

<The Gaming Room>

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 05/14/2023 | Andres Trujillo | Created a UML diagram, explained the purpose of software design solutions involved. |
| 2.0 | 06/02/2023 | Andres Trujillo | Evaluated server side, client side, and development tools and requirements. |
| 3.0 | 06/14/2023 | Andres Trujillo | Recommendations added. |

**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 problem: Creative Technology Solutions is updating The Gaming Room. Currently an Android application that consists of players within a team working together to guess what is being drawn. Instead of drawing on the application, the teams will be presented with images from a large database the clues as to what the winning phrase, title or object is being represented on screen. This game will be called Draw it or Lose it. The goal is to create an application that hosts multiple teams consisting of a varying number of players that is web-based and serves multiple platforms instead of their current distribution, Draw it Lose it made for Android phones. Creating code in the Java Development Kit will ensure that this game can be webhosted and run on multiple platforms as well as having a library of code to ease and ensure rapid deployment. This holds tremendous value because you can easily migrate the game to another platform by installing the Java Runtime Environment which is a multi-platform interpreter capable of running Java applications across many different computing environments such as MacOS, Linux, and Windows. Allowing the game to reach multiple gaming communities through one software deployment to the cloud where the game can be downloaded and accessed easily.

## Requirements

**Business requirements:**

The application must be web-based therefore it needs to be hosted in the cloud for multiple users around the world to create teams. The team names must be unique, as well as the players’ names within each team must be unique. The application needs to have access to a large database in order to host the multiple pictures used to represent the clue, phrase or thing that the players must guess in order to win.

**Technical Requirements:**

The server needs to be able to have enough memory to handle many players and be easily scalable up and down while the game is running to accommodate the growing player base. There also needs to be a database that can store images and a programmer to develop the code for the requests and queries to the database on behalf of the game. There can only be one game service that keeps track of teams and their current game states, such as points and player ID’s. The same will be true for each player and game because the user will want to join a specific game which will have the list of teams competing and the players on each team. This will allow multiple games to be run across the gaming service.

## [Design Constraints](#_2et92p0)

The design constraints posed by our problem of creating a new web-based game consist of computing resources, the time it takes to deploy and the security of the application, such as preventing sensitive information to be published by the application. Having a web-based game comes with budget concerns if millions of users start playing the game, because cloud-based solutions will require monthly payments for access to their computing infrastructure. If the game goes down because of limitations to this tier-based payment system, it will affect the users gaming experience. The team names need to be unique because the players logging in and out will want to regroup with their team and this will allow teams to be searched by name for the best user experience. The software will check each game, team and player name to ensure it is unique. The design constraints imposed by this problem will be to find an efficient and speedy way to find names among the multiple list’s possibly being search consecutively.

## [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)

Starting from the top left we have a Program Driver class that uses a Singleton Tester class to test and make sure our Game Service singleton was properly created. The next class diagram shown to the right of the singleton tester is the Entity class which holds in memory our unique ID and name for our Game, Team, and Player classes. This abstraction allows the programmers to hide details pertinent to these other objects’ attributes. This type of inheritance is important because it will be able to keep code isolated to their intended classes, therefore keeping them easily readable and extendable even further in the future. The Entity class has a private constructor with empty parameters because it will not allow an Entity to be created without specific parameters such as the next line denotes. That next line is the public default constructor with two parameters for the Player, Game or Team classes. Containing the ID and the name of their extended classes.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | There aren’t many strengths to hosting a web based software solution on the Mac except that it has strong security, a GUI and a knowledgeable team able to assist you. But its purpose is for personal computing. | Linux is free and open sourced which means the community centered around improving Linux moves at a much more rapid pace. There are many servers currently running on Linux which make it the go to for web-based application solutions. | Windows Server OS is very customizable and has received many improvements as they continue dominating the world markets but ultimately it is a more costly solution than Linux due to various factors such as technical expertise and cost prohibition. | Mobile devices would be great to host a web-based solution but the CPU power and the memory it holds would ultimately be its downfall. Not an insignificant number of processes can run on mobile devices, but this aims to be suited for personal use. |
| **Client Side** | The software development characteristics that pertain to Mac users is that there needs to be an aesthetic that conforms to apples design methodologies so that the user can feel at home with the product. MacOS is easy to learn and easy to teach making it great for clients and enabling programmers to compete in this space. | Linux is difficult to master and even more so with the added customizations and security vulnerabilities found every day making it difficult for clients to use but a very good tool for software developers. The only plus side for clients is the price it takes to run. | Windows is perfect for the client, being both complex and realistically simple enough for the hobbyist programmer. You can use advanced tools and settings within this operating system or not at all allowing flexible use. Easy to learn yet difficult to master. | Easy to learn for clients and has tons of features able to help users navigate the city or bus routes but very minimal support for computer programmers. |
| **Development Tools** | Xcode is great for generating unit tests and its built into the IDE. You can use major IDE code editors such as Code::Blocks, Eclipse and my personal favorite. Vim. Languages able to run are C, C++, Java. Python, JavaScript, and Swift. | Linux has vi and improved text editor and can also run famous distributions such as JetBrains suite which is similar to the Visual Studio platform. And can run C, C++, C#, Python, JavaScript, Java and machine code. | Windows has Visual Studio Suite which is fantastic and VsCode which is equally as good but not as robust and is very lightweight. Can run C, C++, C#, Python, JavaScript, Java and machine code. | Not many programming languages are supported except to enhance user experience through the browser such as JavaScript for videos. But can run Java applications and C applications due to the Linux environment. The exception to the support that it must be downloaded from the Google Play Store. |

## Recommendations

1. **Operating Platform**: The best operating platform to deploy the gaming service for the Gaming Room is Linux because it is a small lightweight system capable of running Java from the Java Runtime Environment. Multiple different operating systems will be able to access this service because it will be solely responsible for running the game application.
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.> The operating system architectures inherent in the Linux operating system is that it can be used as a server which is connected to the internet. Through this gateway the Linux server will be able to process incoming requests from players to see if they can play within the Gaming room. Once the operating system architecture is connected to the internet and being a valid gaming service to our game will we be able to search for our game or team to join. The games currently being played will be in memory and have concurrent users attaching themselves to the process. This is handled by the CPU and the direct read access memory. If the game is old and not in memory the CPU will retrieve it from the hard disk and load it into memory. Verifying that the game exists or creating a new game with a unique name.
3. **Storage Management**: An appropriate storage management system would be to have multiple storage units capable of serving the platform and accessible by the operating platform/system. This can be a server with multiple hard disks to be able to store multiple gaming instances and to potentially store a database in case the database is in house instead of in the cloud being accessed through an application programmable interface. The file system used could be ext4 because it can support large files and file systems up to 16 terabytes, with unlimited sub directories. An algorithm that can help with the management of team, player, game names is to save them in alphabetical order to perform binary search on the characters.
4. **Memory Management**: The memory management techniques employed by the game Draw It or Lose It will enable the player to find his team quickly by storing the data effectively through the use of algorithms. By being able to binary search the names will enable the program to find team, player and game names much faster.
5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

Distributed software is a large and vast business model developed to prevent program outages by mitigation techniques such as redundancy. This redundant model of programming allows the user experience to be uninterrupted if something happens to the server because distributed systems can rely on other computers to serve information. This abstraction allows the programmer to program for one platform such as Linux and have it deployed immediately.

1. **Security**:

User security is handled differently by each platform, currently MacOS has a touch bar that allows users to login with their fingerprint and this works as a two-factor authentication method to be used alongside their password (1st factor is password, 2nd factor is fingerprint). Other security techniques are that passwords must be long enough to satisfy certain requirements such as 16 characters long with at least one special character, more than one digit and not be a common phrase(gibberish). User information can be protected by encrypting the data as it is sent across the internet with the use of a public and private key. All these platforms use these security features. Window’s Lenovo lineup of personal computers uses fingerprint like MacOS. Linux can be made compatible with fingerprint readers and now becomes standard in the Ubuntu distribution.