Department of Electrical, Computer, and Software Engineering

Part IV Research Project

Final Report

Project Number: 92

Voluntarily: A matchmaking system for volunteering

Darcy Cox

Partner: Cyrus Raitava-Kumar

Supervisor: Andrew Meads

25/09/2019

Declaration of Originality

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

[illegible]

Name: Darcy Cox

ABSTRACT

Digital technology content will soon be added to the New Zealand curriculum, posing a significant challenge for teachers from non-technical backgrounds as they try to teach this content. Voluntarily is an open-source software project that hopes to assist teachers with this transition to the new digital curriculum by connecting them with volunteers from relevant industries. This will bring expert voices into the classroom, helping the students to effectively learn these new technical concepts that the teachers are not necessarily comfortable with. Voluntarily needs a way of facilitating this connection between volunteers and teachers, which this part IV project has implemented. A literature review of classical matchmaking techniques was performed, outlining the core concepts in the literature and their usefulness to Voluntarily. This project's resulting implementation facilitates the connection of volunteers and teachers by applying matchmaking techniques from the literature to produce three main features: a tagging engine that helps with classifying information; rich searching and filtering functionalities to help volunteers browse opportunities and easily find what they need; and recommended opportunities presented to the users on their home page, based on information they have set in their user profile. The API endpoints exposing these features were performance tested to understand how the implementation behaves at scale. The results were mostly satisfactory, but there is significant performance degradation at the highest scale that was tested, suggesting some further work is needed for the implementation to be useful once Voluntarily is used by many schools around New Zealand.

LIST OF FIGURES

1. Introduction

Voluntarily is an open-source software project built using modern web technologies that aims to bring together schoolteachers and volunteers from the technology industry. The project assists schools with teaching digital content in light of recent changes to the New Zealand curriculum, where schools are now required to teach technology skills to students from a young age right through to the end of their schooling. As teachers typically aren't skilled in these areas of technology, Voluntarily helps them find volunteers who are knowledgeable in these areas and can help run certain activities to teach their students these concepts. To be a useful product, Voluntarily must provide an easy way for teachers to find the right volunteers that they need and for volunteers to find opportunities that are most desirable to them. To implement this effectively, a matchmaking system is needed which facilitates the connection of teachers and volunteers. This part IV project focusses on applying matchmaking techniques to Voluntarily.

Matchmaking in its most basic form is the process of finding a pairing between two parties such that both parties are satisfied based on their respective needs. In the case of Voluntarily, these two parties are teachers and volunteers. Teachers are satisfied when they can find volunteers who will get along well with the children in their class and have the required skills to complete a given activity. Volunteers on the other hand are interested in finding opportunities that meet certain criteria such as having a convenient date, being nearby, and being aligned with their skills. The implementation arising from this project makes it easier for volunteers and teachers to find each other by applying matchmaking techniques to the existing Voluntarily code base.

1.1. Research Intent

As there is significant pre-existing literature on matchmaking techniques across many different application domains, this project's intent is to gain a good understanding of these classic techniques that have been around for a long time and adapt them to a modern context within the Voluntarily code base. This adaptation requires dealing with both a unique application domain (a volunteering system) as well as a technology stack that matchmaking is not typically applied in (modern JavaScript rather than more mature languages such as C++ [1]). To be successful, these matchmaking techniques must make it as easy as possible for the teachers and volunteers to find a suitable match. In the case of Voluntarily, because both parties needing to be matched are real users, the matchmaking techniques must also be applied in such a way that they provide a positive user experience.

Throughout the timeline of this project, Voluntarily was a start-up project still in its infancy. Being in its very early stages, the platform evolved rapidly as a more thorough understanding of the users and the context of use was gained, changing the priorities of the system. Much of this project's research involved keeping up with the priorities of Voluntarily and understanding how to best apply matchmaking techniques to the evolving product. This required frequent communication with the Voluntarily team as well as staying heavily involved in the developers' community.

Finally, upon completing an implementation of matchmaking in Voluntarily, the implementation must be tested for both correctness and performance. The correctness of the matchmaking is highly important as it directly correlates to the usefulness of Voluntarily in that both teachers and volunteers must be able to find relevant matches. The performance testing is also essential due to the inherent non-scalable nature of classic matchmaking techniques. These tests will provide an objective understanding of how successful the project has been and make clear any future adjustments to be addressed.

1.2. Division of Work

Throughout most of this project, the required work was completed as a team effort between the project partners either through in-person pair programming or through regular communication using online video calls. When it seemed necessary to divide tasks between the partners, this work would be split up into front-end and back-end work, which could be implemented separately and then combined to complete the necessary integration of the two.

2. Literature Review

Prior to implementing this project, a literature review was performed to gain an understanding of the existing matchmaking techniques and their applications [2]. This section outlines the key points from the literature review and discusses the usefulness of matchmaking techniques found in the literature when applying them to the Voluntarily.

2.1. Matchmaking in general

In its general form, matchmaking can be described as the "process of finding an appropriate provider for a requester through a middle agent" [3]. In this definition, a "provider" refers to an entity that offers something, a "requester" is an entity that is looking for something offered by providers, and the "middle agent" (matchmaker) is the system that facilitates this matchmaking to assist the requesters in finding a suitable provider [3]. This generalised conceptualisation of matchmaking is summarised in fig. 1.

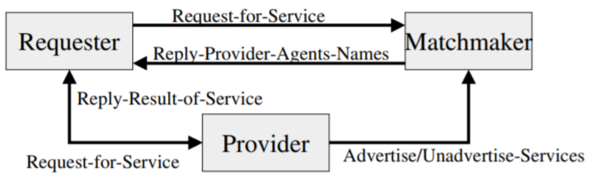


Fig. 1 A general conceptualisation of a matchmaking system [3]

This definition of matchmaking can be adapted to many different application domains by considering the providers and requestors as different entities (such as teachers and volunteers in the case of Voluntarily). Because of this, it is relatively easy to adapt classic matchmaking techniques in the literature to a new domain, which was very beneficial for this project.

2.2. Practical applications of matchmaking

As discussed in the previously published literature review [2], the two most dominant applications of matchmaking are found in electronic marketplaces [4] and web service discovery [5]. While these two applications are by far the most common, matchmaking techniques have also been applied to different domains such as job advertisements [6] and live kidney donations [7]. Due to its ability to be considered at a very generalised, abstract level, matchmaking has undoubtedly been applied to many other domains, including ones that do not appear in the literature for proprietary reasons (such as Tinder or other online dating applications).

Studying the existing applications of matchmaking enabled a better understanding of how to adapt matchmaking techniques to a modern context such as Voluntarily.

2.3. Matchmaking techniques

There are many matchmaking techniques found in the literature which can be organised into the categories of information classification, ranking and matching, optimisations, and types of match. Each of these categories addresses a specific sub-problem within the matchmaking field and were either found to be useful in the project's implementation or are worth considering for future work on this project.

2.3.1. Information classification

A matchmaking system must have some way of understanding the nature of the entities it is matching in order to determine when a suitable match has been found. The two main options are a syntactical approach [8] or a semantic approach [9].

A semantic approach allows a matchmaking system to understand the meaning behind natural language descriptions. This involves using an ontology, which is a technique that achieves a shared understanding within a system by defining concepts of a given domain, their properties, and the relationships between them [9]. Ontologies are typically implemented using "description logics", which are a form of knowledge representation languages [1]. Ontologies exist as an entire research field on their own and typically have complex implementations that were deemed out of scope given the timeline of this project.

This project's matchmaking implementation uses a syntactic approach, where attributes are associated with the entities to be matched without an understanding of the semantics behind these attributes. In order to determine a suitable match, simple constraint checks are performed based on what the user has requested.

2.3.2. Ranking and matching

A useful distinction within the literature is that between ranking and matching. Matching can be considered the process of determining if a given pair of entities is a suitable match based on a set of constraints; it is a simple yes or no answer. Ranking is the process of taking a set of valid matches and ordering them from best to worst in terms of how well they match [5]. For Voluntarily, the matching process is often as simple as constructing the right database query to find all valid matches. Therefore, this project focussed much more on the ranking process, which is essential for ensuring the users are only presented with the most suitable matches and are not overwhelmed with too many options.

2.3.3. Optimisations

Matching and ranking techniques inherently involve a large number of comparisons due to the need to check all possible matches to see how they conform to certain constraints. This causes the classical techniques to have runtimes that do not scale well. The prior literature review [2] discussed several optimisation techniques that help reduce the runtimes, including clustering and the notion of hard and soft constraints. Given the timeline of this project and the current scope of Voluntarily, these optimisations were not applied in the implementation but would be a good candidate for future work.

2.3.4. Type of match

As discussed in [2], the literature distinguishes between three types of match: partial, potential, and exact. A partial match arises when a pair of entities satisfies some constraints, but explicitly does not satisfy others. A potential match only satisfies some constraints, but it is unclear whether the other constraints are satisfied, implying further clarification is required. An exact match perfectly satisfies all constraints. The implementation in Voluntarily uses the concept of potential matching, as it optimistically assumes the absence of information on all constraints is enough to be considered a match.

The other useful distinction found in the literature is that of a stalled match and an immediate match. Immediate matching is done only once at the time of the request, while stalled matching can be performed periodically to find a match without user involvement [10]. This project's implementation uses immediate matching due to its lower complexity to guarantee that Voluntarily could quickly be provided with useful matchmaking techniques. However, stalled matching would be very useful in future as it could minimise user involvement and ensure teachers and volunteers eventually find a match for what they need.

3. Voluntarily

Voluntarily is an initiative started by the Pam Fergusson Charitable Trust with support from many leading New Zealand companies. It aims to seamlessly connect teachers, volunteers, and content providers (who provide useful resources such as lesson plans) as the New Zealand curriculum introduces digital technology content. Teachers typically do not have technical expertise, so Voluntarily will help them bring expert voices into the classroom to teach the new content.

The users of Voluntarily at the time of completing this project consist of teachers and volunteers. If a teacher needs help with a certain lesson, they may log into the system and create an "opportunity", specifying what skills are required, where and when the opportunity will take place, and other information that volunteers might need to know before signing up. This opportunity will be added into the database, allowing volunteers to view it. The volunteers using the system are typically people from tech companies who want or need to complete several volunteer days as part of their contractual obligation with their company. Voluntarily allows the volunteers to browse opportunities and, once they have found one that they are interested in signing up for, can express their interest. On expressing an interest, the teacher who requested the opportunity is sent an e-mail with the volunteer's details and they can decide if this person is suitable for their needs.

The application is built using JavaScript in both the front end and back end. More specifically, the front end uses the popular UI framework React in combination with Redux, a popular state management tool. The backend utilises the server framework Express and a library called NextJS to support server-side rendering. MongoDB is used for the database-tier due to its non-constraining nature, allowing the data model to evolve rapidly as the Voluntarily project grows, without the overhead of writing database scripts to modify the data to comply with each new schema.

4. Project Outcomes

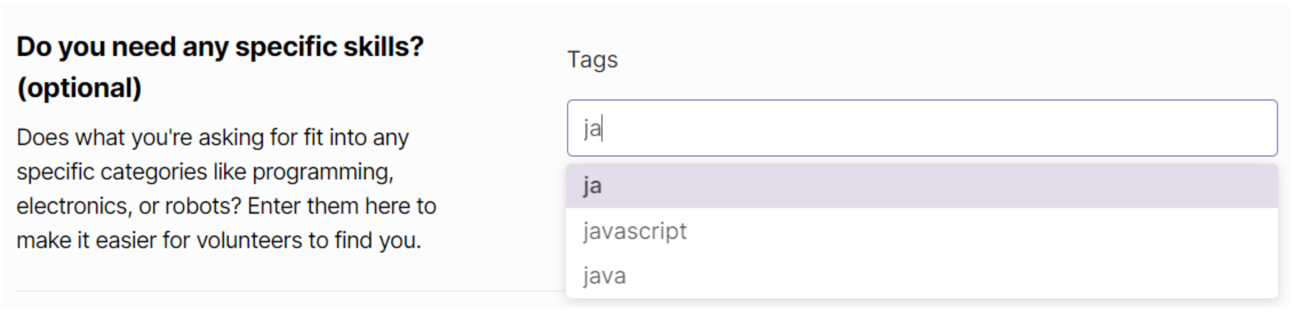
As this project took place while Voluntarily was in its infancy, much of the focus was related to assisting volunteers in finding opportunities that are most applicable to them. This is because in the minimum viable product, volunteers are the drivers of the system and initiate matchmaking by finding opportunities and expressing their interest. This project assists volunteers with finding relevant opportunities in three different ways: a tagging engine that helps classify information, a user interface allowing browsing of opportunities, and a feature that recommends opportunities to volunteers on their home page.

4.1. Tagging Engine

4.1.1. Current Approach

This project opted to use tags as the primary mechanism of information classification. Tags in Voluntarily are stored in their own MongoDB collection, each having its own ID property and a single value, which is the name of the tag. Opportunities have a tags property which is an array of IDs, each referencing an element from the tags collection. The tagging engine has been designed so that the vocabulary of the tags builds up over time as more tags are entered into the system, placing trust in the users of Voluntarily to define the common terms used across the system.

Tags relating to an opportunity are specified by the teacher requesting the opportunity either at creation time or on subsequent edits. Within the form in the front end, there is a field for specifying the tags required. This tag input field allows the user to add multiple tags to a single opportunity and offers helpful suggestions as the user types, based on the currently existing terms in the database (fig. 2). Teachers can select existing tags but may also type new tags that don't yet exist in the database. The logic implemented in the back end upon saving opportunities has tag initialisation middleware that executes before the save to database function is run. This middleware analyses the specified tags and checks if they already exist. If a tag does not exist, the tag is created, otherwise its ID is taken as is, preventing duplication of tags. It is expected that as time goes on there will be less new tags created, while in the early stages of the system being live there will be many new tags due to the vocabulary not yet existing when few opportunities have been requested.


Do you need any specific skills? (optional)

Does what you're asking for fit into any specific categories like programming, electronics, or robots? Enter them here to make it easier for volunteers to find you.

Fig. 2. The tag input field with autocomplete active

The decision to store tags as a separate entity within Voluntarily was made in order to avoid duplicating the same information across many opportunities in the database. Having a tag entity allows multiple opportunities to simply store a reference to this entity that contains all the information. Currently, this information stored within the tag entity is no more than a string value representing the name of the tag. This approach pre-emptively supports the addition of more powerful features such as aliasing (discussed in section 4.1.2).

Tagging is an effective way of classifying information and for this reason it is the main driver behind the matchmaking techniques applied to Voluntarily. Due to its effectiveness, tagging has since been added to other entities in Voluntarily such as "activities" (an upcoming feature that enables teachers to create pre-designed opportunities by "content providers" such as OMGTech!) and people (i.e. users) to allow easy discovery of these entities.

4.1.2. Future Tagging Extensions

The tagging implementation resulting from this project is effective for classifying information and provides a lot of value to the users of Voluntarily. However, there are features that were not implemented in this project that could further improve the tagging engine.

Aliasing would allow associating several values with the same tag, giving the system a semantic understanding of the vocabulary and allowing for more powerful classification of information. For example, two opportunities might be classified with the tags "year 12" and "level 2" respectively. As these terms are essentially describing the same concept, they could be aliased as a single tag, ensuring that both opportunities would appear in a search based on either of the terms.

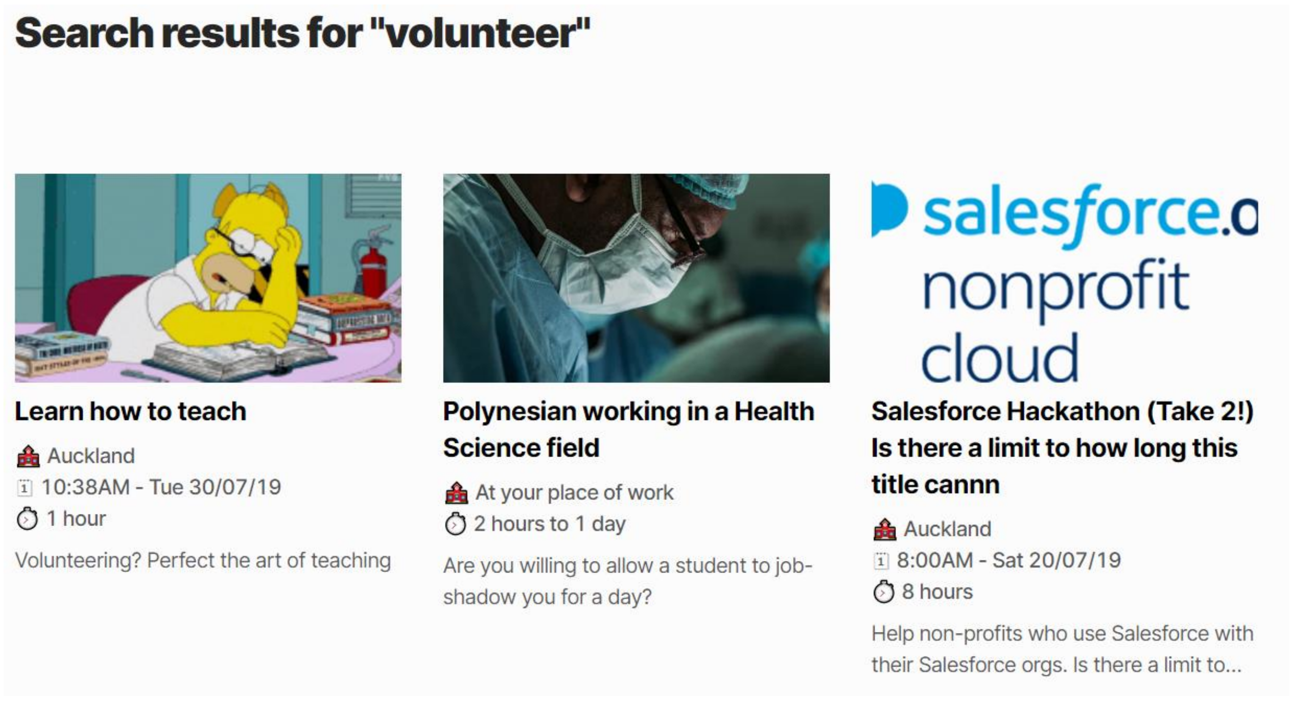
With the current implementation of the tagging engine, there is no easy way to manage the existing tags other than by directly interacting with the Voluntarily database. It would be very useful to provide a webpage that allows an administrator to manage tags through a polished user interface instead of a database client. Tag management could involve creating aliases between terms that are clearly intended to mean the same thing, creating new tags to assist in building the shared vocabulary, deleting tags to avoid offensive terms, or general clean-up.

4.2. Browsing Opportunities

For a volunteer to find an opportunity that they are interested in, they must be able to browse the existing opportunities in such a way that they can specify properties they are looking for. To accommodate for users who wish to find opportunities in this manual way, this project has added searching and filtering functionalities for opportunities in Voluntarily.

4.2.1. Searching

Across all pages in Voluntarily, the header component contains a search bar where users may enter a search term to query the existing opportunities. On confirming this search, the user is directed to a page that shows all opportunities returned in the search results (fig. 3). This search functionality was implemented in the backend of the application to fully utilise the performance of MongoDB queries and to keep the frontend logic as simple as sending an API call containing the search term and rendering the opportunities included in the response.


Search results for "volunteer"

Learn how to teach
Auckland
10:38AM - Tue 30/07/19
1 hour
Volunteering? Perfect the art of teaching

Polynesian working in a Health Science field
At your place of work
2 hours to 1 day
Are you willing to allow a student to job-shadow you for a day?

Salesforce Hackathon (Take 2!) Is there a limit to how long this title cannn
Auckland
8:00AM - Sat 20/07/19
8 hours
Help non-profits who use Salesforce with their Salesforce orgs. Is there a limit to...

Fig. 3. The results page showing matching opportunities after a search is performed

The search API supports searching for a direct term that appears in the opportunity in either its title, subtitle, or description. This ensures that if the search term directly matches one of these properties of an opportunity then that opportunity will be included in the response. The API also treats the search term as an array of keywords, rather than a direct search string. These keywords are matched against tags in the database, then all opportunities are found that include these tags and are added to the response. Implementing the search in this way accommodates users who are trying to find a specific opportunity as well as users who are looking for opportunities based on general keywords.

4.2.2. Filtering

To enhance the search functionality, filtering has been implemented within the search results page so that users can specify further criteria for opportunities they want to view. The currently implemented filters include a date filter and a location filter. As with the search functionality, these filters are implemented within the backend of Voluntarily and are specified as extra query parameters in the API call.

The date filter allows the user to select a specific date, week, month, or date range, giving them plenty of flexibility in further filtering the opportunities shown in the search results. The location filter (fig. 4) populates a selection component with all existing territories and regions within New Zealand, which is the same list that appears when selecting a location in the opportunity creation form. Once a filter is applied, a new request is sent to the search API including the filter values, and the matching opportunities are returned. If a region was selected (a broad location containing territories), opportunities with that region or territories within that region as their location will be included. If a territory is selected, then only opportunities with that territory as their location are selected because a territory is more specific than a region. This approach ensures all relevant opportunities are returned even if the location isn't an exact match, due to the possibility for some locations to be within others.

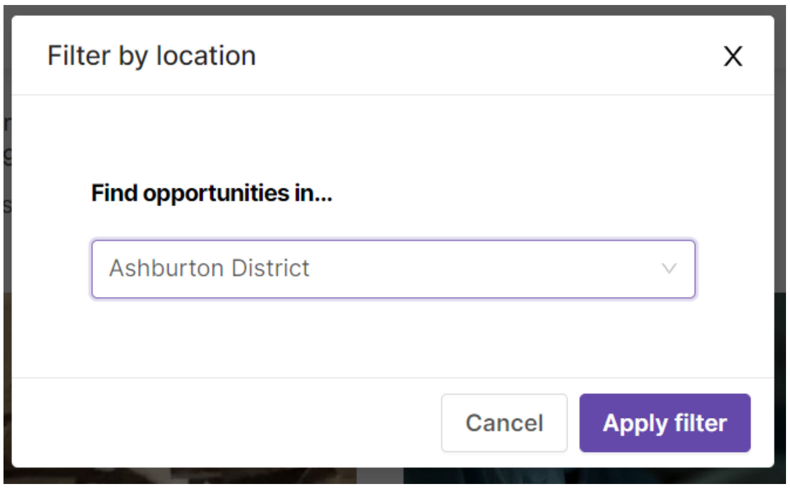


Fig. 4. The location filter selector

4.3. Recommended Opportunities

To require even less effort from volunteers in finding what they need, opportunities deemed relevant to the currently logged in volunteer are displayed to them on their home page. Relevant opportunities are determined based on certain information specified in the user's profile, such as their location and skills. These recommended opportunities are ranked based on how likely they are to meet the volunteer's needs and are presented in this order.

4.3.1. Based on skills

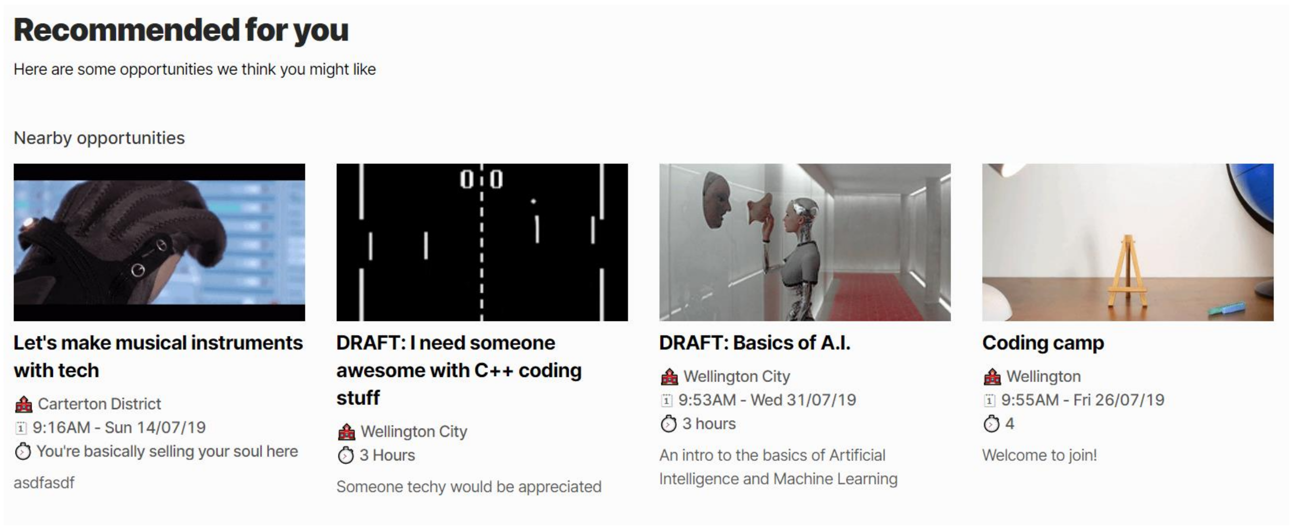
Volunteers can set skills in their profile using terms from the tagging engine as discussed in section 4.1. When generating the recommendations based on skills, the volunteer's skills are compared with the tags associated with each opportunity in the database. All opportunities with at least one matching skill are considered, and the top ten matches are returned based on the number of skills in common with what the volunteer has (fig. 5).

```
getSkillsRecommendations(skills):
  ops = all opportunities from DB with at least
  one tag matching one of the given skills

  for each op in ops:
    op.count = 0
    for each tag associated with op:
      if tag is in skills:
        op.count = op.count + 1
      end
    end
  end

  sort ops with highest counts first
  return first 10 elements in ops
end
```

Fig. 5. Pseudocode for determining relevant opportunities based on skills

4.3.2. Based on location

Volunteers may also set their location in their user profile, allowing the system to recommend nearby opportunities (fig. 6). All opportunities within the region that the user resides in are considered. These opportunities are ranked in order of how specifically they match the user's location, meaning if an opportunity matches at the territory level this opportunity will rank higher than another that matches because it is within the same region (which is a broader area).



Recommended for you

Here are some opportunities we think you might like

Nearby opportunities

Let's make musical instruments with tech
Carterton District
9:16AM - Sun 14/07/19
You're basically selling your soul here
asdfasdf

DRAFT: I need someone awesome with C++ coding stuff
Wellington City
3 Hours
Someone techy would be appreciated

DRAFT: Basics of A.I.
Wellington City
9:53AM - Wed 31/07/19
3 hours
An intro to the basics of Artificial Intelligence and Machine Learning

Coding camp
Wellington
9:55AM - Fri 26/07/19
4
Welcome to join!

Fig. 6. Nearby opportunities recommended based on the user's location

5. Project Experience

Implementing this part IV project during the early stages of Voluntarily's development was interesting due to the rapid pace at which the priorities and the implementation of the product changed. For example, about a month into the project, the Voluntarily code base migrated to a completely new repository using a more powerful framework, Next JS. These rapid changes within Voluntarily led to a decrease in the scope of this project. If an entire matchmaking system was developed on its own without integrating it into the code base frequently, by the time the implementation was complete it would likely no longer meet the needs of Voluntarily and need to be reworked. Having to fit the project scope into the timeline of Voluntarily's development and align the implementation with their goals is the reason why this project focussed on several smaller features contributing to matchmaking in Voluntarily rather than a complete matchmaking system.

The development work was often performed with the Voluntarily team present such as at their hackathons and during the university break at their office. This was very beneficial for ensuring the next steps of the project were aligned with Voluntarily's goals and that they would find them useful in the system. This also enabled frequent feedback on the project's progression and was useful for redefining the scope of the project to ensure value would be provided to Voluntarily by the time the project was complete.

5.1. Hackathons

As Voluntarily is a team that only consists of two paid developers (due to it being a charitable initiative), much of the development work relied on people volunteering their time to work on the project. About once every two months, Voluntarily would host a hackathon where anyone, experienced or not, was welcome to come along and work on some features to add to the platform. These would often take place over a whole weekend, giving plenty of time to make significant progress on major features. Much of the work done throughout the project was completed at these hackathons. We found these very beneficial to productivity due to the presence of the entire Voluntarily team and many other experienced developers who could offer advice and assist with the planning to ensure the implemented features would be useful to the users.

During the timeline of this project, we attended three of these hackathons. The first hackathon was hosted at Datacom and had around fifteen volunteers present to help progress the Voluntarily platform. This first hackathon was very early on in the development stage of Voluntarily and the work done was related to implementing back end functionality to allow volunteers to express interest in an opportunity, which was a necessary addition before we started working on matchmaking features. The work achieved in this first hackathon set the next goal for the project: to make it easier for volunteers to find opportunities so that they can then express their interest. The second hackathon, hosted at the Unleash Space, was arguably the most significant for this project as it is where the tagging engine and a significant portion of the search functionality was implemented. The third and final hackathon was again hosted at Datacom, and our goal for the weekend was to extend the tagging engine so that it could be added to other entities in the system. During this weekend, tags were added to the in-progress “activity” database schema. It was encouraging to see the Voluntarily team wanting to add tags to other entities, confirming the tagging engine’s usefulness to the system.

6. Results & Discussions

6.1. Unit testing

The usefulness of this project’s implementation relies on it behaving correctly so that volunteers can find opportunities that they are interested in. To ensure correctness, unit tests were written for each code change or addition that needed to be added to the Voluntarily code base. These tests were written throughout the course of the project due to all changes needing to go through the review process used by Voluntarily's development team. Changes would not be accepted unless a significant portion of the new code was covered by the test suite, including both front end and back end code. Due to this rigorous review process throughout the project, we are confident that all features behave as expected and that volunteers can successfully find the opportunities they are looking for.

6.2. Performance testing

As well as testing for correctness, performance testing was also carried out to understand how the implementation behaves at scale. Scale is an important factor for this project due to the eventual size of the Voluntarily platform as it is used by more schools around New Zealand. If the implementation quickly becomes unresponsive when a lot of data is stored in the system, the users will become frustrated and this project will have caused more issues than it has solved. The results from this testing will provide an objective way of discussing the efficiency of the implemented features in this project.

6.2.1. Method

The features implemented in this project consist mostly of back end functionalities that are exposed to the front end through an API endpoint. The front end simply sends requests to the server to retrieve certain information and displays it accordingly. For this reason, only the backend code was performance tested. The API endpoints for searching and for generating recommendations were tested, with the number of tags and opportunities stored in the database being changed to see how these affected the response time.

Three main scenarios were tested: the search response time when the search term is five existing tags separated by spaces, and the recommendations response time when the requesting user has 10 skills and 20 skills. The recommendations endpoint was tested with these two scenarios because the response depends on the current user, whereas the search endpoint will return the same response no matter who the requester is. Also, having more skills associated with a user is expected to make the response time longer due to the algorithm used (fig.

5). Testing this endpoint with 10 and 20 user skills will give a good idea of the performance for a typical user and a user with the worst-case scenario (it is thought unlikely that a user will enter more than 20 skills).

Opportunities and tags were generated randomly to setup the data before running each test. For each test, opportunities were associated with a set number of tags, which could be any word in a set of 200 random words. This setup was chosen to simulate as realistic a scenario as possible; limiting the tags to 200 total terms accounts for the fact that when there are many tags in the system, users are far less likely to create new ones. This means with many opportunities, there will always be opportunities sharing the same tags, which is exactly what would happen in the live version of Voluntarily.

Each of the three scenarios were tested with 10, 100, 1000, and 10000 opportunities in the system, where the opportunities were associated with 10, 20, and 30 tags. This resulted in 12 data points for each scenario that allow discussion of how both the number of opportunities and tags in the system affect the performance of these endpoints. The results for the three tested scenarios can be found in appendix A (fig. 7, fig. 8, fig. 9).

6.2.2. Results

As expected, there is a clear trend of increasing response time as the number of opportunities and tags increases. For both the search endpoint and the recommendations endpoint at 1000 opportunities, the response time is well under a second, which is considered acceptable performance. However, at 10,000 opportunities, the response time increases significantly. For the search endpoint, the worst performance at this number of opportunities is when each opportunity has 30 tags, at around 1.7 seconds. The recommendations endpoint's performance degrades much more significantly, with the worst cases being 10.9 seconds and 24 seconds when there are 30 tags for each opportunity and the requester has 10 and 20 skills respectively. This would usually be considered unacceptable performance; however, when accounting for the fact that the recommendations endpoint is only used once on the initial home page load and that it will take some time before Voluntarily will store close to 10,000 opportunities, the performance of the recommendations endpoint is passable.

Overall, the results from the performance testing are acceptable for now given that Voluntarily is still in early stages of adoption and will not yet be pushing this project's implementation to the limit. The fast performance degradation suggests some work is required before the matchmaking features implemented in this project will be suitable for Voluntarily once it reaches many users around the country. The most likely reason for this inefficiency at scale is the array of tag ids stored in the opportunity schema. Using a join table would be far more efficient as database indexing could be taken advantage of and scanning potentially large arrays of tags across many opportunities can be avoided.

7. Conclusions & Future Directions

This project has provided a lot of value to the Voluntarily platform and has significantly sped up its early development process. Through the three primary features of this implementation, Volunteers can easily find suitable opportunities for them to participate in. The tagging engine associates opportunities with keywords that belong to a vocabulary built up over time by the users. Rich searching and filtering functionality allows volunteers to browse opportunities. To involve even less effort from the volunteers, recommendations are presented to them on their home page based on the skills and location set in their user profile.

As many of the features of this project are powered by the tagging engine, future work focussing on this would be beneficial. Section 4.1.2 outlined possible future improvements to the tagging engine which would achieve more powerful information classification and resulting searches and recommendations. The performance testing results discussed in section 6.2.2 suggest that the efficiency of the implemented features for this project should be improved in future to ensure these features can operate at scale as Voluntarily grows.

Finally, while this project has successfully applied matchmaking techniques to assist volunteers with finding relevant opportunities, little work has been done to apply these techniques in the other direction and assist teachers finding volunteers. Adding features to assist teachers would render this project more complete as it would provide value to both the primary users of Voluntarily and be one step closer to a fully-fledged matchmaking system bringing volunteers and teachers together.

Acknowledgements

We would like to thank our supervisor Andrew Meads and the Voluntarily team leads Andrew Watkins and Walter Lim for supporting us throughout this project and making this project a success.

References

[1] T. D. Noia, E. D. Sciascio, F. M. Donini, and M. Mongiello, "A System for Principled Matchmaking in an Electronic Marketplace," vol. 8, no. 4, pp. 9–37, Jul. 2004.

[2] D. Cox, "Matchmaking Systems and Algorithms for Voluntarily", 2019

[3] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace," vol. 5, no. 2, pp. 173–203, Jun. 2002.

[4] S. Agarwal and S. Lamparter, "SMART - a semantic matchmaking portal for electronic markets," 2005, pp. 405– 408.

[5] J. Yan and J. Piao, Towards QoS-Based Web Services Discovery. pp. 200–210.

[6] S. C. (Politecnico di Bari, Bari, Italy, and s.colucci@poliba.it), "A Formal Approach to Ontology-Based Semantic Match of Skills Descriptions," vol. 9.

[7] S. L. Saidman, A. E. Roth, T. Sonmez, M. U. Unver, and F. L. Delmonico, "Increasing the Opportunity of Live Kidney Donation by Matching for Two- and Three-Way Exchanges," vol. 81, no. 5, pp. 773–782, Mar. 2006.

[8] J. Bao and J. Xia, "A hybrid algorithm for service matchmaking based on ontology approach." pp. 2420–2424, 2017.

[9] U. Bellur, H. Vadodaria, and A. Gupta, "Semantic Matchmaking Algorithms."

[10] W. Kondro, G. Vogel, D. Normile, A. Lawler, and D. Malakoff, "Matchmaking," vol. 290, no. 5492, p. 685, Oct. 2000.

Appendix A

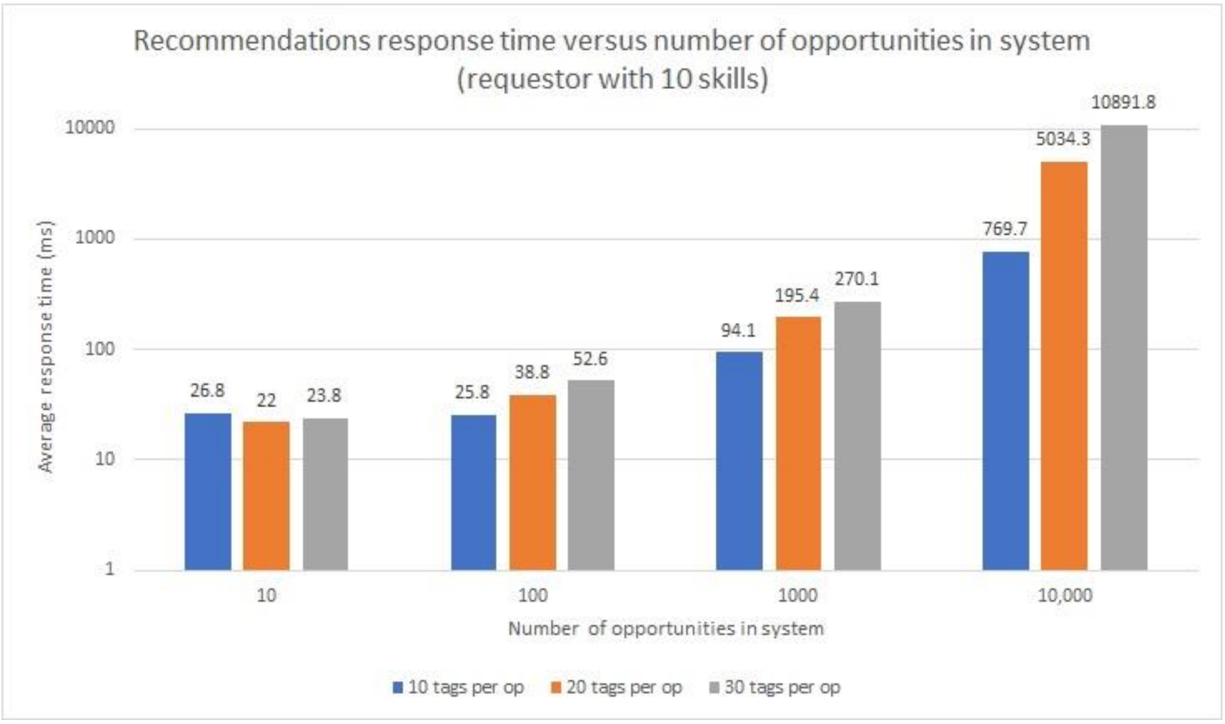


Fig. 7. Graph showing how recommendations endpoint response time changes with data in the system when requester has 10 skills.

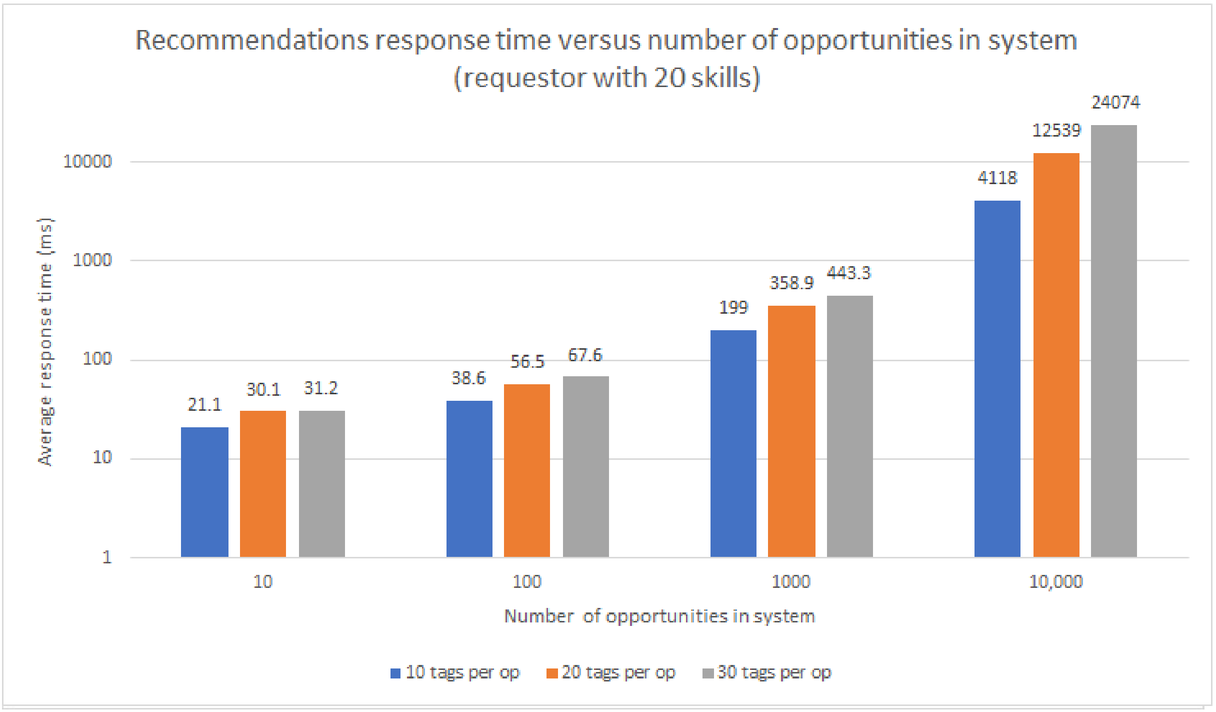


Fig. 8. Graph showing how recommendations endpoint response time changes with data in the system when requester has 20 skills.

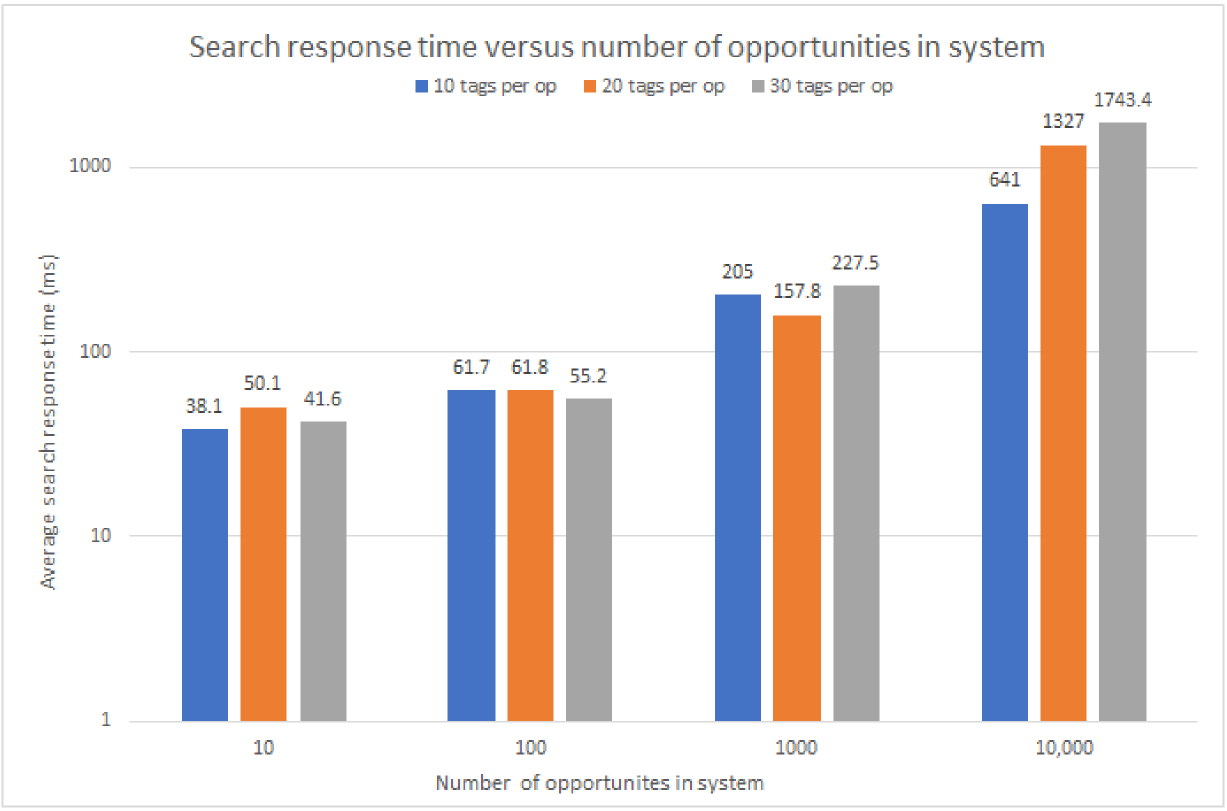


Fig. 9. Graph showing how search endpoint response time changes with data in the system.