

Draw It or Lose It

# Software Design Template

Version 1.2

## Table of Contents

[Software Design Template 1](#_Toc75036067)

[Table of Contents 2](#_Toc75036068)

[Document Revision History 2](#_Toc75036069)

[Executive Summary 3](#_Toc75036070)

[Requirements 3](#_Toc75036071)

[Design Constraints 3](#_Toc75036072)

[Rationale 3](#_Toc75036073)

[Evaluation 4](#_Toc75036074)

[Recommendations 5](#_Toc75036075)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/22/2021 | Tony Chiesa | Initial creation of software design document. |
| 1.1 | 06/05/2021 | Tony Chiesa | Added Evaluation |
| 1.2 | 6/19/2021 | Tony Chiesa | Recommendations |

## Executive Summary

The Gaming Room wants to develop a web-based game based on their current Android app, Draw It or Lose It. The client is looking to establish the environment and streamline the development process.

## Requirements

The Gaming Room wants the web-based game that serves multiple platforms. They want a game to have the ability to have one or more teams involved. Each team will have multiple players assigned to it. Game and team names must be unique to allow users 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## Design Constraints

Draw It or Lose It is similar to the 1980’s gameshow Win, Lose or Draw, with the clients unique personal twist. The app will render images from a large library of stock drawings as “clues” for the game. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## Rationale

Based on the clients’ requirements and design constraints the main technical and development constraints will be setting up the environment to meet the needs of the client, making sure that the game functionality meets all standards, and that the game can only exist once in memory at any given time.

## Evaluation

Server-side Evaluations:

* Mac: Top of the line hardware and high levels of security. Flexible terminal commands to configure the server, access, or make changes.
* Linux: The most cost-effective solution because of the open-source nature of the operating system.
* Windows: Top of the line hardware. Has the most software and additional technologies available compared to the other operating systems.
* Mobile Devices: It's better if the server is immobile and can be tracked at a single place. Specifications are better in other devices.

Client-side Evaluations:

During the development process to ensure the application is compatible with all web browser platforms and mobile devices it is important to consider additional time spent cross-browser testing and making sure code is efficient.

* Mac: Moderate expertise and time required. Cost similar to Windows.
* Linux: Maximum expertise and time required. Minimum cost.
* Windows: Minimum expertise and time required. Cost similar to Mac.
* Mobile Devices: Provides flexibility to clients or even developers to see updates at any place. Slightly more difficult to implement than other devices.

Development tools:

* Common languages and frameworks: HTML/CSS/JavaScript and supporting libraries (Bootstrap, Materialize, React, Angular) to support the frontend and general-purpose programming languages like Python, Java, C++, PHP, Ruby on Rails.
* Tools: PyCharm, Eclipse, Visual Studio, Github, Notepad++, Databases (e.g. MongoDB, SQL, Cassandra). These languages/tools are available for all OS.

## 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 recommendation we are making is Windows for 3 reasons. It is less expensive than other operating systems, you can develop for multiple platforms using tools like Visual Studio, Windows will run on most hardware platforms.
2. **Operating Systems Architectures**: Multi-server architecture with load balancing capabilities to ensure affective capacity planning as the web application grows to more users.
3. **Storage Management**: Our recommendation is to utilize a RAID 10 configuration that combines disk mirroring and disk striping to protect data. We recommend starting with 8 480GB SSD disks.
4. **Memory Management**: We recommend to use Contiguous memory allocation within Windows OS because it is a simple memory management technique that all types of computer memory can be used for the application.
5. **Distributed Systems and Networks**: We will accomplish the communication between various platforms by developing using Java because it can be utilized on multi-platforms. The network that connects the devices should be on a mirrored network, for redundancy, that functions on a fiber optic backbone. To prepare for outages, a RAID configuration for hard drives and redundancy built into the hardware for outages.
6. **Security**: We recommend using the “Best Practices” recommended by Microsoft for Windows servers such as user account server hardening, network security configuration (firewall), and general security settings.