

<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 | <mm/dd/yy> | <Your-Name> | <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](#_sbfa50wo7nsh)

We want to create a drawing game where the teams must guess what is being drawn to win. The twist to the game is that the players are not actually drawing the image, but a stock image is going to be rendered in over time, at 30 seconds it will be completely rendered in. There will be four rounds and each round will be one minute long. If the time finishes and the team didn’t guess the answer the remaining teams will have an opportunity to guess, they will have a 15 second limit.

## [Design Constraints](#_2et92p0)

There are some constraints when it comes to building an app for a web-based environment.

1. We need to make sure that the application can communicate with several users on the web
2. Rendering the picture slowly until the 30 seconds hit will be challenging
3. The web-based environment needs to have access to the stock images that will be used
4. To help with this constraints we will need to use a language that can be easily used in the web environment and can read files

## [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 Entity class is a parent class to Game, Team, and Player. The arrows going from Game, Team and Player to Entity show us that it has inheritance. GameService can only use Games attributes, the line connecting GameService to Game shows 0…\*, this means that there can not be a GameService if there is no game, the same thing means for Game with Team, Team with Player.

**"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** | Mac has some great advantages when it comes to web-based software. Mac makes it much easier to create features with Win group, compared to Windows. A disadvantage that Mac has is that it runs on a limited number of devices, which in this case is only Apple products, Mac also can’t run some programs that Windows can. | Linux is powerful, stable, and very easy to use. This is good because it would ensure us that the application, we make would be safe and not fail. | Windows is one of the most used and well-known platforms used. Windows is well known for its web hosting advantages along with Linux. | Mobile Devices have gained popularity when it comes to games, especially simple games like the drawing game in question. A big disadvantage that mobile devices has is the process it takes to get the game available on the app store. It will require a much bigger team and a wider variety of skills. |
| **Client Side** | Although Mac is very protected and is widely used, Mac is only used on Apple devices. So, using Mac for this program will require the users to have an apple device which is not ideal. | Linux is widely used and is considered one of the best operating systems. When it comes to cost Linux is a free system all you would need is a computer that runs Linux, which is common. | Using windows will allow more people to be able to use the application. It is also a great platform for web sites and windows apps. | Designing this app for mobile devices will come with some costs, although almost everyone in this world has access to a mobile device. Phones and tablets have more limited storage and screen size than a computer would. |
| **Development Tools** | The language that is most used for Mac is C, so that what would probably be used. | Linux can be used with almost any language so picking out a language to use if we use Linux won’t be too difficult. | Windows has visual studios which is a great IDE for C++ which is the language that is mostly used in windows. C++ is one of the most used languages in the world. | Using a mobile device will limit our use of languages, when it comes to IOS we can use Swiftic and for Android, Java is the official language. |

## 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**: A good operating platform that can be used is Windows, since it is more widely used, and you don’t need to own a specific brand of computer. Windows will give us the range and ability to create this app for the web.
2. **Operating Systems Architectures**: Windows can run in a 64 bit and a 32 bit; windows also use a layered design.
3. **Storage Management**: An appropriate storage is an SDD this will allow for faster loads and the memory is stored on the software.
4. **Memory Management**: Windows uses a 32 bit and 64 bits, both have their virtual address, only downside is that a 32-bit process cannot get memory from a 64 bit.
5. **Distributed Systems and Networks**: Games that are played online share a database between players, and each player has their own physical database. We would need to connect the shared database with the players.
6. **Security**: Security is a huge thing when it comes to an application and web sites. When developing a game, you want to secure the data and protect it from your competitors. Most importantly you want to protect all your user’s data in case a hacker breaches the system.