

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc1991911052)

[Table of Contents 1](#_Toc1969902260)

[Document Revision History 2](#_Toc1479931151)

[Executive Summary 2](#_Toc928704740)

[Requirements 3](#_Toc349971323)

[Design Constraints 3](#_Toc1449612968)

[System Architecture View 3](#_Toc788594786)

[Domain Model 3](#_Toc1177920301)

[Evaluation 4](#_Toc341323352)

[Recommendations 6](#_Toc1414378141)

## 

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/04/24 | Dathan Pompa | Created the Executive Summary, Design Constraints and explained the Domain Model |
| 1.1 | 02/10/24 | Dathan Pompa | Completed the Evaluation portion of the documentation. |
| 1.2 | 02/28/24 | Dathan Pompa | Completed the recommendations section. |

**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.

## 

The Gaming Room wants to make a web-based game run on multiple platforms that will be called Draw it or Lose it. It’ll be modeled after their current android app. Changes will need to be made to the android app to account for this game becoming web-based so that it can run on all platforms.

## Requirements

Games need 1 or more teams to run.

Each team will have multiple players.

Game and Team names must be unique for users to check availability.

Each game instance can exist once in memory at a time.

Game must run on multiple platforms.

Games last 4 rounds, each round lasting 1 minute at most.

## [Design Constraints](#_2et92p0)

Since only one game instance can exist in memory, we need to make this a singleton game. This game needs to run on all platforms excluding android since it already runs on it. New code needs to be written for the other platforms it needs to run on, since this game is going to be a web-based distributed environment, technologies that revolve around it must be considered. Server connections and if the current server can support the other platforms. Redesigning for the android app will need to happen so that it can connect to the web-based game.

## [System Architecture View](#_ilbxbyevv6b6)

The system architecture of this game will need to be based off a web-based environment that can run on multiple platforms. We will need computers for the developers to create the game on and use technologies that allow them to test their game in the environments they are making it for. Docker is a good application for developers to utilize to make programs in different environments. When the user is interacting with the game, they will be sending data to the server which will interact with the service layer to have any services operate or make changes, then send the data back to the user. Since it is web-based, the device constraint does not need to be considered because operations will be server-side instead of client.

## 

**For our UML, it starts at our GameService class which has a relationship none to many with Game class. Game class inherits from Entity and has a relationship, 0 to many with Team class. Team class inherits from Entity and has a relationship with Player class which is also 0 to many. Game, Team, and Player class all inherit from Entity, which shows inheritance, they inherit name and id. Entity is a super class because of it. GameService, Game and Team are “has a” relationships because GameService has a game, game has a team, and Team has a player, this lets me know that this is an aggregation OOP principle.**

**"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.**

## 

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Macs have OS X server which makes web hosting simple for a user to set up and start hosting. Unfortunately, Mac has decided to discontinue their support for MacOS Server, so you will have to look for other solutions. Mac uses UNIX as their system which is like Linux.  An advantage is apple being known for being simple and easy to use and how apple products interact with other apple products. Mac offers server-based deployment that can utilize Apache server or use MAMP to host. A big con for Mac is that its support is discontinued, so this means longevity for Mac will not fit any business or software need. | The terminal in Linux is basically the same, but it's cheaper to use because system requirements are low. Linux is more secure, making it an advantage and making it the most preferred choice in web hosting. It is more secure because viruses need to get system level access, but since the root account owns system related files, viruses can get delated easily as it would only affect the user. Since not many users use a Linux machine, finding support for drivers and other applications will be difficult when web hosting. | Terminal is more complicated to use, but software will make it easier to connect and configure to the server. An advantage to windows is its use of resources can be high which gives the user more processing power and less loading times. Windows has lots of support in case problems occur because of its large user base. Just like apple, Windows offers integration within its own ecosystem making it easier for companies with windows computers to connect. One disadvantage of Windows as web host is the fact that there are many viruses that make it susceptible. As well as frequent updates and maintenance will cost more. | It is better to have a sever that is not mobile because you will need to be able to track it down to make changes when needed. More people use mobile than any other platform making it popular amongst users and it being portable is a plus also. It is also the cheapest option. A disadvantage is that there are so many different options with phones so user hardware can vary drastically as well as it being secure is not great, same as windows. |
| **Client Side** | Mac computers cost more compared to other windows computers that are similar in performance and configuration. Time depends on the Development team, if they are used to mac then it will be easy for the team to develop tools and edit code seamlessly. Mac computers are known for being great at developing front-end tools and for making great UI because of their design focused ecosystem along with their high-quality displays. Mac provides tools such as spaces to organize multiple desktops, side-car for ecosystem based utility (it lets you use your phone or Ipad as a second display), Mission control for managing applications and windows that are running and Time Machine for regular backs-ups to keep your data safe from disaster. Mac has support for most major web browsers such as, Chrome, Safari (being their native browser), Firefox, and Opera. Macs can run into compatibility issues though for browsers like Microsoft edge, even though Mac is actively supporting it, it is not going to up as up to date as the version in Windows. For expertise, being proficient in Mac is easier now days than any other platform so I would say it's easier. | When developing on client side on linux computers, it is very efficient, similar to Mac, which is a UNIX based system. Linux has support for lots of development tools like VS code, IntelliJ, Eclipse, etc. And great package handlers too, it just depends on the type of system you are using. If you have an arch linux based system you'll want pacman or for debian based, you'll want apt. When it comes to browser compatibility, this is where youll run into some problems, it does not have support for every browser. The reason people like to develop on linux is the freedom of customization and flexibility. The skill set required is quite high, you can start to develop on linux and will enjoy it, until you run into a compatibility issue, this is where you really need to know what system you have, to solve the problem. Development cost can become high depending on if the developer needs to make multiple tools to allow support of their software. | Development on Windows is made simple with robust stacks such as .NET or Node.js for client side. Windows computers are going to be budget friendly ranging from very expensive due to very powerful performance and cheap due to being very efficient. In recent years, PC hardware has become cheaper with their power increasing. There are tons of bang for your buck lists out there that will suit your needs in any target such as, students, developers, and professionals who need powerful machines for complex computation. That could be video editing or even needing to compile amazing programs like AI that use lots of resources. There is lots of support on the web to help anyone as well as lots of tools available. Popular choices of package managers like npm that can help you install node.js or an alternative is yarn. Lots of front end frameworks for windows like Svelte or Angular. As well as many IDE’s to choose from, VS Code, Visual studio, and Eclipse. Many web browsers work on windows, but for safari, you could run into compatibility issues. | Cost is the lowest because of how cheap phones can be, Time can be longer than mac and windows. Many times developers will create mobile first applications because of many users will have a phone and check how it works on there first. You could lean towards a PWA or progressive web app which will give you a native feel. Mobile development use hybrid mobile apps that use web technologies and are packaged as native app with frameworks, such as react native or apache cordova. Expertise is moderate just because you are limited so using mobile applications that will allow you to develop, so you would not have as much freedom as you would on computers, but it is still possible. |
| **Development Tools** | Swift is a common language used in Mac as well as Java and C++. IDEs like Xcode are popular for mac users and other tools can be used for great front-end development like SceneBuilder. Xcode offers Java support so we can choose this, and we can use Eclipse for our IDE. Apache Maven for the build tools, use git for version control, TestNG for testing framework, but it offers more like parallel execution which is neat. Apache Tomcat for our webserver, we can also use VisualVM to monitor and troubleshoot our application. | Linux can work with most languages. C++ and Java along with web-based languages like JavaScript and python. Lets use NetBeans for Linux because its open source that will support java development. We can use Gradle for our build tools and automation. Let's use GitHub desktop for our version control. Junit is a great unit test framework for java and Lets use Jetty for our web server and servlet container. | Windows has many programming languages at its disposal such as C++, Java, JavaScript, Python. Multiple IDEs and tools as well because so many users use Windows that developers create tools for Windows first than other platforms. IDEs like visual studio, VScode and Eclipse work. For version control we can use Git which works on mac, linux and windows. We can also use Source tree which is my personal favorite. We can use IntelliJ IDEA for our IDE and utilize Apache Maven for our build tools. For our application servers we can use Apache Tomcat, which is a webserver and servlet container, its a popular choice for java web apps. | For IOS mobile we have Xcode, swift, UIkit, CocoaPods for dependency and TestFlight for testing ios apps just to name a few. UI kit for making UI on IOS apps and swift is their modern language used for all their devices. On android we have android studio, java, AndroidSDK, Gradle for build automation and testing. AndroidSDK gives us tools and APIs that assist with making apps for android and java commonly used for making android apps, but Kotlin is officially supported by Google for android development. Since our app is Multiplatform, we should look at React Native, Flutter for our UI, Ionic so we can access device features. |

## 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**: I recommend The Gaming Room use Linux as their operating platform. It is an open-source Operating platform that has lots of community support. Linux can save you money on operating costs due to it being open source and usually free to use. Linux is the best choice with its combination of performance, security, community support, reliability, scalability, and being cost effective.
2. **Operating Systems Architectures**: We can combine Linux with a multi-tier architecture or n-tier architecture because it will separate the tiers that will have their own tasks to handle. This approach will give The Gaming Room maintainability and scalability by separating the concerns of each tier. The tiers being separated are the presentation, application and data; presentation focusing on the UI elements, application will focus on the RESTAPI’s/other services required, and our data tier will focus on the storage.
3. **Storage Management**: For our choice of storage, I recommend The Gaming Room choose an RDBMS or relational database management system because it will allow them to store and get data in an efficient manner. Aside from that positive feature, it will allow them to have a well-structured database allowing for easy management when changes need to occur.
4. **Memory Management**: Since our operating system is Linux, let's note how well it will manage our memory. It uses a paging system that will allow us to swap unused memory back into our drives. This technology will help optimize our application and overall improve its performance.
5. **Distributed Systems and Networks**: I recommend The Gaming Room use load balancers, CDN’s, Microservices with APIs as a gateway and message brokers as these will allow us to create a reliable, high performance and scalable infrastructure. Message Brokers will allow our application to communicate asynchronously between our different components, Microservices with an API gateway is a great approach because it will allow our application to be broken down into smaller services that all get handled through the gateway. A CDN or content delivery network will allow us to cache and send static assets like our images to our end users which will decrease the latency within our application. Finally, our load balancers, this is important because we can distribute incoming traffic to multiple web servers, this is important because it will prevent any single server from being overloaded.
6. **Security**: For security, we will implement encryption, authentication, authorization, network segmentation and access controls. Features like encryption for HTTPS, SSL, TLS protocols and access control as well as firewalls. It helps will intrusion detection and prevents it with IDS or IPS systems. These recommendations will help ensure all data transmitted from user to server is secured and will not be accessible to others. Linux will have built-in features like encryption, secure communication and access controls, but the other security measures should be used as added defense from online attackers.