PROJECT RESEARCH DOCUMENT

X00144631 Ryan Deering
Project Supervisor: Gary Clynch

Contents

Section 1: Detailed Discussion

The idea in question is a social media platform clone, on the lines of 'Instagram' or 'Flickr', catered towards artists and photographers, with a focus on empowering their audience. Upon discussions with friends who are artists and photographers in their field, there is a concern that despite social media platforms like 'Instagram' exist to share their work, it does not help monetise their work. The key idea is to introduce a payment engine through the application to help artists receive money for their work if their work is sought to be bought. This could be through receiving prints or buying a high-quality digital image through the application. This could even include the right to buy the rights to the image, to use elsewhere.

The use cases are very standard 'social media' affair. Add post, delete post, edit post, like post, etc. There will be a social media 'feed' to see the latest posts of the people you follow. Posts must be uploaded in the form of images; a post can have many different images. There should be an option to buy the rights of the photo, think of a musician being paid for their right to use a song in a movie. Or to buy a print of the image in different sizes, this can be authenticated through the likes of PayPal or Stripe implementation. When posting, the user that is making the post can determine whether they want to allow the rights to be bought or to allow a person to buy prints of the image.

There will be three classes of users predominantly, registered user, moderator, and administrator. The idea is the registered user has access to certain things that an unregistered user does not, for example: submitting a post, liking a post, deleting a post. A moderator should have the ability to moderate content, like deleting certain posts or comments. An administrator will have extended functionality compared to a moderator, being able to see additional statistics and backend options in an admin panel.

There will be user profiles along with the social media feed, with the person's posts being displayed on their profile, along with any additional captions. Along with a follow function to keep to date with that profile's posts.

Of course, with different types of users existing, there will be an authentication/login system to allow for these different permissions to be enabled. The user account system as well as authentication will be the starting point for the project, along with drafting up UML and use cases.

The audience is focused on artists and photographers, however, it is still a standard social media website, so anyone who wishes to register can do so. But the focus is on empowering artists and their work.

Section 2: Existing Applications in this Domain:

Project Idea:	Instagram:	Flickr:	Shutterstock:
+ Follow accounts + Submit posts + Like posts + Commenting + Monetisation + Geared towards photography and art	+ Follow accounts + Submit posts + Like posts + Commenting - Monetisation - Geared towards photography	+ Follow accounts + Submit posts + Like posts + Commenting - Monetisation - Geared towards photography	- Follow people - Submit posts - Like posts - Commenting + Monetisation + Geared towards photography, art, video

Section 3: Platform, Technologies, and Libraries:

The plan is to use a Microsoft-orientated stack. I want to use Microsoft Azure services to host the application, as well as Azure DevOps to help manage development. I have experience with both during my course, but I am most familiar with Google Cloud. Google Cloud and Microsoft Azure share a lot of the same features, but it may be a great opportunity to get familiar with a different cloud platform.

The plan is to use Blazor WebAssembly, an up-and-coming single page framework from Microsoft that is based on .NET. It allows for web UIs to be developed using C# instead of JavaScript. *(Microsoft, 2020)* This way the backend can be .NET as well, sharing the server-side and client-side logic to be written in .NET. It runs the client-side code in the browser using WebAssembly. If JavaScript is necessary, it can interoperate with JavaScript, enabling the ability to call libraries. So, while JavaScript is not completely avoided, Blazor can be used to *enhance* JavaScript.

Generally, the idea of a full-stack .NET development process is intriguing. This is an interesting technology and if it takes off, it will be beneficial to get early experience with it if C# or Microsoft houses begin to adopt it. But it is a bet, a low risk one albeit. While we could use a 'safer' and more popular technology like React *(Stack Overflow, 2020),* it adds some risk and excitement to use an unproven technology compared to a safer option. We decided to use a Microsoft stack as the MDSN tends to have very comprehensive and reliable documentation. The author used .NET Core in their internship a bit and found it to be very beneficial and useful along with using C#. We anticipate we will be using the Newtonsoft, XUnit libraries a lot during development. Newtonsoft, to handle serialization and deserialization of JSON.

Section 4 Risks:

The main risk mostly revolves around the Blazor framework. It is a new technology, and not quite a mature framework. I have experience in .NET Core but there may be challenges that will present themselves due to the framework being relatively new. So, there is the risk of being a 'guinea pig' for unexpected or little documented errors that may arise. There are more resources for a framework like Angular compared to Blazor. Blazor is constantly being tinkered with to build a better product for its age. There are also fewer resources and documentation compared to something like React, which has a lot more usage and documentation on websites like Stack Overflow. *(Stack Overflow, 2020)* The tooling is immature as it is freshly out of preview. But it can be used in production, and IDEs such as Visual Studio and do have support. It is also very fast to build CI/CD pipelines, which will help for test-driven development. However, with the cons of using Blazor, it is arguable that the pros outweigh the cons considering: the ability to write a modern web app using C#, built-in form validation, routing, and data-fetching, using tools like Visual Studio and VS Code, as well as using the same Blazor component model on Windows *and* mobile (in future).

Time allocation is a potential risk. Wasting too much time on a feature, then having to push features intended for an iteration to another iteration. Planning articulately and estimating the amount of time to allocate for certain features is very important if I am to develop efficiently and to manage work on the project along with studying for my modules. Of course, spending too much time on a feature that does not require more than a certain amount of development time is inevitable, it is important to be careful to be aware of that. This could lead to failure of being able to deliver on time, so it is important to continuously evaluate the risks during development and if a feature can't be finished on time, to inform the supervisor of the project, Gary Clynch. This is especially important despite the project being a solo endeavour, there is only the author to work on these components of the project, so it is important to allocate our time efficiently to work on what the components that need work. It is also hard to ignore the possibility of delays due to the ongoing COVID-19 pandemic.

It is important to also manage the requirements carefully to avoid feature bloat. While changing requirements during development are accepted in an agile framework of development, it is key to not be unrealistic and not put in features that may take a long time to develop and hurt the development of other components of the project. This will avoid rewrites, changes to the schedule or overloaded sprints. It is important to not change the schedule to balance study with other modules.

There is the risk of poor-quality code, not keeping align with a certain standard of quality and not being tested enough. It is important to keep a good amount of code coverage to prevent this from happening.

References:

(Microsoft) *Blazor | Build client web apps with C# | .NET*. (n.d.). From https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

Newest "blazor" Questions - Stack Overflow. (n.d.). From https://stackoverflow.com/questions/tagged/blazor

Newest "reactjs" Questions - Stack Overflow. (n.d.). From https://stackoverflow.com/questions/tagged/reactjs