

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

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

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

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

[**System Architecture View**](#_3dy6vkm)3

[**Domain Model**](#_1t3h5sf)3

[**Evaluation**](#_2s8eyo1)3

[**Recommendations**](#_3rdcrjn)5

## [Document Revision History](#_3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/17/21 | Brian Gonzalez | <Brief description of changes in this revision> |

**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](#_2et92p0)

The Gaming Room wants to create a web-based game that will be able to run across multiple platforms. This game is going to be called “Draw It or Lose It” where players are competing to guess what is being drawn. The players are put into teams in which then a picture is taken from a library, which requires the team to guess what it may be before the timer runs out. This is currently only available on Android devices with hopes to make this multiplatform.

## [Design Constraints](#_tyjcwt)

For team based games, more than one team needs to be involved. Each team has to account for multiple people. The team and game names must be unique in order to ensure there will not be duplicate team names or games. Only one possible instance of the game can exist at one time. All of this has to be accounted for while running on multiple platforms. Luckily, this game already is running on Android software, already eliminating one portion of the battle. This makes getting it to run on iOS a little bit easier as we have the groundwork for another mobile device. This leaves the plethora of other operating systems like Windows or Linux. We are going to have to find a way to implement the existing code and port it over to these operating systems, or, write the code within the operation system guidelines such as Swift for Apple.

## [System Architecture View](#_3dy6vkm)

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](#_1t3h5sf)

In the provided UML diagram, we have Entity being a SuperClass. This is due to it creating a relationship between “Game”, “Team”, and “Player” classes, This means they all inherit the same information such as “name” and “id” from the Entity Superclass. Team and Player classes also have a “has a” type, which is labeled in UML as aggregation. This means that one instance of a class has a reference to another instance in another class. The diagram displays also that GameService has a reference to Games, while Games has a reference to Team and team to Player. We can also see encapsulation provided in the diagram as Entity is protecting data that is received in the program. It is limiting public access to what is being inputted, thus, protecting the users who are playing.

**"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](#_2s8eyo1)

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** | Apple, being the tech giants that they are, has their own server that people who use their services can use. Mac OS X Server is only $20, so when it comes to budgeting this is a very viable option here. | Linux is where things can be a little dicey. There are many different server options here for Linux and being an open source platform a lot of these servers are open sourced as well. They are also very inexpensive. Finding someone who is familiar with Linux is a majority of the battle as it is not as popular as the other operating software is. | Windows, like Apple, also has their own server. It is called Windows Server and, according to Microsoft, it seems to be on the costly side. There may be third party options that are cheaper, but with Windows Server, you're really paying for a fully functioning out of the box server. | Running a server on a mobile device is the least cost of the options mentioned. But, being that it's a mobile device, and the operating power is far less than those of full fledged desktops, this may not be the smartest option. |
| **Client Side** | There’s not much cost in terms of operating system as Mac is not an open source platform. The cost will ultimately come down to being able to have someone who knows this OS well so that the time it will take to implement changes will not be long. | Cost for Linux would be relatively low as being open source makes things cheaper. Time and expertise will take the bulk of the work as finding someone who is adept at Linux will require time and patience since Linux is not commonly used. | The cost would be just setting up Windows to the settings to code the applications. For time, it would be easier to find people who will be able to code with Windows as Windows is something that is widely used. | Cost is not an issue here and mobile device experience is easy to find as everyone has a smart phone these days. The only problem would be investing time in this market as there are multiple operating systems across multiple phones and there would be a need to understand each and every one of them. |
| **Development Tools** | Swift is the most commonly used language when it comes to coding Apple applications. | The primary dev tools to use for Linux are Eclipse and Atom. | The primarily used coding application is Eclipse or Visual Studio as the flexibility between the two is great in terms of what languages are available to use. | iOS apps are typically on the same boat as Mac as they are both developed in Swift. |

## 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’d recommend that The Gaming Room would start on primarily Windows devices. Windows is widely used and understood amongst a majority of programmers and users so this operating system is familiar.
2. **Operating Systems Architectures**: Windows provides the ability for all windows based apps the use of a GUI, or a Graphical User Interface. This also allows the users the ability to show all sorts of system information and resources. Due to the various software packages they have available, the experience for each user is tailored to what they need it for. Developers who use Windows will find themselves being able to use a vast array of languages as well as being able to use the built in Command Prompt Powershell for server configuration.
3. **Storage Management**: Dealing with Windows, especially with Windows 10, there is a feature that they have implemented called Storage Sense. What this does is it pretty much optimizes the storage that you have for your computer. It will get rid of unwanted or unused files, temp files in order to ensure you have the space needed at all times. There are also cloud capabilities for Windows if that is something needed as well. It also helps that there is a built in file directory so that developers can save their projects and files in marked places for use whenever they need it.
4. **Memory Management**: With Windows, the memory management offered allows you to allocate a certain amount of memory to an application allowing full optimization for an application. Being in this case the game is going to require a large selection of pictures to display, this can be useful with being able to fully allocate the photos, at random, ti the reserved space on the RAM.
5. **Distributed Systems and Networks**: There are many common issues in terms of distributed systems and networks. Common problems start when there are a lot of games and players queuing at once. This offers a lot of server congestion and can ultimately result in lagging of the network. A lot of gaming companies don't really allocate enough server space a majority of times upon releases, which leads to a really bad launch.
6. **Security**: Windows come with a strong built in security software. Unlike things like McAfee and Norton, which is almost the same as installing bloatware onto your computer, Windows Security periodically scans and assesses your system upon it being on. It makes sure you or your files are not compromised. It also has a lock screen in which requires a password or pin in order to be able to use the computer.