

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/28/24 | Sean Tinneny | Filling in missing design |
| 1.1 | 02/11/24 | Sean Tinneny | Extra considerations |

**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](#_heading=h.35nkun2)

This design sets out to create the ‘Draw it or lose it game’. This game will need to be a web app that needs to run on any operating system that it comes in contact with.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_heading=h.1ksv4uv)

Legal Constraints - The webapp will load images for guesses and these images must comply with copyright laws

Multi-Platform - This webapp should be able to be run on any operating system it comes in contact with on any platforms

Multiplayer - This is a team game so there is a need for allowing multiple players doing the same thing at the same time

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

First off, this is a java application which should allow for all platforms the ability to run it. When digging into the diagram, the team/player system will be run from the game service class and the game class. This should allow the game to know what teams and player objects are tracked to it so that no teams will be able to see other games going on. Then, all of them are attached as an entity which allows for tracking of game to teams/players.

**"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](#_heading=h.z337ya)

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 as a server is not the most ideal. While you can use the terminal as bash the existing terminal is not as helpful as a bash terminal would be. With this being a web app, you would need to confirm all proper ports are open and that you can make sure all proper ports are open and security is still up. | Linux would be the best way to host a web application. There is little to no overhead when running so all of the systems processing power can be focussed on running the game. Also the port/security system is a lot easier to work with because you have access to all of the security. | Windows servers are not bad but just like mac they are an os that is a little bit more beefy to run. They also use their own shell, powershell, which is good but not as good as a bash shell in linux. The port system and security systems are also pretty stringent and could get in the way when opening up to unknown devices. | I do not believe that any web application should be hosted on a mobile device. There could be local games that are hosted on dedicated servers that allow people to play games locally. |
| **Client Side** | Since this is a web application a mac should not run into any issues with running it. As long as the browser doesn’t get any security issues when trying to connect to the server it should be straightforward with joining the game. | Web apps just like mac would act the same unless there is an unsecure connection issue when trying to connect to the hosting server. | I would say that windows would be the best to join a webapp with. I think that is allows for the most connections without security problem warnings. | Because this is a webapp there would just need to be a mobile view of the sight once in game and then this may be the best possible way to play the game. |
| **Development Tools** | Development on a mac is great if doing it in swift but other languages can be a little less straightforward. There are many existing IDE softwares that exist that would make it the same as any os but the environment would need to be setup for it. | While just like mac you need to setup the linux environment to develop, the low processing requirements make it the better OS to develop. You have access to a bash shell and just about any path on the box. | Development on a windows computer is the best if the language is c++ but other than that it is mac like with needing to setup the path to handle the new languages. | Mobile development is possible but should not be something to rely upon. |

## 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 think that the best operating platform would be a linux server distro
2. **Operating Systems Architectures**: All operating systems will be able to run the software
3. **Storage Management**:This storage management will be handled inside the linux server distro and file system
4. **Memory Management**: Linux will handle memory the best because it will handle it with the processes. When a process closes, the memory used will be freed up. There can also be limits set to each game to control the memory usage.
5. **Distributed Systems and Networks**: network connections and opening ports is easy on linux. As long as the linux server has an internet connection it will be accessible to any device.
6. **Security**: With the built in security in linux allowing devices access to the location of the running game process.

Extra considerations

* **Windows**:
  + Has server distribution - Yes
  + Price: Not cheaper than linux but less expensive than mac server
  + Server side
    - Shell is powershell
    - most programs are built for windows
  + Client Side
    - Great ui
    - usable out of the box with little to no changes
  + Level of expertise
    - Middle of the road in difficulty most of the environment setup is intuitive enough to jump right in
* **Macintosh**:
  + Has server distribution - Yes
  + Price: EXPENSIVE
  + Server side
    - shell is more bash like
    - homebrew allows for many programs to be installed for mac from windows
    - less control of processes
    - expensive server distribution
  + Client side
    - great ui
    - useable out of the box
  + Level of expertise
    - Mac server is a harder to setup environment. To develop you would need to setup the environment with the development tools and compilers that would be required
* **Linux**:
  + Has server distribution - Yes
  + Price: free to expensive for enterprise options
  + Server side
    - best shell integration
    - processes are highly trackable
    - better control of hardware
  + Client side
    - harder to understand ui
    - heavily configurable
    - many linux distributions for every use case
  + Level of expertise
    - Depending on the distribution the ease of entry ranges from very difficult to windows level ease of setup, development and understanding.
* Development tools
  + IDE
    - Since this is a android application the IDE will need to work with either c++, c, java, python
    - Depending on if you use open source text editors and install public compilers there will be little to no licencing fees
  + Dev Team
    - Front end
    - Back end