

Draw it or Loose it

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 03/19/2023 | Nisarg Patel | Initial draft |
| 2.0 | 04/02/2023 | Nisarg Patel | Updated Evaluation |
| 3.0 | 04/16/2023 | Nisarg Patel | Updated Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Draw it or Lose it currently exists only as an android application, the goal is to have it exist as a web application so that it can be accessed across a number of clients. This process will require a rebuild of the application and its fringe in its new environment which will come with significant development costs.

## [Design Constraints](#_2et92p0)

Some constraints that come with web applications is dealing with client device specifics. The application will be accessed on a variety of screen sizes from compact phones to widescreen TVs which leads to resolution issues, access issues, and interaction difficulties. In addition, the web app can also be accessed from a variety of browsers which will impact the client experience based on the technologies used in development and whether or not they are supported by the specific browser.

## [Domain Model](#_8h2ehzxfam4o)

The main processes of the application are controlled by the Program Driver class which instantiates all class, executes logic, and interacts with the user. As it will eventually become an extensive class, there is a tester built for it to validate logic any logic. The Game Service class works to create and serve the game for the driver and where the backend of the game lives. Since there should be only 1 instance of the game at a time, this class is created with a singleton pattern to ensure multiple instances are not executed simultaneously . The Game, Team, and Player classes inherit from Entity which serves as a boilerplate for objects, defining characteristics which are shared between them all.

**"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** | There are a few tools that are designed specifically for Mac that do not integrate well with the other OS but there are no other direct advantages of using Mac for hosting. | The Linux kernel is amongst the most secure and its capable integrating with most tools that its competitors are without any direct downsides. | Windows is not the best for integrating many development tools/environments but its easy to create VMs for the other environments so it’s a viable tool. | Mobile devices are not the best environment to host applications. |
| **Client Side** | Safari is a browser that is unique to MacOS users which needs to be accounted for. The other popular browsers are shared between the other operating systems. | The primary browser for Linux is Firefox which shares some of the market with Chrome. Neither are unique to Linux so there should be no additional costs or development time associated. | A minority of Windows users that use Edge as a browser which is somewhat unique to Windows that needs to be accounted for. | All browsers on Apple mobile devices use webkit so accounting for traffic from is streamlined. Android apps actually use the drivers of their browsers so there will be some variability between clients. |
| **Development Tools** | The development tools in MacOS are quite advanced, especially for design and related front end work. Homebrew is a really good package manager available on MacOS and it pairs nicely with Xcode.  Macs are also quite expensive, so outfitting each developer with a MAC would add significant costs. | The development environment in Linux is well established and there are a wide array of compatible tools available. VIM is especially useful due to its ubiquitousness within the Linux environment. Linux has a steep learning curve, and developers need to be familiar with command-line interfaces to use it effectively. | Windows has a pretty blank template for development work, it does not come out of the box with any tools, however there are many designed with windows in mind such as VS and VS Code which are a powerhouse for development work. Windows is the most widely used operating system in the world, which means it has a large support community and a wide range of development tools available. | Mobile devices are not designed with development work in mind, and there exist very few built applications that are capable of efficient work, however it serves as a great gateway for team communication, as well as serving notifications of job failures, system outages, or other issues. |

## 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**: The ideal operating platform is Linux. Its kernel is the most secure and its widely accessible via VM’s and SSH to development in other platforms.
2. **Operating Systems Architectures**: Linux’s system architecture can be best described as monolithic. There are a subset of interdependent components that work with each other through function calls. Since the individual processes are separated tasks such as memory management can be optimized far better than on other OS.
3. **Storage Management**: While the webpage can store some amounts of user data, the bulk of it needs to be stored in the backend in a database. AWS offers great compatibility, customization and pricing options for data warehousing in its S3 product. S3 serves as just a place to store your data, but when used in conjunction with Redshift/Athena (other AWS products) querying, managing, and transforming the collected data is made easy. There is excellent compatibility with Linux as the core of AWS operates within a Linux environment.
4. **Memory Management**: Demand paging and virtual memory are fundamental implementations which manage memory for Linux. The use of virtual memory pointers allows for optimized efficiency by separating physical memory details from applications. This means that the application has to worry less about the memory management since some of its inherently controlled by the OS.
5. **Distributed Systems and Networks**: A distributed software is a system that operates on multiple machines within the same environment, or through cloud computing. This is safer since it allows for continuous operations if one or more of the machines experience difficulties. Operating on the same network allows for streamlined communications between the machines and allows them to share resources, data, and information. Use of the cloud does take away some responsibility of managing metal, but takes away some level of control over outages from the 3rd party managing your cloud.
6. **Security**: Linux has a very secure kernel which provides the first layer of security. Other ways to protect the users is through well-defined IAM roles which restrict access users with any level of system access, and the final is through the use of the HTTPS protocol and use of valid certs.