

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/7/2023 | Cooper David | Changes present are: Document revision history, executive summary, design constraints, system architecture, domain model, and recommendations. |

**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 company we took on as a client, The Game Room, wants to make a web-based game. The game is only on android now, but they want to expand to different OS’s. The game is called Draw it or lose it, a game based off of a TV game show called Win, Lose, or Draw.

## Requirements

I will cover the requirements more down below, as I would be pretty much be repeating myself when talking about the design constraints.

## [Design Constraints](#_2et92p0)

Here are the design constraints for the idea:

* The game must be web-based so that many can easily access it
* The game would have 4 one minute rounds each
* The concept of the game is to have team members try and guess part of a puzzle of sorts
* Drawings must be ready and complete by a 30 second time limit
* If the team member doesn’t guess the answer before the time limit, then the leftover teams have 1 guess each to solve the puzzle within a 15 second time limit.
* The game needs to be able to have 1 or multiple teams within a game
* Each team will need multiple players in it
* Game and team names need to be unique, and the game needs to check to make sure there aren’t duplicate team names
* Only one instance of the game can be in existence at a time.

## [Domain Model](#_8h2ehzxfam4o)

The Entity class is essentially our handler for everything but the ProgramDriver and SingletonTester, and GameService classes. The other three inherit characteristics from the Entity class. The four classes created: Game, Team, Player, and GameService, all reference each other. The ProgramDriver class is how we create or build the project to meet the requirements of our clients. From the ProgramDriver class, we can access all the classes that have been made, and execute them as well. And there’s also the SingletonTester class. For each class that is being used, it allows the project to run in line with our design constraints, and it allows for there to be more than one team and players, while having only one game session active at the same time.

**"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 is easy to work on server side and also accessible. It also has an easy to use interface as well. | Linux is cost effective to develop on, however it’s harder to navigate, but most things are done through command prompts/shells. | The server side of this is expensive, but has a command prompt and a good user GUI. | This one is more difficult to develop on, especially since you have to keep in mind all the different user specs. |
| **Client Side** | This OS needs a basic understanding of how computers work. It’s also one of the most expensive OS’s you’ll find out there. | The client needs to know how to operate a command console, and this is one of the more expert OS’s to use. | Windows is one of the easier to use OS’s for clients. That being said, Windows is also expensive, compare windows costs to Mac OS’s. | This OS is one of the easiest ones for clients to use. That being said, users may not be able to run certain programs on their phones, like some iOS apps don’t work on Android and vice versa. |
| **Development Tools** | Languages that Mac supports is usually JavaScript and CSS, both are HTML Based. But, GitHub, Pycharm, visual studio and Swift can also be used on Mac. | Linux supports HTML based languages like CSS and JavaScript, but also supports languages like Ruby and Python. | Windows supports the same languages as Linux and Mac, but also Eclipse and other languages/IDES. | Same as Linux, but this also supports c++ and php. |

## 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 feel like Linux would be a good choice for this project. This is because android, one of the most common OS’s on Mobile Devices, is based on a type of Linux. And I feel like things would transition easier due to that fact.
2. **Operating Systems Architectures**: Linux is a very customizable OS, with a lot of it’s key features being through command prompts and it’s vast customizability for an OS. Though, maybe something more user friendly like Ubuntu would be best for creating programs within.
3. **Storage Management**: In linux, you can use the df and du commands to manage your storage. Df is used to check the disk usage of the file system, and du is used to analyze files/directories and how much storage they are taking up.
4. **Memory Management**: Linux often divides up it’s memory into separate pages with their own addresses. The Virtual Memory manager in Linux is the thing that manages said memory pages.
5. **Distributed Systems and Networks**: Network based multiplayer interactions such as games that are accessed on the internet typically have a database shared with the players that distribute and work with one another on the network. Currently, network game developers have to implement the shared database and the multiplayer communications by themselves.
6. **Security**: Often, linux users are required to have passcodes if they want to access certain parts of their computer. For example, the term “sudo” in anything in a command prompt requires a user passcode in order to progress. Also, only administrators can have access to sensitive parts of the OS.