Department of Electrical, Computer, and Software Engineering

Part IV Research Project

Final Report

Project Number: #92

Three-way matchmaking system for Voluntari.ly.

Author: Cyrus Raitava-Kumar

Project Partner: Darcy Cox

Project Supervisor: Andrew Meads

Date: 27/09/19

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Name: Cyrus Raitava-Kumar

ABSTRACT

As digital technologies play an increasing role in modern society, it is more important than ever that adequate education in this field is provided to all students in need. Voluntari.ly is an online platform that aims to address this problem by connecting industry volunteers with students and high school teachers, alongside course content providers. At the heart of this platform, and this research project, lies the need for an effective matchmaking system. This system aims to facilitate and optimise contact between volunteers, and teachers and students. Research into existing literature lead towards the design of a tagging engine; an implementation based on the concept of information classification. Entities within this system may be matched based on the common concept of a tag. The implementation of this engine is syntactically-based, and performance tests have been carried out to verify that the predicted load of the system can be handled. Future work on this project may include clustering and/or indexing of database information for faster lookup. This is alongside the usage of semantically-related 'aliasing', to allow for developed reasoning from the matchmaking component.

LIST OF FIGURES

1. Introduction

The aim of this project was to construct and evaluate a functional prototype of a three-way matchmaking system for Voluntari.ly. Voluntari.ly is a charitable initiative that aims to connect NZ schools with corporate volunteers and digital technology course content providers, which together make up the three different parties to the platform. The system was created in an effort to address the barriers preventing current experts in digital technology, from aiding teachers in educating burgeoning generations of eager-to-learn students. In a society becoming increasingly dependent on digital/information technology, competency in moving past digital consumption onto digital innovation is incredibly valuable. This value was one of the driving motivations of the project, alongside being able to convert the pre-existing manual matchmaking process, into something much more efficient and scalable. An effective matchmaking service within Voluntari.ly allows for corporate volunteers and content providers to better target their expertise to schools in need, and furthermore allows teachers to get support specially catered to them and their students.

The project goal was to research, evaluate, and apply classical and/or modern 'matchmaking' techniques in the context of Voluntari.ly, in an effort to make it significantly easier for teachers and volunteers to use the system. For this report, we will consider 'matchmaking' in its general form to be a class of information retrieval as described by Munz et al. [1], with the process being of "identifying similar or compatible entities". Ultimately, this project strived to hybridise and extend the cutting-edge developments made within the matchmaking problem domain, into an elegant solution hosted within the Voluntari.ly platform.

1.1. Research Intent

Academically, our research aimed at identifying valuable and novel matchmaking algorithms found within various external literature, in order to discern and aggregate particularly applicable approaches, given Voluntari.ly's context. Onwards from this, further investigation into each alternative approach was to be made, with the intention of combining the best elements from our research, into a distilled, final matchmaking approach. Finally, following the actual selection and implementation of a solution, critical assessment of the approach would allow for detailed insight into its various comparative benefits and disadvantages, and overall usability within the platform.

1.2. Division of Work

Most, if not all of the work done on this project was carried out in tandem and shared equally between the author and his project partner (Darcy). Meetings with our supervisor were carried out on a weekly basis during the semester, with weekend-long hackathons being attended together once every few months. Features were completed via a mix of both co-located and remote pair-programming sessions, as well as periods of individual, yet equally shared work, following meetings for planning and work delegation.

2. Literature Review

Of the specific applications of matchmaking algorithms and techniques reviewed in literature, two common themes revealed themselves; the use of matchmaking within web service discovery, and separately within electronic/virtual marketplaces. These will be discussed first, followed by an analysis of other recurring points of interest found within our process of literature review. The actual utilization of the different patterns determined will also be explored, alongside the reasoning behind why some were more applicable than others. It is of note that the following section is a condensed summation of a previous work of the author [2].

2.1. Web Service Discovery, and Information Representation

Web service discovery is a problem set within matchmaking, where the aim is to identify the most suitable services from a pool of potential choices, given a set of predefined requirements and/or constraints [3]. In a paper analysing a tree-based approach to service discovery, the concept of a ‘classad’ (classified advertisement) was introduced as part of a potential solution to the problem [4]. A classad defines what information is or isn’t parsed, in the evaluation of a request and advertisements of resources. This is particularly critical, as it represents the fundamental infrastructure under which two different entities in a matchmaking system may be compared, and subsequently matched. The choice of how information is represented in matchmaking systems may substantially change the algorithmic approach taken; a study into service matchmaking based on a ‘hybrid’ algorithm, presented the two large algorithmic branches of syntactic versus semantic matchmaking, and the differences between them in how data was or wasn’t parsed [5]. In syntactic matchmaking (based principally on leveraging the pure structure of words to describe entities), information representation adheres to a stricter, more well agreed upon structure within the system. In semantic

matchmaking (based more on the holistic meaning communicated by entity representation), the concept of a 'logic-based ontology' is introduced – a liberally defined structure in which functional *and* non-functional attributes of a service can be communicated. An important part of the core development of Voluntari.ly's matchmaking system, was deciding on an equivalent classad structure to represent its core entities. The structure needed to be simple and easy to integrate, whilst still allowing for adequate future complexity to be built on top of it. The resultant tagging engine that was built, had careful consideration put into it, as a result of reading into syntactic versus semantic matchmaking. Any external entity that has a relationship with a given tag (e.g. a volunteer, or opportunity), does so by storing an association to its ID. In this respect, Web Service Discovery was highly relevant to Voluntari.ly, as it allowed us to see the benefits and negatives of different forms of information interpretation; a central facet to the matchmaking problem domain.

2.2. Electronic/Virtual Marketplaces, and Ranking

Yet another underlying theme of the academic work reviewed, was that of matchmaking algorithms used within electronic and/or virtual marketplaces. Here, matchmaking was defined generally as "the process of searching the space of possible matches … [in order to find] all the supplies that can fulfil the demand to some extent and identifying the most promising ones" [6]. Tommaso Di Noia et al. go on to present an algorithm that contains an important facet not previously described; ranking of matches, and the degrees of proximity potential matches may have, to a 'perfect match'. The generalized algorithm for ranking was presented as follows (written personally, in pseudocode);

Rank Pseudo-code

```
for every potential match:

    assign match a rank value of 0;

    for every comparable attribute:
        if resource attribute value does not equal request specified attribute value:
            add 1 to the match's rank value;

// Rank matches from closest (lowest rank value) to furthest (highest rank value)
```

Figure 1: Ranking algorithm pseudocode

As described by J. Yan and J. Piao [3], matching involves finding advertisements that meets the requirements of the consumer, as opposed to ranking, which presents them to the consumer in an order that best fits their needs. Being able to determine a 'best fit' match from a series of plausible matches, is imperative in giving the

requester of a system freedom to apply any constraints they may internally have, that were not expressible through the inputs for the algorithm. This is as opposed to making decisions for the user themselves, as to what the best match is. Although electronic and/or virtual marketplaces may in themselves not be particularly similar to our context, the thoughts and reasoning behind the inclusion of ranking were directly transferrable to Voluntari.ly. Sorting the algorithm's results to show matches based on predictions of most to least relevant, aids in mitigating the amount of time the user spends on identifying a satisfactory match and increases the application's usability. A variation of the above algorithm was directly applied to the project, as a feature for recommending volunteering opportunities to corporate volunteers. This feature built on both the research made into ranking, as well as information representation. As such, ranking proved to be an important facet within matchmaking, and was well incorporated within the Voluntari.ly platform.

2.3. Algorithmic Optimizations

Once the heart of the algorithm has been implemented, the next question is obviously how performant is the implementation. According to JayaShree and Chandrasekar [7], the inclusion of a preliminary process of *clustering* the resources to be advertised for matching, can significantly lower computational run-time. When coupled with a pre-filtering step within the algorithm, this clustering can cut the search space down significantly, as Yahyaoui, Almulla, and Own [8] discuss. For the purposes of creating the minimum viable product, high performance benchmarks were not necessary for the development of Voluntari.ly within the project. Performance analysis is nonetheless important and will be discussed later, alongside potential optimisations, inspired by this section of the literature reviewed. The runtime and scalability of an algorithm are commonly seen as metrics for its quality and usability and were used to examine the main algorithms around the tagging engine. Alternative algorithmic designs which allow for better incorporation of optimisations in these areas, are of greater future interest, beyond the scope of this project.

2.4. Optimal matching relative to individual pairs, versus groups of pairs

According to Li [9], there seem to exist two common classes of approaches when analysing 'matchmaking algorithms'; one class of which includes classic algorithms like 'Gale-Shapley', where the best match is found across a *group* of pairs, and other alternatives, where the best match is found for a singular pair. For the

purposes of Voluntari.ly, it seemed most appropriate to develop an algorithm that optimises matches based on a *single* pair's request, at any given one time – given the nature of the platform as it stands grouping and batching of requests seemed inappropriate, simply to return optimal *groups* of matches. It's for this reason that the usage of the tagging engine, in both searching and recommending opportunities and activities, is performed on a per-request basis. Continual recalculation of the optimal matching of *all* of the volunteers simultaneously, with all of the opportunities and content providers, would be too unwieldy, unscalable, and furthermore unnecessary.

2.5. Three-way versus two-way matchmaking

In the process of literature review, we found it extremely troubling trying to find matchmaking algorithms that were tailored to three-party matching. Nearer to the end of the review, we both realised that although three-party algorithms were few and far between, most of the other classical algorithms were simple and modular enough, to extend to our needs. This resulted in variations of two-party matchmaking algorithms, extended to work for three parties. This allowed application of the older matchmaking techniques of previous other systems, to our specific, novel and niche context.

2.6. Hard versus Soft Constraints

When analysing a resource/request pair, and comparing via a set of particular attributes, it is important to decide which constraints are *most* vital to satisfy, and least vital to satisfy [10]. A hard constraint, as explained by Field et .al [11], is one which is used to determine in a binary way, whether or not a match can be satisfied. A soft constraint, comparatively, is one which is not necessary to satisfy for the purposes of being matched but can be used in the process of ranking matches that have satisfied all of the hard constraints. Providing a sense of prioritization for different bases of comparison, allowed for another dimension of specificity for satisfying a request in Voluntari.ly. An example of this exists in the searching and filtering steps involved with volunteers finding relevant opportunities.

3. Voluntari.ly Motivations

Voluntari.ly's primary motivation is to make it easy for volunteers and high schools to match with each other, with the intention of the volunteers bringing a real-world context to the education of technology. This has the

flow-on motivation of increasing the quality of competency in tech in New Zealand, and empowering young students to believe they may have awesome futures in tech. Other alternative solutions to this exist, in the form of services such as HelpTank and Community Comms Collective, however neither make efforts to *automate* their core matchmaking processes. In order to achieve the scale that Voluntari.ly's goal demands, it is imperative that the matchmaking process have as few manual aspects as possible. It is for this reason that Voluntari.ly is necessary, and why research into novel ways of *automated* matchmaking was core to this project.

4. Project Outcomes

This section will discuss the outcomes of the work done on Voluntari.ly over the course of the project, and how they relate to the literature reviewed. It will also critically explain how users of the platform interact with the features produced through the project.

4.1. Searching, Filtering and indicating Interest in opportunities

Corporate volunteers are one of the key stakeholders of Voluntari.ly. When a volunteer logs onto the platform, they can browse a list of all of the active opportunities at any given time.

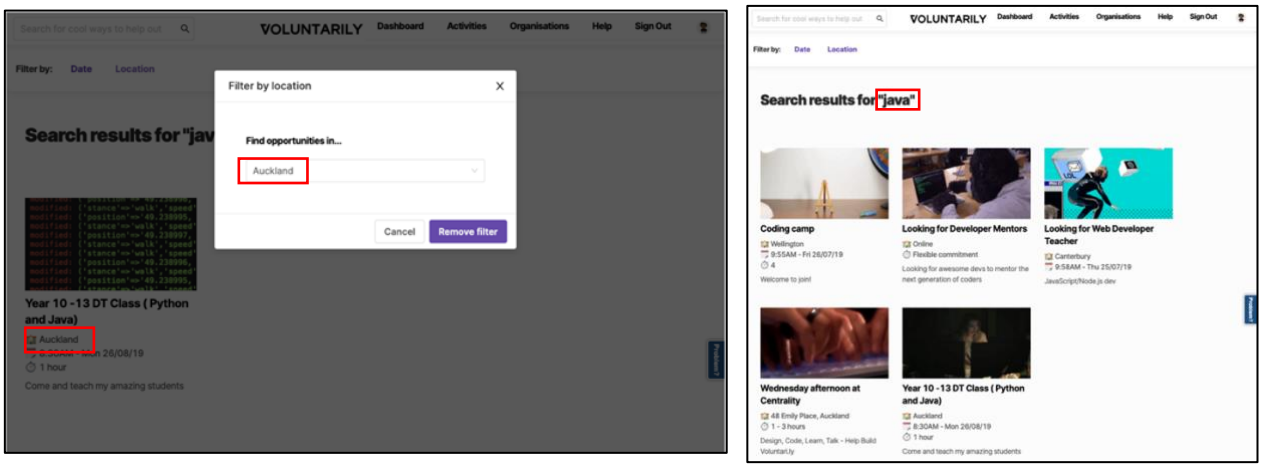

Figure 2: Filtering of opportunities by location, and key words

Volunteers also have the ability to search for opportunities based on key words, or filter opportunities by date and/or location. These can be seen, emphasized in red in the figure above.

Searching checks whether a given input is contained in either the title, subtitle, or description of all opportunities, and then returns a list of any identified opportunities. Onwards from this, the user can then choose to filter this set of results, via an input date or location (as seen in figure 3). Ultimately, this is all in an attempt to help ease the user in finding an opportunity that best suits them, that they may then volunteer for. Following identification of an appropriate opportunity, a user may click on the opportunity card to be directed to further details, including a description of what the opportunity may entail, as well as a link to the organiser's profile, who they can contact for further questions.

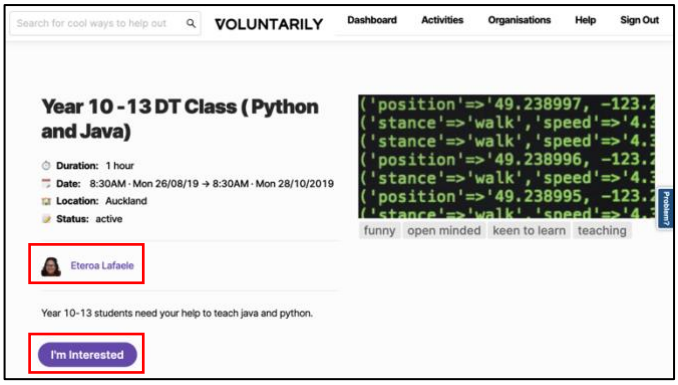

Figure 3: Opportunity details, and expressing interest

They may then express that they are interested in volunteering – this signifies they are free and keen on lending their expertise, at which point it is up to the opportunity organizer, to finalise any remaining details of whether the volunteer attends. This can be seen as user-driven matching and acts as a simple yet effective aid in the matchmaking process of volunteers finding opportunities.

4.2. Calculation of recommended opportunities

In an effort to ease the way in which appropriate opportunities are identified by volunteers, work was done towards the system being able to recommend relevant opportunities to a given user. These recommendations are based on the user's location and skills, according to their profile. New Zealand as a country is split up into several different regions, with each region containing several territories. For a given user's location, opportunities will be recommended with precedence to any contained in the same territory, followed by any contained in the same outer region. For a given set of a user's skills, the top 10 most relevant active opportunities are found and shown. The underlying algorithm for this is similar to that shown in Figure 1,

whereby a ‘score’ is associated with each opportunity. This score is proportional to the relevance of an opportunity and is calculated as the size of the intersection between the user’s skills, and the opportunity’s tags. This holds higher relevance to the literature reviewed prior to the project, as it is a form of automated matching, driven by the system itself. The searching and filtering functionality works in conjunction with the

Figure 4: Adding skills to profiles

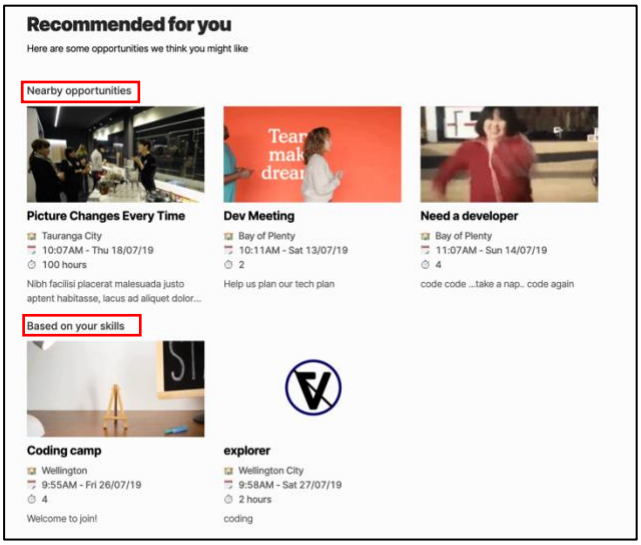

Figure 5: Opportunities recommended by user location and skills

recommending of opportunities, to provide a streamlined way in which volunteers sign up for opportunities. The overall intended result of this was to increase the quality of pairings between volunteers and opportunities and remove as many barriers to user engagement as possible, to ensure high uptake.

4.3. The Core Tagging Engine of the Matchmaking System

To facilitate matchmaking, a tagging engine was developed to link relations between different entities. This approach gave us the freedom to further delve into either syntactic or semantic matchmaking, with tags being leveraged to deduce relations in either case.

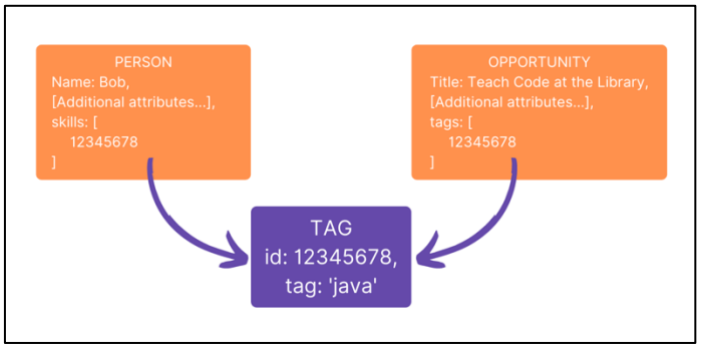

Figure 6: Entity relation expression through tags

Tags were introduced as an attribute to the person, activity and opportunity entities, with tag entries all being shared under the same single universal table in the database. This form of information classification allowed entities to have many-to-many relations with tags, allowing tailoring of complexity to best suit the system in the present, and the future. A tag entity has an associated identification attribute (ID) to ensure its uniqueness, and a 'tag' attribute to represent its human-readable form. An opportunity, activity or person associates themselves to a tag, by containing a reference to it internally, in the form of a tag's identification attribute. One intentional consequence of the chosen approach, was the ability to easily relate different entities to the same logical concept.

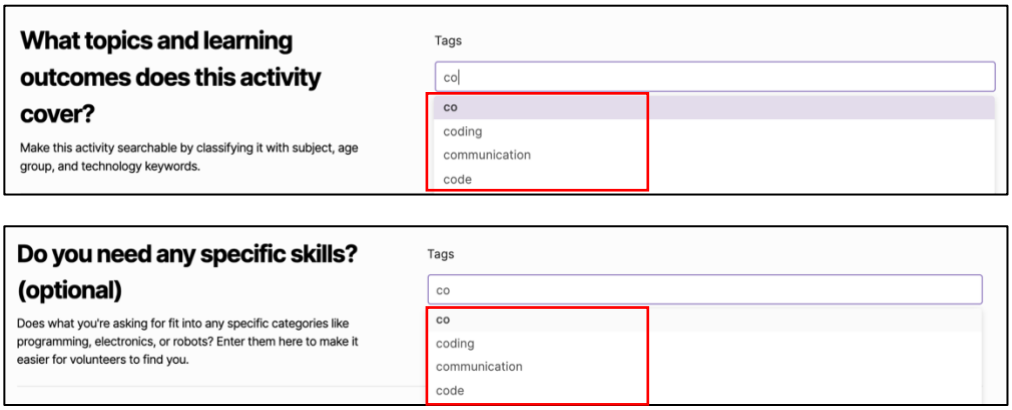

Figure 7: Identical tagging engine functionality for different entities

As shown above in Figure 8, when creating either activities or opportunities, users are able to add tags to either, with the autocomplete function retrieving the same set of data in both cases. This reduced data duplication, as common concepts such as ‘coding’ or ‘communication’, that are relatable to opportunities, would feasibly also be relatable to activities, and people. As more and more users engage with the platform, a strong vocabulary of tags builds up as a result of entries being commonly used, and new ones being continually added.

In order to ensure that such a vocabulary would build over time with no duplicate information stored in the database, it was important to ensure tags were created in a particular way. Essentially, when a tag is associated with an entity, an initial check is made to see whether that tag already exists in the database. If so, the ID of

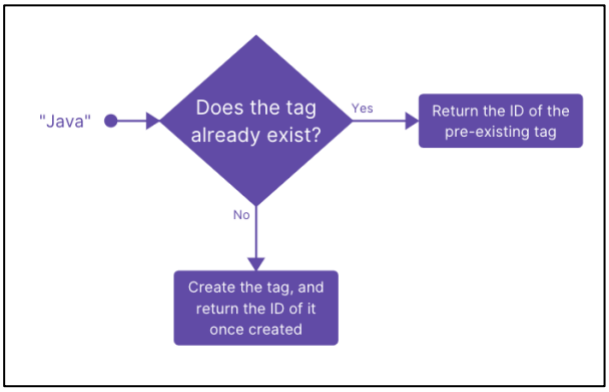

Figure 8: Control flow graph for tag creation

the tag is used as a reference within the entity, and if not, then the tag is first created in the ‘tags’ database table, and the resultant ID associated with it is returned. This ensured every tag of every opportunity, activity and person, was valid, and could be universally, consistently used.

As mentioned previously, matchmaking involving the tagging engine is purely *syntactic* – based on the structure of the words themselves, not their related meanings. Usage of logic-based ontologies for semantic matchmaking was avoided, as the desired outcomes of the matchmaking component were plausible using a simpler syntactic approach, with less overhead. Additional benefits from a semantic approach could include a more customized experience, with less restrictions around the exact structure of tags introduced into the system. This can be achieved with the current syntactic system, via ‘aliasing’ – potential future work that

will be discussed later in the report. Ultimately the tagging engine was built as a means of facilitating both simple and complex matchmaking, in a way that was as intuitive as possible. It is important to note that the engine was designed in such a way as to minimise overhead for initial development, whilst maximising the freedom to adapt the approach in the future. In designing this approach, a large portion of the literature reviewed around information representation was leveraged.

5. Project Experiences

This section will discuss the experience of working on this project, and the successes and difficulties felt within its development processes. It will examine the numerous ways in which work was actually carried out, including sprint work in a physical office, hackathon attendance, and the novelty of working in a start-up environment, among others.

5.1. Struggles and successes of working in start-up environment

Voluntari.ly began as an initiative from the Pam Fergusson Charitable Trust, with an assortment of backing companies, and a few developers and designers. As such, the project in its infancy created a 'start-up' like environment initially, where many large decisions around chosen technologies, and subsequent development approaches were made frequently. Additionally, onboarding of new members of the project happened continually, as development of the platform picked up momentum.

One struggle in particular of working in such an environment, was the slight misalignment of the nature of the Part IV Research Project, with the nature of Voluntari.ly. That is to say, for the purposes of the research project, it was important to have clear, unambiguous goals to be able to work towards over the course of the year. This was difficult to incorporate into the Voluntari.ly team due to their practicing of agile methodologies, where requirements were never particularly constrained or set, and were constantly re-prioritised based on value and relevance. What this consequently meant was that sometimes the highest priority work within Voluntari.ly, was not directly in scope for our project, but was necessary to work on nonetheless. The nature of the project meant several developers were working on it at once – work was not constrained to just the project partners. As such, it was easy to feel an implicit pressure to devote the same time and effort to the project that everyone else was, despite both of us experiencing different time constraints

as full-time students. Our position and stake in the project also meant that care had to be taken in the balance between fulfilling our roles as students and fulfilling our roles as contributors to the platform. The conflict between which side to favour was difficult to handle and made choosing which tasks to devote effort to problematic sometimes.

5.2. Week-to-week work processes

In terms of our work processes on a weekly basis, our cadence varied depending on our capacity. This worked well, as our processes adapted to allow us to both work as efficiently and effectively as possible, with a degree of freedom to account for extracurricular commitments.

In the beginning of the project during the first semester, weekly meetings were held with our supervisor to ensure he was up-to-date with our progress. The actual tasks to be done were gathered from a Jira board created for Voluntari.ly, that was linked to the actual state of the codebase. Remote work was done, with various video calls over the week being used to facilitate delegation and coordination of tasks, to ensure we were both on the same page. This allowed both of us to work to a flexible schedule, whilst ensuring we still made steady progress in development.

During the inter-semester break, a significant amount of time was dedicated to the project – around three days (24 hours) per week were spent physically in the Voluntari.ly office in 48 Emily Place, Auckland, working closely with Andrew Watkins and Walter Lim (two core members of Voluntari.ly). This allowed us to drastically shorten the feedback loop of guidance from the employees at Voluntari.ly, resulting in greater amounts of high-quality work being achieved. Being physically around the other employees of the platform allowed us to establish better interpersonal relations with them, as well as carry out in-person code reviews. These allowed us to learn facets of the codebase quickly with ease, priming us well for development both through the inter-semester break, but also into the second semester, and final half of the project.

Finally, in the second semester, we chose to add a little more rigidity to our work processes, in the hopes of giving ourselves time to perform qualitative analysis of our work. This resulted in employing a semi-agile approach to work, with our weekly supervisor meetings transitioning into more of a review of the work done

in the ‘sprint’ gone by. This was followed immediately by a meeting between both project partners, to plan out the next sprint of committed work. This allowed us to quickly take our own direction to further develop the matchmaking component of the platform specifically. Upon certain tasks being identified and assigned for the week, we would get our plan reviewed by Andrew and Walter, to ensure our work was still aligned with the direction of the system. This process worked particularly well, as it also gave us time to reflect on what things were blocking our progress and allowed us to allocate time to plan against any impedances fully. The approach allowed us to finish up our work on the matchmaking system, whilst still having time to carry out performance testing, and objective analysis.

5.3. Hackathons – struggles and successes

One important facet of the project, was its reliance on the work of volunteers, and its inclusion of hackathons. These hackathons were essentially two-day workshops where intensive collaboration took place, to get entire features integrated into the codebase, with usable software as the result. Several hackathons took place over the year, and we were able to attend a few of them.

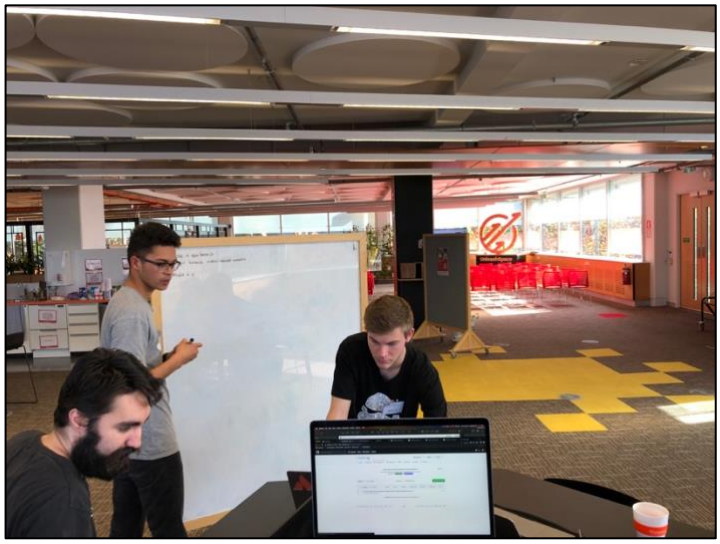

Figure 9: Voluntari.ly development at a hackathon

The hackathons provided great potential for significant development and allowed us to directly collaborate with passionate developers and designers. The events allowed us to expose our ideas to a widely varied pool of experience, giving us the benefits of fresh perspective, and wisdom, simultaneously. Hackathons also

facilitated minimisation of context switching, allowing larger goals to be met, at the cost of sustained, concentrated efforts.

A struggle with these hackathons, was the added responsibility of empowering others to work on our planned features. At the events, we frequently took on the roles of leaders of a subsection of the developers, to address whatever features or fixes the matchmaking system component most urgently needed. We found it difficult to create tasks that were challenging yet not impossible, for newly on-boarded volunteering developers. It was difficult to also be seen as the 'resident expert' on matchmaking from less experienced developers, when in truth a lot of the concepts – particularly in the early stages - were being learnt on the fly by both of us, in the hackathons themselves. A very brief summary of the achievements of each of the attended hackathons is below.

5.3.1. Hackathon 1: Datacom

Work was done on the concept of a volunteer having an 'interest' in an opportunity, via an intermediary join table. This join table contained identification numbers uniquely linking to a person, and an opportunity.

5.3.2. Hackathon 2: Unleash Space

The initial approach and design of the tagging engine was brainstormed and incorporated, in addition to the integration of such tags into external entities. Rudimentary filtering of opportunities was also achieved here, allowing users to finally categorise the information required for matchmaking.

5.3.3. Hackathon 3: Datacom II

Here, the concept of an activity was fleshed out, as well as similarly being able to search and filter activities based on input. Functionality to be able to create an opportunity from an activity was also added, finalizing yet another core bond in the heart of the matchmaking system.

Ultimately the hackathons provided stressful yet productive environments, with differing levels of expertise in the teams we led, making it difficult to sustain any consistent progress pace. In spite of this, it was incredibly rewarding to not only be able to present whole features to the rest of the team, but also to see how less experienced developers became independent in their contributions over time.

6. Results and Discussion

This section will discuss the importance of both user testing and performance testing to the platform, and the methodologies behind both. It will then go on to address the results of qualitative analysis made on the matchmaking features within the project, and where potential further improvements may lie based on this.

6.1. Unit Testing

Throughout the entire project, unit testing of all additional work developed was compulsory, when applicable. The Voluntari.ly codebase incorporated third-party tools such as Travis CI, and CodeCov, to allow for the automation of testing standards – when a pull request was opened, a build would automatically be created and analysed, to see how much of the additional code added was covered by the collection of test cases. Code was 'covered' by a test, if it was executed as a result of a test suite. This inclusion acted as an effective preliminary check, which – when coupled with protected branches that enforce the usage of pull requests – ensured a high standard in testable code. The majority of pull requests reviewed were done so by experienced developers in the team, who were able to identify whether or not components were adequately and validly tested. Ultimately unit testing led to a higher quality project, with greater assurance that the observed behaviour matched the intended behaviour.

6.2. Performance Testing

Once the main features of the project were complete, we decided it was important to qualitatively test our code; both for the sake of seeing how performant the matchmaking system was given its context, as well as being able to identify areas for future improvement. The underlying motivation for this came as a result of the literature review – matchmaking algorithms that return optimal solutions tend to have poor time complexities, due to their nature of *comparing* all potential matches [12]. We were interested to see how well our algorithmic approach would stand under load, and how feasibly scalable our components were for future usage.

6.2.1. Performance Testing Methodologies and Discussion

Our methodology for performance testing, involved measuring the system's time to respond under varying levels of simulated load. Due to the core logic of the matchmaking system existing on the server-side of the codebase, we decided that performance testing of the client-side section of the project would yield little useful

data and would more likely be testing the strengths and weaknesses of React, which was the frontend framework used. Given this, we decided to test two main server-side API endpoints – one which represented the search functionality for identifying suitable opportunities, and another which allowed a user to be recommended opportunities to them by the system.

Our aim was to create conditions for the project, that were as realistic as possible – one way we achieved this, was in the way that we placed the system under load. We chose to control the number of tags, and number of opportunities in the database, as our independent variables. We scaled up the numbers of opportunities by factors of 10, from 10, to 100, 1000, and 10,000. For each four of these scenarios, we scaled up the number of tags associated to each opportunity, linearly from 10, to 20, and 30. For the purposes of testing the search endpoint this was sufficient, however the recommendations endpoint required an extra dimension of information, in the form of associated skills particular to the user requesting the information. We chose to test two general scenarios to cover this, of a user having 10 associated skills, and 20 associated skills.

The way in which we populated this data was specific as well – our initial approach involved scripts that created a completely new batch of tags with every additional opportunity, with no two opportunities having a single shared tag. This did not take into account the fact that the vocabulary of tags in the system in production was purposed to build up over time, with many tags being reused via the autocomplete component (see Figure 8). We thus pivoted the seeding scripts that we wrote to associate tags with a new opportunity, to instead source tags from a constant, potential pool of 200 different values. This better modelled the expected, realistic state of the system under use.

Once the fixtures were set, we recorded and averaged the time it took between sending a request to each of the endpoints, and getting a valid response. This was achieved via an npm package called loadtest, which allowed the specification of the total number of requests to send, alongside how many requests per second to send. It would then provide a measurement for the average period of latency, from which we constructed our results.

Our choices of independent variables accurately reflected which conditions would be variable under user load, whilst still being strongly relevant to the endpoints under test. Under supervision from Andrew Watkins, we

chose *not* to address testing the load of multiple simultaneous users, as this was deemed significantly less realistic than simply focusing on the load of several opportunities and tags.

6.2.2. Performance Testing Results

After aggregating the data from our testing, one can see the results in Figures 11 and 12, below.

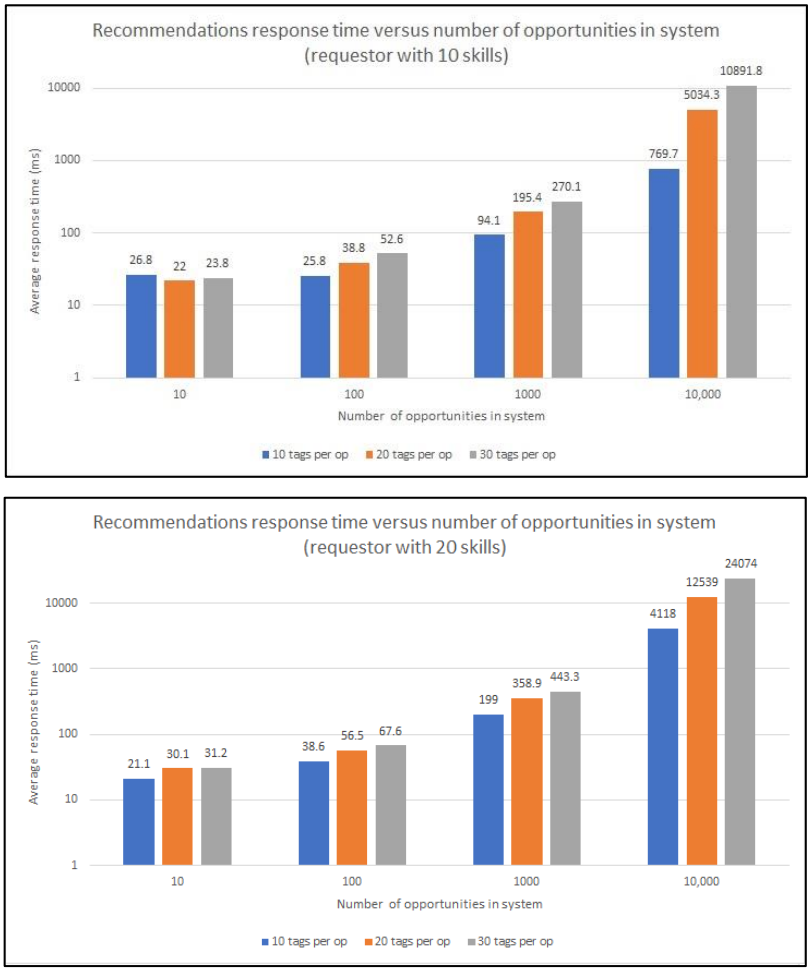

Figure 10: Performance testing results of the recommendations endpoint

The recommendations endpoint itself, looks through each opportunity in the system, and ranks them according to the size of the intersection of its tags, with the skills of the user. This relies on the algorithm scanning through and comparing each of the user's skills, with each and every single tag associated to every opportunity – in the worst-case scenario above, this results in 6 million comparisons. The runtime of the search endpoint does not degenerate as badly – the database's querying tools are leveraged to check whether the input search term is

contained in one of three attributes for each opportunity, resulting in a maximum number of 30,000 ‘contains’ operations.

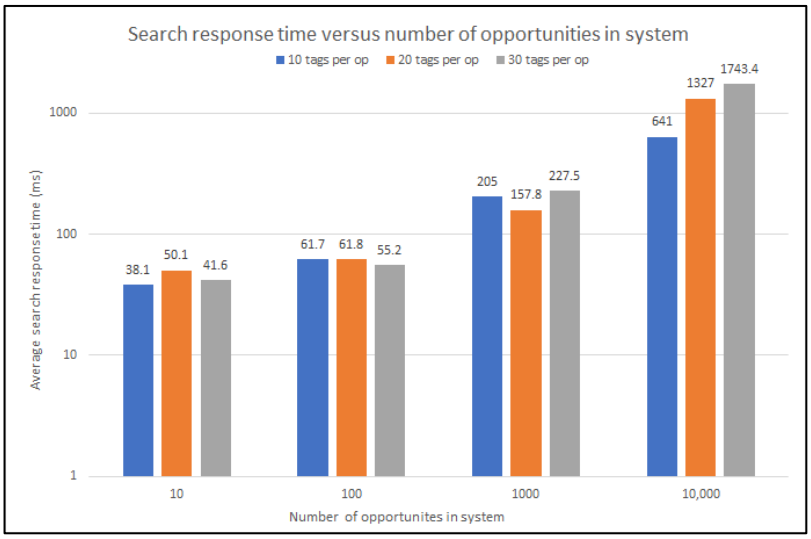

Figure 11: Performance testing results of the search endpoint

As expected, generally as each of the controlled variables increase in scale, so too does the average response time of the endpoints themselves. This behaviour is acceptable for the purposes of the minimum viable product – for example, Voluntari.ly does not feasibly expect to have a user with 20 input skills, seek recommended opportunities from a pool of 10,000, with 30 tags associated to each opportunity, anytime soon. The purposes of testing at such extreme levels was simply to note how the system degenerates as scale increases, and whether there were any causes of concern due to premature failure, of which there are none – the system still works, albeit very slowly. It is important to note that on the lower scales of load, the system’s behaviour more closely resembles that of the database itself, in terms of response time. This may explain the slight inconsistencies in results, found at the level of 10 opportunities, for each of the three test suites above.

6.2.3. Weaknesses and potential solutions identified

It is important to note the inherent limitations using certain technologies can have, for further optimisations – MongoDB was chosen by Voluntari.ly as a database program due to its relatively painless setup, and lack of relational constraints, yet nonetheless has its own weaknesses. Any further improvements to the runtime of both the recommendations and search endpoints however, are partially restricted by the performance and behaviour of their dependencies, of which MongoDB is the main one. A potential future workaround to this, could involve the incorporation of clustering and indexing of opportunities based on associated tags, as

identified in the literature review [7]. This optimisation would be better suited to the recommendations endpoint due to the constrained values of tags and could cut the search space down by removing the analysis of any opportunities that *aren't* at all related to a given tag.

Overall, performance testing allowed us to identify oversights in algorithmic design, as well as potential optimisations that could be employed in the future. For the time being, the performance of the matchmaking component is suitable, and will remain that way until the projected number of active users scales to a significantly higher level.

7. Conclusions & Future Directions

The future of Voluntari.ly is incredibly promising – following the conclusion of this research project, an initial full release to the public is planned for January 2020. The research into existing relevant literature, as well as similar systems, highlighted the need for a simple, yet effective and easily extensible approach to matchmaking in Voluntari.ly's context. The result of this was a syntactically-based matchmaking system built on a form of information classification, formalised as a tagging engine. The design of this engine was based on syntactic comparisons, yet future research into 'aliasing' could allow for the system to be able to semantically link concepts, which could be sustained through user input. Performance testing of the engine revealed its current behaviour is acceptable for the minimum viable product, and that potential optimisations lie in the clustering and indexing of opportunities based on their tags. Ultimately the purpose behind Voluntari.ly is a reputable one, and it has been incredibly fulfilling working on such a technically complex system. As new ways are found to leverage classical matchmaking techniques, it is hoped that Voluntari.ly may exist as a stepping stone to more efficient and effective matching algorithms, and a higher quality future for the education of technology in New Zealand.

Acknowledgements

We would like to acknowledge the hard work and efforts contributed by Andrew Meads, as the continuously supportive supervisor of this project. Andrew Watkins and Walter Lim have also been incredible visionaries, in helping realise how the matchmaking system best fits within Voluntari.ly. The Pam Fergusson Charitable Trust, and its founder Vaughan Fergusson must also be thanked for helping bring the platform into reality, as

well as all of the companies that provided backing behind them. The author would also like to thank Darcy Cox for being a great project partner, and finally his family and friends for their support in him pursuing Software Engineering.

References

[1] U. Schmid, L. Berle, M. Munz, K. Stein, and M. Sticht, "How Similar is What I Get to What I Want: Matchmaking for Mobility Support," *Computational Approaches to Analogical Reasoning: Current Trends Studies in Computational Intelligence*, pp. 263–287, 2014.

[2] C. E. O. Raitava-Kumar, "Literature Review and Statement of Research Intent; Project Number: #92 Three-way matchmaking system for Voluntari.ly."

[3] J. Yan and J. Piao, "Towards QoS-Based Web Services Discovery," *Service-Oriented Computing – ICSOC 2007 Lecture Notes in Computer Science*, pp. 200–210, 2009.

[4] M. R. Islam, M. Z. Islam, and N. Leyla, "A tree-based approach to matchmaking algorithms for resource discovery," *International Journal of Network Management*, vol. 18, no. 5, pp. 427–436, 2008.

[5] J. Bao and J. Xia, "A hybrid algorithm for service matchmaking based on ontology approach," *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2017.

[6] T. D. Noia, E. D. Sciascio, F. M. Donini, and M. Mongiello, "A system for principled matchmaking in an electronic marketplace," *Proceedings of the twelfth international conference on World Wide Web - WWW 03*, 2003.

[7] K. JayaShree, M. Chandrasekar, "Clustering Web services for effective service discovery", *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 5, no. 10, pp. 2500-2503, October 2016.

[8] H. Yahyaoui, M. Almulla, and H. S. Own, "A novel non-functional matchmaking approach between fuzzy user queries and real world web services based on rough sets," *Future Generation Computer Systems*, vol. 35, pp. 27–38, 2014.

[9] Li, Charles, "Devising a framework for efficient and accurate matchmaking and real-time web communication." *Technical Report TR2016-798*, Dartmouth College, Hanover, 2016,< http://www. cs. dartmouth. edu/reports/TR2016-798-rev2. pdf.

[10] U. Bellur, H. Vadodaria, and A. Gupt, "Semantic Matchmaking Algorithms," *Greedy Algorithms*, 2008.

[11] S. Field, C. Facciorusso, Y. Hoffner, A. Schade, and M. Stolze, "Design Criteria for a Virtual Market Place (ViMP)," *Research and Advanced Technology for Digital Libraries Lecture Notes in Computer Science*, pp. 819–832, 1998.

[12] K. Kritikos and D. Plexousakis, "Novel Optimal and Scalable Nonfunctional Service Matchmaking Techniques," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 614–627, 2014.