

**Draw It or Lose It – Web Based**

# **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 | 1/23/2021 | John Mascelli | Original |
| 2.0 | 2/5/2021 | John Mascelli | Updated Evaluation |
| 3.0 | 2/17/2021 | John Mascelli | Updated Recommendation |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room (GR) hired Creative Technical Solutions (CTS) to develop a web-based game version of its popular, Android-based “Draw It or Lose It” (DIoLI) application. This new web-based version allows GR to reach a much larger segment of its potential customer. The new software design will allow users to log on to a the DIoLI server and either create a new unique team or join an existing team. The software will support multiple players, multiple teams, and multiple games at the same time. CTS will use an Agile DevOps software development process to code, test and deploy the application. The hardware design and procurement process will start once all software application decisions complete. CTS intends to deliver the entire solution by 30 November 2021.

## [Design Constraints](#_2et92p0)

1. DIoLI is a web-based game and will need to work with a large range of operating systems on both mobile and fixed devices. JAVA code will be used to the largest extent practicable since it works well with most these operating systems – Mac, Windows, Android, etc.…
2. DIoLI must allow many players, teams, and games at any time. Code will use a multithread process (each game has its own process or thread, but shares common code, data, and files).
3. Every player and team must each have a unique ID. Using unique player and team IDs allows players to form teams and collaborate with friends and relatives to use the game. CTS will use a Singleton class design for players and teams to ensure that only 1 instance of each exists at any time.
4. Large design library will be used to provide a gaming experience with limited repeatability. Images will render at a steady rate and will be fully displayed after 1 minute. DIoLI code will use a best-of-class photo API and will wrap it with timer that will enable a steadily increasing percentage of the photo to display to all players within each game - from 0 percent of the image displayed at Sec0, to 100 percent of the image displayed at Sec59.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here.

## [Domain Model](#_8h2ehzxfam4o)

The below UML diagram provides a graphical view of the proposed high-level software architecture. The Entity class is what is referred to as the superclass. The Entity class will include common attributes and functions that will be inherited by the subclasses Game, Team, and Player. Each of these three subclasses include 0 to many associations with each other – Game to Team and Team to Player. This architecture allows the system to support multiple players, teams, and games at any instant. That is, a single game instance may have multiple teams and these teams may have multiple players. But a player can only be on a single team at any time, and a team can only be in a single game at any time.

The GameService class will have a 0 to many association with a specific Game instance. This allows the software to support multiple games at any instant and will be implemented via a multithread construct.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | **Advantages:** scalable memory, fast, efficient, supports multiple users, lots of memory, good tech, and admin support  **Weaknesses:** **expensive**, processors not upgradable  **Server-based option:** Yes | **Advantages:** Open source (**inexpensive**), fairly secure, scalable, remote admin.  **Weaknesses:** limited tech support, infrequent updates  **Server-based option:** Yes | **Advantages:** extensive admin and tech support, easy to use, frequent upgrades  **Weaknesses:** **expensive**, resource intensive, use by multiple users, not very secure  **Server-based option:**  yes | **Not a viable option.** Mobile devices will not have enough memory, power, or web connectivity to meet the needs of a server.  **Server-based option:**  No |
| **Client Side** | **Cost:** Moderate  **Time:** Dependent on the application  **Expertise:** Moderately trained, fairly ubiquitous  **App dev process dependencies**: yes, handle by coding mostly w/JAVA which is cross platform | **Cost:** Least Expensive  **Time:** Dependent on the application  **Expertise:** Highly trained expertise needed  **App dev process dependencies:** yes, handle by coding mostly w/JAVA which is cross platform | **Cost:** Moderate  **Time:** Dependent on the application  **Expertise:** Moderately trained, ubiquitous  **App dev process dependencies:** yes, handle by coding mostly w/JAVA which is cross platform | Not practical. |
| **Development Tools** | **Programming Languages:** C/C++, JAVA, SQL  **IDE/Tools:** AppCode ($600), CLion ($200), Xcode ($100)  **Dev Teams Needed:** single | **Programming Languages**: C/C++, JAVA, Python  **IDE/Tools:** Eclipse (free), Sublime ($70)  **Dev Teams Needed:** multiple | **Programming Languages:** C/C++, JAVA, Python,  **IDE/Tools:** Visual Studio ($550), NetBeans ($600), Eclipse (free)  **Dev Teams Needed:** single | **Programming Languages:** Java, C/C++/C#, Swift, Kotlin  **IDE/Tools:** Webstorm ($650)  **Dev Teams Needed:** multiple |

## Recommendations

1. **Operating Platform**: CTS recommends the use of a Windows-based operating environment. The choice of this environment balances cost, tech support, security, scalability and allows GR to expand capacity to support the expansion of GR’s suite of applications to a web-based environment.
2. **Operating Systems Architectures**: CTS recommends a Hybrid systems architecture for this application. The hybrid structure allows GR to procure a cost-effective off-the-shelf shelf solution that is highly supported, easy to administer, customizable, scalable, and expandable which allow GR to expand its suite of web-based games.
3. **Storage Management**: CTS recommends Windows Storage Management. Not only is it designed for Windows-based systems running Windows OS, but it also includes and API that supports C, C++, JAVA, Python, and other commonly used programming languages which will allow GR to chose new and future software applications based on what is best for the application knowing that Storage Management will support that choice.
4. **Memory Management**: The Windows Server OS will use the compression technique to manage memory when demand on the system is high. Compression minimizes unused RAM data which frees up unused storage and allows more processes to run simultaneously and increases system performance.
5. **Distributed Systems and Networks**: To enable seamless communication between all systems, mitigate the frequency of network outages, CTS recommends using a multi-tiered system. Multi-tiered networks provide redundancy which mitigates network outages, are very scalable to support future GR growth, and are faster since you can have servers closer to users which minimizes latency.
6. **Security**: The Windows suite of products provides a high-level of security. Windows security professionals are constantly monitoring networks for intrusions, bugs, etc. and providing OS security updates to eliminate these threats. It is also quick to respond to network threats without impacting network communication by moving service to other areas that may not be experiencing the same threat.