

Draw It or Lost It

# **CS 230 Project Software Design Template**

Version 1.2

## 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)4

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/19/20 | John Ledbetter | Executive Summary, Design Constraints, Domain Model Update. |
| 1.1 | 10/04/20 | John Ledbetter | Updates to Evaluation |
| 1.2 | 10/17/20 | John Ledbetter | Recommendations Added |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to expand their game Draw It or Lost It from an Android base only application to multiple platforms. An initial goal of a Web based app is being requested.

## [Design Constraints](#_2et92p0)

* Visual
  + Font Types
  + Color Scheme
* Server
  + Cloud Based
* Web-Based
  + Incorporate HTML
  + Container for Game
* Scalable to more platforms.
  + Server should be capable of communicating to clients on various platforms.

## [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 GameService will implement multiple entities of derived classes. Each class will have a name and id with functions to call those. The derived class have a hierarchy of Game->Team->Player where a game needs to be implemented first before team and then team before player.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The macOS server requires a Mac device to run. There is currently no new server grade hardware being made and it is meant only to support Apple OS. All other hardware has to be a mac for communication between devices. | The Linux kernel has a variety of Server OS options from paid to free. Paid version providing a more stable environment and free versions providing new open source tools. Can be installed on almost any device. Not intuitive to use and requires a separate GUI installed if not using a remote terminal to access. | Windows Server is a paid for license. Comes with GUI and is specifically designed for server management. Has the ability to run Linux and Windows containers side by. Power tools built in for use with Windows Azure cloud environment. | Though it is possible to run a server environment on a mobile device using a Linux kernel OS. It is not advised to do so for any major projects. A mobile device with a server on it would be portable, but will not have a way to maintain a constant connection to the internet and will not have the ability to handle large amounts of traffic. |
| **Client Side** | Macs use the Safari web browser. Requirements for a web page include conforming to W3C standards and the use of JS, HTML, and CSS with no third-party plugins. Overall, easily met standards with the possibility of additional training to convert from Java to JavaScript. | Linux is more Java driven, but not limited to it. Initial setup may increase time, but future development will have reduced time. Has access to various software for controlled access and communication between devices. | Windows GUI is understood and usable for a wide range of users and is a general basis of learning to perform task on. Windows supports a number of Web Browsers and comes with a large level of documentation to support even the most difficult of task. | Mobile Devices have different available web browser. A number of them are dependent on the brand of the device. iPhone for example have Safari installed, but can additionally have Google Chrome installed. |
| **Development Tools** | Mac uses the XCode IDE for development. XCode does support the use of Java and other language for application development. | Linux has a variety of package for development tools. Including, but not limited to Java IDE, C++ Dev Tools, NodeJS for building and running JavaScript, and python. | Windows, like Linux, has a variety of downloadable IDEs. Visual Studio and Eclipse for example. | Mobile Device applications are more limited to Androids IDE and Apples XCode for development. |

## 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**: A Linux Kernel based server is recommended for the Gaming Room for their Web-Server based game.
2. **Operating Systems Architectures**: The Linux kernel uses scheduling for controlling different task in the OS and their use of the CPU of the server. This is important for the Web-server as the developers have a variety of options for reducing CPU usage and prioritizing various task related to the functions of the Application on the server.
3. **Storage Management**: A dedicated database server is recommended for the storage of all client-user information. Relevant application data should be stored in the file system of the Web-Server. This includes the application and images used in the game. A motion could be made to have original copies of the images saved to the DB server and a sync service made to have copies loaded/removed to the Web-servers. In-effect creating a single source location of the images to be maintained.
4. **Memory Management**: Linux takes advantage of NUMA memory management. In Non-Uniform Memory Access (NUMA), system memory is divided into zones (called nodes), which are allocated to particular CPUs or sockets. Access to memory that is local to a CPU is faster than memory connected to remote CPUs on that system.1  Other memory management techniques should be handled by the application. Application request instances need to be small, efficient and removed from memory once completed at the application level. The idea of only updating elements on the client side web-page is made here versus sending all elements to the client. Once these requests from the client are made (like logging in), the information sent should be removed from memory and not be stored.
5. **Distributed Systems and Networks**: Clients will access the application through a network gateway that will handle the load balancing of all client-server connections. A gateway will allow for one or more Web-servers to be stood up and will manage the number of connections to each Web-server. This setup allows for minimalized downtime during upgrades and maintenance.
6. **Security**: Security on a Linux servers can be controlled through a variety of options. Minimalized OS installation, which is an installation type that only installs what is needed to run the serve. Along with user options to install addition applications. There are a variety of firewall tools, from directly accessing the built in IP Tables, to installing a Network Manger tool. User access control to the servers can be maintain with Active Directory, and Identity Management. User access through the client can be controlled with a RESTful API. A RESTful API can be used as a foundation to access the Web-Server contents through user authentication.

References:

1. Chapter 9. NUMA Red Hat Enterprise Linux 6. (2017, October 3). Red Hat Customer Portal. https://access.redhat.com/documentation/en-us/red\_hat\_enterprise\_linux/6/html/virtualization\_tuning\_and\_optimization\_guide/chap-virtualization\_tuning\_optimization\_guide-numa