Identity and Access Management using Graph Database

Content

Introduction

Identity and Access Management (IAM) is a critical aspect of modern information systems, as it helps to ensure that the right individuals have access to the right resources at the right times. One approach to implementing IAM is using a graph database, which is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.

In this approach, the graph database is used to store and manage the identities of users and the resources they are allowed to access. Each node in the graph represents a user, a resource, or a permission, and the edges between nodes represent the relationships between them. For example, an edge between a user node and a resource node represents the user's permission to access the resource.

The use of a graph database allows for a flexible and scalable IAM solution that can handle a large number of users, resources, and permissions, and can easily be adapted to changing business requirements. Querying the graph can be used to quickly determine the permissions of a specific user, to identify all users with access to a particular resource, and to determine if a requested access is allowed based on the relationships stored in the graph.

Motivation

Complex, densely connected access control schemes covering trillions of parties and resources can be stored in a graph database. Both hierarchical and non-hierarchical structures are supported by its richly and variably structured data architecture, and its extendable property model enables obtaining detailed metadata on each system component.

Lookups in a graph database over vast, complicated structures happen in moments rather than minutes or hours thanks to a query engine that can traverse millions of relationships per second.
A graph database access control system supports both top-down and bottom-up queries, just like in network and IT operations.

Graph database-powered access control and authorization systems are particularly useful for content management, federated authorization services, social networking preferences, and software as a service (SaaS) offerings compared to relational databases, they achieve performance improvements of minutes to milliseconds.

Problem Statement

Leverage graph database capabilities and functions to map important and complex relationships describing identity and access management ecosystem within an organization and visualize/query the same on web.

Area of Application

Healthcare Identity Management : It could help healthcare providers manage access to patient records, comply with regulatory requirements, and ensure that only authorized individuals can access sensitive information.

Banking and Financial sector : It could help financial firms enforce access policies, comply with regulations such as KYC (know your customer), and prevent fraud and other security threats.

Education sector : It can help manage access control to student records, grading systems, and other sensitive educational data.

Government sector : It can be used to manage access control to government systems, databases, and applications to ensure authorized access and reduce the risk of cyber attacks and data breaches.

Enterprise Access Management : An IAM system based on a graph database could help organizations manage access to sensitive data, systems, and applications, enforcing policies based on user attributes, job roles, and more.

Literature Review

Traditionally, IAM has been managed using relational databases, which store data in tables and use SQL to retrieve information. However, these databases can be inflexible and inefficient for managing IAM information, particularly in large and complex organizations [1].

Recently, there has been a growing interest in using Graph Databases for IAM. Graph Databases store data in a graph structure consisting of nodes (entities) and edges (relationships between entities), providing a flexible and scalable representation of relationships between entities [2].

IAM is a critical aspect of securing information systems and protecting sensitive data [3]. The need for effective IAM management has increased significantly in recent years due to the growing complexity of organizations, the increasing number of users, and the increasing importance of information security.

A graph-based IAM visualization system that uses graph algorithms to analyze the relationships between users, groups, and permissions and generate interactive visualizations of the IAM information [4]. The visualization system was able to provide a clear and intuitive representation of IAM information, enabling administrators to make informed decisions about IAM policies.

Literature Review

The IAM visualization system will provide numerous useful features for its management. For example, administrators will be able to view information about users, such as their name, username, email address, as well as information about groups, such as the memberships of users, the privileges granted [5]. The system will also allow administrators to make changes to the IAM information, such as adding or removing users from groups, granting or revoking privileges, creating or deleting groups.

The need for IAM in organizations is driven by a number of factors, such as increasing regulatory requirements, the rise of cloud computing, and the growth of mobile and IoT devices. A literature review of IAM typically highlights the importance of IAM and its impact on organizations' security and compliance posture.

A variety of technologies are available for implementing IAM, including directory services, access management systems, identity management systems, and single sign-on (SSO) solutions.

Implementing IAM can be challenging for organizations, due to a number of factors such as complexity, cost, and compatibility with existing systems. A literature review of IAM typically covers these challenges, as well as potential solutions for overcoming them [6].

SWOT Analysis

STRENGTH

Graph Database architecture is designed to scale horizontally, which can be used for IAM systems to handle large volumes of data as the number of users and resources grow. Also, its particularly well-suited for managing complex relationships between users, roles, resources, and permissions.

WEAKNESS

While Graph Database flexibility and graph-based approach can be an advantage, they can also make the system more complex and difficult to manage, particularly for teams that are not familiar with graph databases.

OPPORTUNITY

Graph Database can help IAM systems provide more personalized and efficient user experiences by better managing user access and permissions. It's ability to manage complex relationships can better secure data by enforcing access controls and minimizing the risk of data breaches.

THREAT

There are other IAM solutions on the market, and organizations may choose to use a different solution instead of Graph database and adopting this database for IAM may lead to vendor lock-in, which can make it difficult to switch to a different solution in the future.

Objective

Main Objective:

Use Graph Database for IAM to manage user identities, access to resources, and permissions in a scalable, secure, and efficient way.

Sub-Objective:

•To let organizations handle identity and access management with optimized processing.

•To implement graph database for problem solving.

•To implement a web-based identity and access management system with authentication.

Methodology

Reference Software model:- Iterative model

- An iterative life cycle model focuses on an initial, basic set of user features before gradually adding complexity and a wider range of features until the desired system is finished.
- This approach avoids having to start with a thorough definition of requirements.
- The principle of incremental development will also be frequently and interchangeably applied when using the iterative technique.

We aim to build an Identity and access management system visualization using graph database. Neo4j is the preferred graph database that will be used in this project. A website with login authentication for individual users will be made using standard web development technologies like react, Mongodb, node etc. Each user will have to upload a dataset in a preferable format through which, visualization will be done and graph database will be setup. Query options can be implemented using standard neo4j query language: Cypher. Any hosting or deployment will be handled using firebase.

Methodology

Steps for implementation :

1. Modelling the Identity Domain: First, define the entities and relationships that are relevant to the IAM domain. For example, entities such as users, groups, roles, and permissions, and relationships such as "belongs to" or "has access to."

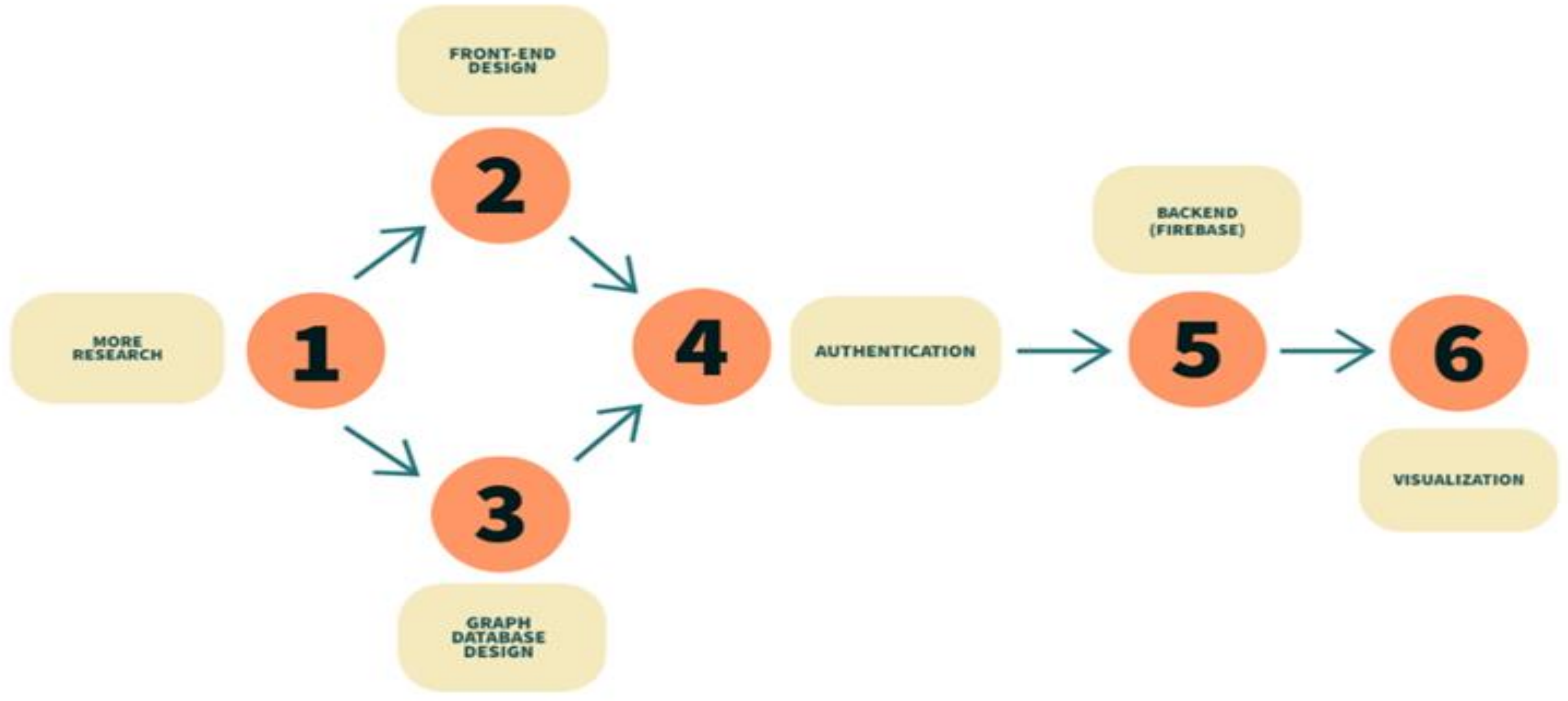
2. Storing Identity Data: Next, store the identity data in the graph database. Each entity is represented as a node, and each relationship as an edge connecting two nodes.

3.Defining Access Control Rules: Define the access control rules that determine who can access what resources. These rules can be expressed in the form of Cypher queries, the query language used in graph databases.

4. Implementing Authentication and Authorization: Implement the authentication and authorization processes that determine whether a user is allowed to access a resource. This can be done by executing Cypher queries to check if the user meets the conditions specified in the access control rules.

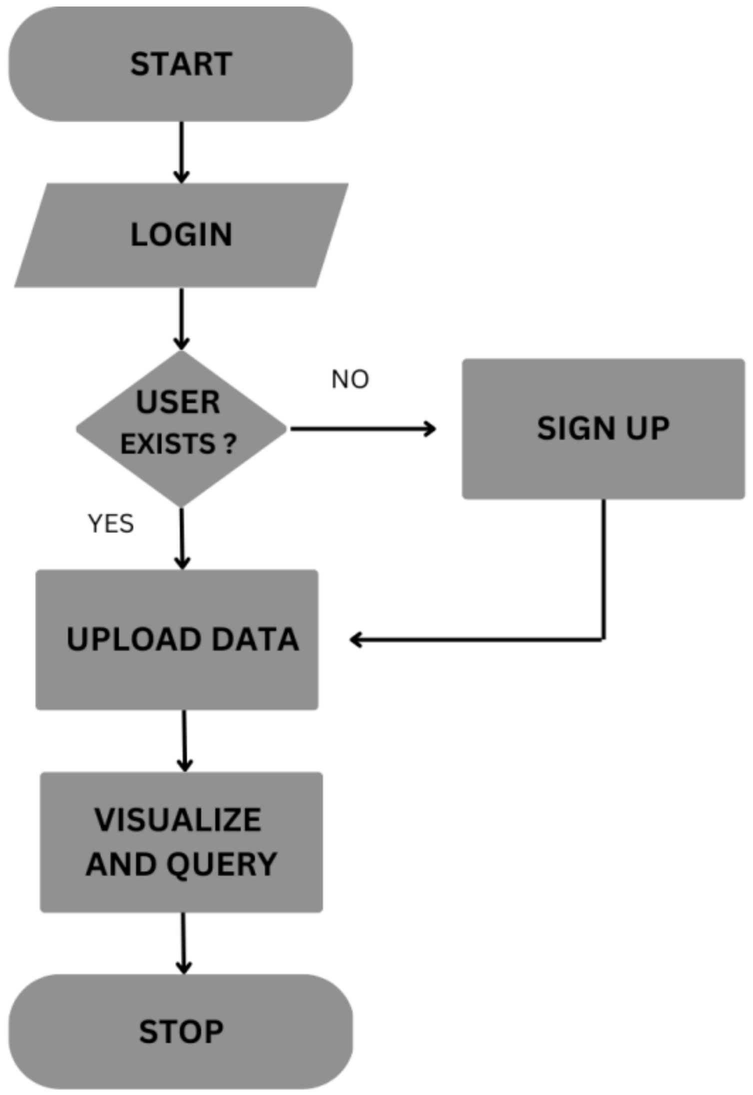
5. Auditing and Reporting: Finally, implement auditing and reporting functionality to keep track of who has accessed what resources and when. This can be done by storing the access logs in the graph database and using Cypher queries to generate reports.

Methodology



Methodology Chart

Data Model


START
LOGIN
USER EXISTS ?
NO
SIGN UP
YES
UPLOAD DATA
VISUALIZE AND QUERY
STOP

Working Model

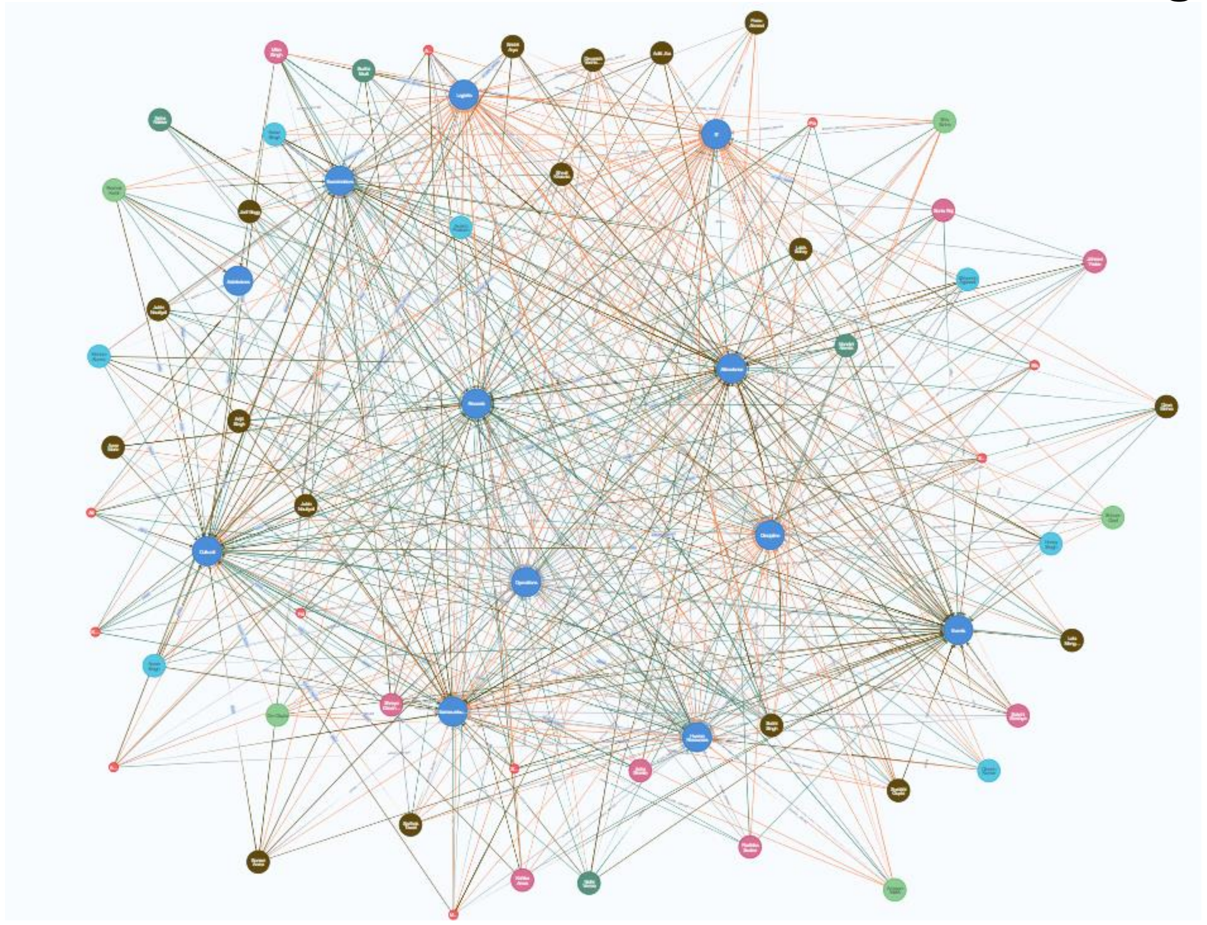
Requirement analysis (Link of SRS)

Software Requirement Specification (SRS)

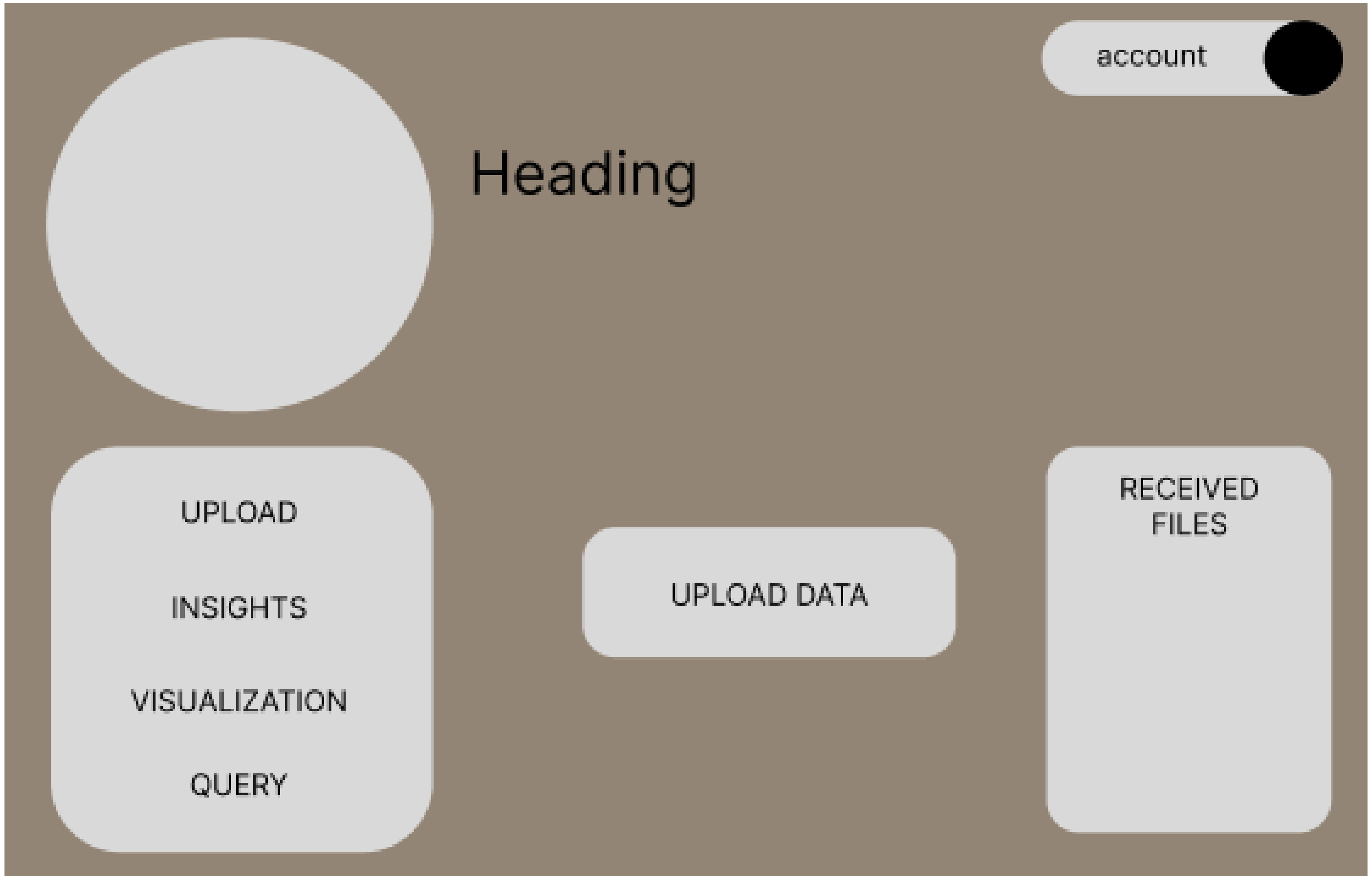
Working Model

Working Module:

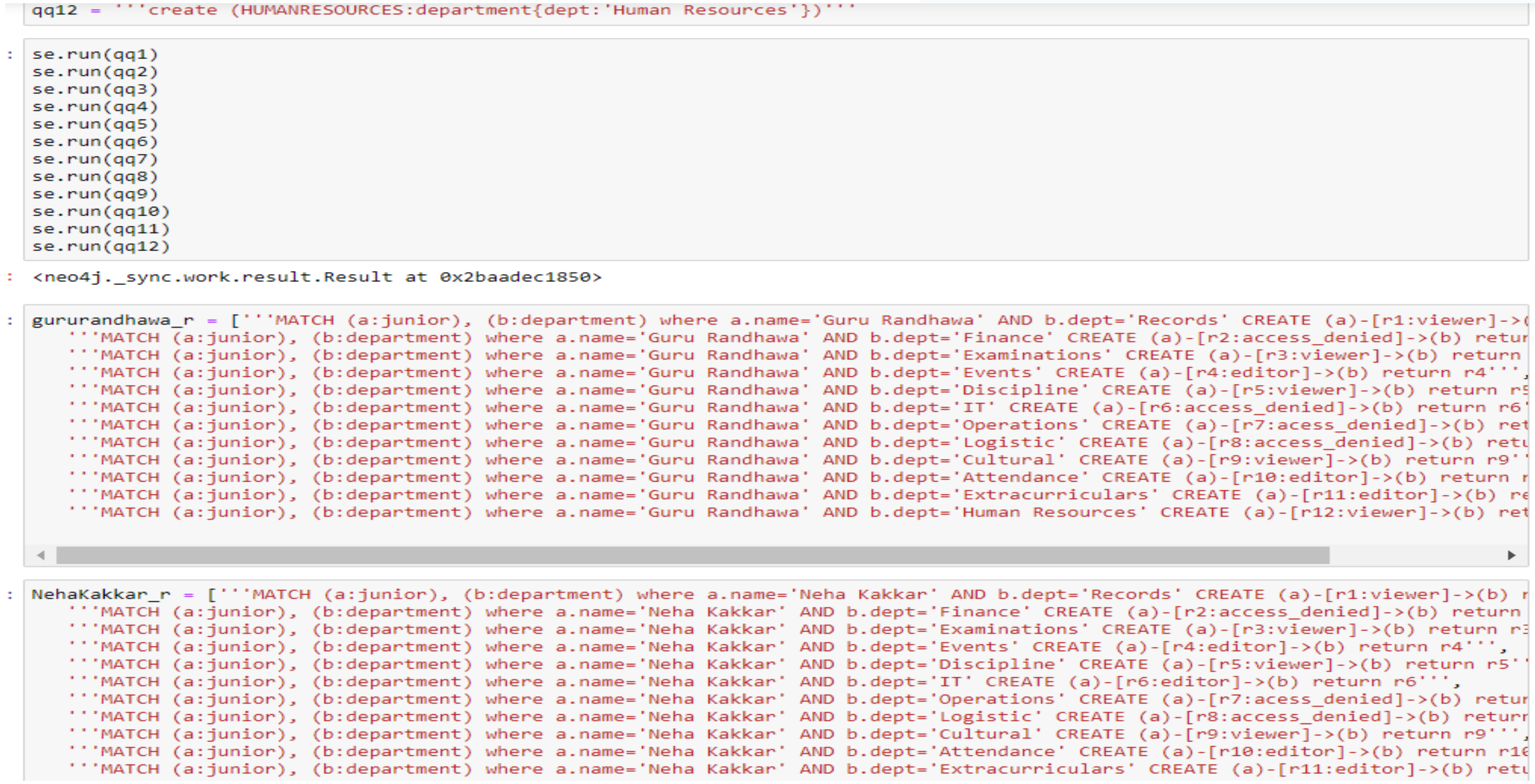
In our Project, we provide a platform for visualization for the complex data by the use of graph database(Neo4j), in this the user/organization upload their data in our website and run the query, so that our model can visualize their data in form of graphs.



Working Model



Working Model



```
qq12 = '''create (HUMANRESOURCES:department{dept:'Human Resources'})'''
```

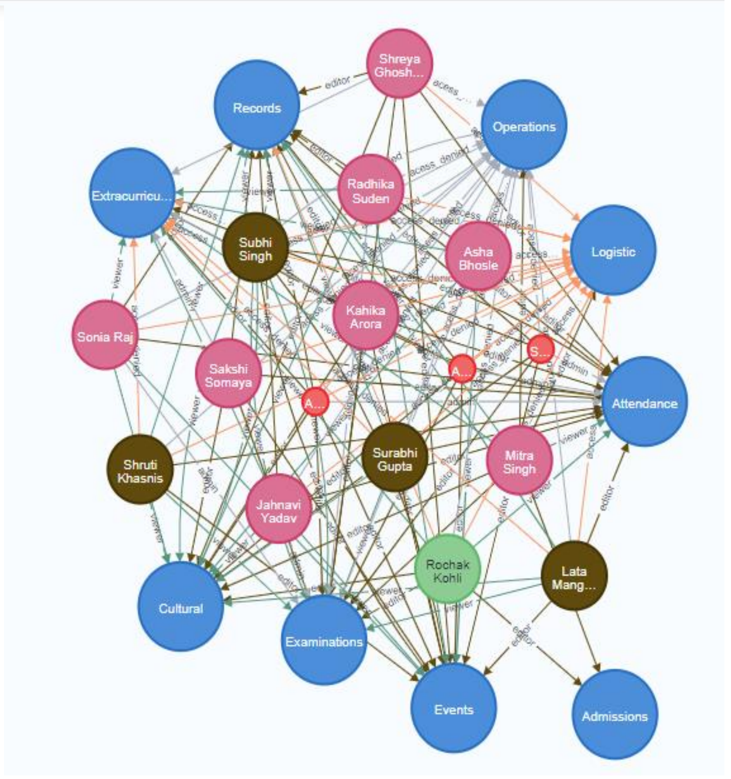
```
: se.run(qq1)
  se.run(qq2)
  se.run(qq3)
  se.run(qq4)
  se.run(qq5)
  se.run(qq6)
  se.run(qq7)
  se.run(qq8)
  se.run(qq9)
  se.run(qq10)
  se.run(qq11)
  se.run(qq12)
```

```
: <neo4j._sync.work.result.Result at 0x2baadec1850>
```

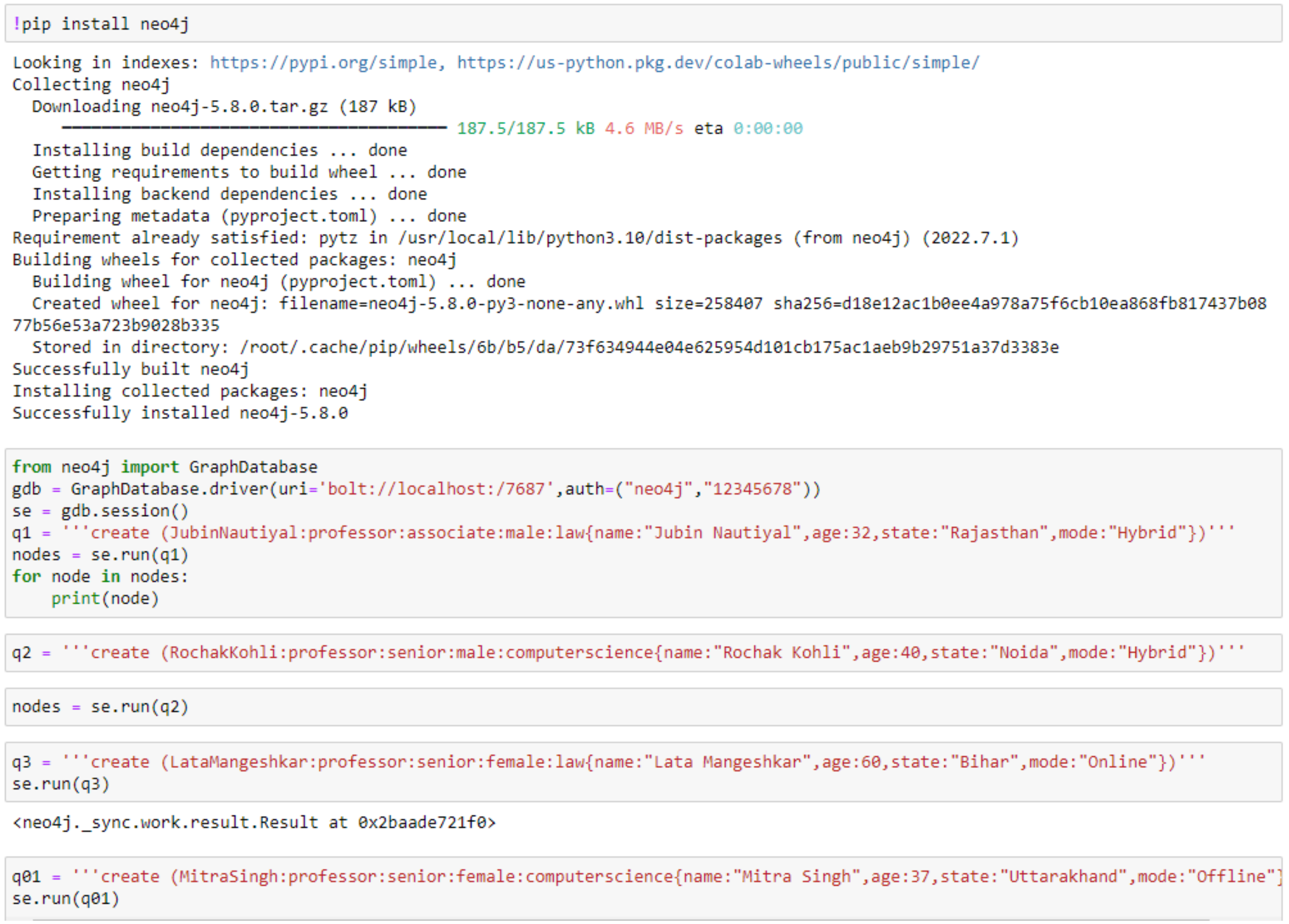
```
: gururandhawa_r = ['''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Records' CREATE (a)-[r1:viewer]->(
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Finance' CREATE (a)-[r2:access_denied]->(b) retur
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Examinations' CREATE (a)-[r3:viewer]->(b) return
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Events' CREATE (a)-[r4:editor]->(b) return r4''',
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Discipline' CREATE (a)-[r5:viewer]->(b) return r5
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='IT' CREATE (a)-[r6:access_denied]->(b) return r6'
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Operations' CREATE (a)-[r7:acess_denied]->(b) ret
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Logistic' CREATE (a)-[r8:access_denied]->(b) retu
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Cultural' CREATE (a)-[r9:viewer]->(b) return r9''
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Attendance' CREATE (a)-[r10:editor]->(b) return r
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Extracurriculars' CREATE (a)-[r11:editor]->(b) re
      '''MATCH (a:junior), (b:department) where a.name='Guru Randhawa' AND b.dept='Human Resources' CREATE (a)-[r12:viewer]->(b) ret
```

```
: NehaKakkar_r = ['''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Records' CREATE (a)-[r1:viewer]->(b) r
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Finance' CREATE (a)-[r2:access_denied]->(b) return
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Examinations' CREATE (a)-[r3:viewer]->(b) return r3
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Events' CREATE (a)-[r4:editor]->(b) return r4''',
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Discipline' CREATE (a)-[r5:viewer]->(b) return r5''
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='IT' CREATE (a)-[r6:editor]->(b) return r6''',
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Operations' CREATE (a)-[r7:acess_denied]->(b) retur
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Logistic' CREATE (a)-[r8:access_denied]->(b) return
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Cultural' CREATE (a)-[r9:viewer]->(b) return r9''',
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Attendance' CREATE (a)-[r10:editor]->(b) return r10
      '''MATCH (a:junior), (b:department) where a.name='Neha Kakkar' AND b.dept='Extracurriculars' CREATE (a)-[r11:editor]->(b) retu
```

```
MATCH p=(a:senior)-[:editor]→() return p
```



Working Model



```
!pip install neo4j
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting neo4j
  Downloading neo4j-5.8.0.tar.gz (187 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 187.5/187.5 kB 4.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pytz in /usr/local/lib/python3.10/dist-packages (from neo4j) (2022.7.1)
Building wheels for collected packages: neo4j
  Building wheel for neo4j (pyproject.toml) ... done
  Created wheel for neo4j: filename=neo4j-5.8.0-py3-none-any.whl size=258407 sha256=d18e12ac1b0ee4a978a75f6cb10ea868fb817437b08
77b56e53a723b9028b335
  Stored in directory: /root/.cache/pip/wheels/6b/b5/da/73f634944e04e625954d101cb175ac1aeb9b29751a37d3383e
Successfully built neo4j
Installing collected packages: neo4j
Successfully installed neo4j-5.8.0
```

```
from neo4j import GraphDatabase
gdb = GraphDatabase.driver(uri='bolt://localhost:/7687',auth=("neo4j","12345678"))
se = gdb.session()
q1 = '''create (JubinNautiyal:professor:associate:male:law{name:"Jubin Nautiyal",age:32,state:"Rajasthan",mode:"Hybrid"})'''
nodes = se.run(q1)
for node in nodes:
    print(node)
```

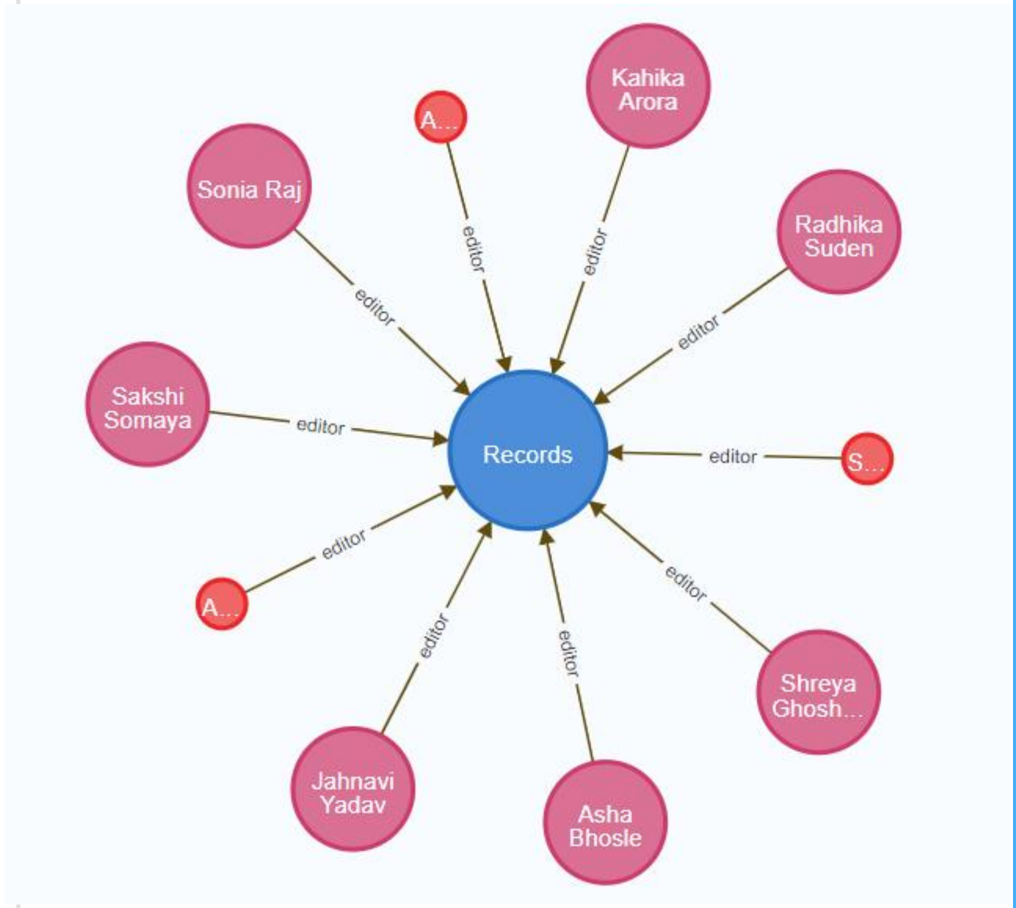
```
q2 = '''create (RochakKohli:professor:senior:male:computerscience{name:"Rochak Kohli",age:40,state:"Noida",mode:"Hybrid"})'''
```

```
nodes = se.run(q2)
```

```
q3 = '''create (LataMangeshkar:professor:senior:female:law{name:"Lata Mangeshkar",age:60,state:"Bihar",mode:"Online"})'''
se.run(q3)
```

```
<neo4j._sync.work.result.Result at 0x2baade721f0>
```

```
q01 = '''create (MitraSingh:professor:senior:female:computerscience{name:"Mitra Singh",age:37,state:"Uttarakhand",mode:"Offline"}
se.run(q01)
```



Working Model

Attained Deliverables:

Objectives	Status
Identify the project requirements and goals	Completed
Learn the required skills	Completed
Plan the system design and architecture	Completed
Creating a user interface	Completed
Design the graph database schema	Completed
Create Neo4j Database and integrate	Partially Completed

Reference

[1] Martin Gaedke, Johannes Meinecke, and Martin Nussbaumer. (2005). “A modeling approach to federated identity and access management”. In Special interest tracks and posters of the 14th international conference on World Wide Web 2005.

[2] Wong, C. Y., Lee, C. Y., & Lee, W. B. “A graph-based identity and access management system”. Journal of Network and Computer Applications 2019.

[3] Fan, X., Wang, Y., & Li, X. “A graph-based identity and access management visualization system”. Journal of Information Security and Applications 2021.

[4] Mohammed, Ishaq Azhar. “Intelligent authentication for identity and access management”. A review paper, Iraqi Journal for Computers and informatics 2013.

[5] Deepak H. Sharma, C.A. Dhote, Manish M. Potey. “Identity and Access Management as Security-as-a-Service from Clouds”. 7th International Conference on Communication 2016

[6] M. A. Thakur and R. Gaikwad. “User identity and Access Management trends in IT infrastructure- an overview”. International Conference on Pervasive Computing (ICPC) 2015.

UPES
UNIVERSITY OF TOMORROW

Thank You