

<Name-of-Software-Application>

# **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 | 5/28/22 | Adam Benoit | Added executive summary, design constraints, and domain model. |
| 1.1 | 6/05/22 | Adam Benoit | Added evaluations for server, client and development tools |
| 1.2 | 6/18/22 | Adam Benoit | Added 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.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is a successful company looking to expand their successful gaming android app Draw IT or Lose IT based off the 1980s game show Win Lose or Draw into a web-based games. Their hope in making the game web based is to serve more platforms based on the current game. The web-based version will need to be able to host multiple games and have multiple teams competing simultaneously, so each team name and game instance needs to be unique. The game consists of a large library that the game pics an image at random for the teams to guess what the image is. The Gaming Room needs help with setting up the environment and streamlining development. A solution to the client’s problem would be to host the environment on a server reachable through the web. This would allow all instances to be hosted and cross checked for duplicate names across multiple platforms.

## [Design Constraints](#_2et92p0)

You design constraint will be that the game already has a successful app that the web-based version should follow for format and layout. This is for branding purpose to not confuse the already established player group. We are currently unaware of any budget restrictions which can heavily impact design and structure of the web-based platform. Some constraints imposed by The Gaming Room are that the game be designed with one or more teams involved. They also want teams to be assigned with multiple players. The game most also recognizes when a name is already in use and not allow duplication of names or team names. They also want each game to have only one instance in the memory at any given time this can be accomplished by assigning games with an identifier. Security will be difficult with the web-based distribution with so many computers and potential accessing the game there is concern about the integrity and resource allocation if to many users flood the system. Should implement a system that verifies players through their email so that players can’t create infinite number of accounts to compromise the server. Will have to get hardware that can support a large player base and not compromise speed for the users or disconnects. This might be difficult based on budget restrictions. There are currently no constraints based on software or operating systems. These will be looked at and outline with their benefits and downsides of each later in the packet.

## [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)

One of the OOP principles demonstrated in the diagram would be inheritance. This shown by the arrows linking the one class to another class. One example of this would be the arrow pointing from either the team, player or game class pointing to the Entity class. This means that the Entity class is the parent class that can pass to the other 3 previously mentioned classes. Another principle would be portability of any of the classes. The classes are able to be moved and still maintain all that it is to be used in other programs. The diagram also shows encapsulation by the use of making attributes private protecting them from outside access and not allowing outside changes to influence 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** | * More expensive than other options * Better security * Has cloud support but higher cost needed for better performance | * Open sourced * Software is free only cost would be the hardware * More difficult to use * Most functions through a command line environment | * Cheaper than mac * Has cloud support * More widely used than mac or Linux * Less secure open to more attacks | * The least secure * Not as powerful as the others |
| **Client Side** | * Program will be used through a web browser * Will need an apple expert to look into requirements for safari or if another web browser will be needed * Safari has rules regarding CSS and HTML that may cause problems for clients | * Similar to mac operating system * No extra precautions or considerations needed with the use of a modern web browser | * Can run with Microsoft web browsers * Should be supported by all popular web browsers * No added cost, time or expertise with supporting a web based client * Supports rest framework | * Less processing capability all processing/logic must be done server side * expertise in developing for optimal user experience may be required * Will take more time for different phones for example android and iPhone ios |
| **Development Tools** | * One development team should be enough * Can use objective c, swift or java for development * For server development can use java runtime environment * Will need a web hosting application, XAMPP is an example of one * Eclipse is a popular free option for ide for java related development * Could use ReactJS or any other js framework for the web client site | * One development team should be enough * Python is popular with linux but is not oop language can cause issues with game * Python is more beginner friendly lower costs and faster development * Java may be best to use but requires more lines of code taking a longer time in development * Eclipse is a free option for ide for java and pycharm is a free option for python * Requirements are very similar to macOS | * One development team should be enough * There are options for language to use: java, python, C#(can be used to make an app) * Plenty of ide’s to pick from that have free options to save money, eclipse is an example * May have to buy license for more features if needed * Windows is optimized for .NET framework | * May need 2 development teams one for android platforms and the other for ios * Can use one if budget allows for licensing of tools to translate for both platforms * Ios mainly uses Swift * Android uses java * Java has plenty of free ides to pick from eclipse is a popular choice |

## 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 operating platform that is best suited for Draw it or Lose it that will allow it to expand to multiple other computing platforms I believe to be a Linus operating system. Linux are open source allowing modifications to be at the kernel level of the software that adjustments can be made to make the software more compatible. This is unlike windows and mac operating systems because they are not open source and do allow for adjustments at the kernel level and would limit the options for platforms that the game could run on. Linux may be more complicated and may take longer to set up but the pros of having the flexibility with an open-source operating system out way the possible added set up time.
2. **Operating Systems Architectures**: I believe that a 3-tier system would be the best option over a 2-tier system. In a two-tier system having the server house the data base and the application can lead to load balancing issues. By having a three-tier system we help reduce load balance issue. The tiers would be as follows, starting at the first tier we would have the client. At the client we have the presentation logic. This level is what gives users control with the game and allows them to input commands and input validation also occurs. The next tier would be where the application would be housed and ran. Then in the next tier would be the database the would store all important data like the photographs in cycle and user data. This would allow the application to pull from the database and temporally use the photographs and recycle on a set timer and pull new ones. This would also only users that are currently signed being seen by the application tier and not over loading it with every account created.
3. **Storage Management**: My recommendation for storage is to have solid state storage over hard disk drive. The added performance that a solid state can give over a hard disk out weighs any possible failure the solid state might have. But to avoid any possible loss of failure I would recommend doing a raid 5 array on all levels we control. This would be 4 or 5 SSD’s built into an array and would allow for one drive to fail and still maintain full function. I would also recommend a separate back that is performed at a predetermined interval and stored in a separate location.
4. **Memory Management**: Linux helps with memory management by taking advantage of virtual memory. This is when the operating system uses a disk as an extension of the physical ram installed in a device. This slows the size of usable memory to grow correspondingly to the demand. The kernel in the operating system will write contents of a currently unused block of memory to the hard disk so that the memory can be used for another purpose. The downside to this would be that writing to a disk and rewriting over is slower than ram would process it. So to avoid this I would recommend a setup that has no less than 64 GB of ram. Ideally 128 or even 256 GB would be optimal with the option to add more in the future as the demand from players grows.
5. **Distributed Systems and Networks**: I would recommend to avoid outages or drop in connection to the web to have a back service provider for connection to the internet. For example, we could have Verizon FiOS as primary and comcast Xfinity as a secondary connection (connection companies may vary based on server location). This would allow if our primary went down, we would still have a backup connection that would still allow user to connect to the game. Also, we could allocate after so much stress over the primary connection that the second connection can become live reducing stress on the primary. I would also recommend fiber optics cabling between the database tier and the application tier to improve speed and communication between the two. I would also recommend a REST api setup for this project. This would allow the application to be transferred and played on web services that support it across multiple platforms.
6. **Security**: Linux is a very secure operating system with segmented working environments. This segmentation of the working environments allows for higher protection against virus attacks. This is a leg up on windows with windows deferring not to have as much segmentation in their operating system. Also, with Linux holding a smaller portion of the market when compared to windows or mac is another reason it is more secure with less users using the operating system making it more secure. Another bonus of Linux is that everything inside of a Linux operating system is treated as a file. Even mounted drives are obtained via a special file in the built-in file system. This allows read write and execute access to be assigned on a file-by-file basis. Allowing for files to only be allowed access by certain access by certain aspects of the program allowing it to block spread if a virus were to get in. For network security we will have complete HTTPS support for secure web browsers.