

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 10/13/23 | Jared Baker | * Altered cover page * Wrote executive summary * Wrote design constraints * Cleared system architecture view * Wrote domain model UML description * Wrote Evaluation Table |
| 1.1 | 10/16/23 | Jared Baker | * Revised Evaluation Table * Wrote General development information above Evaluation Table * Added FIXME to Recommendations to rewrite for project 3 submission |
| 1.2 | 10/19/23 | Jared Baker | * Revised Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is working with Creative Technology Solutions to develop a web-based game based on their current game, Draw It or Lose It. Draw It or Lose It is only available as an android app, but the new web-based game must serve multiple platforms.

The mechanics of the game are to have teams of multiple people guessing what image is being drawn with clue stock images being rendered over 30 seconds as clues. If the team does not guess correctly before one minute expires, the other teams may offer one guess each to solve the puzzle within 15 seconds. The game consists of four rounds.

The rationale for solutions to these requirements is described below.

## Requirements

* The game must support one or more teams
* Each team will have multiple players assigned
* Game and team names must be unique
* Users must be able to check whether a name is in use when choosing a team name
* Only one instance of the game can exist in memory at any given time (Singleton Pattern)

## [Design Constraints](#_2et92p0)

The game must run on multiple platforms, so we must develop with a widely supported programming language. We must also keep the restricted technical specifications of less powerful platforms, such as mobile devices, in mind.

The game must support one game, with multiple teams of multiple players all with unique names. To achieve this, we must make use of lists and input validation checks against pre-existing names. A singleton pattern should be used to ensure only one instance of the game exists at a time.

## [System Architecture View](#_ilbxbyevv6b6)

Not Applicable.

## [Domain Model](#_8h2ehzxfam4o)

The Entity superclass is a foundation for the Game, Team, and Player classes, which all inherit Entity’s variables and methods. There is a HAS-A relationship between GameService, Game, Team, and Player. This means that the unique GameService object (ensured by the singleton pattern) has a Game, which has a Team, which has a Player (or multiple). The entire package is run on the ProgramDriver that contains main() and calls for GameService to instantiate the only instance of the GameService.

**"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)

The game will use a server-style configuration for hosting the website and must be able to scale up to thousands of players across multiple platforms. A RESTful API can be used to allow for different languages across different operating systems to communicate with one server. The open-source JavaScript framework React-Native is REST based and would be a fantastic option for developing the client-side for iOS, Android, and web. The reusability and compatibility of the code for cross-platform development has made it an immensely popular choice in the development community. Multiple development teams can be avoided using a highly compatible framework like React-Native. AWS cloud services offer support across all operating systems below. They provide many services and tools that make AWS a strong choice for any development environment decision.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | User-friendly GUI and highly configurable server environment. Mac has a large library of command line tools. Only runs on relatively limited hardware. Mac includes a built-in web server component called Apache, but Mac is not very popular for enterprise-level deployments. Apache is known for being very flexible and robust.  AWS cloud solutions are available | Command shell for simple configuration. Much more cost efficient than Mac and Windows. Linux servers are regarded as very secure and reliable with the wide variety of hardening practices available. Linux is a popular choice for web development and is the operating system of choice for many web hosting providers. Nginx is popular for its scalability and management of a large number of simultaneous connections. | User-friendly GUI. Command prompt. Windows provides Internet Information Services (IIS) as web server software. IIS has modular architecture, centralized management, scalability, and many security features such as SSL/TLS data transmission. This OS is widely used for web hosting and is popular for websites using Microsoft technologies. | Hosting a server on a mobile device has a much larger list of cons than pros. Coding and managing the backend with less functionality, worse hardware, and more unique problems to craft solutions for would be unnecessarily time consuming and result in a worse product. Using a mobile device to host a server over a traditional operating system is not something I would recommend. |
| **Client Side** | Expensive platform for consumers. User-friendly, but requires a reasonable amount of time to learn to navigate the system. It has the most compatibility issues of the options, but that should not be an issue for a web-based game. | Niche consumer base. Linux data is required to use OS. Linux is not user friendly, and it takes the most time to learn between the options. Linux is almost never seen in a context outside of technology professionals and enthusiasts. | Wide range of price for consumers. User-friendly, but requires time to learn to navigate the system. Most widely used platform with extensive support for software development. | Great variability in operating systems. Virtually everyone already has one. Screen aspect ratios vary wildly and will require a more specific interface designed for touch screen use as well as the smallest potential screen. |
| **Development Tools** | Popular IDEs: Xcode, Visual Studio Code, IntelliJ IDEA, Android Studio, Eclipse, PyCharm, NetBeans, Sublime Text, Atom, and Swift.  Xcode is free for macOS developers. | Popular IDEs: Visual Studio Code, IntelliJ IDEA, Eclipse, Android Studio, PyCharm, NetBeans, Code::Blocks, Geany, BlueJ, Anjuta.  Most Linux IDEs are open source or have free community editions with no direct licensing costs.  Note: Linux has the most freedom and options for development. | Popular IDEs: Visual Studio, Visual Studio Code, Eclipse, IntelliJ IDEA, Android Studio, PyCharm, NetBeans, Code::Blocks, Dev-C++, BlueJ.  Visual Studio and PyCharm have free and paid versions with additional features and support. | Popular IDEs: Android Studio, Xcode, Visual Studio, IntelliJ IDEA, Flutter/Dart, React-Native, Swift, AppCode, Xamarin, PhoneGap.  Android Studio is free, and Visual Studio (for Xamarin or React-Native) has a free version as well as paid versions with additional features and support. |

## 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 client-side applications will be created for multiple platforms, and I would recommend React-Native framework to work with the client-server REST style API. For the server-side I would recommend using Mac. IOS is the most restrictive platform we need to run on and will require Apple’s software development kit. Other SDKs are freely available on Mac.
2. **Operating Systems Architectures**: A REST API will be implemeted for the client-server framework. The Flask framework in Python is simple, scalable, and RESTful. It is lightweight and well documented with a solid set of libraries to back it up. This would be run on a Mac 64-bit server.
3. **Storage Management**: Cloud-based storage for the server would be ideal. Potentially multiple locations depending on the concern for latency in very distant areas of the world. The game is image based and should utilize compression techniques to reduce network load. Amazon AWS Bucket has ample features for the task and has competitive pricing.
4. **Memory Management**: Memory leaks can be managed using Xcode on MacOS. The Xcode memory debugger automatically detects leaks and assists in patching them. It can also be used to identify abandoned memory.
5. **Distributed Systems and Networks**: The Apple app store, Google Play store, and external links pointing to the game’s web page will be the primary distributors for the game. The game should have a blog on a platform such as Twitter or Facebook to keep a log of development changes, announcements, community connection, and advertisements.
6. **Security**: Encryption, data / input validation, and exception handling should be fully developed for the game. A RESTful API with proper authentication should be used for account credentials in the game. Authorization permissions should also be managed properly. No client should have any access to information on the server or other client’s applications.