DOCKSUSHARE PROJECT PROPOSAL

RYAN DOCKSTADER

CONTACT INFORMATION

Ryan Dockstader

- **Email:** rdockstader@byui.edu
- **Phone: (**435) 799-7237

ABSTRACT

DocksUShare is a program that is designed to share discovery online, between prosecutors and defenders office. It gives the lawyers and attorneys the ability to easily share and access any discovery to make it easier to enact the law.

BACKGROUND

I have been working in software surrounding legal proceedings for about 5 years now, along side many wonderful co-workers. A little while ago one of my coworkers approached me, knowing I was going to college to become a software engineering, and asked if I would be interested in working with him on building a new type of software. This software would allow people to share large files, easily, over the internet.

At the time that we started working on this project, there wasn't anything out there that could process large discovery files, but there were laws coming out in many states that providing discovery within a small amount of time was mandatory, and some even going to the extent that it must be provided digitally.

I have been working on this project for a little over a year at this point. It was originally built (with me as it's sole developer) on the MEAN stack (MongoDB, Express.js, Angular 5, Node.js). However, with the recent launch of .net core 3 I went about switching the backend to run on .net core 3.0, and upgraded to 3.1 (the LTS version) last week. With the new upgrade I also switched from running on MongoDB to running MS SQL which has had some significant speed advantages for the type of data I am storing.

The backend change over has allowed for many front end changes that haven't been made yet. Such as breaking up the cases into folders, access to background running services, Hubs, more consistent error handling, and several other things.

DESCRIPTION

The application will be a progressive web application based on the Angular framework, using several angular modules including RxJs, NgRx, and Angular Materials. The front end should use an easy to use GUI to load cases, upload files, share those cases/files, and download individual files, folders, and entire cases. The system will also notify "authorized users" (users outside the agency that have access to certain cases/folders/files) of any changes or additions to the files they have access to. Simply put it's a browser based document management system that is made to allow for easy, time based sharing with eternal users.

TARGET AUDIENCE

The target audience for this application is government-based entities involved in litigation procedures. This means the primary users for will have a wide range but will generally be between 25-65 and have at minimum some college experience. They will be mildly familiar with technology but may not use more then a smart phone in their every day lives. They will be a mix of males and females. They will use a range of devices from smart phones to full desktops to view the cases and files.

SUCCESS CRITERIA

For this application to be successful, a user will need to be able to create a case, add authorized users to the case, and deploy many various file sizes (very small to more then 10 GB single file size) onto a case. The system should then email the authorized users that there is a new case available for them that they can easily access.

SCOPE

The project will consist of a backend REST based API, and a front end PWA/SPA.

The backend will serve CRUD endpoints for Cases, Case Category Types, Log Records, Media Folders, Media Items, and Users. The backend will allow for uploads/downloads of media files, media folders (zipped), and cases (zipped).

The front end will provide a simple GUI to make requests to the available CRUD endpoints, as well as uploading, and downloading files/folders/cases. It will also allow you to view certain file types (namely photos, videos, and pdf documents) directly in the browser for ease of use.

SIGNIFICANCE

This project is significant, because of its size (which is relatively large for a single developer of my experience) and because of its importance. First, I'd like to speak towards its size. For a full team of developers of varying experience this would be a pretty easy project. There would be some people to work on the features of the REST API, and I'm sure some of them would have some reasonable amount of experience in .net Core, or at least C#. Then there would be at least one if not a team of people working more on the front end. However, as a single developer with very small full project experience this application has already been a massive learning experience, and has taken a considerable amount of time.

Second, I would like to speak towards the importance of the project. Currently in states that are mandated to share discovery electronically are commonly sharing it with compact disks, USB flash drives, or giving others access to their internal networks. All of these methods have some extreme security risks and lack convenience. This program has the vision of making the process of discovery painless and secure. With a quick and easy way to share discovery, while being able to track, and prove that the other side of the trial received said discovery criminals will no longer get away by a simple formality, and innocent people will no longer be charged with crimes they did not commit by the same simple formality.

NEW COMPUTER SCIENCE CONCEPTS

This application is rich with new concepts. I've been learning along with angular since about version 5, but I haven't done anything significant with it. At most I've had a small application with a couple of services that reached out to a REST API backend. This application is going to be large enough that it will require state

management. It is also dealing with file uploads, and downloads. I will also be working on keeping the Angular Framework running on the latest supported version which I have never tried to do before.

Another major new component for me is the backend is in C#, specifically asp.net core 3.1, which up until a month or so ago I had never worked extensively in. This framework follows the MVC standard, but does a lot of things for you. That is one of the biggest problems that I've ran into so far with .net core, is that there is a lot of stuff behind the scenes going on, and it always takes me at least some research to determine how to modify that behavior An example of this is with large file uploads/downloads, there was a max size. It took me quite some time to find it, and it ended up being a setting stored in the startup file, that was defaulted but could be overwritten. I also recently had to learn about .net core hubs using SignalR which required a bit of a depth search into web sockets and how those work, and how to hook that up to an Angular based web application.

Large files also come with many other problems and areas of concern. One that came up this last week is keeping a computer awake through a browser while uploading/downloading large files. If it takes longer then the computer sleep time, and the user isn't using the computer the computer will go to sleep and kill the connection. It appears they may be some Web API's that I can use to hook into chrome to inform the machine that it should not go to sleep, but that is a topic that I'm going to have to research out.

This application also *must* be secure. I will need to learn about file encryption, SSL certificates, HTTPS handshakes, and I'm sure many other security aspects that are unknown to me at this point. Due to the sensitive nature of the information the application will contain, I have a sense of obligation to do whatever I can to protect that data from the people that should not have it.

INTERESTINGNESS

At this stage in my development career, I'm pretty interested in C# and Angular independently. I have a brother who has invested a lot of time and energy research and developing in C#, and everything he has told me about it has me very excited to get more and more familiar with it. I also really like the roadmap for .net that was announced recently at .net conf (https://www.dotnetconf.net/). I'd like to eventually look further into the Blazor stuff, but at this time I'm pretty happy with Angular.

Angular has been something that has been extremely interesting to learn about, and has been a framework that I keep coming back to, to dive deeper into the framework, learn about new versions, and figure out how to work with the various Web API's within the framework.

Another aspect of this that is interesting to me is deployment. In school, they don't really talk too much about deployment which I think is a real problem. With this application I *must* figure out the deployment pipeline and figure out how to make it as optimal as possible. Visual Studio has a lot of very powerful tools to make deployment very easy, even to a remote environment. I'm very excited to learn about all of this, and really optimize my deployments to test environments, and eventually production environments.

On top of my fascination with both languages and the deployment pipeline I'm working in, I also really think this product can fix a real problem out there. No matter where anyone sits on the whole criminal justice stuff, I think we can all agree that nobody likes it when a known criminal gets off on a technicality, or when an innocent person is wrongly convicted. I think this application, if done correctly, can help with that.

TASKS AND SCHEDULE

TASKS

PRIORITY 1 (MUST HAVE)

1. Fix 504 Error Messages
2. Keep computer awake when uploading/downloading files
3. The user should know what uploads/downloads/zips they are waiting on, and what their status is
4. Log files should be searchable by case, item, or user
5. Log records should be added when an email is sent
6. Log records should be added when a user downloads (instead of just views) a media item
7. The application should not fill up the server's hard drive when zipping files
8. The application should track the file size of individual files
9. The user should be allowed to create folders and sub folders in cases
10. The application should auto refresh tokens when users are actively using the software/uploading media items
11. Retry Button for failed uploads/downloads
12. Video should stream instead of downloading the entire video before it's watchable. Video player should be able to fast forward and rewind similar to YouTube, Netflix, etc. functionality
13. Add Google analytics to main web site
14. Allow admins to view how much storage is being used
15. Backend files should be deleted when a media item is deleted

PRIORITY 2 (NICE TO HAVE)

1. Each API endpoint should have its own test, to verify security and functionality without human checking
2. Show authorized users what is new on cases since the last time they viewed the case. They should also have a download button to download all these new items at once.
3. Compressed copies of files for streaming (I.E. convert videos and photos to smaller resolution and store the large and small file. One for streaming, one for downloading)
4. Create an installer that will connect to the database, spin up a new database, new app server, etc. This should also be able to connect to an existing instance and upgrade it.
5. The application should allow an admin to resend an invite email
6. Add arrows so the user can click through media files
7. The single upload and bulk upload screens should be consolidated into one. Allowing the user to selected multiple files, and modify their names/types/descriptions/etc.
8. The application should tell the admin when a user has setup their account
9. The application should hold deleted files/cases for 30 days before it deletes backend files
10. Admins should be able to filter and sort users by names, emails and user types, isActive and isRegistered
11. Active checkbox should be checked by default when adding new users

PRIORITY 3 (IF I HAVE EXTRA TIME)

- Contact Us form on Main page Contact Us
- Add FAQ Section to the main page site, to show frequently asked questions
- The application should work with gestures, and look good on phones

SCHEDULE

WEEK 1

- Fix 504 Error Messages
    - Remove AWS Beanstalk from deployment stack
    - Remove EBS from deployment stack
    - Remove Docker from deployment Stack
    - Replace AWS Certificates with LetsEncrypt Certificates
    - Deploy a SingnalR Hub that signals when a zip file is ready to be downloaded
    - Modify MediaFolder endpoints to have a request and download endpoint for zips
    - Create a ZipService as a backend service to monitor the ZipQueue in the Database
    - Modify the Front End to work with the request system

WEEK 2

- Keep computer awake when uploading/downloading files
    - Research the web background task API in an attempt to keep the computer awake when downloading/uploading files
- General User Experience
    - Improve front end error handling. Display error code in all error messages
    - Authorized users should be auto-deleted when deleting a user
    - List Items preventing deletion of User
- The user should know what uploads/downloads/zips they are waiting on, and what their status is
    - Deploy a list of pending zips on the front end
    - Deploy a list of pending downloads, and their status to the front end
    - Setup Progress bars for all zips/uploads/downloads where possible

WEEK 3

- Log Files should be searchable by case, item or user
- Log records should be added when an email is sent
- Log records should be added when a user downloads (instead of just views) a media item

WEEK 4

- The application should not fill up the server's hard drive when zipping files
- The application should track the file size of individual files

WEEK 5

- The user should be allowed to create folders and sub folders in cases
- Retry button for uploads/downloads

WEEK 6

- The application should auto refresh tokens when users are actively using the software/uploading media items
- Add Google analytics to main web site
- Allow admins to view how much storage is being used

WEEK 7

- Video should stream instead of downloading the entire video before it's watchable
- Backend files should be deleted when a media item is deleted

WEEK 8

- Each API endpoint should have it own test, to verify security and functionality

WEEK 9

- Show authorized users what is new on cases since last time they viewed the case
- Download button for all new media items

WEEK 10

- Compressed files for streaming (this needs to be tracked for file size purposes as well)
- The application should allow an admin to resend an invite email
- The application should tell the admin when a user has setup their account
- Add arrows so the user can click through media files
- Allow the next and previous keyboard keys to move through media items

WEEK 11

- Create an installer/upgrader to provision, setup a database, and deploy an application with given parameters

WEEK 12

- The single upload and bulk uploads screens should be consolidated
- The admin should be able to filter and sort users by names, emails user types, isActive, and isRegistered
- Active checkbox should be check by default when adding new users

WEEK 13

- The application should hold deleted files and cases for 30 days before it permanently deletes them

REQUIRED RESOURCES WITH COSTS

This application will be hosted in the AWS government cloud (the east servers, specifically). This will incur the following costs:

Item	*Quantity*	*Billing Period*	*Total Price per Billing Period*	*Total Yearly Cost*
AWS S3 Storage	2 TB	Monthly	$79.89	$958.68
AWS EC2 t3.large Server	1	Yearly	$567.00	$567.00
AWS EC2 t3.micro Server	2	Yearly	$71.00	$142.00
AWS Route 52 Domain	1	Yearly	$12.00	$12.00
AWS Hosted Zone	1	Monthly	$0.50	$6.00
DNS Queries	?	.50/million/month	$1.00	$12.00
AWS EBS	100 GB	Monthly	$10.00	$120.00
			Total:	$1817.68

There is also a requirement to have a good developer machine with decent screens, but I already have this equipment available to me personally. IDE Software, SSL certificates, and a couple other small things have been found and acquired at no cost.