

Job Seeker Portal Report

Yashawant Parab(11012130)
Aravindhana Dhanaraj(11012025)
Siddharth Bammani(11011885)
Sandeep RK(11011854)
Sreenath Gopi(11012060)

1. Introduction

Being in a digitized environment, recruitment should be time-saving, cost-effective and at the same time should search out the qualified candidates across the globe. An aspirant can see things not only in terms of the job but in terms of role, skill, location, company and wage. Quick safe and easy to use. Finding potential candidates from the employers perspective.

Technologies Used:

- MongoDB
 - Document-Oriented NoSQL Database.
 - Simple & High Performance.
 - Uses JSON Format to Store Data. JSON is a lightweight data-interchange format.
- Neo4J
 - High Performant Read and Write Scalability
 - Easy to learn and use.
- Python
 - Easy to Expand Existing Data.

2. Organisation

Development Phase	Nov	Dec	Jan	Feb	Mar	Involved
User Story Identification						Yashawant Parab, Aravindhan Dhanaraj, Siddharth Bammani, Sandeep RK, Sreenath Gopi
Requirement Gathering and Analysis						Yashawant Parab, Aravindhan Dhanaraj, Siddharth Bammani, Sandeep RK, Sreenath Gopi
Coding						Yashawant Parab, Siddharth Bammani
Implementation & Deployment						Yashawant Parab, Siddharth Bammani

Content Generation					Yashawant Parab, Aravindhan Dhanaraj, Siddharth Bammani, Sandeep RK, Sreenath Gopi
Report Creation					Yashawant Parab, Aravindhan Dhanaraj, Siddharth Bammani, Sandeep RK, Sreenath Gopi
Presentation Creation					Yashawant Parab, Aravindhan Dhanaraj, Siddharth Bammani, Sreenath Gopi

Activities:

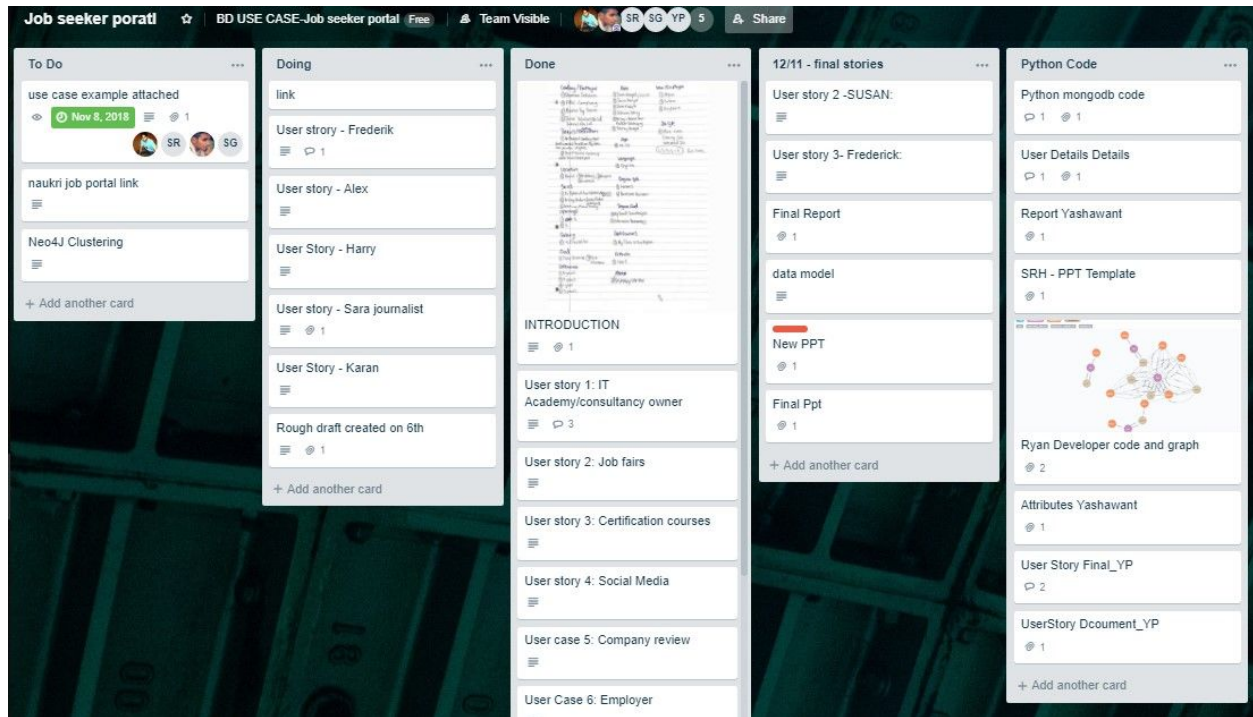
The team is divided into two parts initially. Siddharth and Yashwant were taken responsibility of development of the data model. Everyone else is involved in rest of the activities.

1. Development (Coding & Implementation)
2. Requirement Gathering & Analysis (Content & Tool Identification)

Tools Used:

Weekly two meetings took place. Mainly the meeting is scheduled via whatsapp group. Content and other activity updates were added to the trello board. Google drive is used for editing documents online by multiple contributors. Also project resources were shared between people via google drive.

1. Trello
2. Whatsapp
3. Google Drive



Trello Used For Tracking Task Status and Updates

3. Use case in detail (By Sreenath Gopi)

For this project, we have come up with 5 different user stories which explain different scenarios from a job seeker's perspective and an employer's perspective.

Following user stories are taken to design the data models.

USER STORY 1:

"ABC analytics located at XYZ is looking for talented people to be part of their company, they have 10 openings for Data engineers. They seek a job portal to hire people conveniently."

Explanation:

"ABC company is an analytics company that helps retailers transform big data into valuable insights. It is located at xyz location and has 10 openings for data engineers. They are specifically looking for talents which have data engineering skills and experience.

Requirements deduced :

- Company/Employer

-
- Company Description
 - Location
 - Skills
 - No of Openings
 - Field
 - Experience
 - Role

USER STORY 2:

“Ryan an engineering graduate and a fresher wants to know about job fairs being held near his location, so that he can prepare prior hand and manage his schedule to attend that fair.”

Explanation:

Ryan just completed his graduation in computer engineering and he is looking for job opportunities to kick start his career. He is aware that many job fairs take place in and around his location. Since it is not feasible for him to attend all the job fairs he would like to filter out the job fairs which are happening near to his location. In this way, he would be able to plan and attend the maximum number of job fairs effectively.

Requirements deduced:

- Job fair Location
- Distance to Location
- Companies attending the fair
- Vacancies in the company
- Skills required for the vacancy

USER STORY 3:

“Susan an experienced Business intelligence Analyst in SAP is looking for certification courses related to her profession which would help her for a promotion.”

Explanation:

Susan has 6 years of experience in Business Intelligence and is working in SAP currently. In order to get a promotion from her current role, she finds herself in a situation where she would require to get certified in BI courses. She hopes this would land her a promotion or better job opportunities in a different company.

Requirements deduced:

- Field
- Years of Experience
- Role
- Certification courses
- Current company
- Employers
- Skills

USER STORY 4:

“Frederik a bachelor degree holder on Information technology who received an interview call from ABC a software testing company. He wants to learn more about the company and job role so that he is well prepared and gets the job.”

Explanation:

Frederik is a graduate in Information Technology is looking for a software testing job. After many tries, he was called up for an interview for the role of software tester by a company named ABC. He has experienced failure in his previous interviews due to the lack of knowledge about the company and the job role. He is seeking a job portal where the details of the company and job role are available to him directly through his application status. In this way, he expects to know more about the company and job role by the time of his interview.

Requirements deduced:

- User name / User ID
- Application ID
- Job ID
- Company Name

- Company Details
- Job Description

USER STORY 5:

"As a manager at Local Toy Store, I noticed so many kids coming along with parents during weekends and evening hours. We need 3 part-time employees who are having minimum 1 year to maximum 3 years of experience in any of the following positions - Billing section, Product demo section, Product cataloguing section. The pay will be €10 per/hour during weekdays and €14 per/hour during weekends. We require profile/resume of employees between the age of 18-25 who speaks and writes English."

Explanation:

A manager in a toy store is looking for 3 part-time employees. He would prefer employees aged between 18-25 since he feels they would be able to communicate and handle things smoothly with the type of customers he expects. Being in a city with a lot of diversity in the language he would like to have employees who can read and write English well. Also, he expects them to have experience of working in the billing section or product demo section or, product cataloguing. A reasonable pay of 10€/hour on weekdays and 14€/hour on weekends are offered.

Requirements deduced:

- Age
- Experience
- Languages spoken
- Expected wage
- Location

4. Databases(By Sandheep RK)

For Jobseeker portal project, we have used 2 databases.

1. Graph Database -> Neo4J
2. Document Database -> MongoDB

Graph Database focus on relationships. The best way to picture a graph store is like a mathematical graph where you have edges and vertices. With a graph store data is stored as Nodes and Relationships. A Node is basically a noun, a person, place, thing, entity, etc. A relationship is a one- or two-way connection between two nodes. A node could be people and a relationship could be a two-way friendship.

We can usually apply metadata documents to nodes and relationships, as well as labels. We can label a node. We can add data to the nodes, it's schema-less and very flexible. Nodes store data about each entity in the database, Relationships describe a relationship between nodes, and a Property is simply the node on the opposite end of the relationship. Whereas a traditional database stores a description of each possible relationship in foreign key fields or junction tables, graph databases allow for virtually any relationship to be defined on the fly.

Document Stores store data in "documents", typically XML or JSON documents. They're typically schemaless, so each document can contain any data that you want them to have and you can change it on the fly. Documents contain key-value pairs, which can be any sort of value, array, or even another document.

This is mainly for schema-less, JSON/XML semi-structure storage. It's still KV store, but it adds the collection concepts to have better hierarchy organization of the data, and the value of the object is usually in JSON format.

In our Project, the following User stories use the graph database.

User Story 2: For this User Story, We can see that we have data which can be stored as Key Value Pair. So the option is to either go for **Key-Value database** like Redis or **Graph database** like Neo4J.

The key-value database does not manage complex relationships. If we just need to store fairly static data and don't have to do much processing on it, a simple key-value database could be the way to go.

But In this user story, we have relationships to be built between the Jobseekers and the Job fairs held in and around the city. Again, the relationship needs to be built between the Companies and the job fairs in which they are participating. So, we have nodes User with the properties such as Name, Degree, Qualification etc and Job fairs with the properties such as City, Distance etc. **Therefore**, going for the Graph Database seem to be the fair choice to go with.

For the rest of the User Stories (1,3,4,5), We have Used Document Database.

For User Story 1, Document database is an obvious choice because there is a key-value pair between the Run-time Infotech and the Jobseekers but the properties and the

requirement of the Jobseekers had to be stored in documents and Job details and the company details also had to be stored in the document as to fetch the required list of candidates on the fly using query based on the job requirements of the company.

For User Story 3, Like User Story 1, the data of the Jobseeker and the data of the company providing the certification courses have to be stored in the collections for accessing the desired result about the certification courses based on the requirement of the user which is Jobseeker's data.

For User Story 4 and 5, There are data related to Users, Companies, Job details, Company details, distance, preferred language, Salary, Work timings etc and all of these data had to be stored as a document to build the relationship between the collections and query the data.

Therefore, Document Database was the most certain choice of database for the above User Stories.

5 Data models (By Yashawant Parab and Siddharth Bammani)

5.1 Mapping of requirements to data models:

To interact with Embedded documents and Graph Relationship we used MongoDB and Neo4j Databases for our user stories.

The key decision to design the data model in MongoDB application revolves around the structure documents and how the relations represents between data. But Since MongoDB does not support joins between two collections or separate documents, we are using graph data model as well. Nowadays, most of the data exists in the form of the relationship between different objects and more often, the relationship between the data is more valuable than the data itself. Neo4j provide features like transactional integrity and operational availability. MongoDB is a schema less document-oriented storage database in which one collection holds different documents. Whereas in Neo4j provide cypher query language to represent the graph visually.

In MongoDB the data is denormalized and stored with related documents to remove the need of Joins. For this MongoDB use manual and DBRefs methods to relating documents.

In Neo4j follows as ACID properties and is transactional database which makes the database as scalable and reliable i.e., we can scale the database by increasing the number of reads/writes, and the volume without effecting the query processing speed and data integrity.

We Use Manual Referencing method where `_id` (MongoDB creates a unique index on the `_id` field during the creation of a collection()) field change as per the user story. The key use of this manual referencing is, we can use first document as a reference in the second document.

According to the User Stories, we use Embedded document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. Array fields for example; Organization Job Details, Open Positions in Toy Store, Locations where companies are situated are stored in different collections with the flexible `_id` field for easy referencing. The `_id` field act as a primary key in document-based database.

For the second User story we use graph database, because the relation between the object entities are useful rather the object itself. The data objects are created as nodes, the properties are mentioned in the data nodes and the relationship between the nodes are retrieved to find the results.

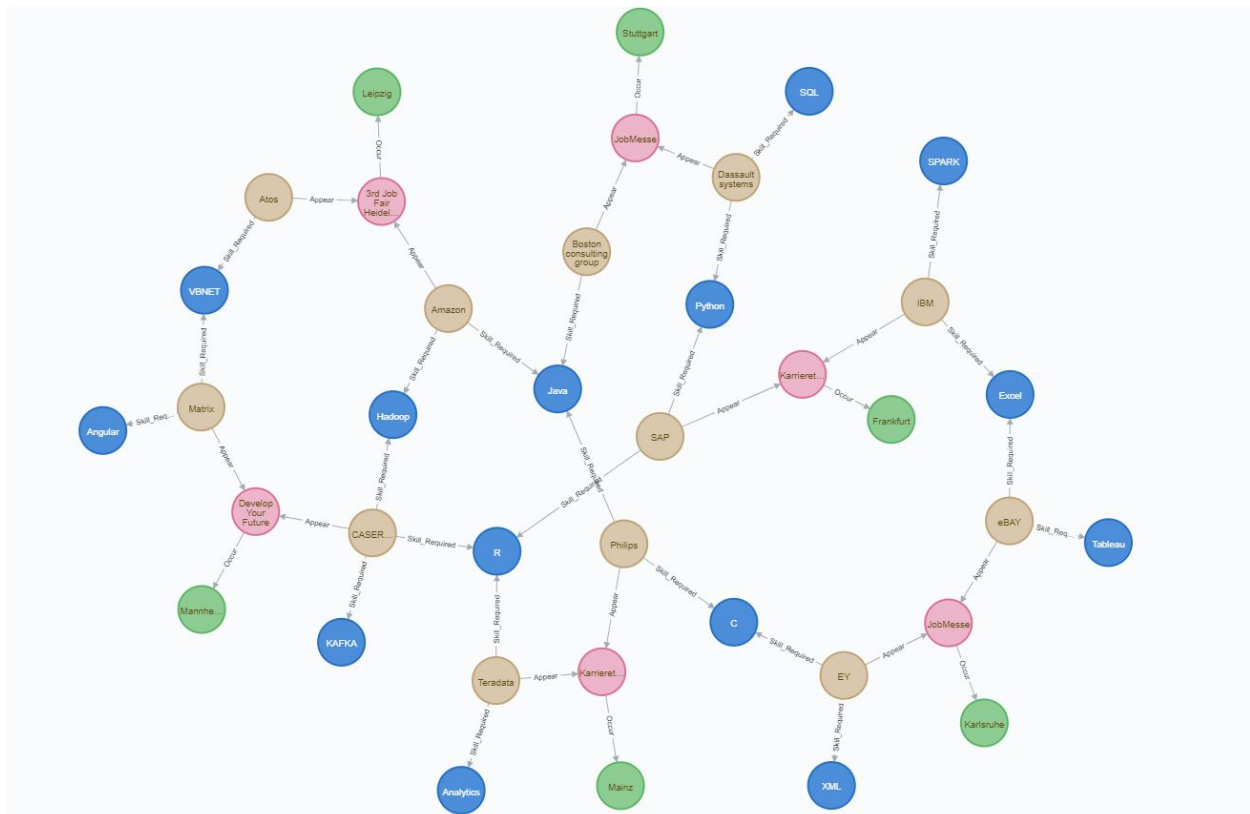
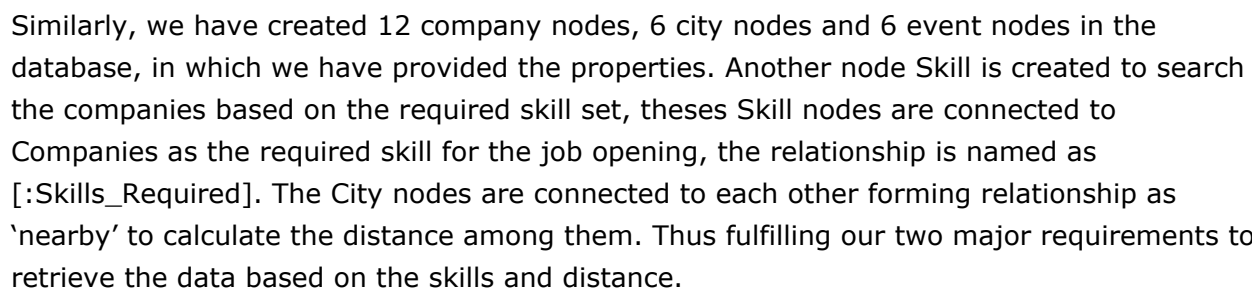
In this Job Portal, we have fields like Gender, Application Status which also modeled by using the key value database which is Redis.

5.2 Graph data model (By Siddharth Bammani)

We are using Neo4j to model the data in the form of graph, the nodes of the graph depict the entities while the relationships depict the association of these nodes. The graph data model represents the data in nodes, relationships and properties.

We are creating three main categories of nodes i.e. companies, job fair events and the cities, in company node we are passing the company details and job opening details and location as property, in city node we are mentioning the properties event, location, longitude and latitude, and in Event node we are mentioning the city name, location, date of event and Name of the event as property.

Below graph represents the relation between the nodes, here the companies naming IBM and SAP are appearing in the Job fair event which has label 'Karrieretag' and it is going to occur in Frankfurt city.



5.3 Document data model (By Yashawant Parab)

MongoDB use data structure with name-value pairs like in JSON. For example, Employee object has attribute First_Name, Email_id, Skills where Skills is a list of details.

Document for Employee in MongoDB will be like:

```
{
    _id : "User_001"
    First_Name : "Albert",
    Email_id :
    "albert.einstein@rocketmail.com",
    Skills : ["Physics", "Mathematics "]
}
```

Also, excessive JOINS can be avoided by saving data in form of Array and Documents (Embedded) inside a document.

Document for Application Status and Organization data in MongoDB :

In Organization collection we stored all Employers details with the number of openings as well as the job descriptions. The employee application status is stored in application collection where application collection contains User data, organization data and the employee application status. Employee application rejection and acceptance is indicated after the employee got rejection or acceptance from the organization.

Also, as per the Toy store user case we have created two different collections as part time job seeker and the toy store. Where toy store manager is looking for the candidates with specific requirements. This search query performs by using lookup aggregation query on part time job seeker to get the candidate data according to the store manager requirement.

All the data fixed in the document manner with the JSON serialization format, which is main advantage of No-SQL database over SQL database. A single write operation can insert or update the data for an entity. MongoDB based on the Atomicity property, where no single write operation affects more than one collection.

6. Implementation (By Yashawant Parab and Siddharth Bammani)

We perform Integration mainly into two databases MongoDB and Neo4j. Also, we use Python and JSON formatter to input dummy values in database.

User Story: 1 (By Yashawant Parab)

We use \$match aggregate function.

In this user story employer search for the candidate who has at least 5 years of experience in data science and, he is good in programming language skills like python and R.

Python Script:

To establish a connection to MongoDB with PyMongo, we use the MongoClient class. All the user details fill into db.UserDetails collection.

```

5 @author: yash
6 """
7
8 from pymongo import MongoClient
9 from getpass import getpass
10 try:
11     conn = MongoClient()
12     print("Connected successfully!!!")
13 except:
14     print("Could not connect to MongoDB")
15
16 db = conn.Job_Portal_Group
17 collection = db.UserDetails
18

```

To input the data, we use python input script:

```

20 #User Credentials Database
21 User_Name = input("Enter Your User_Name: ")
22 Password = getpass("Enter Your Password: ")
23 First_Name = input("Enter Your First_Name: ")
24 Last_Name = input("Enter Your Last_Name: ")
25 Email_id = input("Enter Your Email Address: ")
26 Phone_Number = input("Enter Your Phone_Number: ")
27 Date_of_Birth = input("Enter Your Date_of_Birth: ")
28 Place_of_Birth = input("Enter Your Place_of_Birth: ")
29 Experience = input("Are you a Fresher or Professional: ")
30 Highest_Degree = input("Enter Your Highest Qualification: ")
31 Degree_Course = input("Enter Your Course Name: ")
32 University = input("Enter Your University Name: ")
33 Passing_year = input("Enter Your Year Of Passing: ")
34 Current_Company_Name = input("Enter Current Company Name: ")
35 Current_Designation = input("Enter Current Designation: ")
36 Years_Of_Experience = input("Enter Your Years Of Experience: ")
37 Current_Job_Location = input("Enter Your Current Job Location: ")
38 Current_Job_Salary = input("Enter Your Salary range : ")
39 Expected_Job_Salary = input("Enter Your Salary Amount You Expect : ")
40 Skills1 = input("Select skills: C++, Python ,JAVA, SQL, MySQL: ")

```

Data in MongoDB is represented (and stored) using JSON-style documents. To insert a document into a collection we use `insert_many()` method. After inserting the document, the list of collection has been created on the server.

```

45 collection.insert_many(
46     [
47         {
48             "UserName" : User_Name,
49             "Password" : Password,
50             "First_Name" : First_Name,
51             "Last_Name" : Last_Name,
52             "Email_id" : Email_id,
53             "Phone_Number" : Phone_Number,
54             "Date_of_Birth" : Date_of_Birth,
55             "Place_of_Birth" : Place_of_Birth,
56             "Experience" : Experience,
57             "Highest_Degree" : Highest_Degree,
58             "Degree_Course" : Degree_Course,
59             "University" : University,
60             "Passing_year" : Passing_year,
61             "Current_Company_Name" : Current_Company_Name,
62             "Current_Designation" : Current_Designation,
63             "Years_Of_Experience" : Years_Of_Experience,
64             "Current_Job_Location" : Current_Job_Location,
65             "Current_Job_Salary" : Current_Job_Salary,
66             "Expected_Job_Salary" : Expected_Job_Salary,
67             "Skills" : [Skills1, Skills2]
68         },
69     ])

```

User Story 2 (By Siddharth Bammani)

We are using Neo4j for this user story, the relations between the company and the user can be implemented more effectively. The user Ryan is looking for job fairs being held near the cities where he lives and can also get the company details as well as look for the job openings for that fair.

We are creating three main categories of nodes i.e. companies, job fair events and the cities, in company node we are passing the company details and job opening details and location as property, in city node we are mentioning the properties event, location, longitude and latitude, and in Event node we are mentioning the city name, location, date of event and Name of the event as property.

Creating City nodes:

```
CREATE (Heidelberg:City {City_Name: "Heidelberg", Event: "Job Fair"})
```

Adding in latitude and longitude as properties in the existing city nodes, below is just for one city similarly adding it for the rest of the cities:

```
MATCH (n:City)
```

```
WHERE n.City_Name = 'Mannheim'
```

```
SET n.latitude = 49.4875
```

```
SET n.longitude = 8.4660
```

```
RETURN n
```

Creating Company nodes:

Below mentioned cypher query is just for one company similarly we have created 12 company nodes.

```
CREATE (SAP:Company {Company_Name: 'SAP', Description: 'Provides the technical business analysis for software solutions for Equities and Equities Derivatives, Fixed Income, FX applications used by Investment Banks. Collaborates with business, development, quality assurance, and project/program management technology stakeholders towards the development of legacy and mobile applications', Position: 'Business Analyst', Openings: '4', Location: 'Frankfurt', Skills: ["R", "Data manipulation", "Statistics", "java", "NoSQL"], Experience: '1', Salary: 'NA'})
```

Creating Event Nodes:

Below mentioned cypher query is just for one event similarly we have created 6 event nodes.

```
CREATE (JobFair1:Event {Name: "Develop Your Future", Location: "Mannheim", Event: "Job Fair", Date: "06-02-2019", Specialization: "Science, Informatics, Engineering"})
```

Creating Skill node:

```
CREATE (R:Skill {Skill_Name: "R"})
```

```
CREATE (Python:Skill {Skill_Name: "Python"})
```

We are creating the data model such that the user can search for jobs and jobs fairs based on the skills and the distance.

We have created the relations between the cities, events and company. Below are the queries for creating the relations.

Create relation between Job fair event and Company:

```
MATCH(a:Event{Location:"Frankfurt"}),(b:Company{Company_Name:"SAP"})
```

```
CREATE (b)-[:Appear]->(a)
```

Create relation between City and Event:


```
MATCH(a:City{City_Name:"Leipzig"}),(b:Event{Location:"Leipzig"})
```

```
CREATE (b)-[:Occur]->(a)
```

User Story 3 (By Yashawant Parab)

This user story contains the services which Job Portal is providing to candidate. User story mainly depend on the course certification depending on the specific domain. We use JSON format insert query form MongoDB console to build data into database.

```
> db.Services.insertMany([ {"_id" : "Service_001", "Course_Domain" : "Big Data", "Course_Certification" : "Cloudera Certified Professional", "Cost" : "300Euro"}, {"_id" : "Service_002", "Course_Domain" : "Big Data", "Course_Certification" : "Intellipaat Big Data Hadoop Certification", "Cost" : "380Euro"}, {"_id" : "Service_003", "Course_Domain" : "Big Data", "Course_Certification" : "Microsoft's MCSE in Data Management and Analytics", "Cost" : "350Euro"}, {"_id" : "Service_004", "Course_Domain" : "Big Data", "Course_Certification" : "Hortonworks Hadoop Certification", "Cost" : "410Euro"}, {"_id" : "Service_005", "Course_Domain" : "Big Data", "Course_Certification" : "MongoDB Certified Developer Exam", "Cost" : "450Euro"}, {"_id" : "Service_006", "Course_Domain" : "Big Data", "Course_Certification" : "EMC Data Science and Big Data Analytics Certifications", "Cost" : "470Euro"}, {"_id" : "Service_007", "Course_Domain" : "Big Data", "Course_Certification" : "SAS Certified Data Scientist", "Cost" : "255Euro"}, {"_id" : "Service_008", "Course_Domain" : "Big Data", "Course_Certification" : "Data Science Council of America Certification", "Cost" : "300Euro"}, {"_id" : "Service_009", "Course_Domain" : "Data Science", "Course_Certification" : "Cloudera Certified Professional: Data Scientist (CCP: DS)", "Cost" : "135Euro"}, {"_id" : "Service_010", "Course_Domain" : "Data Science", "Course_Certification" : "EMC Data Science Associate", "Cost" : "220Euro"} ])
```

User Story 4 (By Yashawant Parab)

Based on the employer and employee section, employer can post the open positions from various domain where as employee can search suitable job position according to the Job skills, Company domain, salary, location and apply via job portal. In this user story Frederik has applied various company and he received an interview call from Capgemini company, and he wants to learn about the company for interview.

We have created Two different collections where Organization collection has all organization data including job positions, job description, Company description and Application Collection has Candidate application status where he/she selected or not.

Organization Collection:


```
> db.Organization.insertMany( [ { "Organization_Name" : "TCS", "Location" : ["Frankfurt", "Heidelberg", "Munich", "Berlin"], "Open_Position" : "Data Analyst", "Job_id_Org" : "Job001", "Job_Details" : [ { "Required_Skills" : ["Python", "R"], "Desired_skillSet" : ["Mathematics", "PowerPointPresentation"], "About_Position" : ["Good In Data Science", "Expert in Statistics"], "Salary" : "7lack", "Work_Location" : ["Heidelberg", "Frankfurt", "Munich", "Berlin"], "Employment_Type" : ["Working Student", "Part-Time", "Internship", "Full-Time"], "Min_Experience" : 5, "Company_Description" : "Tata Consultancy Services Limited (TCS) is an Indian multinational information technology (IT) service, consulting company headquartered in Mumbai, Maharashtra. It is part of the Tata Group and operates in 46 countries. TCS is one of the largest Indian companies by market capitalization.", "Job_Description" : "Analytical Skills: Data analysts work with large amounts of data: facts, figures, and number crunching. You will need to see through the data and analyze it to find conclusions. Communication Skills: Data analysts are often called to present their findings, or translate the data into an understandable document. You will need to write and speak clearly, easily communicating complex ideas. Critical Thinking: Data analysts must look at the numbers, trends, and data and come to new conclusions based on the findings. Attention to Detail: Data is precise. Data analysts have to make sure they are vigilant in their analysis to come to correct conclusions. Math Skills: Data analysts need math skills to estimate numerical data." } ] }, { "Organization_Name" : "IBM", "Location" : ["Bremen", "C
```

Application Collection:

```
> db.Application.insertMany( [ { "UserName" : "JoyMoni", "Job_id_Org" : "Job001", "Application_Status" : "Reject" }, { "UserName" : "Thomas_Lofar", "Job_id_Org" : "Job002", "Application_Status" : "Accept" }, { "UserName" : "PranavThamboli", "Job_id_Org" : "Job003", "Application_Status" : "Interview Call" }, { "UserName" : "RahulS", "Job_id_Org" : "Job004", "Application_Status" : "Reject" }, { "UserName" : "Johnson", "Job_id_Org" : "Job005", "Application_Status" : "Waiting" }, { "UserName" : "Frederik", "Job_id_Org" : "Job016", "Application_Status" : "Interview Call" }, { "UserName" : "Siddhesh_Marathe", "Job_id_Org" : "Job006", "Application_Status" : "Accept" }, { "UserName" : "Piyush", "Job_id_Org" : "Job007", "Application_Status" : "Accept" } ] )
```

User Story 5 (By Yashawant Parab)

For part-time students and local stores, Job portal has separate section where part time student can apply separately to specific part time job and store can post their vacancy with the short job-part time job requirement. For this user story we have create two different collection names as PartTimeJob and Store

PartTimeJob:

```
> db.PartTimeJob.insertMany([ { "_id" : "PT001", "Name" : "Yashawant", "Employee_Age" : 25, "Speaking_Language": ["English","German","Hindi"], "Writing_language" : ["English","Hindi"], "Experience" : 1, "Expected_Salary" : 10 }, { "_id" : "PT002", "Name" : "Sid", "Employee_Age" : 18, "Speaking_Language": ["English","German"], "Writing_language" : ["English","German"], "Experience" : 2, "Expected_Salary" : 14 }, { "_id" : "PT003", "Name" : "Kumar", "Employee_Age" : 30, "Speaking_Language": ["English","German","Hindi"], "Writing_language" : ["English","Hindi"], "Experience" : 3, "Expected_Salary" : 15 }, { "_id" : "PT004", "Name" : "Sundar", "Employee_Age" : 15, "Speaking_Language": ["English","German","Hindi"], "Writing_language" : ["English","German","Hindi"], "Experience" : 6, "Expected_Salary" : 11 }, { "_id" : "PT005", "Name" : "Piyush", "Employee_Age" : 19, "Speaking_Language": ["English","German"], "Writing_language" : ["English","German"], "Experience" : 2, "Expected_Salary" : 12 }, { "_id" : "PT006", "Name" : "Bhushan", "Employee_Age" : 24, "Speaking_Language": ["English","German","Hindi"], "Writing_language" : ["English","Hindi"], "Experience" : 3, "Expected_Salary" : 10 }, { "_id" : "PT007", "Name" : "Valentina", "Employee_Age" : 23, "Speaking_Language": ["English","Hindi"], "Writing_language" : ["English","Hindi"], "Experience" : 2, "Expected_Salary" : 10 } ] )
```

Store Collection:

```
> db.ToyStore.insertMany( [ { "_id": "TD001", "Store_Name" : "TKMax", "Required_Employers" : 3, "Open_P
osition" :{ "Billing_Section" : 1, "Product_Demo_Section" : 1, "Product_Cataloguing": 1}, "Salary": 10 ,
"Required_Experience" : 3,"Required_Language": "English" }, { "_id": "TD002", "Store_Name" : "Game Shop
", "Required_Employers" : 2, "Open_Position" :{ "Billing_Section" : 0, "Product_Demo_Section" : 1, "Pro
duct_Cataloguing": 1}, "Salary": 12, "Required_Experience" : 2, "Required_Language": "German"}, { "_id":
"TD003", "Store_Name" : "Hamley", "Required_Employers" : 1, "Open_Position" :{ "Billing_Section" : 0,
"Product_Demo_Section" : 0, "Product_Cataloguing": 1}, "Salary": 11, "Required_Experience" : 2, "Requi
red_Language": "English"}, { "_id": "TD004", "Store_Name" : "Planet Shop", "Required_Employers" : 5, "O
pen_Position" :{ "Billing_Section" : 2, "Product_Demo_Section" : 2, "Product_Cataloguing": 1}, "Salary":
10, "Required_Experience" :2,"Required_Language": "English"}})
```

7. Queries

User Story 1 (**By Yashawant Parab**)

We use \$match aggregate function.

In this user story employer search for the candidate who has at least 5 years of experience in data science and, he is good in programming language skills like python and R.

```
//User Story 1 where Employer search for the candidate
db.UserDetails.aggregate(
  [ { $match : { $and : { (Years_Of_Experience :5), ( Skills : "Python"), (Skills : "Mathematical Domain Expert") } } } ]
)
```

Highest_Degree	Degree_Course	University	Passing_year	Current_Compan	Current_Designat	Years_Of_Experier	Current_Job_Loca	Current_Job_Salar	Expected_Job_Sal	Skills
1 Masters	Big Data A...	University ...	2018.0	SAP	Data Scientist	5.0	Berlin	61000.0	850000.0	[3 elements]

Alternatively, we use find query to get this user story details

```
//Alternative find query for User Story 1
db.UserDetails.find((Years_Of_Experience :5 , Skills : "Python", Skills : "Mathematical Domain Expert"))
```

Highest_Degree	Degree_Course	University	Passing_year	Current_Compan	Current_Designat	Years_Of_Experier	Current_Job_Loca	Current_Job_Salar	Expected_Job_Sal	Skills
1 Masters	Big Data A...	University ...	2018.0	SAP	Data Scientist	5.0	Berlin	61000.0	850000.0	[3 elements]

User story 2 (**By Siddharth Bammani**)

Ryan is a master's graduate in Big data and Data Analytics, and he is looking for job opportunities, he wants to know about job fairs being held in and around his city.




Query 1: To find the companies that match the skill set requirements to that of the user searching for job.

Query	MATCH (a:(User)-[:Skilled_In]->(b:Skill))<-[:Skill_Required]-(c:Company)-[:Appear]->(d:Event) RETURN a,c,d
-------	--

Query 2: To find the closest city near the user where the job fairs are being conducted and distance is less than 100 km.

Query	<pre> MATCH (t:City)-[:Nearby]->(o:User) WITH point({ longitude: t.longitude, latitude: t.latitude }) AS Citypoint, point({ longitude: o.longitude, latitude: o.latitude }) AS Userpoint, t.City_Name as City, t.latitude as Latitude, t.longitude as Longitude WHERE (toInteger(distance(Userpoint, Citypoint)/1000) < 100) RETURN (toInteger(distance(Userpoint, Citypoint)/1000)) as KM, City, Latitude, Longitude </pre>
--------------	--

Below is the result of the above query which provide the city names along with the distance.

\$ MATCH (t:City)-[:Nearby]->(o:User) WITH point({ longitude: t.longitude, latitude: t.l...				
	KM	City	Latitude	Longitude
	34	"Mannheim"	49.4875	8.466
	27	"Karlsruhe"	49.0069	8.4037
	62	"Stuttgart"	48.7758	9.1829
	92	"Mainz"	49.9929	8.2473
Started streaming 4 records after 1 ms and completed after 2 ms.				

Query 3 To find the Job fairs being held in the cities near the user and to retrieve the Company details.

Query	<pre> MATCH (a:User)-[:Skilled_In]->(b:Skill)-[:Skill_Required]-(c:Company)-[:Appear]->(d:Event)-[:Occur]->(e:City) RETURN a,c,d,e </pre>
--------------	--

We retrieve the city, event and the company nodes after running the 1st and the 3rd query.

User Story 3 **(By Yashawant Parab)**

Susan has experience in Data Analysis, and she is looking for the options to upgrade her skills in big data visualization. So, she wants the related certifications for big data and their cost as well.

Job Portal providing services which include the certification parameters, resume building, Aptitude preparation etc. As per the user story we mainly focused on the services which has certification and respective cost.

Candidate is looking for the big data domain certification and willing to pay 800 Euro.

```
db.Services.aggregate(
  [{ $match : { $and : [{ Course_Domain : "Big Data"}, { Cost : { $lt: "800" } } ] } } ] )
```

Services 0.001sec.

_id	Course_Domain	Course_Certification	Cost
1 Service_001	Big Data	Cloudera Certified Professional	300Euro
2 Service_002	Big Data	Intellipaat Big Data Hadoop Certification	380Euro
3 Service_003	Big Data	Microsoft's MCSE in Data Management and Analytics	350Euro
4 Service_004	Big Data	Hortonworks Hadoop Certification	410Euro
5 Service_005	Big Data	MongoDB Certified Developer Exam	450Euro
6 Service_006	Big Data	EMC Data Science and Big Data Analytics Certifications	470Euro
7 Service_007	Big Data	SAS Certified Data Scientist	255Euro
8 Service_008	Big Data	Data Science Council of America Certification	300Euro

User Story 4 (By Yashawant Parab)

Frederik is IT Graduate and he applied to various companies from job portal and he received interview call from Capgemini company. Now he wants to learn about the company and job description.

We use two separate collection and aggregate these collections to get the application status of user. With the help of this query application collection will populate the candidate result where company has accepted or not.

Application collection has Job_id_Org field which is reference to the organization collection and candidate can prepare to his interview. With Find and \$match query we get all this data from different collection.

```
db.Application.aggregate([
  { $lookup:
    {from: "UserDetails",
     localField: "UserName",
     foreignField: "UserName",
     as: "ApplicationData"}} ] )
```

Application 0.001 sec.

Key	Value	Type
> (1) ObjectId("5c84421dff4537ca97ff16ff")	{ 5 fields }	Object
> (2) ObjectId("5c84421dff4537ca97ff1700")	{ 5 fields }	Object
> (3) ObjectId("5c84421dff4537ca97ff1701")	{ 5 fields }	Object
> (4) ObjectId("5c84421dff4537ca97ff1702")	{ 5 fields }	Object
> (5) ObjectId("5c84421dff4537ca97ff1703")	{ 5 fields }	Object
> (6) ObjectId("5c84421dff4537ca97ff1704")	{ 5 fields }	Object
_id	ObjectId("5c84421dff4537ca97ff1704")	ObjectId
UserName	Frederik	String
Job_id_Org	Job016	String
Application_Status	Interview Call	String
ApplicationData	[1 element]	Array

Now Candidate Frederik wants to check Job Description and Company Details where he has selected. From "Job_id_Org" we will get the company details from Organization collection.

Key	Value	Type
(1) Organization011	{ 6 fields }	Object
_id	Organization011	String
Organization_Name	Capgemini	String
Location	[3 elements]	Array
Open_Position	Software Engineer	String
Job_id_Org	Job016	String
Job_Details	[1 element]	Array
[0]	{ 9 fields }	Object
Required_Skills	[3 elements]	Array
Desired_skillSet	[2 elements]	Array
About_Position	[1 element]	Array
Salary	6 lack	String
Work_Location	[3 elements]	Array
Employment_Type	[4 elements]	Array
Min_Experience	0.0	Double
Company_Description	A global leader in consulting, technology and outsourcing services...	String
Job_Description	Managing the software build, release and deployment process. Abl...	String

User Story 5 (**By Yashawant Parab**)

Toy Store manager released few vacancies for part time student with their mandatory requirements. For this user story we have two different collection. Part time student / candidate has uploaded their skills and details in PartTimeJob collection and Store manager posted vacancies in Store collection.

Query:

```
db.PartTimeJob.aggregate([
  {
    $match: { $and: [ {Employee_Age: { $gte: 18, $lte: 25 } },
      {Experience : { $gte: 1 , $lte :3} } ,
      {Speaking_Language : { $all: [ "English" ] }},
      {Expected_Salary : { $gte : 10, $lte: 12}}] }
  },
  {
    $lookup:
```

```

    {from: "Store",
    localField: "Experience",
    foreignField: "Required_Experience",
    as: "FilteredData"}}})

```

New Connectionlocalhost:27017Job_Portal_DataModel

```
db.PartTimeJob.aggregate([
{
    $match: { $and: [ { Employee_Age: { $gte: 18, $lte: 25 } },
    { Experience : { $gte: 1, $lte :3 } },
    { Speaking_Language : { $all: [ "English" ] } },
    { Expected_Salary : { $gte : 10, $lte: 12} }
    ] }
},
{
    $lookup:
    {
        from: "Store",
        localField: "Experience",
        foreignField: "Required_Experience",
        as: "FilteredData"
    }
}]
]
```

PartTimeJob0.051 sec.

<

8. Evaluation

The requirements collected from user stories were fully served. The following table shows the requirement for each specific user story, data model used and the requirements fulfilled.

	Requirements	Data Model Used	Status
Use Case 1	Years of Experience Skills Candidate Name	Document Data Model	Requirement Completely Fulfilled
Use Case 2	Location Role Employee	Graph Data Model	Requirement Completely Fulfilled

	Skill Company Event		
Use Case 3	Domain Certification Cost	Document Data Model	Requirement Completely Fulfilled
Use Case 4	User Details Application_id Organization Details	Document Data Model	Requirement Completely Fulfilled
Use Case 5	Employee Age Experience Languages spoken Expected salary	Document Data Model	Requirement Completely Fulfilled

To enhance the data models further few more features have been identified. Since most of the user stories share common requirements an table has been created with all the identified features. Its been broken into three columns namely Job Seeker, Employer and Administrator.

Job Seeker	Employer	Administrator
Save/Bookmark jobs	Advertise job vacancies instantly	Multiple administrator accounts
Receive jobs via email	Edit/Delete job vacancies	Upgrade/Downgrade agency accounts
Enable/Disable email alerts	Upload company logo	Add unlimited pages of content
Retrieve login information via email	Hide contact details on job postings	Create multiple job categories with subcategories
Build online CV	Company logo displayed by job postings	Add/Edit jobs for each specific agency
Upload Word/PDF CV	Specify application rules for each job posting	Create countries with regions and cities
Apply to job vacancies	Receive application alerts via email	Specify working hours, contracts and experience levels
Upload personal photo	View/Delete applicant CV's	View/Edit/Delete job seekers/agencies

Forward job vacancy	Save/Shortlist jobseeker CV's	Email individual job seekers/agencies
---------------------	-------------------------------	---------------------------------------

9. Bibliography

- <https://neo4j.com/>
- <https://www.mongodb.com/>
- <https://trello.com/>
- <https://drive.google.com/>
- [Mongo Db Advantages over SQL and Other database](#)
- https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm
- https://www.tutorialspoint.com/mongodb/mongodb_overview.htm

Link Python and mongodb

- <https://dzone.com/articles/getting-started-with-python-and-mongodb>
- <https://www.datacamp.com/community/tutorials/introduction-mongodb-python>
- <http://api.mongodb.com/python/current/tutorial.html>
- <https://scalegrid.io/blog/getting-started-with-python-and-mongodb-2/>

MongoDB Queries on array, Aggregate, in query

- <https://docs.mongodb.com/manual/tutorial/query-arrays/#query-for-an-element-by-the-array-index-position>
- [Create Input and Output using Python for User Story 1](#)
- https://en.wikibooks.org/wiki/Python_Programming/Input_and_Output
- https://www.w3schools.com/python/python_mongodb_insert.asp
- <https://stackoverflow.com/questions/42469800/select-from-dropdown-list-in-flask>

Sites used for Neo4j data model and working on the queries.

- https://www.tutorialspoint.com/neo4j/neo4j_set_clause.htm
- <https://www.geeksforgeeks.org/acid-properties-in-dbms/>

For finding the distance between city queries:

-
- <https://neo4j.com/docs/cypher-manual/current/functions/spatial/>
 - <https://neo4j.com/docs/cypher-manual/current/clauses/>
 - <https://neo4j.com/docs/cypher-manual/current/execution-plans/shortestpath-planning/>
 - <https://www.youtube.com/watch?v=I0bTa5Kp7Wg>
 - <https://conferences.oreilly.com/oscon/oscon2011/public/schedule/detail/19822>
 - <https://www.youtube.com/watch?v=Vfz--5hX5qs>

Deleting property of a node:

- <https://stackoverflow.com/questions/18010551/delete-property-from-neo4j-graph>