**Project Research Document**

**RenCloud**

**X00202628 Vlad Pocris**

**Section 1**

As of the first feature, I am going to use the After Effects app, which is known to be a VFX application that usually requires people to wait a lot of time if looking for the best quality and framerate. Even if it might sound easy, there are no such applications on the market; all of them require either an internet connection throughout the process or hardware. By implementing the way I plan, I can reduce not only the wait times but also the use of any resources such as network, computing, or graphics power.

The way it is going to be built is simple: the use of the APIs in order to build necessary components for each user, such as NFS, which will be directly connected to the virtual machine that will render for the user, offering a very easy and simple way of getting your project processed. All you have to do is just drag and drop your folder with project files, and after that, you can choose from the settings you have available and start the rendering process. All of this will be done in the cloud, and you don’t even need to have the app opened; you will be able to monitor progress when you need by just opening the app or even not check at all because the output file is going to be generated in the same directory (Attached NFS) in an output folder.

The project not only solves a specific problem and helps drastically content creators, but also addresses security and privacy since every instance will be created for each user separately. Another advantage is the room for developing: there are many resource-intensive applications that require lots of hardware power and time to render. It can be customized as much as you like, from setting the rendering settings to setting the virtual machine settings that you need. The login process is going to be one of the most advanced with many options such as Google sign-in, Facebook, and two-factor authentication.

There is also perfect room for subscriptions you could implement many advantages. While the project seems to be interesting, this is not the only advantage, I work with new technologies that I have never worked with before, which will give me an important skill set. The user interface is going to be a modern-type interface.

I am confident that this approach will be very good and the app could become very valuable since I am myself an editor which faces the same problems and understands the pain of the people that doesn’t have last generation laptops or PC’s.

I plan to create the first prototype as desktop app for Windows, which will be made compatible with phones/Mac/Iphone in case of success. I have a very clear view of how, what technologies to use, what are the requirements and what security issues should be addressed.

**Section 2**

<https://dropandrender.com/>

<https://render.st/>

<https://www.plainlyvideos.com/>

<https://turborender.com/>

The websites above are each render farms, each of them have their own strengths but there is definitely 1 weakness that I was able to spot in every single solution, which is user interface not being friendly at all, the only one which tends to be friendly is [turborender](https://turborender.com/) and it also has many similarities to what I am looking for to build.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Features** | **Drop&Render** | **RenderST** | **Plainly** | **TurboRender** | **Rencloud** |
| Name | 8 | 6 | ? | 10 | 10 |
| UserInterface | 3 | 4 | 2 | 8 | 10 |
| Technology | 8 | 8 | 8 | 8 | 10 |
| Instalation | 4 | 5 | 4 | 10 | 10 |
| Pricing | 10 | 5 | 6 | 7 | 10 |
| Diversity | 7 | 7 | 8 | 8 | 4 |
| ProblemSolve | 8 | 8 | 8 | 10 | 10 |

These table showcases the strengths and weaknesses of each of the implementation, every single technology got very unfriendly user interfaces, me as a video editor was not able to understand from first look at it how could I actually use it to render which they intend to do, which in my opinion is the most important thing it the implementations are trying to solve, the diversity also is a huge problem, most of them are used just for a couple of software and that’s pretty much it when you could add some diversity and more apps since it follows the same concept. From the cloud perspective the only one implementation that claims to do that is Drop&Render which is very nice since they also implemented a pay-as-you-go pricing, yet still not really understandable because it is labelled as different subscriptions, and the installation and use of the apps is terrible and complicated, the only one that looks easy to use is TurboRender with slightly good interface because it just doesn’t make sense the limitation of 10GB and use of FTP if is larger when you could implement that differently which showcases that they did not use any Cloud technology. I could go on more but ½ page was said in the section criteria.

**Section 3**

I plan to use either Azure or AWS services with C# and .Net8 framework to build a very user friendly app. I began building the user interface using Windows Forms, and managed to make use of some built-in dll’s for designing. Mainly I look for VM’s, NFS, LogIn services, I will still add many technologies if needed but I believe in simplicity of the solution which then will allow it to be developed further. As for now this is my picks, solution sounds easy, very user friendly with no need for IT extra knowledge to use it.

I really wish I could use AWS but I think I will stick with Azure since they provide with free credits for student and aswell it would be a nice to acquire new skills since I haven’t used Azure services before. And it also give’s me Microsoft Enterprise Visual Code subscription which is awesome!

Use Cases and Logical Architecture

* XID: X00202628
* Name: Vlad Pocris
* Project Title: RenCloud

Section 1: Each Use Case:

|  |  |
| --- | --- |
| Title (goal) | Login |
| Primary Actor | User/Content creator |
| Story | Suppose the user want’s to log in for the first time. He can register by using the app or he can login if already done so. |

|  |  |
| --- | --- |
| Title (goal) | Uploading Files |
| Primary Actor | User/Content creator |
| Story | Suppose the user want’s to upload the project files, He could press a button that would manually mount the NFS volume to it’s computer device |

|  |  |
| --- | --- |
| Title (goal) | Specifying the render settings |
| Primary Actor | User/Content creator |
| Story | The user should pick the render settings before starting the render process. |

|  |  |
| --- | --- |
| Title (goal) | Starting/Finishing Render |
| Primary Actor | User/Content creator |
| Story | After successful upload the user wants to start the render process. He press the start button and wait for a notification that would confirm the status. |

|  |  |
| --- | --- |
| Title (goal) | Payment |
| Primary Actor | User/Content creator |
| Story | How much would the user have to pay using the service? Pay-As-You-Go implementation derived from prices from the cloud. |

Section 2: Logical Architecture

LOGIN

A diagram of a cloud computing system

Description automatically generated

Files Upload

A computer screen shot of a cloud

Description automatically generated

Render Settings

A computer screen shot of a cloud

Description automatically generated

Start/Finishing Render

A computer network diagram with text

Description automatically generated with medium confidence

Payment

A screenshot of a computer

Description automatically generated

**Section 4**

I see no risks what so ever, but if we talk about security concerns then I will continue with that.

Since this is a processing project the implementation could really be bad if not worked out secure, providing every user with their own NFS and VM could be a disadvantage, but I tailor more security and privacy rather than performance at this stage, by doing that there would be 0 risk from the security and data perspective to be leaked/breached with use of malicious software.

What I depend on is Cloud services, everything should be done in the cloud and no information should be extracted from the user because there is no need for that to solve the current problem.

Illegal content might be a risk if uploaded, which should be revised, but I already have a fix for that.