

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <09/13/23> | Jennifer Dominguez Garcia | Added information to executive summary, design constraints,domain model, evaluation, and to recommendations. |
| 1.1 | 09/29/23 | Jennifer Dominguez Garcia | Add information and edit to client and server side. Development tools. Evaluating the needs |

**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](#_35nkun2)

The Gaming Room wants to make Draw It or Lose it available on different platforms. It is currently only available on Android, they want it to be available on different Operating Systems. The game will be four rounds with each being a minute long. If the playing team does not guess within thirty seconds, the opposite team gets fifteen seconds to give one guess.

## Requirements

Only one game can be going at one time(only one team participating at once), teams should be of similar or same size, game must be able to run on different platforms, Teams must have different names to help differentiate between the teams,.

## [Design Constraints](#_1ksv4uv)

cross platform; making it available on different devices other than just android, have the same app throughout or different apps with the same concept designed in each operating system.

Ability to have multiple teams each consisting of multiple people.

allow the different users to give their team names different names to help establish the different teams

## [System Architecture View](#_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](#_2jxsxqh)

Based on the UML diagram it can be seen that there is a relationship created by the entity class between game, team, and player classes. Based on the arrows it can interpreted that they inherit information from Entity. The programDriver class being pointed to the SingletonTester shows that this will be used to test the code. GameService class holds all the information in the game, all that makes up the code and the different details that make up the class whether public or private. The different connections help show association with the number showing how many connections there are.

**"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](#_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** | Using MacOs will create an advantage for apple users. Easy to use and pretty accessible. weaknesses are it relies on other programs and could be costly. flexible, not the best for hosting services  Updates:  connect browser with node.js or flask, roll own server by running HTML. cost would be minimal but other requirements would be needed to increase security to ensure only authenticated sessions have access | Advantages is it is good at hosting services. Cost effective. catches security problems. Disadvantages is it harder to use and might not meet requirements  updates:  keeping the software up to date, get rid of unnecessary accessories. Keeping a unique and fortified password, allowing only authenticated systems to gain access, adding an extra layer of security by having more than one step verification | Has a lot of resources attached to it. Most people are comfortable with using windows. disadvantages is the safety and susceptibility isnt the best.  updates:  similar to mac, use node.js or flask. Minimal cost as well but does increase with added power and needs for coding. the code is portable throughout different methods. | More available and has more reach. more portable than the rest. Limited in what can be done and is not as secure  updates:  would not be the best option, would require multiple apps that would be used during development. The resources implemented on a mobile device would not be enough, would require more work than it would to work on a mac,linux,or windows |
| **Client Side** | The price for the user is somewhat inexpensive. Need to know how to use, great expertise. Depending on level depends on the time, usually low.  updates: Easy to use, requires some time to learn how to use | Expert level expertise required, not entry level. Reasonable cost. a lot of time required  updates:  most useful for developers and people who have experience, takes a lot of learning | barely any expertise needed and time needed. Cost similar to that of mac but more expensive than Linux  updates:  many unique features that can be easily used and convert to different computer system. Great for websites aswell as app developing | flexible for clients to see updates. Harder to implement services  updates:can have the app anywhere, available in different sizes depending on the device ex; tablet vs ipad. ability differs depending on the device |
| **Development Tools** | can run all programming languages including Javascript, CSS, and html. Can use different applications only available on mac to run Python such as pycharm, java, etc.  updates:flask, websockets,node.js, mySQL, | can run all programming languages the same as mac. works with eclipse,etc.  updates:  node, flask. Javascript, Amazon AWS, heroku | Can run all programming languages as well. Has more tools available to develop, has support as well  updates:  javascript, HTML,mySQL, different developmental tools, node, | Is able to support all programming languages such as HTML, javascript, etc.  updates:  can use different websites or apps that would allow the code to be developed. safari, google chrome, etc. |

## 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 would recommend using Windows as the operating platform. There are more advantages to using it and it is easy to navigate.
2. **Operating Systems Architectures**:Windows has many services such as graphics,messaging, internet connection, ability to store files, and run software. These features make it easier to develop games
3. **Storage Management**:Windows has the ability to manage storage. Information can be moved to different locations while keeping track of said information. Able to keep track of the different files and the memory that is being used
4. **Memory Management**:Windows has the ability to store files in a folder making the storing of the different files for the game easier to locate. Storing all the different files including the different images for the game in one place. Keeping all the files secure.
5. **Distributed Systems and Networks**:To make sure the game is still cross-play there are different ways to publish the game so that it can be published on different platforms and for them to run smoothly. different aspects will have to be taken into consideration to ensure that the game runs well on the different platforms. Adjusting as needed.
6. **Security**:Windows is very secure and helps detect any safety breaches. There is built-in software protection. More security can be applied such as antiviruses and being safe on the use of the game.