need to be able to write code in multiple languages and be familiar will several frameworks in the industry. * The Application will need to be written as a PC game. * Use Responsive Web Design to create an Adaptive and Responsive web application. * Use the Desktop-first Design. * Create fluid layouts, media queries, & scripts that can reformat webpages and Markup automatically and instantaneously. |  |  |  |
| **Development Tools** | Languages:   * Swift   IDEs:   * Xcode * Visual Studio   Frameworks:   * Xcode * Cocoa   Server OS Tools:   * N/A | Languages:   * C * C++ * C#   IDEs:   * Visual Studios * NetBeans * Eclipse * PyCharm   Frameworks:   * Windows UI Library * .NET   Server OS Tools:   * Bootstrap & Wirefy: Free and open source. * Adaptive Images: Automatically renders images to the right size for user, free. * Webflow: Handles code aspect of the design to allow for focus on developing the user experience, free & paid versions. * CrossBrowserTesting & Resizer: Free testing tool.   LDAP & Cloud: Yes, compatible. | Languages:   * Python * C++ * Java   IDEs:   * Atom * Eclipse * IntelliJ   Frameworks:   * Gnome * KDE   Server OS Tools:   * Bootstrap & Wirefy: Free and open source. * Adaptive Images: Automatically renders images to the right size for user, free. * Webflow: Handles code aspect of the design to allow for focus on developing the user experience, free & paid versions. * CrossBrowserTesting & Resizer: Free testing tool.   LDAP & Cloud: Yes, compatible. | Android:  Languages:   * Java * Kotlin   IDEs:   * Android Studio * Droid Script * CppDroid   Frameworks:   * Flutter * Xamarin * Corona   IOS:  Languages:   * Swift   IDEs:   * Xcode * Visual Studio   Frameworks:   * SwiftUI * React Native   Server OS Tools:   * N/A |

## Recommendations

1. **Operating Platform**: We will be using Linux for development on this project. Particularly, we will be using the Ubuntu Server OS.
2. **Operating Systems Architectures**: We will be using a three-tier architecture, as well as designing a web-based game application for The Gaming Room. We will use the customers device, Oracle Cloud services, and an Ubuntu physical server to set up a distributed system for the game. This will allow the game’s logic to be run client-side which will conserve memory for the company but will require each user to have the required amount of memory to run the game properly.

The cloud-side will be responsible for game play, real time memory usage, and the storage needed to hold the games data such as audio, video, pictures, high scores, user game data & the game’s code. By utilizing FaaS, Function as a Service, we will be utilizing serverless technology. This will allow the game to be event driven and triggered by user’s actions. This will only allow use of data on an as needed basis, which will save memory resources and costs to the company. The fact that these services open due to user activity, and close because of inactivity, offers another layer of security as well for the users. We will also use SaaS, Software as a Service, to run the game and store its data in the cloud.

The Ubuntu Server will use the MySQL Database to securely store usernames and passwords, as well as other critical user information. All web communications will be made via the Apache Server and will be secured using SSL or TLS. There will also be a RESTful API in place to insure communication travels in the manner that it should for this application.

1. **Storage Management**: Game data storage management will be handled by the cloud, SaaS. Server storage is conserved by using the Ubuntu Server, which only requires 2gb of hard drive to run.
2. **Memory Management**: Game will not be able to be played on mobile devices with less than 4gb of RAM and a 5G connection, or on non-mobile devices with less than 8gb of RAM. FaaS will be used in the cloud because it is event based, auto scalable, and only uses resources on a needed basis. Ubuntu Server only requires 512mb of RAM to run on the server-side.
3. **Distributed Systems and Networks**: Distributed systems facilitate sharing resources amongst users. The communication between objects and processes is established through message passing. In a distributed system, we can have all components RUN concurrently with each other and at the same time FAIL independently of each other. Because of this, a distributed system is extremely reliable, efficient, and fault tolerant. Also, since the game logic takes place on the user’s device, it is extremely cheap for our client to and new users and upscale resources when needed.
4. **Security**: We will be implementing several components to ensure security of the user and client’s data. HTTPS SSL (Secure Socket Layer) will be used to encrypt the link between the webserver and browser. There will also be encryption used on data traveling across the server, or in the database, and a server firewall will be used. During the application development process, privilege rights will be established, and cloud services will also provide an extra layer of security. Draw It or Lose It will also have a strict password policy and use two-factor authentication.