PROJECT PROPOSAL – Sample

UNSW SYDNEY

SkillStash

Automated System for Job and Employees Search

Background/Problem Domain

Nowadays the recruitment process is very sophisticated and complex for all participants: potential employees, hiring companies, and recruiters. We have identified major reasons why this is happening in the industry.

High variety of skills

Many people have very diverse experience and often their skills vary in many subjects and fields. While many portals allow candidates to choose ‘up to three’ core skills, some people can count dozens. This issue is very important, as it expands the skillsets to a potentially unlimited number of fields.

Unstructured data in CVs

Most CVs are structured in a personalised way, which makes it very difficult to extract any data from them. It is crucial for employers and recruiters to see the previous experience and educational background in order to judge a candidate’s skills.

Evidence for the skills

Unfortunately, many candidates understand that recruiting companies and employers will not be able to verify the skills of potential employees, so they exaggerate or even falsify their skills. This leads to wasted time during recruitment, and unfair hiring.

Overly specific requirements for candidates

Employers tend to demand a lot from their candidates, and very often requirements are overly specific not only in terms of hard skills, but also in terms of soft skills. This leads to increased difficulty finding a matching candidate.

Recruiters are dependent on their networks

Recruiters have limited professional networks, and have to spent time constantly updating their databases, so they lack time to focus on the quality of candidates and job offers.

Existing Systems and Drawbacks

LinkedIn

LinkedIn is a social network focused on careers and business. Potential candidates can post their work experience and educational background, and add skillsets to their profile. Potential employers and recruiters can post job listings, fill out the profile of their company, and search for candidates. The drawback of the system is that users can only see profiles within the 3rd degree connection – if they are connected with somebody, who is connected with somebody, who is connected with that person. This limits the search outcome a lot.

In addition, the skills that a candidate chooses to put on their profile are not structured: candidate can put any word to be their skill. This way companies or recruiters have difficulty choosing the key words for their searches, considering all possible synonyms and acronyms.

Seek

On this portal users can upload their CV and manually add their education and experience, with an additional feature to add their list of skills, though these skills have no structure and no autosearch. Jobs can be found by city, salary range, and category. Employers cannot choose required proficiency levels for the candidate.

Ribit

On this platform candidates can do the same as on Seek, but with some limitations: even though the skills are categorized, candidates cannot choose more than five of them.

Employers also are limited in their search with the skillsets.

Aim

We are aiming to create an easy to use system that considers the priorities of all participants: candidates, employers and recruiters. The system must gather skills from candidates, offering the choice from a categorized and structured list of options. Employers can post job requirements, based on the skills that they need, the necessary level of proficiency for each skill, and the required experience. This can also be done in a structured way.

Recruiters should be able to use the system as a great search tool for jobs and candidates, be able to mark their proposed candidate to track their performance and be able to prove a referral.

With regards to the technical system requirements, we have chosen three main target criteria to aim for. The system should be:

1) Fast

All types of users shouldn't spend their precious time waiting for searchet results. The system should perform searches fast enough for good usability. Three seconds delay per search is the maximum latency.

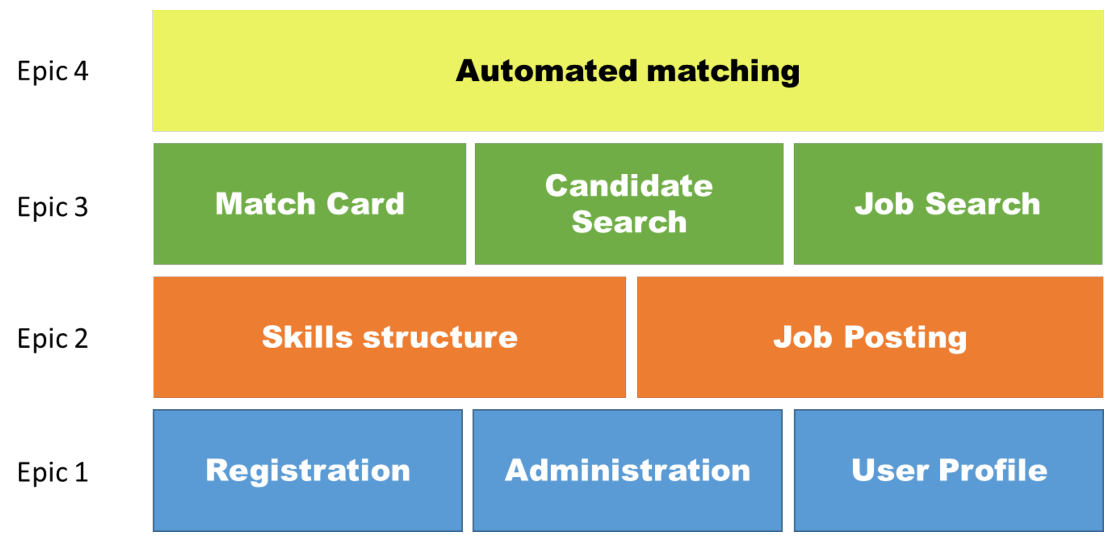
2) Accurate

The system should perform the search with a reasonable accuracy: 90% of returned results should be similar to human search (as the recruiter would have chosen the same profile).

3) Comprehensive

The system's usability and the interface should be intuitive and cause no frustration or misunderstanding for the users. Users shouldn't require more than 20 minutes to fill in their profile, 2 minutes to perform a job search, or 20 minutes to create a job post.

Epics



1) Registration, Admin Backend, User Profile

This epic is aimed at creating a baseline for the future application. The major parts are set up: registration, basic skills selection, user profile, administration function: adding possible skills.

We have chosen the Django REST framework for implementation, as it has many web-browsable APIs, is highly customizable, well documented, and trusted internationally.

This epic will give us experience using the framework, give us an understanding of its limits and capabilities, and give an idea of possible ways to develop the other epics of the project.

Pages to implement:

- Registration (front end + back end)
- Administration (back end)
- User profile (front end + back end)

2) Job Posting, Sophistication of Skills Structure

At this epic skill structure is defined and implemented, along with adding the functionality for employers or for recruiters to post a job.

The front end part will be implemented for most of the pages and existing functionality.

Skill structure is meant to be categorized, the top categories will be fixed and split into following:

- Technical & Knowledge
- Leadership & Management
- Communication
- Analysis & Research
- Personality

Users should be able to enter their estimated proficiency at each skill, and provide evidence to prove that they have the skill. Possible forms of evidence are diplomas/certificates, online certificates, confirmed work experience, books that they have read, training that they have done, or endorsements from other system users.

Both employers and recruiters should have access to the job posting functionality. Each job post should contain:

- Title of Position
- City
- Description
- Contact person
- Salary
- Required skills with the proficiency level and options for proof of the skills

Pages to implement:

- Homepage (Front end + back end)
- Job Posting (Front end + back end)
- Update skills selection (Front end + back end)

3) Match creation for an employer and candidate, candidate search, job search

This epic is supposed to create functionality to find a potential 'match' between an employer and a candidate. This match can be created by any type of user and will be presented as 'created by' that user. Every match must be bound to a job post, and have one candidate. This match can be accepted or declined by any side at any time. The status of the match can be one of the following:

- 'in progress'
- 'declined by candidate'
- 'declined by employer'
- 'hired'

Matches are visible to the employer who posted the job, the candidate in the match, and the creator of the match.

Candidate search is available for recruiters and employers. Candidate search should consider city and a skillset, which requires filling up the 3P fields:

- Priority
- Proficiency
- Provability

By setting the skill priorities, the employer or recruiter can emphasize which skills are critical to the position and which are not. This is especially relevant for soft skills and communication skills. Proficiency will represent the skill level and provability is the way the employer is willing to allow the candidate to prove their skill.

Skills with the highest priority are considered essential, so the search function will only match candidates who possess all of the highest priority skills. For each lower priority skill that the candidate possesses, they will be assigned a score depending on their proficiency and the priority of the skill. The system will select candidates based on their total scores.

Job search will have a similar process structure: even though it will be possible to search by the title of the job, it will be highly recommended to search according to the skills: this way candidates may find opportunities that they had never thought about before. Job search will also include a city field, company field, and salary range.

Pages to implement:

- Match (Front end + back end)
- Job Search (Front end + back end)
- Candidate Search (Front end + back end)

4) Automated matching

At the last epic for each job post the system will filter the possible candidates, and for each candidate the system will offer potential job postings.

These potential matches will use the same interface as the manual ones, but the creator will be the system.

For candidate recommendation:

This will be performed by adjusting the priorities to different ranks and finding the matching candidates according to their skillsets.

For job recommendation:

Once a candidate registered and filled in their skills (updated the skills), the system will scan potential roles in the city and create potential matches, which will be displayed for the candidate. The same matches will appear for the job posting creators, thus, each job post can be updated with newly registered candidates.

Pages to implement:

- Match (Updated back end)

Project Methodology

The project will be implemented using an adapted version of Scrum methodology of software development.

Scrum is one of a selection of agile development methods - which have in common a set of principles that put the main value in flexibility, interaction, and collaboration (*Manifesto for Agile Software Development 2001*).

Scrum utilises the agile development guidelines and produces a procedure for fast and iterative development of a functional product.

Due to the specifics of this being a University project, and not being implemented in a context of a company, there are going to be some minor differences from the traditional Scrum methodology.

Workflow & Tools

The general work progress, backlogs, task breakdown and assignment are being tracked via Trello.

The code is stored in a BitBucket repository, and more development-specific tasks are being tracked and assigned via Issues there.

The Scrum sprint duration is 2 weeks, with a review meeting at the end, and a discussion with the designated mentor.

There is also a weekly group meeting in person, and a group chat on WhatsApp for daily discussions.

Project Timeframe

Week	Date	Epic	Activity
Week 2	31 July		Initiation, group formation, topic chosen
	1 August		First meeting, roles assigned, accounts created, first hands on Django
Week 3	7 August	1	Follow up on first implementations,
	8 August		Team Meeting 2, Epics discussion, proposal discussion
	12 August		Proposal submitted
Week 4	14 August		Epic 1 Finished, testing, tasks for Epic 2 assigned
	15 August	2	Team Meeting 3, Epic 2 Started, follow up on questions
	19 August		Online follow up on progress
Week 5	21 August		Follow up on progress, testing, documenting
	22 August		Team Meeting 4, possible pivots and changes discussion
	26 August		Online follow up on progress
Week 6	28 August		Progress demo
	29 August	3	Team Meeting 5, start of Epic 3, tasks assignment
	2 September		Online Follow up on progress
Week 7	4 September		Testing, discussion on changes
	5 September		Team Meeting 6, Documenting
	9 September		Online Follow up on progress
Week 8	11 September		Progress demo
	12 September		Team Meeting 7, Task assigned for Epic 4
	16 September		Online Follow up on progress
Week 9	18 September	4	Changes discussion, testing Epic 3, Epic 4 start
	19 September		Team Meeting 8, Demo scenario discussion, follow up on progress
	23 September		Online Follow up on progress
Week 10	25 September		Teaching recess

	26 September		Teaching recess
	30 September		Teaching recess
Week 11	2 October		Dry run Demo, testing
	3 October		Team Meeting 9, Report preparation, gathering all the documentation
	7 October		Dry run Demo
Week 12	TBD		Demo
	TBD		Report Submitted

Team

Our team is represented by four postgraduate computer science students with complementary experience. We have most of the required skills covered for the successful project delivery.

Team Member 1

Software developer with more than 20 years commercial experience, mainly developing financial applications. Highly proficient at C++ programming, and also comfortable programming in Python, C#, or Java. Has some knowledge of many other programming languages. Has a solid understanding of common data structures and algorithms. Has a basic understanding of web technologies, but no experience developing web applications.

Team Member 2

Proficient in Python, with prior experience in C/C++ and C#. Some experience in plain PHP with HTML/CSS/JavaScript (as part of several undergraduate degree projects), but none on the production scale. Has written client/server applications in C++. More familiarity with back-end, rather than front-end development.

Highly experienced in Trello, and was therefore assigned the role of Scrum Master for this project.

Team Member 3

Software developer and data analyst with over 5 years of experience in technology. Fluent in Python and SQL; most experience comes from Machine Learning and AI projects. Has experience in developing the back-end of web-applications. Highly efficient in building the methodologies and architecture of systems.

Team Member 4

Has 3 years working experience as Java engineer, familiar with web development. Key skills: HTML/CSS/NodeJS/Vue for front-end, J2EE/Django for back-end side.

References

- *COMP9900 Information Technology Project* Lecture Slides & Materials, 2018
- Manifesto for Agile Software Development, 2001, <agilemanifesto.org>
- LinkedIn, <linkedin.com>
- Seek, <seek.com.au>
- Ribit, <ribit.net>