­­

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/14/2022 | Alexander Garcia | Initial summary of requested service and possible plan to achieve goals set out by client. |

**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)

The client, The Gaming Room, wants to develop a web-based game that serves multiple platforms called draw it or lose it. The game consists of showing an image and having the players guess the title before the time expires. We will help them set up the environment and design the game application.

## [Design Constraints](#_2et92p0)

-The game will run on multiple platforms

-The game needs to have the ability to have multiple teams

-The teams will have multiple players

-Each team and player will have a unique identifier

-The game will exist only once in memory

-Each instance of the game will require a unique identifier

## [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)

The program driver uses the singleton tester to ensure a singleton is working properly. The GameService class is a singleton which can have none or more Game objects in it, and it will keep track of these games in a list and will ensure there is only one of each in memory. The Game object will have a list of none or more Team objects and will ensure there is no repeated id or name. The Team object will have a list of the Player object. The Game, Player, and Team objects will all inherit from the Entity object and will all have their own id and name. They will be able to reference down to each other from the GameService class which will keep track of which id to use next for each of them.

**"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** | Popular for web hosting. Less preferred.  Is upgradeable with various options for web hosting. Pricing will vary depending on needs, but will be generally as expensive as windows | Open source makes sure flaws are found quickly. Less apps to support hosting. Secure and preferred. Seems to be slightly less expensive than mac or windows. | Susceptible to malware. Has more resources behind it. Higher costs. Cost will vary with requirements , but will generally cost similar to mac. | Will need support for different phone os. Less secure, and less powerful. Wider use. Possible to use third party to host such as AWS, will need to be knowledgeable about this. |
| **Client Side** | Moderate cost, and time, but can have good user experience. Some expertise specific to Mac required but less so than linux. Since browsers are cross platform, focusing on working with specific browsers may be a better idea than applications for OS. | Highest amount of expertise, very specific to linux. Will cost less due to the open source nature. May indicate a larger time requirement. Although since linux uses same browser as others, its better to focus on a webpage. | Less specific expertise required, with a still medium cost and time. With good user experience. Will still be a better idea to focus on browser specific development rather than OS specific. | More flexible for users, which means there needs to be a higher expertise in development and a broader expertise. This will cause more time and money to be required. Will need to develop apps with proper formatting for different phones. |
| **Development Tools** | Mac has own language and developer tools using swift and XCode. XCode is only available on mac computers, so you would need mac computers to do the development. Can also use other development tools for JS, html, etc. Can always download third party ides. Licensing will depend on what kind of software is written. | You can use eclipse, visual studio or many other ides. Most languages are available and can be run. Not swift. Mainly done in C. Can also use IDE to work on web apps. Licensing will depend on what kind of software is written. | You can use eclipse, visual studio or many other ides. Most languages are available and can be run. Not swift. Easier to use than linux and runs the same stuff. Can also use IDE to work on web apps. Licensing will depend on what kind of software is written. | Depending on the mobile OS, can use swift for IOS which you will need an apple device to create, or react for android. If you want to do both in one there is react native. Will need a variety of phones to test apps, or be able to run emulators. Licensing will depend on what kind of software is written. React native has an MIT license which is almost open source with slight restrictions. |

## 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 of the options listed, windows would be a good place to start developing before expanding to other operating systems. It has a nice balance of medium costs and lower expertise requirements as well as a good amount of developer tools along with good support in place. Although if any options were available the AWS platform has lots of options and tools that could be helpful.
2. **Operating Systems Architectures**: Windows has a modular architecture which consists of two main layers, the user mode, and kernel mode. Each mode will have a different level of access the system resources. AWS uses a Linux operating system which is security focused, stable and has high performance execution to develop and run applications.
3. **Storage Management**: Windows has a built-in storage feature that allows files to be managed and inspected for their use and size. Using this with the file path system allows you to create and access projects easily. AWS has a variety of options for storage and has systems in their platform to manage storage effectively.
4. **Memory Management**: Windows processes create their own virtual address which has a certain amount of memory, so that while the game is running it can use this space for more efficient use. While designing the game you can create folders within a designated folder to hold the different required assets, but still keep the project in a single area. You can simply open the main folder with your IDE to view the project as a whole. AWS has a similar system in place where it will let you choose how much memory you want for your server and will use your space inside a container.
5. **Distributed Systems and Networks**: The benefit of having the software be a distributed system is that it can have much of the data stored on the user side of the software and keep some of the functionality only on the server side. The main issues would be with the responsiveness due to fluctuations on connectivity. This can also be affected by the actual state of the servers themselves, if the server goes down due to a physical problem the game would become unplayable. The benefit would be that it would enable the functionality to be fixed and updated easily as it only needs to be updated at the server not on each individual device.
6. **Security**: For securing any important client data the client may want to invest in some type of end to end encryption while sending or receiving sensitive data within the gaming system. As far as for windows it does have a built in malware protection software that scans the system and any incoming data. AWS has automatic security scans and built in systems to ensure security.