Draw It or Lose It

CS 230 Project Software Design Template

Version 4.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
4.0	04/23/23	Jesenia Roberts	Added journal

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 is looking to integrate their mobile app “Draw it or Lose it” for the web. They are looking to replicate this game as a web application that will be able to maintain multiplayer rounds of their game, assure unique usernames, and allow for only one instance of the game to exist at a time.

Requirements

The necessary requirements will be the ability to host multiplayer, the formation of teams, input checking and comparing to keep all names unique (including a message informing the user to try another name if their original choice is already taken), the implementation of the Singleton Pattern with unique identifiers added to assist in this process.

Design Constraints

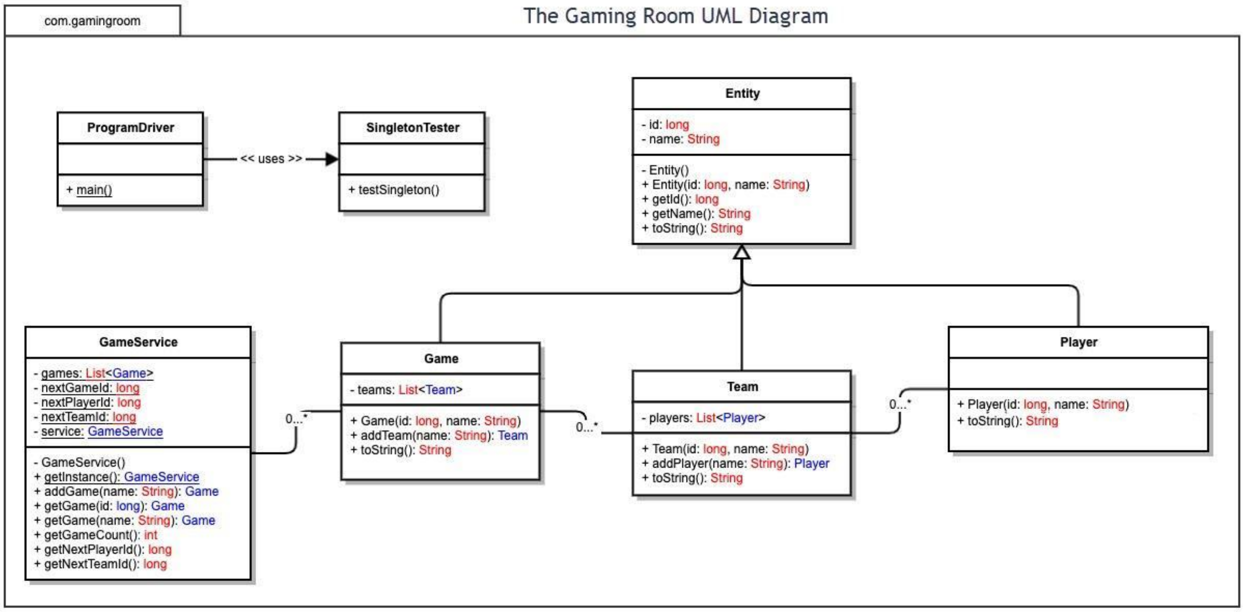
The web application will need flexible design and media queries throughout the code in order to maintain an aesthetically pleasing UI throughout various monitor screen sizes. For the web implementation it must be kept in mind that monitor screens vary in size more significantly than Android mobile screens. Additionally, loading times when pulling photos from the stock library is an additional concern, and as such, this process should be as optimized as possible. Another consideration in terms of performance is CPU usage and how the web app will be able to run while other applications are running, tabs are open, and other additional processes are at work. Data sanitization is another concern when creating a web-based application, and steps should be taken to avoid any user input that could potentially break the game. In summary, in terms of design constraints, there must be a focus on, flexible UI design, optimization of loading times for images (especially for many teams) as well as general optimization of resources, and data sanitization.

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

Team, Game, and Player all extend the Entity class and as a result, will have access to all of its public and protected variables and methods. GameService has a none-to-many relationship with Game, with an unbroken line that implies an association between the two and not a dependency (which will continue to be the case throughout the rest of the diagram). From this point on, Game to Team and Team to Player all have a none-to-many relationship. ProgramDriver holds the main() method in which the program is run, and it uses the SingletonTester to ensure that the code follows the SingletonPattern.
Object-oriented concepts include instance variables, classes, constructors, a driver class, accessors, and inheritance.



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	Hosting on Mac is the most expensive option of the three operating systems, but it is debatably the most developer friendly. The resources used to pay for the hosting on Mac could be negated by money saved by not needing to outsource experts in Linux. Mac servers also tend to remain stable when facing a high volume of activity, which is an asset for maintaining the website.	While Linux is the safest option of the three to host a web application on, it requires more expertise to navigate building for Linux, including working in the Linux command line. However, Linux is also the ideal choice for scripting, which if that is necessary for the game (it may be depending on how the images are pulled from the stock library), could be an asset. On a financial note, Linux is free.	Windows is easier to use and more people tend to have experience with it, which could make development easier on the team. It does cost money to host on Windows which will need to be taken into consideration.	An advantage of mobile development is the speed of the application. With the app needing to pull images and upload them to multiple screens in order for this game to work, this is an effective aspect to have. An issue with mobile development is storage, phones tend to have very restricted storage and so the app would need to take up a small amount of memory in order to not inevitably be deleted due to lack of space.

Client Side	Mac tends to be more stable than Windows and less virus-prone. Navigating the complexities of setting up a multiplayer environment may be easiest to do with a Mac for the programmers, given Apple's focus on simplicity.	While the actual programming will be more difficult to manage, the actual results of how the app would be implemented through Linux are overall positive. It will be cheaper, more secure, and will have an easier time managing traffic on the website.	Windows has issues with crashing when the volume of activity is too high. That being said, the number of users that are anticipated to play the game at a time should be taken into consideration when choosing an operating system to host on. If the application will be Windows-based, then hosting on Windows is ideal as some operating systems would not support it.	With mobile devices, depending on the operating systems and whether or not The Gaming Room would prefer to have a native or hybrid app, the app may need to be programmed in multiple languages. If a native app is implemented, the game will need to be programmed in Swift and in Java to reach Android and IOS. This will require programmers will expertise in both languages, the time to develop both, and the resources to afford the team.
Development Tools	Relevant programming languages would be HTML, CSS, PHP, JavaScript, and Swift. Tools that could be used are Sublime, Visual Studio Code, and Atom (these can be used for all of the following operating systems as well).	Relevant programming languages would be HTML, CSS, PHP, JavaScript, Python, C/C++, and Swift.	Relevant programming languages would be HTML, CSS, PHP, JavaScript, Ruby, Python, Java, and NodeJS.	In order to program in both Swift and Android for a native application, the team would need both an IDE such as Android Studio and one for IOS such as AppCode or Xcode. If The Gaming Room chooses to move forward with a hybrid app, it would be beneficial to write it in a widespread language like Java or JavaScript. [1]

[1]

Recommendations

1. **Operating Platform**: Given that The Gaming Room has the necessary resources at this time to hire a team that is proficient in Linux development, Linux is the ideal platform to host on. This is because of its security, ability to handle web traffic, as well as that it is free, which would save money long term.

2. **Operating Systems Architectures**: The application will use client-server architecture as it allows a variety of devices to be able to access the server with ease. In this architecture format, a number of clients connect to one server using RESTful standards. The client will be designed to send and receive the same type of information to the server regardless of what device the client is running on. A separate client will be required for each operating system, but only one server will be needed.

3. **Storage Management**: A way to best optimize storage is to stockpile the images in a database and pull them from that database when the rounds start. Additionally, the UI that will be used consistently throughout the game should be kept in device storage where it will be most easily accessible.

4. **Memory Management**: The necessary images should be loaded into memory before the start of the round, and disposed of when the round is over to keep as much of the short-term memory open as possible.

5. **Distributed Systems and Networks**:
   Two components in this system will be the client and the server which will split tasks between themselves. The client will be responsible for displaying information, sending requests, and taking in user input. The server is tasked with serving the images to the client and possibly keeping track of game information such as score and time limits. This particular game will need an online connection in order to be played, so outages and connectivity issues would render it momentarily unusable. All platforms would need their own client as mentioned above but will be able to communicate through the server.

6. **Security**: On the client side, user information can be protected by having a login system in which a unique username and valid password are required in order to access the account. As an additional security measure, dual authentication can be utilized for additional validation that it is the user of the account logging in. Data sanitization is also a must as it will prevent users with

Resources:

ESDSMarketing Team at ESDS Software Solutions Limited. (2014, March 18). *ESDS*. India's Best Cloud Hosting Service Provider |. Retrieved March 17, 2023, from https://www.esds.co.in/blog/battle-of-web-hosting-platforms-mac-vs-linux-vs-windows/

VP-Cart. (2020, September 23). *Windows vs linux hosting, the advantages and disadvantages*. VP. Retrieved March 17, 2023, from https://www.vpcart.com/blog/windows-vs-linux-hosting-the-advantages-and-disadvantages/

malicious intent to exploit the program. As Linux is the best fit for security purposes, this further justifies hosting Draw It or Lose It on that platform. When using Linux to host the following should be maintained: a firewall protecting data, an updated server, and the use of a Secure Shell key to access the server. [2]

Journal:

Briefly summarize The Gaming Room client and their software requirements. Who was the client? What type of software did they want you to design?

The Gaming Room wanted help integrating their mobile app Draw It or Lose It for the web. They wanted to know how best to design and host this for the best security and performance.

What did you do particularly well in developing this documentation?

In this documentation, I am particularly proud of the recommendations as at this point the material all began to click for me, and I was confident that what I was writing was correct.

What about the process of working through a design document did you find helpful when developing the code?

I have programmed clients and servers before at my previous university, however learning about how they work made me better understand why we design them the way that we do. That was something that had not been explained to me until this course.

If you could choose one part of your work on these documents to revise, what would you pick? How would you improve it?

If I could revise anything it would be the executive summary as at that point I knew the least and now I could write something more descriptive for it.

How did you interpret the user's needs and implement them into your software design? Why is it so important to consider the user's needs when designing?

I interpreted the user's needs by understanding what is necessary for the application to run smoothly, be well functioning, and be accessible to the users. It is important to consider the user's needs because ultimately, it needs to run in a way that makes them want to use the application. If the user rejects the application, then it is not completing its purpose.

How did you approach designing software? What techniques or strategies would you use in the future to analyze and design a similar software application?

[2] Mullins, P. (n.d.). *7 steps to securing your linux server*. Opensource.com. Retrieved April 14, 2023, from https://opensource.com/article/19/10/linux-server-security

The way I approached designing software was to map out what was needed in a UML diagram, then declare the variables and make shells for the methods (adding TODO comments), then go back in and figure out functionality, testing it along the way. In the future, I will definitely continue to use UML diagrams, although my process otherwise may change if I find a better one.