

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/11/2023 | Dawson Kennedy | Updated Executive Summary, Evaluation and Recomendation. |

**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](#_heading=h.35nkun2)

Creative Technologies Solutions (CTS) is excited to partner with The Gaming Room to develop a web-based version of their popular game, Draw It or Lose it. The objective is to expand the current Android app to a multi-platform web game that supports multiple teams and players. CTS aims to streamline the development process, ensuring uniqueness in game and team names, and implementing a robust system architecture to meet the client’s needs.

## Requirements

1. Game Functionality
   1. Ability to support one or more teams.
   2. Each team can have multiple players.
   3. Unique names for games and teams to avoid duplication.
   4. Only one instance of the game should exist in memory at a time.
2. User Interaction:
   1. Users can create and join games.
   2. Teams and players can be added to the game.
   3. Checking for uniqueness of game and team names during creation.
3. System Architecture:
   1. Web-based distribution environment.
   2. Singleton pattern for GameService to ensure only one instance exists.
   3. Iterator pattern for checking unique names during game team creation.

## [Design Constraints](#_heading=h.1ksv4uv)

1. Web-Based Distribution Environment:
   1. Consideration for potential latency and bandwidth issues.
   2. Address security and data integrity concerns in a distributed environment.
   3. Plan for scalability to support multiple concurrent users.
2. Singleton Pattern:
   1. Implement an iterator pattern for checking unique names during game and team creation.
   2. Efficiently search for existing instances with the same name.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

1. Entity:
   1. Serves as the base class for common attributes such as ‘id’ and ‘name’ shared by Game, Team and Player
2. Game:
   1. Inherits from Entity and includes list of teams.
   2. Provides the methods for adding a team and generating a string representation.
3. Team:
   1. Inherits from Entity and includes player list.
   2. Provides methods for adding players
4. Player
   1. Inherits from Entity and represents individual players.

**"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](#_heading=h.z337ya)

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** | **Characteristics:**  UNIX-based operating system.  Excellent user interface.  Supports various applications and frameworks.  **Advantages:**  Robust development environment.  Well integrated with open source tools.  Suitable for small to medium sized web applications.  **Weaknesses:**  Limited availability of commercial software.  Not as commonly used in enterprise lever server deployments. | **Characteristics:**  Open source operating system with various distributions.  Highly customizable and configurable.  Widely used for server hosting.  **Advantages:**  Excellent stability and performance  Cost effective  Broad support for server applications and frameworks  **Weaknesses:**  May have a bit of a steeper learning curve for some users.  Limited support for certain commercial applications. | **Characteristics:**  Proprietary operating system developed by Microsoft.  Widely used  Supports a range of server applications.  **Advantages:**  Seamless integration with Microsoft technologies.  Extensive commercial software support  User friendly interface.  **Weaknesses:**  Higher licensing cost  Generally considered less secure than Linux. | **Characteristics:**  Various operating systems.  Limited resources compared to traditional servers  **Advantages:**  Widely used  ideal for lightweight, mobile focused web based applications.  **Weaknesses:**  Limited processing, power and memory.  may have connectivity issues. |
| **Client Side** | Cost - Generally Mac development tools can be expensive  Time - Development time may be influenced by the need of macOS specific features.  Expertise - Requires knowledge of macOS development tools and frameworks. | Cost - Often open source reducing cost.  Time- Development may be influenced by compatibility with Linux Libraries and dependencies.  Expertise- Requires expertise in Linux development and command line tools. | Cost - Development tools vary in cost. Can also factory in Windows license for expense.  Time- Development may be influenced by compatibility with Windows specific features.  Expertise- Requires knowledge of Windows development tools and API’s. | Cost - Development tools may have an assorted cost.  Time- Development may be influenced by platform specific requirements and testing.  Expertise- Requires expertise in mobile development frameworks. |
| **Development Tools** | Xcode (IDE) , Git (Version Control), XCTest (Testing), Swift, Homebrew. | VSCode, Git, JUnit, C, C++, Python, Ruby, Eclipse. | Visual Studios, Git, NUnit, .NET, C#, C++, VS Code. | Xcode (iOS), Android Studio (Android), Git, XCTest, Espresso, Swift, Flutter/Dart SDK. |

## 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**: We Based Distribution Environment.
2. **Operating Systems Architectures**: Client side - Users device (i.e. Windows, Linux, Mac and Mobile Device.) Server Side - Linux based server architecture for hosting the web based game.
3. **Storage Management**: A real database management system like MySQL or PostgreSQL.
4. **Memory Management**: Make it simple. Do automatic memory collection. It would help reduce the risk of memory related issues and make it easier on developers.
5. **Distributed Systems and Networks**:
   1. Connectivity - Ensure robust error handling for connectivity.
   2. Outages - Implement mechanism for graceful degradation during network outages.
   3. Load Balancing - Distribute incoming traffic efficiently across multiple servers for scalability.
6. **Security**: SSL Encryption and OAuth Authentication.
   1. SSL Encryption - Secures data transmitted over network.
   2. OAuth - Provides secure, token based authentication for users.