

<Leonard Foy >

# **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 | 08/01/25 | Leonard Foy | Changes done here include a change in the cover page  and additional information covering the indicated  information was included in the Document Revision  History table and below each heade  Changes made in this version include an updated cover page and the addition of required evaluation content under each relevant section. These updates are also reflected in the Document Revision History table. |

## 

### ****Executive Summary****

The Gaming Room plans to develop a web-based version of its current game, Draw It or Lose It, which is currently available only on Android. The web-based game will support multiple platforms and deliver gameplay by rendering images from a large library of stock drawings. Instead of players creating their own drawings, teams will compete to guess the correct answer based on these visual clues.

### ****Requirements****

The client has outlined the following software requirements for the game application:

* The game must support one or more teams per session.
* Each team will consist of multiple players.
* Game and team names must be unique to avoid duplication and allow name availability checks.
* Only one instance of the game should exist in memory at any given time.
* This can be ensured by generating unique identifiers for each game, team, or player instance.

## [Design Constraints](#_2et92p0)

*Draw It or Lose It* is inspired by the 1980s game show *Win, Lose or Draw*, where teams compete to guess what is being illustrated. Instead of players drawing, the game uses stock images rendered from a large library as visual clues. Each game has four rounds, with each round lasting one minute. The drawing appears gradually and is fully displayed at the 30-second mark. If the active team doesn’t guess the answer in time, the other teams each get one 15-second chance to submit a guess.

## [System Architecture View](#_ilbxbyevv6b6)

The game must support multiple platforms. Only one instance of the game should run at a time, and all game and team names must be unique.

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram outlines the structure of The Gaming Room’s system. It shows classes, attributes, methods, and their relationships.

* Attributes store key data for each class.
* “+” indicates public access; “–” indicates private.
* Game, Team, and Player inherit from the Entity superclass, which provides shared functionality.

**"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.**

**A table of software testing

AI-generated content may be incorrect.**

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Systems Architectures**: Windows supports applications with a Graphical User Interface (GUI) and provides services for accessing system resources, graphics, multimedia, messaging, and web services.

**Storage Management**: Windows includes built-in storage tools like Storage Sense, which help manage files such as game progress, user profiles, and images. This ensures game data can be easily accessed across platforms.

**Memory Management**: User and game data would be securely stored and managed by Windows’ storage system to support smooth gameplay.

**Distributed Systems and Networks**: Since the game will connect players across platforms, Windows is ideal—it supports network communication and multiplayer features easily.

**Security** Windows has built-in security features. Adding login systems, like username/password authentication and role-based access, will help protect user information across all platforms.