

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 05/30/21 | Dominic Cintolo | Initial release. Adds summary, requirements, constraints, and rationale. |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions has been contracted by The Gaming Room to develop a web-based game based on their current Android application, “Draw It or Lost It”. This new version of the game will run on multiple platforms. This project will require establishing a development environment to streamline The Gaming Room’s efforts.

## [Design Constraints](#_2et92p0)

* The MERN stack will be used to develop the application to be cross-platform
* Mac OS development environment
* Linux servers for hosting the application
* Apache reverse proxy for performance

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

The UML class diagram below shows the relationship between the different classes. The “Entity” class is a base class that holds attributes common among the three main classes. The “Game”, “Team”, and “Player” make use of inheritance to allow for a reusable and modular approach to organizing certain pieces of information that all classes will use. A “GameService” will hold zero to many “Games”, which in turn can have zero to many “Teams” that are comprised of zero to many “Players”. Both the “GameService” and “Entity” constructors are private. This allows us to follow the singleton pattern for enforcing one of the client’s requirements – only one instance of a game can exist in memory at any given time. The “ProgramDriver” makes use of the “SingletonTester” to run the application.



## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac servers are costly in terms of both hardware and software. These are less common in data centers, and thus will have less support overall. | Linux based servers will support a wide range of hardware, use no-cost, high quality software, and support many different configurations. | Windows based servers will support most hardware on the market. Licensing will be expensive and additional security concerns apply. | Mobile devices are a poor choice for server side hardware. These devices are low powered in comparison, offer less modularity, and less repair-ability. |
| **Client Side** | Macs are more common in enterprise environments today, with many IT departments deploying and supporting them. Many developers have familiarity with this operating system and its ecosystem of tools. | The proposed tooling is supported on Linux. These systems will be less supported in an enterprise environment. Inconsistencies may arise if the team does not standardize on a distribution. This will require developers be familiar with Linux and common tools/systems. | Most tooling will support Windows out of the box. Windows is ubiquitous in enterprise environments, commonly deployed by IT teams. Most developers have familiarity with this platform. | Mobile devices are not ideal for full-fledged enterprise development. While they are useful for remote work, they lack support for many tools, lack power to run fully integrated development environments, and require additional tooling to be productive. |
| **Developm**  **ent Tools** | The MongoDB, Express.js, React.js, and Node.js (MERN) stack will be used to developm the application. IntelliJ will be used as an IDE due to past Android usage. | The MongoDB, Express.js, React.js, and Node.js (MERN) stack will be used to developm the application. IntelliJ will be used as an IDE due to past Android usage. | The MongoDB, Express.js, React.js, and Node.js (MERN) stack will be used to developm the application. IntelliJ will be used as an IDE due to past Android usage. | The MongoDB, Express.js, React.js, and Node.js (MERN) stack will be used to developm the application. IntelliJ will be used as an IDE due to past Android usage. |

**Recommendations**

1. **Operating Platform**: Debian Linux will be used server side to host the application. The target client platforms are Windows, Mac OS, Linux, iOS, and Android.
2. **Operating Systems Architectures**: Debian has been chosen for the server operating system for several reasons – out of the box support for all necessary tooling to host a progressive web app developed using the MERN stack, support for the most popular web and proxy servers, high level of stability, a notable positive record with security and stability, large array of hardware and cloud vendor support, and high quality documentation. The system will also be highly configurable to address the needs of different work loads. The client platforms were selected for their wide-spread use and support for running progressive web apps.
3. **Storage Management**: MongoDB will be used for storage. Mongo uses binary JSON documents with schema, allowing for tight integration with the javascript stack. Mongo is also highly flexible due to not having a predefined schema, is horizontally scalable, very well documented, and supports simple query syntax. It also comes with performance benefits given that much of its data is stored in RAM.
4. **Memory Management**: The Linux kernel directly exposes the administrator to numerous configurable parameters for tuning memory, including its “swappiness”, page sizes, the out of memory killer, over-committing, and much more. This will allow for maximizing the usage of available resources during times of high usage. The Linux kernel, as most major operating systems do, supports virtual memory to provide an abstractions of the fine details of physical memory. This comes with increased security by protecting and controlling the sharing of memory between processes.
5. **Distributed Systems and Networks**: MongoDB supports Replica Sets. Replica Sets may be distributed across two or more data centers. This allows for many different combinations of replica sets to balance the need for high availability and the hosting costs. Replica Set Elections occur when a primary becomes unavailable, with a new primary automatically selected. Once the former primary comes back online, write operations can revert to the original. Apache httpd web servers can be clustered for high availability. Using heartbeat, we can ensure that there are always servers available to serve the web application.
6. **Security**: Being a progressive web app, data will be stored server-side. Data can be further protected by making use of SELinux to add contexts on top of the already robust permissions system supported by the Linux kernel. The permissions may be further extended with attributes to further control administrators working on the back-end. Many tools exist for securing, logging, and auditing access on this platform.