Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
2.0	12/12/2021	Elizabeth Danzberger	Completed recommendations - Finalized document
1.5	11/28/2021	Elizabeth Danzberger	Completed Evaluation section
1.0	11/11/2021	Elizabeth Danzberger	Initial draft - Completed the Execute Summary, Design Constraints, and Domain Model sections.

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

The Gaming Room wants to develop a web-based game that is accessible from multiple different operating platforms, and is based on their current game, Draw It or Lose It. The current version is available only on the Android mobile operating system. The current staff at The Gaming Room are unsure of how to go about setting up an environment for the web-based solution.

The solution will be to develop a server application that will run on a dedicated machine (or multiple dedicated machines running simultaneously) that will host game sessions and provide multiplayer functionality. We will also need to develop a web application that runs in a user's web browser, allowing us to use one code base that runs on several platforms.

Design Constraints

Because the game is web-based, it will likely run in a web browser, so the code base must be written in an appropriate language, such as JavaScript (or compiled to JavaScript from another language like C or C++). JavaScript is perfectly capable of powering web-based games, but is really our only option when it comes to choosing a programming language, hence I've labeled it a technical constraint.

Once the environment is set up and the game is running, those at The Gaming Room may need to be educated on maintenance and such. This could be a business constraint, because time and even money would be needed to teach the employees.

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

The following UML Diagram depicts the relationship between all the classes used in the application.

The Entity class is never used by itself. Instead, it holds information that multiple other classes use. These child classes inherit the properties and methods from the Entity class, thus exhibiting inheritance as an OOP principle. The classes which inherit from Entity are the Game, Team, and Player classes.

Almost every class depicted in the UML class diagram exhibits encapsulation as an OOP principle. All of the properties are private, with public accessor and mutator methods.

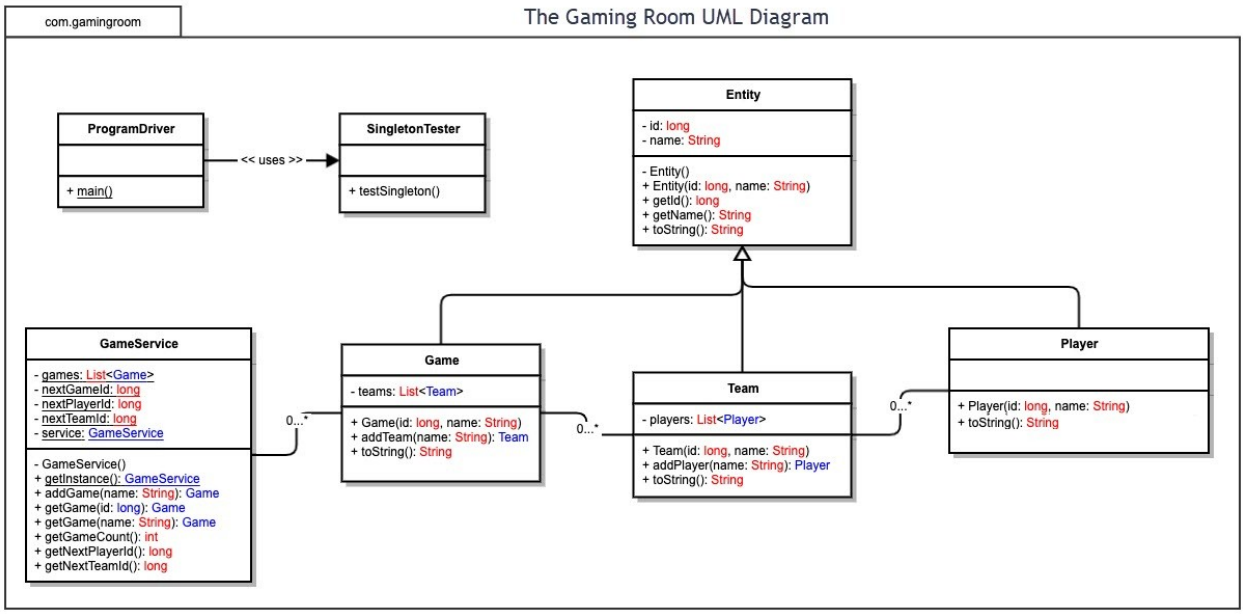
The ProgramDriver class contains the main method and is the starting point for the application. The ProgramDriver class also creates an instance of the SingletonTester class for use in the main method.

The SingletonTester class provides only one public method, used for testing the singleton design concept.

The GameService class keeps track of the currently running game sessions. Game sessions cannot exist without the GameService class, but the GameService class can exist without any running game sessions. Thus, zero or more instances of the Game class can exist in relation to the GameService class.

The Game, Team, and Player classes are related to each other, as well. An instance of the Player class can exist in the context of a team; in other words, an instance of the Team class can exist even if there are no players on the team, but a player needs a team. Therefore, zero or more instances of the Player class can exist in relation to the Team class. Likewise with the Team class in the context of the Game class. Zero or more instances of the Team class can be instantiated in relation to the Game class.

The Game, Team and Player classes also exhbit polymorphism as an OOP principle. They all override the toString() method which they each inherit from the Entity class.



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.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

Development Requirements	**Mac**	**Linux**	**Windows**	**Mobile Devices**
Server Side	I do not think MacOS would be a suitable operating system for a server. There is a MacOS server implementation, but I do not think it serves the same purposes as a typical server hosting a web application. Machines that run MacOS are also very expensive, so it would not make much sense financially.	Linux would be a great option for running a server, as there are distributions specifically designed to function as servers, such as Ubuntu Server. Because most distributions of Linux are completely free, the only thing to consider financially would be server hardware, since the server must be able to handle thousands of simultaneous connections.	Microsoft offers Windows Server, which would be suitable for hosting a web server to host the REST API for the game. While services like Azure are free in the beginning, they would not be suitable for longterm use as they can get pretty expensive.	It would be the opposite of smart to host a website from a mobile device in a production environment. They are not suitable for this.
Client Side	Because the game will be run in the browser, a native macOS application will not need to be developed. This saves money, because most of the time you	Linux is capable of running most of the most popular web browsers, such as Chrome, Firefox, and even Microsoft Edge. Because most Linux	Windows is capable of running almost any web browser, but primarly Microsoft Edge and even Internet Explorer. Although	Devices that run iOS typically use only the Safari web browser, although there are other options such as Chrome and Edge. However, the majority of users use the

Development Requirements	**Mac**	**Linux**	**Windows**	**Mobile Devices**
	need to own a macOS machine yourself in order to develop a native macOS application. The application will need to support the Safari browser, as this is the default and primary web browser on most macOS devices. This means taking into consideration any Safari specific capabilities or requirements.	distributions are free, you can obtain it for free and test all browsers on Linux to ensure the game works properly, so cost in not necessarily an issue here. There are not really any Linux specifics to take into consideration, as you are mostly limited by the web browser itself, and not the operating system, as webpages run in a sandbox.	Internet Explorer is very outdated, it is still recommended to offer support for this browser. Therefore, the game will need to be developed with Internet Explorer in mind and tested on the browser itself. This requires a Windows machine, so cost may be somewhat of an issue here, as a license for the operating system can be well over $100.	default web browser, Safari. Therefore, in order to maintain proper support for iOS devices, the Safari web browser needs to be taken into consideration. Android devices typically run Google Chrome, but a lot of Android devices run vendor-specific web browsers, such as the Samsung web browser on Samsung devices. The broad variety of web browsers will need to be taken into consideration, although I do not think these web browsers vary too much that a separate code base will need to be written.
Development Tools	Because the game will run in a web browser, it will be built using HTML, CSS, and JavaScript. These are all cross platform	Because the game will run in a web browser, it will be built using HTML, CSS, and JavaScript. These are all cross platform	Because the game will run in a web browser, it will be built using HTML, CSS, and JavaScript. These are all cross platform	Because the game will run in a web browser, it will be built using HTML, CSS, and JavaScript. These are all cross platform

Development Requirements	**Mac**	**Linux**	**Windows**	**Mobile Devices**
	as they are interpreted and run by the web browser itself, not the operating system. Because macOS applications are typically written in Objective-C or Swift, and taking into consideration that the game will be run in a web browser (likely Safari), the use of these macOS-proprietary programming languages will not be needed. This allows developers with proficiency in the common web languages to work on the game. There are several free and lightweight IDEs that are perfect for web development that run on every major operating system, such	as they are interpreted and run by the web browser itself, not the operating system. Applications on Linux can be written in a variety of different programming languages, mostly C++, but because the game will run in a web browser, the use of a programming language other than JavaScript will likely not be required (client-side, at least). So, a separate team of developers with a particular skillset will not be required. There are so many IDEs that run on Linux, because Linux is a popular system among developers, so it will not be a problem	as they are interpreted and run by the web browser itself, not the operating system. Most Windows applications are written in either C++, C#, or Visual Basic. However, since the game will run in a web browser, these languages will not be required, so an extra team of developers will likely not be required. Windows is probably the second most used system for application development, so there are many IDEs and development tools available for use, that are completely free. Cost for such things is not a concern.	as they are interpreted and run by the web browser itself, not the operating system. 90% of Android applications are written in Java, but because the application will be run in the user's web browser, the use of Java will be limited to the server. Client code will only need to be written in JavaScript, which is unrelated to Java. Android Studio is a common IDE for developing Android applications, and I'm sure it could be used for web development, but this IDE runs in a desktop environment, not a mobile environment. Mobile devices are capable of running IDEs

Development Requirements	Mac	Linux	Windows	Mobile Devices
	as Atom and Visual Studio Code. Thus, the cost of development tools will likely be relatively low.	finding one that is cost-efficient (there are many free ones).		designed for mobile devices, but developing a website (or any other type of application, for that matter) is very difficult because they are not suitable for such things.

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**:
   Linux would be the best operating platform. There are distributions of Linux specifically designed to be ran as a server, such as Ubuntu Server. It is free and extremely versatile, and is very expandable. It's also lightweight and portable, allowing it to be run on almost any hardware configuration, allowing the hardware to be upgraded, downgraded easily if necessary.

2. **Operating Systems Architectures**:
   The architecture of the Linux operating system is very straightforward. It contains four layers: hardware, kernel, shell, and utilities. The hardware layer includes peripherals, such as keyboards, mice, GPU, etc. The kernel is the core component of the operating system and interacts directly with the hardware layer. The shell is a sort of interface that allows you to interact with the shell without exposing you to all of its complexities. It allows you to run commands and interact with the kernel this way. Utilities are software components, like programs the user has installed, and other things.

3. **Storage Management**:
   Because not much asset data has to be stored, the only thing needed to be considered is other types of data, like user data. Even so, the required amount of storage is very low. Not a lot of storage will be needed, and in my opinion, can be incorporated normally, without the use of a RAID array or anything similar, unless redundency is desired (increasing the amount of storage mediums needed).

4. **Memory Management**: Memory management will be important for this project, because although the amount of storage required to host all of the assets and user data, etc. is very low, there will be thousands of concurrent users accessing this information. That information must be loaded information in order to ensure that it's readily available. Linux is very lightweight and by itself does not need much memory to run, but the server will need to have at least 16 gigabytes of RAM readily available in order to store necessary image files and user data that is being pulled from a database.

5. **Distributed Systems and Networks**: The server will be hosting a REST API that clients (users' web browsers) submit requests to, and the REST API will send a response with the desired data. The REST API will be built with Java, and allows specific endpoints to be controlled and only allow access to those with proper authority. Using a REST API is a really great solution for a game like this, because it is easy to make HTTP requests from the client side and once the information is received by the client it is readily available for use, and it minimizes the need for socket programming, etc.

6. **Security**: Linux is a very secure platform on which to host a server. Although it is open-source at its core, various server distributions provide the maximum amount of security possible, and the number of bugs is significantly reduced due to the open-source nature of the platform. Because the server will be running a REST API, users and other entities require the proper authentication in order to access server-side resources. Without this, the resources are unavailable to the entity requesting them. Full control over these things means that user data, as well as game data stored on the server is only accessed when needed, and by those with proper authority. This helps guard against malicious entities from obtaining information. If one entity were to be compromised, due to the nature of the environment, minimal damage would be done because the user only has the minimum amount of permissions they require to play successfully. Because these protections are implemented server-side, it applies to all platforms the users are playing on.