

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 09/17/2021 | Sarah C Jodrey | Address development of web-based game for The Gaming Room |
| 2.0 | 10/02/2021 | Sarah C Jodrey | Updated evaluation section. |
| 3.0 | 10/16/2021 | Sarah C Jodrey | Updated recommendations section. |

**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 Gaming Room has requested the development of a web-based game centered on their current Android app, Draw It or Lose It. This creation must serve multiple platforms.

At any given time, only one instance of the game can exist in memory as ensured by implementing the singleton pattern. Each game will allow for one or more teams, consisting of multiple players. The game and team names must be unique and allow access for verification of uniqueness when choosing a new name which will be verified using the iterator pattern.

## [Design Constraints](#_2et92p0)

* Current code is written for Android mobile application only. To run the web-based application on Windows, Linux, Mac and mobile devices a base code will be written in HTML and JavaScript.
* Must be cross-browser compatible. The game must be playable on any system, and in any browser, including iOS, Android, Chrome, Firefox, Mozilla, Safari, and Internet Explorer.
* Test for compatibility and functionality across multiple browsers so that any issues may be addressed.

## [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 ProgramDriver class houses the main function which drives the entire game. This uses the SingletonTester class, which contains the singleton pattern, to verify that only one instance of the game is being run at a given time.

The Entity class demonstrates abstraction and encapsulation as it houses the common attributes and methods of its subclasses. It is the super/ parent class to the Game, Team, and Player classes as they inherit the methods and attributes while creating their own. The toString() method in the Game, Team, and Player subclasses exemplify polymorphism as they override the original method in the Entity superclass.

The GameService class can create zero to many instances of the Game class even though only one can exist at a given time. In turn, the Game class can create zero to many instances of the Team class and the Team class can create zero to many instances of the Player class. Inheritance is present as Player does not inherit from Team, but Team inherits from Player, just as Game inherits from Team, but Team does not inherit from Game.

**"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** | Pro: Can host a webserver with FreeBSD.  Pro: User friendly GUI.  Pro: Moderate security with XProtect proprietary antivirus software.  Con: Expensive licensing.  Con: Hosts are harder to find. | Pro: Easy to get running as web-based host.  Pro: Will run on almost any hardware.  Pro: Free licensing.  Pro: Highly secure – customizable security options and features, restricts root access.  Con: Requires a greater level of expertise/steep learning curve.  Con: Some professional programs aren’t equipped with user support. | Pro: Great for running web-based applications.  Pro: High availability of software and support.  Pro: Moderate security with protection built into the OS.  Con: Fee for licensing and apps.  Con: More hardware requirements than other OS.  Con: Not open source. | Pro: Hosting is not required for native applications.  Con: Wireless connectivity only.  Con: Not designed to act as a high-end web-based host.  Con: Least secure due to larger market share and does not required passcodes to install software. |
| **Client Side** | Pro: Expertise level requires accessing browser.  Browsers  Safari – exclusive to Mac but will work on iOS.  Chrome – works on all OS.  Firefox – works on all OS and mobile devices | Pro: Expertise level requires accessing browser.  Browsers  Edge – will not work on Mac or mobile devices.  Chrome – works on all OS and mobile devices.  Firefox – works on all OS and mobile devices. | Pro: Expertise level requires accessing browser.  Browsers  Edge – will not work on Mac or mobile devices.  Chrome – works on all OS and mobile devices.  Firefox – works on all OS and mobile devices. | Pro: Expertise level requires accessing browser.  Con: Experience may vary if client uses more than one/different mobile devices (due to screen size and accessing browser).  Browsers  Safari – exclusive to iOS.  Chrome – default on Android, but also works on iOS and all OS.  Firefox – works on Android and iOS, all OS, and has HTML5 support. |
| **Development Tools** | Front End Languages  HTML, CSS, JavaScript  Backend Languages Java  Web Development IDE  Visual Studio, Atom, and IntelliJ. | Front End Languages  HTML, CSS, JavaScript  Backend Languages  Java  Web Development IDE  Visual Studio, Atom, and IntelliJ.  Pro: Most IDE’s can be used for development. | Front End Languages  HTML, CSS, JavaScript  Backend Languages  Java  Web Development IDE  Visual Studio, Atom, and IntelliJ.  Pro: Can be developed on any platform and use a majority of IDE’s. | Con: Development will need to be non-mobile - occur on a desktop or laptop machine.  Languages  Swift (iOS), Objective C(iOS), and JavaScript (using Hybrid App) |

## 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 Linux operating platform allows for the Draw It or Lose It game to be run on the cloud which is perfect for expansion. As the game changes, or increases in need for data, the AWS cloud can be adjusted to meet those needs. This choice saves money without sacrificing performance or quality.
2. **Operating Systems Architectures**: The Linux architecture has essentially four layers. The kernel is the core component which interacts directly with the hardware. The shell is the interface taking input from the user sending it to the Kernel and taking output from the Kernel to the shell. The outermost layer are the applications, web-browsers, and utilities, IDE’s, running on the shell. The kernel would only need to be customized for the Amazon EC2 (Elastic Compute Cloud).
3. **Storage Management**: For efficiency, the Amazon Simple Storage Service (S3) is recommended. This is a benefit because you are only paying for the storage you use. Since storage is not limited, it can be adjusted as the development expands or contracts. In comparison, storing data locally on a disk could result in paying for more than you need, not having enough, and you run the risk of hardware failures impacting the game.
4. **Memory Management**: The game will need at least 1.6 GB of memory to store the high-resolution images. This memory can be purchased on the cloud. It is scalable, so more, or less, can be purchased depending on need. The amount of memory needed can be easily accessed via the AWS control panel. Setting cache limits, and settings, can help deliver a game experience of consistent speed and quality.
5. **Distributed Systems and Networks**: RESTful is used to communicate between the client and the server for the Draw It or Lose It game. The game will only have one server where the shared game instances are maintained. The client will use the RESTful API to interact with the server.

Using the AWS cloud for the server provides advantages like moving the node to another server that is functioning. In the case of a system outage, another server would start, restoring itself. Connectivity would be relevant to the clients’ network speed but having AWS services in more locations/regions will help address this issue.

1. **Security:** AWS hosting Linux provides manual security tasks to become automated which saves time and ensures basic security measures are adhered to. Any changes made to the AWS requires authentication and authorization based on roles. The role-based authorization and authentication sets limits on who can edit game settings or make significant changes to the code. On the client side, using in-flight data encryption will increase security so that data being sent to and from the server is protected from things like middleman attacks.