What's available today:

1. Resumes available in MySQL database. Current volume of Resumes: 4,000. The below attributes are available:
    a. Candidate's current title and company
    b. # of experience in years
    c. Resume in text format
    d. Candidate's location
2. Jobs available in MySQL database. Current volume of Jobs: 100 The below attributes are available:
    a. Job Title
    b. Job Location
    c. Job Description in text format

Overall goals to be achieved:

Need to build a scoring system that can score resumes against each job. This will enable our application to give job recommendations to candidates and hiring managers. The scoring will be based on:

- Overall matching score
- Technical skills matching score (for each technical skill category)
- Soft skills matching score (for each soft skill category)
- Location matching score
- Education matching score
- Job title matching score
- Industry/domain matching score

The above matching will be done against:

1. Jobs with Resumes - this will provide best matched resumes for selected job
2. Resumes with Jobs - this will provide best matched jobs for selected resume
3. Jobs with Jobs - this will provide similar jobs for selected job
4. Resumes with Resumes - this will provide similar resumes for selected resume

For each of the matching criteria above, the scoring will be calculated and stored in a database table for all combinations. For e.g. if we have 100 jobs and 100 resumes, a table will be created as "job_resume_matches" that will have 10,000 rows. The table will have columns such as job_id, resume_id, score, technical_skills_score, soft_skills_score, education_score, job_title_score, industry_score, career_score. Likewise, there will be other tables as "resume_job_matches", "job_job_matches", "resume_resume_matches" with similar attributes.

Training data:

Since we have some data for existing job and resumes that were manually matched, this dataset can be used for training the model.

What's needed:

As part of this engagement, you will be working with us to achieve all the above-mentioned goals. The goals will be executed in a 2-week sprint period. Each sprint will be scoped out and be delivered accordingly.

The job entails building backend processes in Python that can calculate necessary scores and keep on inserting/updating the scoring tables. As new jobs or resumes get added, the processes will pick up the new records and score accordingly. The expectation is that the process will run every 5 minutes and do the necessary processing. The process should be able to score at least 100 jobs and 100 resumes within 5 minutes.

Additional backend processes may be built to train the model. This process can take longer.

Python 3.8 or higher version will need to be used

Apply best AI/NLP/ML model for scoring. **YOU WILL NEED TO SPECIFY WHAT MODEL YOU ARE GOING TO USE AND WHY PRIOR TO BUILDING THE MODELS.**

What's not needed:

- We already have parsed resumes in text format available. So, there is no need to perform any pdf to text or document to text conversion.
- This is purely a backend processing job. So, creation of python program files with appropriate python libraries will be used.
- No user interface needs to be developed
- No API's need to be developed

MILESTONE #1
Duration: 2 Weeks

Goal: Build basic scoring framework for calculating score of a resume against each job

Scope:

- Apply necessary AI/NLP/ML model to generate overall score for combination of jobs and resumes.
- The following combination of key attributes should be considered for scoring:
    - Job title and Candidate title
    - Technical Skills matched between Job description and Resume
    - Location matched between job and resume

- Create a background process that takes all the jobs and for each job do a scoring calculation with each and every resume and store the results into “job_resume_matches” table. The “job_resume_matches” table will have job_id, resume_id, overall_score and created_date attributes. This job should calculate score and create/update records in this table. The score value should be stored between 0 and 100 - where 0 being least match and 100 being most match.
- Create background process to train the data

Sign-off criteria:

- Appropriate programs are provided (scoring and training) with description on how to run each program
- Upon running the program, the program should populate “job_resume_matches” with necessary fields
- The program should create all combinations of jobs and resumes for eg. if during the time of execution, the database has 100 job records and 5,000 resumes, the “job_resume_matches” should have 500,000 records.
- The overall score for each record should be between 0 and 100
- The accuracy of scoring should be reasonable enough. It will be optimized in later phases.
- The program should be able to process new jobs and resumes and calculate scores for new records only.

MILESTONE #2
Duration: TBD

Goal: Update the overall score (job to resume matches only) based on additional parameters such as Education, Experience, Soft Skills and Industry/Domain

MILESTONE #3
Duration: TBD

Goal: Calculate individual score (job to resume match only) for Job Title, Location, Technical Skills, Soft Skills, Education, Experience, Industry/Domain

FUTURE MILESTONES

- Perform similar scoring as above between each Jobs
- Perform similar scoring as above between each Resumes
- Perform similar scoring as above between Resumes and Jobs
- Extract technical skills from resumes and rank them (classification)
- Extract soft skills from resume and rank them (classification)
- AI based recommended keywords for Jobs and Resumes

APPENDIX

Technical Skills

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
statistics	**machine learning**	**deep learning**	**nlp**	**data engineering**	**databases**	**web**	**programming**	**blockchain**	**operating system**	**virtualization**	**ecommerce**	**crm**	**erp**	**salesforce**	**analytics**	**big data**
statistical models	linear regression	neural network	nlp	aws	mysql	html5	c#	bitcoin	microsoft windows	vmware	magento	salesforce	netsuite		excel	hadoop
statistical modeling	logistic regression	keras	natural language processing	ec2	oracle	css	java	ripple	macos	vsphere	prestashop	siebel	oracle financials		tableau	mapreduce
probability	k means	theano	topic modeling	redshift	sql server	django	c++	ethereum	linux	hyper-v	joomla	servicenow	sap		sas	spark
normal distribution	random forest	face detection	lda	s3	postgres	node.js	javascript	bitcoin cash	unix	qemu	opencart	zoho			spss	pig
poisson distribution	xgboost	neural networks	named entity recognition	docker	nosql	laravel	python	monero		oracle vm	woocommerce	oracle cloud			d3	hive
survival models	svm	convolutional neural network	pos tagging	kubernetes	hbase	react	php	litecoin		virtualbox	shopify	sap			saas	shark
hypothesis testing	naive bayes	recurrent neural network	word2vec	scala	cassandra		objective-c	blockchain		xen					pandas	oozie
bayesian inference	pca	object detection	word embedding	teradata	mongodb		ajax								numpy	zookeeper
factor analysis	decision trees	yolo	lsi	google big query	rdbms		asp.net								scipy	flume
forecasting	svd	gpu	spacy	lambda	bigquery		ruby								spss	mahout
markov chain	ensemble models	cuda	gensim	emr	firebase		golang								spotfire	etl
monte carlo	boltzman machine	tensorflow	nltk	hive			perl								scikit	
		lstm	nmf	hadoop			bash								splunk	
		gan	doct2vec	sql			r								power	
		opencv	cbow	azure			matlab								h20	
		aml	bag of words	heroku			scala									
		scik-it-learn	skip gram													
			bert													
			sentiment analysis													
			chat bot													