

Draw It or Lose It

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 04/17/22 | Alex Jester | Added Recommendations |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

**Executive Summary**

TheGame Room wants to develop the new Android-only app, Draw It or Lose It. The game is based around the concept of users guessing what is being drawn. There should be multiple games, teams, and players per team. The app will need to communicate with a remote service for storing data, and managing time keeping. Realistically there should be a client version and server version for game. The client must ensure they are able to store remote data for the app. Implications for the customer include: having infrastructure set to support database operations, database maintainers, as well as server-side code maintainers.

The app, only being on android, will be designed and maintained in a Java development environment while using as many platform independent modules as possible. Not only will we design the initial implementation of the code, but we can also provide systems maintainers for both the database and client / server-side code if the customer.

**Design Constraints**

Language - The code base will be written purely in Java, but may include other database components for storing data.

Class Design - Programmers must utilize standardized best practices when designing and implementing classes. We will enforce the separation of private and public members and use public setters and getters when dealing with mutable private data.

Singleton Pattern - The class that manages games must utilize the singleton design pattern to ensure that only one game is loaded into memory at any given execution.

Iterator Pattern -Internal search functionality will adhere to a standardized iterator pattern across all classes. This is to ensure code uniformity and maintainability. The iterator pattern established in the 'GameService' class is the pattern standard to be used throughout the code base.

Client-side code - The client-side code must not perform any authentication whatsoever. The client must be able to communicate with the server program to authenticate, validate the game timer, and store data.

Server-side code - The server code must authenticate users to the service, communicate with the client, manage timekeeping, and maintain database connections.

**System Architecture View**

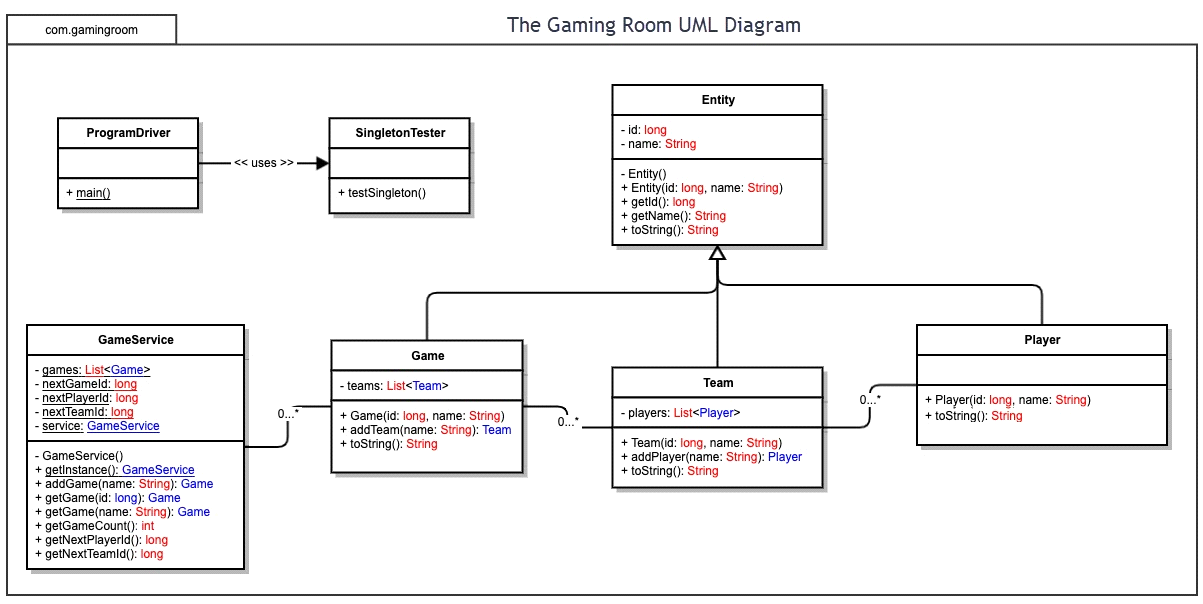
Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

**Domain Model**

Referencing the below UML diagram, I will start by mentioning that the project will in fact be using many different Object-Oriented Programing (OOP) design concepts. Entity is the overarching class that acts as a parent for Game, Team, and Player. The reason for this, is because these three classes all share common members (id, and name). This is a 'Is-A' relationship where Game, Team, and Player is an Entity.

Player, Team, and Game are even further related. A Game can have zero to many Teams and Teams can have zero to many Players. GameService also has zero to many Games. Each class is separated by a horizontal line midway through their respective UML box. The upper portion represents private data members, while the lower portion represents publicly accessible members. According to the design constraints, we will only access private members outside of the class through public members.

Finally, there exists a driver, ProgramDriver and a singleton pattern tester, SingletonTester. ProgramDriver is where the functional flow of the program will exist. Testing classes are intended to ensure that the functionality we desire is successfully obtained.



**Evaluation**

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| **Server Side** | - OS X Server  - $499 - 10 clients  - $999 - unlimited  - Apache WS  - Jboss  - Tomcat  - MySQL  - Scripting  - PKI  - OpenSSL  - User-friendly  - official maintenance and troubleshooting program (more $$$)  Top Pros: User-friendly. Unix-like containing most common Linux features.  Top Cons:  Expensive licensing for most of the same features found in free options. | - Ubuntu Server  - FREE  - Apache WS  - Jboss  - Tomcat  - MySQL  - Linux Kernel  - Scripting  - PKI  - OpenSSL  - Less user-friendly  - extensive community support  Top Pros: Completely Free. Highly customizable environment. Pre-packaged server management software. Extensive community support.  Top Cons:  Less user-friendly, no official support. | - Windows Server 2022  - Standard $1069  - Hybrid Integration  - Software Defined Storage  - Software Defined Networking  - guaranteed official long-term support  Top Pros:  Easier Hybrid integration that enables development on multiple platforms. Guaranteed official long-term support.  Top Cons:  Expensive licensing. | - Why would you want to host a server on a mobile device?  - TinyWeb Server (Android)  - FREE  - TinySSL  - CGI  - poor administrative support  - limited web server capabilities.  Top Pros:  Completely Free. Lightweight web server that is easy to set up.  Top Cons:  There are too many to list, but I’ll name a few. Limited client connections and no official tech support. |
| **Client Side** | - Niche technical expertise to develop MAC apps.  - apps must complete bloated process and pass review to be on store.  - technical support for Mac OS users.  - Potentially large client base. | - No need to publish to an app store.  - Many programming languages supported.  - Easier network programming.  - Potentially small client base. | - Must meet or exceed windows certification standard  - Potentially large client base. | - Pass review to be added to platform’s app store.  - Niche technical expertise for developing mobile apps.  - Should Support multiple mobile platforms (iOS, Android)  - Large client base.  - Hybrid development available. |
| **Development Tools** | IDE: XCode  Languages: Swift, Objective-C, Java. | IDE: Visual Studio, VSCode, CLion, Eclipse  Languages: Swift, C, C++, Java, Perl, Python, Ruby, etc. | IDE: Visual Studio, VSCode, IntelliJ Suite, Eclipse, NetBeans  Languages: Swift, C, C++, Java, Perl, Python, Ruby, .NET integration | IDEs: Android Studio, XCode  Languages: Java, C++, Kotlin, Swift. |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: The operating platform that I recommend to host Draw It or Lose It is Ubuntu Server which is a distribution of the Linux operating system. Linux operating systems boast a 53% market share in the global server operating system market. That staggering number is a testament to the portability, reliability, and general consensus of Linux servers being a good fit.
* **Operating Systems Architectures**: Most Linux operating systems are open source and completely free to use. I am recommending that we host services utilizing Ubuntu Server. Ubuntu is a Long-term support operating system with a dedicated community of motivated engineers. It comes packed full of all the tools necessary to successfully run a server that is capable of supporting cross-platform clients with completely customizable interfaces and security features. What you can do with Linux is truly limitless. The implications of going Linux include no official technical support agencies and you would have to hire certified Linux server administrators.
* **Storage Management**: To go a long with my recommendation of a free Linux Server distribution, I am recommending that you use a free database software to store information as well. There are tons free SQL programs to pick from, but the one that stands out the most is MySQL. MySQL integrates well with Linux and need I remind you is absolutely free. The implications for utilizing free software remain the same. While MySQL is long-term support, there is no official technical support aside from their website’s help manuals. It would be preferred to hire a database manager with knowledge in SQL. The Linux server administrators may know about SQL, but it is not within their wheelhouse to properly integrate and maintain database services.
* **Memory Management**: The code base for Draw It or Lose It will be primarily utilizing Java so all memory management will occur within the Java Virtual Machine (JVM) which acts as an ambassador to the operating system’s memory. Due to the recommended operating platform being Linux, this opens the gateway for internal developers to write modules in other coding languages if need be. When writing code in a more primitive language like C for instance it is paramount that the developers know what they are doing when directly accessing memory. Programs like C give you a lot of power in the sense that you are manipulating memory at the address level. With that being said, if you intend on writing primitive code that directly accesses the operating system’s resources, please do make sure your coders are well qualified.
* **Distributed Systems and Networks**: Having Linux as the server means that it will be much easier to handle cross-platform distributed systems and networks. The server is only concerned with data coming in and out, hosting the service, authenticating users, and managing storage. In the event of an outage, the company should have a plan set forth that includes backup servers and a cold/hot site to work from so that the service is available as much as possible. Ideally the backup server should synchronize with the main server and be physically located in a different geographical location preferably near or inside of the cold/hot sites.
* **Security**: Using Linux as your operating platform can be a double-edged sword. The security can be great, disastrous, or something in between. This is where coordination between all teams should exists. There should be a unified effort to be as secure as possible. You are hosting a server that stores information about users, not just any users, but your clients. The people who are using your service. You owe it to them to not only provide a fantastic gaming experience but also protect their data at all costs. Hire certified developers, system architects, system administrators, testers, and database administrators. Enroll your non-technical employees into security awareness training such as social engineering defense. Require all technical employees to at least be certified in industry areas such as COMPTIA Security+. Taking these actions place your company in a better security posture.