

<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 | 05/21/2022 | Nathaniel Maplethorpe | Executive summary and design constraints, domain model and evaluation. |
| 1.1 | 06/4/2022 | Nathaniel  Maplethorpe | Updated Evaluation |
| 1.2 | 06/18/22 | Nathaniel  Maplethorpe | Updated Reccomendations |

**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)

Our client The Gaming Room would like to develop a web-based game that will be accessible on various different platforms. Currently the game Draw it or Lose it, is only available on android based platform, as of now the game consists of different teams and the object of the game is to guess what images are produced. So it must be able to support multiple teams as well as several players designated too those teams, as well as making it so game being able to exist at a time.

## [Design Constraints](#_2et92p0)

* Capacity for one or more teams at a time.
* Multiple players to one team
* Unique game and team names
* Ability to check if names are in use
* Multiple platform capability

## [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)

For the application draw it or lose it, we added a new entity class which is a base class that will be used to hold the common attributes and behaviors of the program it will also be serving as a parent class to the classes Game, Team, and Player, inheriting from all those classes, GameService also shares with them as well as holding majority of the original methods that we will use to make the game which will allow us to add multiple different players to a team as well as adding multiple teams to a given game, the classes ProgramDriver and SingletonTester are separate as we can see now the programdriver class consists of the main method which is used to call the singletontester in order to test our code, now the singleton tester also checks to see if there’s a single occurrence of the game is running at any given time, this should fufil the games requirement of testing if one game instance exists in the memory.

**"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** | Popular for web hosting.  Flexible terminal allowing you to configure servers better  Expensive compared to the alternatives | Similar to mac but more cost effective  Higher security  Difficulty find applications that support the required needs. | Most software compatibility compared to the other operating systems  Limited hosting platforms, and virus susceptibility  Most popular of the OS | Wider reach giving it better compatibility and more cost effective.  Limited hardware such as ram making it less scalable  Poor security when it comes to the others |
| **Client Side** | Generally need a mac to develop for mac.  Languauges like sdk objective-c are lesser known so harder to find workers.  Cost is similar to windows but has a few extra steps. | Straight forward development usually popular languages such as java c/c++ and python are common.  Generally harder to setup compared to mac and windows  Lowest cost to develop. | Also uses common languages making it easier to find workers.  Generally the least amount of work to setup and run  Shares similar cost to mac. | More flexible to the clients and developers.  Generally more difficult to implement compared to the other devices. |
| **Development Tools** | For mac swift is the most populars option and can use notepad++, and mac can generally run the most popular languauges while less popular than swift can run html, css, javascript for front end, and can run java, python and ruby, for all purposes. | Linux has the ability to use visual studio and multiple other ide’s as well as notepad++, jt can also use the same languauges as mac making it accessible and able to program multiple projects | Windows is easier to run than linux but runs the same ide’s such as visual studio and eclipse, as well as pycharm, while also being able to use notpad++ sharing identical languauges to the other 2 operating systems but is generally easier to setup. | Uses specific libraries for apps that make it easy and secure.  Uses android and swift to create multiple applications that can run on all 3 machines. |

## 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**: personally I think gameroom should be on a windows device seeing as there is farm more software as well as ease of use making it easier to get your project up and running
2. **Operating Systems Architectures**: Windows can use its graphical user interface as well as using its system resources making it easier for the applications to run with less resources as well as making it so they can be used on a user account or a server
3. **Storage Management**: windows has a storage management system that helps manage files on your pc as well as how much it takes up. You may also choose where the save would go making the applications easier to access, but they also have a cloud service that allows for easy placement for large projects.
4. **Memory Management**: Memory allocation will allow for easy storage when it comes to using picture and various other objects outside of the default folders, allowing us to keep the project more secure
5. **Distributed Systems and Networks**: using a client-server distribution stems will allow us to have each application depend on a single server application for the game, in doing so the application can be developed to the clients systems strength, using a strong server network will also be needed as the games success is dependent on having many clients connect to one server
6. **Security**: windows has windows defender built in using this as a security function for any windows system, all we wouldn’t need to do is encrypt all the data that will be sent back and forth.