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	03/29/25	David Brown	Added executive Summary and described the domain model.
2.0	04/08/25	David Brown	Updated summary. Added requirements.
3.0	04/12/25	David Brown	Updated and revised Requirements. Added design constraints and System architectural view. Completed the Evaluation for operating systems and 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

This project is an online game called Draw it or Lose it commissioned by The Gaming Room. This game involves an image that will be slowly generated up to the 30 second mark where it will be fully revealed. If the team currently guessing does not guess the image in time the other teams will have 15 seconds to guess. This involves at least two teams with four rounds of one minute each.

Requirements

The requirements of this game consist of a variety of things.

- You need to have multiplayer team access where one team has access to their turn at a time, unless it is the steal period where the rest of the teams have access.
- You need a library of stock images to be rendered.
- There needs to be a scoring system tied to each of the teams to tally correct guesses.
- There has to be some kind of timer system split up into sections for turns and stealing.
- Some kind of HTML5 user interface should be in place that can be used between all the different platforms.
- This interface needs to include some kind of split display to show the drawings, scores, guesses, and the timer all on the same screen.
- You need to have some kind of server that can interact with all the different systems. A linux based server using Apache or Nginx would be best.
- You would also want to avoid using non licensed images to avoid copyright,

Design Constraints

A well made interface that will work across all platforms requires a lot of testing to consistently fit different screens and browsers. You need to ensure the server you are using is low latency as seconds are important in a game with such short rounds. Network latency or bad synchronizing can cause major problems on the clients end. Because of the amount of image loading you need to find a good balance of quality and speed so as to not crash the system loading too big of an image. If mobile devices are going to have access it needs to be optimized to support lower memory devices. Getting a large library of images could sometimes be expensive to license.

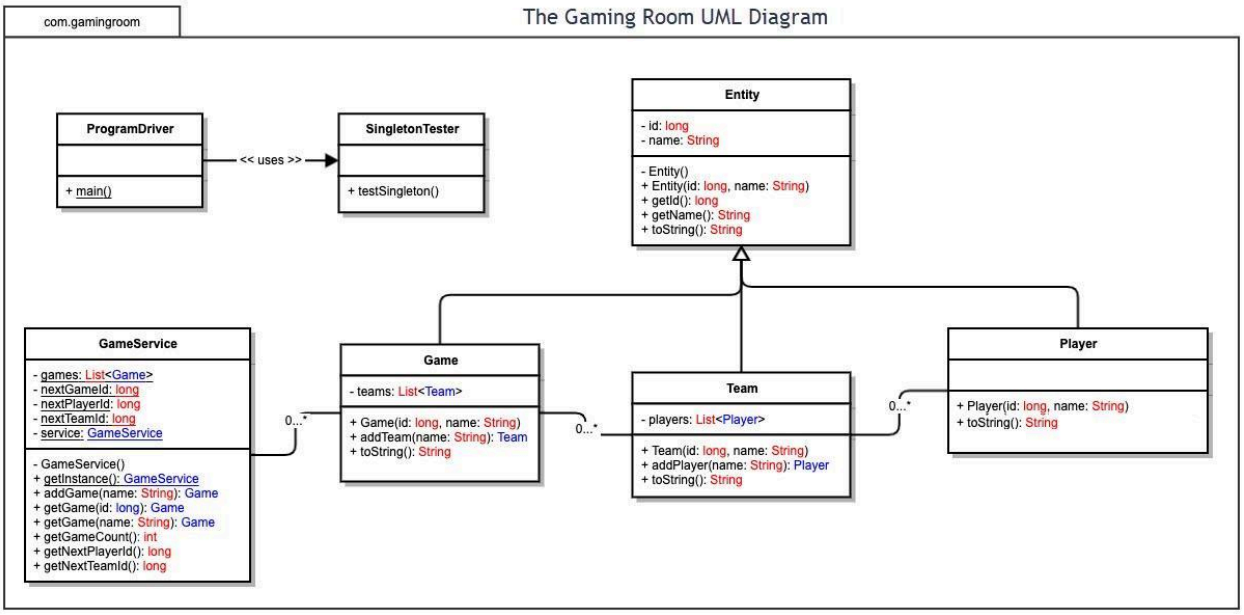
System Architecture View

This system is a layered architecture that consists of a client layer as the frontend, a server layer as the backend, and a data layer to store everything about the game and users. The client layer consists of things like active updates for the players, rendering of the game and images, and user interaction like a menu. The server layer consists of hosting the multiplayer sessions and managing the game logic. The data layer is where all the information is stored. This includes everything about individual sessions as well as user data and game assets like the images and team names.

Domain Model

The entity class below is the basis for all of the game, team, and player classes. Since all three of these items will have names and ID’s using the entity class is useful to get rid of repeating code and having all the name and id calling in one class. The game service class uses all three of these entity based classes to

fill in the necessary information into the game like the names and id's. This allows for individual distinction between these entities when the game is running.



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	Mac servers are Unix-based which do have wide support for web servers and are very friendly, but they aren't very good with large scale and apple is expensive to use.	Linux is open source supporting very scalable things like Apache or Nginx. These are very secure but you need to know how to use them well.	Microsoft also supports Apache and has IIS. These are easier to manage for microsoft as they integrate well with the system but typically are more expensive than linux options.	Mobile devices are very client focused. They are mostly made to tie into other servers and can't host their own web applications themselves.

Client Side	The cost for Mac is fairly high as it requires Mac hardware and special developer accounts. Testing the safari browser takes time and specific expertise in that area.	Linux open source tools are fairly cheap and there are no licensing fees. You will need moderate knowledge of html to test the chrome and firefox browsers, but has minimal extra requirements besides standard testing.	The cost for windows is fairly low as there is a wide variety of tools available with no extra hardware. You need to test on Microsoft Edge as well as chrome and firefox. Very efficient like Linux with no extra expertise needed.	Testing on mobile devices can sometimes be expensive as Apple will charge money and Android uses software like BrowserStack which has a monthly cost. It will take extra time to optimize for all kinds of mobile screen sizes and extra integration for Chrome and Safari mobile versions.
Development Tools	Mac uses HTML5, Java, CSS, Swift, and Node.js. It has development tools like VS code, Xcode, Git, and BrowserStack. Xcode is mac specific which can require IOS specific knowledge to know how to program with it.	Linux uses HTML5, Java, CSS, Python, SQL, and Node.js. It also has more tools like VS code, IntelliJ, Docker, Git, and Apache/Nginx. Linux has more variety when it comes to tools and languages, and many of them are more familiar to developers and generally are free.	Microsoft uses HTML5, Java, CSS, C#, Node.js, Python and SQL. It has similar tools to Linux like VS code, Visual Studio, Git, Docker, and ISS. Many developers are familiar with these tools, but Visual Studio Pro is quite expensive.	Mobile devices use HTML5, Java, CSS, Swift (IOS), and Kotlin (Android).It has limited tools that include Xcode (IOS), Android Studio, VS code, and BrowserStack. You might need specific experience to Test on android and ios specific apps.

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 using Ubuntu is the ideal platform for a game of this type as it has a wide variety of client side support across all the other platforms.

2. **Operating Systems Architectures**: Linux is kernel based where there is a kernel that handles the main functions with extensions to handle extra additions to the system.

3. **Storage Management**: MongoDB and AWS S3 are good options for storing game data and images because they are both cost effective software that integrate very well with Linux to ensure fast loading and rendering of data.

4. **Memory Management**: Linux uses virtual memory paging that maps to Ram or Swap spaces. This allows for efficient memory allocation for more than one session at a time.

5. **Distributed Systems and Networks**: Using WebSocket we can achieve real time updates and use REST APIs to exchange data. This can be hosted on a Linux-based server in AWS. With things like cloud infrastructure and redundancy we can tackle challenges like latency, and synchronization.

6. **Security**: Linux has user permissions that are restricted by a firewall using Server ports to block unauthorized access. Both MongoDB and S3 allow for encryption of stored data making it harder for unauthorized users to get access to game data.