

<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 | 05/17/2023 | Samuel Walters | Design document for first revision |

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.5 | 5/31/2023 | Samuel Walters | Design document for second 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)

This will be a web application that will play the game Draw it or Lose it. Because this will be a web app there will be requirements for streaming through the cloud or some other service. The current design will consist of Java code to create the app and classes.

## Requirements

Must have choice of one or more *unique* teams. Each team with multiple unique players. Only one instance of the game run in memory.

## [Design Constraints](#_2et92p0)

Design requirements as follows:

* Client requires web based to increase audience.
* Each game will be 4 rounds.
* Rounds will be one minute each.
* Drawings must be rendered steadily, and completely rendered at 30 seconds.
* Each game will have one or more teams.
* Each team will have multiple players.
* Each team and player ID must be unique.
* Only one instance of a game will exist in memory. Accomplished through applying unique ID’s to teams, players, and game instance.

## [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)

Entity class is the superclass for the following: Game, Team, and Player class all inherit from Entity. Entity class provides id, which is unique, and ensure that only one instance of each runs in memory. All classes are accessed through ProgramDriver. SingletonTester provides testing functionality to ensure that there is only one instance running in 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** | MacOS has discontinued the macOS Server according to their website. | Linux is an open source solution, and the most popular for servers. Highly customizable with no licensing fees, and a strong community. | Windows server has a licensing fee per year. However there is a GUI, as well as great documentation, and a long term support of up to 10 years from release. | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | Will run either Chromium, or Firefox. Chromium uses Blink engine, and Firefox uses Gecko. MacOS also uses Safari which is WebKit engine based. | Will run either Chromium, or Firefox. Chromium uses Blink engine, and Firefox uses Gecko. | Will run either Chromium, or Firefox. Chromium uses Blink engine, and Firefox uses Gecko. Microsoft edge is available, but is a chromium based browser. | All IOS devices MUST use the WebKit engine. Android browsers include Firefox, and Chrome. |
| **Development Tools** | Developing on MacOS, can use eclipse for Mac, however must run on Apple’s hardware and software. | Will be a learning experience for people used to mac or Windows. However the OS is free and there are no license fees. | Windows is the most popular OS, so likely already used to it. However there are different file structures from Linux and Mac, so depends on the server. | Will use the browser Dev Tools in order to preview how the pages can look for mobile devices. |

## 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 server should run on Linux, with some flavor of Ubuntu. The 22.04.02 LTS will be supported till April 2027, with commercial support offered at a fee from Canonical. All major cloud providers also offer Ubuntu server, which is the recommendation of CTS.
2. **Operating Systems Architectures**: Ubuntu architecture is based on Debian/GNU Linux.
3. **Storage Management**: Storage management for cloud can include elastic storage. For the Database including User ID and other information, there are complete solutions provided for PostgreSQL, which can also include elastic storage, or easily upgradable options.
4. **Memory Management**: The Java Runtime Environment, OpenJDK will need to be installed. Instances of the game will be managed by ensuring the SingletonTester is successful with checking that only one instance is instantiated each time a game is initiated.
5. **Distributed Systems and Networks**: By running through a cloud provider and utilizing multiple nodes, there will be uptime approaching 100%. All the networking will be done by the cloud provider. We will use Terraform, to provision our containers with our app running a microservice based platform. This way we can rapidly scale up or down to save money on cloud costs, as well as reduce latency.
6. **Security**: User information will be stored server-side, leaving no extra information to be stolen. The User will log in, and have access to only their profile. All the different platforms will log in through the web, so there is no extra requirements for client side security. Development teams and organizational wide will be best served by adopting a zero-trust approach to security, to ensure no customer leaks occur.