**April**

Today I continued work on the technical specification which is required to be submitted for the project. It details the work of the project from several areas, from research, implementation and design. I worked on the research, summarising my notes on Optical Character Recognition, on how it works and the reasons for the design choices of the system, switching from a general solution to getting the users to select regions on the invoice. I also summarised my reasoning and justification of the use of a microservices architecture and the benefits over a monolithic system.

I brainstormed ideas out for some other sections, and planned out ideas on what to include as I work on the project. I want to talk about the overall architecture, use cases and how the user will interact with the system and how those interactions work.

**November 18th**

Today I have worked on my Functional Specification for the project. In the functional spec, it is to get us thinking about the functional and non-functional requirements of the system. That is, what we want the project to do and the constraints it has to be done in. In it, I talked about the need for a classifier to recognise an invoice and preform layout analysis using machine learning.

I talked about the business context of the project and how it is from our farm back in Louth. With how invoices are currently managed in the farm, this type of project felt natural to me to want to bring a bit more order so that my Father can spend less time searching and storing invoices manually in his paper filing system and just get on with more business critical tasks to the farm.

For general functionality I talk about the importance for the need to upload invoices, plot graph and have business profiles generated form the info given to the system and how this needs to be searchable. I talk about how this will relate to the layout segmentation of an invoice and the need for an OCR system to accomplish this. I describe the types of user who will use, the people who are interested in the financial matters of the business and how they might possibly need it for expenditure planning in a business, as historical data can be useful for future planning. I give examples of usage and wireframes of what I want it to look like.

![Wireframe of viewing expenditure](/content/images/2016/04/wireframe.png)

Happy with this image. 😀

My supervisor Alex gave me good tips for writing this and that I should really pay attention to the user stories that I submit, that they will form the whole basis of figuring out the design and functionality of the system.

I gave an architecture overview which I will use as a basis when I’m implementing the system.

![Architecture of the system](/content/images/2016/04/architecture.png)

Nearly finished it and shall submit it in a few days once Alex gives the thumbs up.

Refactor or burn

The paradox that haunts me: the more you know, the more you know you don’t know. As I go further along with the development of this project, there is a learning curve to certain technologies. There is always more to learn and it’s a welcome challenge. It is surprising how quickly your paradigm of how something works can change so quickly when you fix a fundamental flaw in your coding.

In doing React with Redux, I just wasn’t getting it. I didn’t understand the differences between Smart and Dumb components, when to use an action, (never mind asynchronous actions), how to manage your applications state or anything. I read them, but I did not have the real-deep understanding that made me ‘get it’. As a result, I was fumbling around with React components, mixing the state of an internal react component with Redux state and making a mess. When trying to make a single change, it was a fire-fighting task. I knew I couldn’t go on and dove deep into research to get a better understanding.

Dan Abramov is a fantastic teacher and also the creator of the Redux library. He made some <https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0#.3bogll27b> excellent articles on helping me gain a better understanding. Alex, my supervisor also sent me a phenomenal flowchart (by Mr. Abramov too) when listening to my qualms of what was happening. <https://github.com/gaearon/react-makes-you-sad> It turns out, I was overcomplicating my solution by using Redux-Form. I didn’t understand Redux itself and I was jumping ahead of the game, expecting a library to solve all my problems. I had to start small and go through the pain of implementing my own solutions, it works out better.

So as a result, I went through my code and refactored and turned my react components into proper Redux components, and it totally paid off. I understand so much better the Flux state (pun intended…?) of mind. Things are so much easier to reason about and it ‘works’ much better. But again, there is an awful lot more to learn but we get better with the more practice we get.