Blog: Job Analytics

Daniel O'Sullivan

Blog 1: 02/11/2017

I am still doing tutorials for the software I will be implementing in my project. At the moment I am doing Django tutorials. I think I will need another while to become comfortable in the framework, as I didn't realise how big it was. Nevertheless, I am really enjoying using it. Today I set out a plan for my project. By plan I mean certain steps I need to carry out to make sure that I have something to show for my work when it comes to the deadline. The plan is to first:

- Implement backend services (server and database). Mainly just to get the database accepting objects and variables.
- Second, I need to work on the CV scanner. I will work on this independently from the project and then will implement it when I am satisfied it is complete.
- I will then work on the front end of the application while trying to integrate all parts of the project.

If for some reason, I don't get to complete one of these tasks effectively, I will still have something to fall back on thanks to this plan. Next steps is to start working on my functional specification.

Blog 2: 16/11/2017

I've made a lot of progress with my functional specification. I have most of the text written segments completed and I am currently working on the various diagrams. The reason why I am taking my time in constructing these diagrams is because I am trying to get them right. I am still completing tutorials for some of the technologies I am using, and don't want to make a final decision on how I implement the system architecture until I am fully familiar with the software I will be using. When I am done with the functional specification the next step will be to set up my backend services (server and database).

Blog 3: 20/11/2017

I have made huge progress on my functional specification. I have added a number of graphs that illustrate how my application works and how it is implemented. Doing the functional specification has made me think about my project in ways I have not thought before. Problems that I didn't think were relevant to my project have appeared as I plan out my application. I'm glad I caught these problems early in the design process, so I can rethink how I am going to implement this project to avoid as many problems as possible. Next step is to finally start coding.

Blog 4: 26/11/2017

My functional specification has been submitted and now it is time to start writing some code. I have started experimenting with Django by making test projects and seeing how everything works. I have

been following a tutorial series on youtube to learn about Django. I understand how the views and templates are connected and how the database connects to Django. I have a made a basic template for the homepage of my web application and now will start to attempt to connect my database. This may be my last blog for a while, as I want to focus on studying for my Christmas Exams. However, I will still continue to work on the project in my spare time.

Blog 5: 15/12/2017

I have connected my database to my web application. However, despite saying in my functional specification that I will be using Google Firebase, I have opted to use a PostgreSQL database instead. I have a few reasons for doing this:

- Google Firebase is a very new piece of technology. Because of this there aren't many tutorials about implementing it with Django that are simple to understand (in my opinion).
- I have read countless of posts in Django forums and community pages about how PostgreSQL is the best database to use with Django. It is also very easy to set up and use.
- The main feature of my project isn't my database, it is the CV parser and machine learning algorithm. The sooner I implement a working database the sooner I can start working on the application's main features. Setting up PostgreSQL was easy and quick and I didn't compromise the quality of a good database.

As I said in my plan in my first blog, I want to implement backend services first. I also think it would be good idea if I implement basic user authenication and applying for and posting a job functionality first as well. This way I can have the foundation of a web application made and then I can add to it as I go along with my project. Keeping this is mind, I need to create a basic user interface as well. I have been doing a lot of ReactJs tutorials and I am starting to feel comfortable using it. So next step is to implement react into my application.

Blog 6: 21/01/2018

I have finished my Christmas exams and I can now solely focus on my project. In my last blog, I said I was going to start implementing ReactJs into my project. However, like my original plan for my database I have decided to not use ReactJs and just use the default Jinja templates that Django provides with Bootstrap CSS. Again I have my reasons for this:

- ReactJs is a very powerful tool. However integrating an npm build pipeline with Django is too complicated for what I am trying to achieve.
- I can still create a good user interface using Django's Jinja logic templates. So I'm not losing anything by not using React.

I have made a homepage using Jinja templates and now I am going to start creating user models for my database.

Blog 7: 26/01/2018

I have written a user model for my database. Django has a default one you can use for user authentication. I have used it but have tweaked it to my liking. At the moment, only employers can create users. Many job posting sites don't allow for applicants to make users, because they should be able to apply for a job without this. So for now I will not be making an applicant user model. If I have time further down the line, I will implement it. Next I will implement the job posting functionality.

Blog 8: 01/02/2018

I have implemented basic job posting functionality. At the moment, all a user can do when posting job is specify:

- The job title
- The employer
- Years experience if any
- Description of the job
- Requirements in detail

Although this is just a basic version, I want to do a lot more with it. I want the requirements for the job to be dynamic i.e. the user can specify how many requirements there are. But doing this with Django model forms is proving difficult. I will research this topic further.

Blog 9: 08/02/2018

I have researched creating dynamic model forms in Django further but have still yet to find a solution to my problem. I have decided to take a break from the dyanmic forms and have started writing code to allow an applicant to apply for a job. As mentioned in my functional specification, the applicant will need to answer certain questions in order for the machine learning prediction algorithm to work. These questions are:

- Were you on a high/medium/low salary in your previous job?
- how satisfied were you in your last job? (from 1 to 10)
- how many hours did you work a month?
- have you ever had a work accident?
- have you had a promotion in the last 5 years?
- how many projects did you work in your previous job?

The user can of course also upload their personal information such as their name, phone number and email address. I have yet to figure out how to upload CVs, but I will be more concerned with that after I start working on the CV parser.

Blog 10: 15/02/2018

I have found the solution to my dynamic forms problem. There is a jQuery plugin for this. However to get it to work I had to refactor my view functions in my views.py file to get the plugin to work. I

had to make my views generic templates. I plan on doing this with most of my views down the line, as they are very simple and powerful. Users can now add and remove fields in the job posting model depending on how many requirements they want to specify.

The jQuery plugin can be found here.

Blog 11: 25/02/2018

I started working on the machine learning algorithm for predicting an applicant's performance. I have written a decision tree model from scratch, just to learn how they work. I won't be using it for my actual project as it is good business practice to implement renowned libraries like Scikit learn. At the moment I am just experimenting with different models and impurity measures. Once I am comfortable writing machine learning models i'll start writing the algorithm I will be using in the application.

Blog 12: 07/03/2018

I have written an prediction algorithm and have integrated it with my application. It is a random forest regressor model using a mean square error criterion. The prediction accuracy is only averaging at 60%. However it was originally at 40%. I improved it by tweaking some of the function parameters Scikit learn provides. Still I am hoping to get it higher. I have been thinking of creating some sort of score metric for employee evaluations and turning my regression problem into a classification problem. I don't know if that will have an affect on the accuracy, but I am keen to try it. These are future plans. At the moment the algorithm is integrated with the application and when someone applies for a job, the algorithm predicts a score and stores it in the database. My next plans are to focus are creating the CV parser.

Blog 13: 12/03/2018

I am still working on the CV parser. I'm only allowing one type of CV format to be used by applicants at the moment. If I have time after I will implement more. But I think only allowing one format makes sense, as no employer wants to receive a poorly structured CV. So only allowing one good format is a good idea in my opinion. At the moment, all the parser can do is split the CV into sections suchs as work experience, skills, personal info etc. How I search for requirements in these sections will be challenging. It depends on the type of requirement as well.

- Searching for skills (such as communication, java, excel etc) will be easy as they are generally just one or two words that can be searched for by iterating through the text in the CV.
- Education requirements are very different to skills. They could be the name of a degree, a certain grade or a module involved with the degree. I haven't figured out a way to do this yet, but regardless users uploading their CVs will have to have an absolutely perfect education section on their CV to comply with the format.
- Another requirement could be a qualification. Like a ceritificate or license. These are usually very specific so searching based on the name of the qualification should be easy.

- Finally, there could an experience requirement. This could be a minimum time working in role or experience working with a particular tool or software. This will be difficult to search for, but one way I have thought of would be to implement some sort of BLEU score.

I am going to continue working on the parser until I get a version working that I am satisfied with.

Blog 14: 21/03/2018

I am still working on the CV parser. Searching for the different requirements have proved more challenging than i thought. I have implemented searching for requirements based on keywords. I am trying implement some sort of BLEU score functionality for experience requirements. I have written one from scratch and have used python's nltk BLEU score function also to compare, but I am struggling to get it working the way I want it to. It seems regardless of whether the CV contains they specified requirements or not, other words are carrying more weight and therefore scores aren't consistent. I will continue to work on improving this.

Blog 15: 04/04/2018

A lot of progress has been made since my last blog. Since my last blogs, I have implemeneted the following functionality:

- CVs are now being read and matching keyword job requirements.
- Users can now delete and edit previous job postings
- Users can now search for job posting based on their title
- User's previous employer posting information is saved for future reuse

The next steps are to:

- Implement functionality that can match experience and education requirements
- Implement a "job finder", which allows applicants to upload their CV and see what requirements they match.
- Continue to refactor

Blog 16: 19/04/2018

I had been hoping to have written this blog sooner but unfortunately I have been struggling to implement features of my project. These are the task I have completed since my last blog:

- Education and Experience requirements can now be matched in CVs
- I have also refactored most of the CV parser functions
- I have added some graphical interfaces using a javascript library known as Chart.js

Despite completing a basic working version of my CV parser I cannot help but feel I can do better. It is a very basic implementation of the idea I originally had when I designed this project. I have been trying to make it more but I am struggling. I will continue to try and add more to do it as well as trying to do the following tasks as well:

- Implement a "job finder", which allows applicants to upload their CV and see what requirements they match
- Save CV info to database models for each applicant
- Implement some sort of resume ranking algorithm (BLEU score, machine learning techniques etc)

Blog 17: 29/04/2018

It's now getting closer to submission date and I am struggling to implement features that I specified in my last blog. I attempted to implement a machine learning resume ranking algorithm but could not implement it. I have decided to focus on making my application a somewhat working project, so I can say I submitted something that is complete. Here are the task that I have done since my last blog:

- The most common skills amongst applicants that have applied for a job are displayed in graph on the user's dashboard
- A small summarised version of an applicants curriculum vitae is shown beside the full contents of their CV
- The application form for a job is now split into 3 multi step forms. When an applicant uploads their CV, their personal information is extracted and displayed in the next step of the form.
- Information extracted from the applicant's cv is now saved to a database model
- Users can now sort their tables of applicants based on job match ratio and other headings in the table

The plan now is to implement minor changes and focus on testing and documentation.

Blog 18: 04/05/2018

Progress on my project is steady. I have started my testing documentation and have implemented minor changes into my application. This is how much testing I plan to do:

- I am writing isolated unit tests for the models, forms and views of my application. I am using the 3rd party python coverage library to see how much of my code I have tested. So far I am at 50%. I plan on getting this figure to at least 70%.
- I will be writing automated accpetance (end-to-end) tests using a libray called Sikuli. I used this on my INTRA placement so I am quite familiar with how to use it. I will write scripts to test tasks such as registering a user, applying for a job, posting a job etc.
- I plan on carrying out user testing. I have organised for two users to test my application; one who works in HR as part of their companies recruitment process and the other a recent graduate who has been using job hiring applications a lot lately.

I plan to deploy my application as soon as I am satisfied with my testing.

Blog 19: 20/05/18

I have complted all my documentation, testing and video walkthrough. My application isn't fully

finsihed but I am proud of what I have built. Now it's time for a good night sleep.