Summer Research Internship 2021

Project Report

MyLinkedin

Case Study of NoSQL Databases for Social Media

Mentor: Prof. PM Jat

Students: Madhavendra Choksi(201801255),
Aayush Patel(201801259)

Dhirubhai Ambani Institute
of Information and Communication Technology

August, 2021

Table of Contents

Problem Statement **2**

Use Cases: **3**

Identified Entities **12**

Entity Relationship Diagram **17**

MongoDB schema **18**

Cassandra Schema: **39**

Queries to be run on database for use cases : **41**

Shell Scripts: **50**

Problem Statement

Understanding a complex existing system , designing and modeling its data requirements into NoSQL database . In our case-study, the system is Linkedin which is basically a social media website used primarily for professional purposes and growing your professional network and connecting with companies and firms .Job application and hiring is also an objective for the users using this system.

Why NoSQL?

MongoDB is a document-oriented NoSQL database that we can use for storing a high volume of data.

Database Management systems of this type use dynamic schemas. This scheme means that you have the option to create records without giving the definition to the structure first. This is the biggest advantage of NoSQL databases and which makes them powerful.MongoDB gives you permission for changing the structure of the records. We can call this document after we add new fields or delete the already existing one.

In today's world of Web Development as we mentioned earlier, .json data format is changing the scene drastically and MongoDB makes the representation in the form of JSON documents. On the other hand, MySQL represents the data in the form of tables, rows, and columns. So it is obvious to use MongoDB over MySQL.

Use Cases:

The first step towards proper data modeling is identifying the Use Cases that the system is required to perform as this helps us to identify which actions are performed by the user and what kind of data model is required for achieving the use case efficiently.

.

1. Login:

1. *Name*: Login
2. Actors Involved: User
3. Precondition: User must already have registered on MyLinkedin.
4. Description: Email id and password of user are taken and verified with the database to authenticate the user.
5. Postcondition: Display message regarding login status after authenticating the user.

2. Signup:

1. Name: Signup
2. Actors: User, Linkedin DB
3. Precondition:User must have a valid email id
4. Description: Basic details of the user such as name, username,password , email , phone number , education , Date of birth is asked and entered into the database.
5. Postcondition: new user object should be created in DB and the details should be stored in it.

3. Add connection:

1. Name: Connection
2. Actors Involved: User
3. Precondition: User must have registered on MyLinkedin.
4. Description: Users can connect to any other user on linkedin, and thereafter be able to view each other's updates.
5. Postcondition: If the account is private, the second user is notified about the connection request, else they are connected.

4. Messaging:

1. Name: Messaging
2. Actors Involved: User
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can send messages to other users.
5. Postcondition: If the account is private, the second user is notified about the message request, else conversation is started..

5. Job positioning:

1. Name: Job positioning.
2. Actors Involved: User
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can search and apply for a job from the list of job opportunities or can contact the recruiter. A user can also add job opportunities for others.
5. Postcondition: The recruiter who has created a job opportunity, is notified when someone has applied for it.

6. Post:

1. Name: Post
2. Actors Involved: User, Linkedin
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can add/delete photo,text,video,event,experience as a post.

7. Edit a Post:

5. Name: Edit Post
6. Actors Involved: User, Linkedin
7. Precondition: User must be logged into his MyLinkedin profile nad should have posted something which is to be edited.
8. Description: A user can edit the content of the post.
9. Postcondition: The database should be updated with the edited content.

8. Like/comment:

1. Name: Like and comment.
2. Actors Involved: User , Linkedin
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can give a like to some other user's post or add a public comment to the post.

9. Unlike a post:

1. Name: Unlike a post.
2. Actors Involved: User , Linkedin
3. Precondition: User must be logged into his MyLinkedin profile and the specific post should already be liked.
4. Description: A user can unlike a post which was already liked by him.
5. Postcondition: The database should be updated with changed like status

10. Share:

1. Name: Share post.
2. Actors Involved: User , Linkedin

3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can share someone else's post on their feed.

11. Search:

1. Name: Search.
2. Actors Involved: User
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user can search for other users, company pages, hashtags and jobs.
5. Postcondition: Search results of related words.

12. Activity:

1. Name: View activity.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: A user should be able to see the log of their post liked, commented, profile viewed, connection requested or connection added.

13. View Connections:

1. Name: View connections.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile
4. Description:A user can view all his connections on linked in .

14. Privacy setting:

1. Name: Private account.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: User should be able to change the privacy setting to 'Private' or 'Public'. If a profile is private, only the connections should be able to view the profile and others can see only their name.

15. Sign out:

1. Name: Sign out.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: User should be able to sign out from his account, so no functionalities can be used further by him, without re - signing in.
5. Postcondition: User won't be able to access his account.

16. Add section:

1. Name:Add section.

2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile
4. Description: Users can add intro,education, background , experience , volunteer work ,skills, accomplishments ,supported languages.
5. Postcondition:This section article should be added to the database.

17. View News feed:
    1. Name:News feed.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: Users should be able to view all the posts of their and their connections such that the newest is shown on top.

18. Add Education:
    1. Name:Education under background.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: Users should be able to add education with the details like passing year, institute, major subjects etc.
    5. Postcodition: Education section should be updated with new education document.

19. Respond to a connection request:
    1. Name: Connection request.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile and have an active connection request.
    4. Description: User should be able to accept or delete a connection request.
    5. Postcondition: If the request is accepted, the connections list is updated with new connection and the list of connection requests is updated with removing the connection request (even if it is declined).

20. Save post:
    1. Name: Save post.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: If a post is visible to the user, the user should be able to save that post and view it in future.
    5. Postcondition: The user should be able to see the post in saved posts section.

21. View profile:
    1. Name: View profile.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.

4. Description:User should be able to see the profile of other users on linkedin.

22. Edit profile intro:

1. Name: Edit profile intro.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description:User should be able to edit their profile intro.
5. Post Condition: New profile intro should be updated to the database and other users should be able to see the updated profile intro.

23. Edit profile photo:

1. Name: Edit profile photo.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description:User should be able to edit their profile photo.
5. Post Condition: New profile photo should be updated to the database and other users should be able to see the updated profile photo.

24. Edit wall photo:

1. Name: Edit wall photo.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: User should be able to edit their wall photo.
5. Post Condition: New wall photo should be updated to the database and other users should be able to see the updated wall photo.

24. Like a comment on a post:

1. Name: Like a comment on a post.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: Users should be able to like comment(s) on a post.
5. Post Condition: Likes count on a comment should be increased.

25. Edit post visibility:

1. Name: Edit who can see the post.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: User can change the post visibility setting to private, public or to only their connections
5. Post Condition: Database to be updated with the new settings.

26. Edit open to settings:
    1. Name: Edit the status for job positioning.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: User can change the open to settings for job positioning, it can be hiring or looking for job.
    5. Post Condition: Database to be updated with the new settings.
27. Edit mute status of a connection:
    1. Name: Edit the mute status of a connection.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile and should be connected to that user what is to be muted.
    4. Description: User can change the mute status for a user, so no posts are visible to the user of that particular user or if already muted can unmute.
    5. Post Condition: Change the mute status in connections list of the user.

28. Follow a page:
    1. Name: Follow a page.
    2. Actors Involved: User, Linkedin db, page.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: User should be able to follow a page and see the posts of the page on the user's news feed.
    5. Post Condition: List of followers should be updated with the name of user and page name should be inserted to the list of pages followed by the user.

29. Follow a hashtag:
    1. Name: Follow a hashtag.
    2. Actors Involved: User, Linkedin db.
    3. Precondition: User must be logged into his MyLinkedin profile.
    4. Description: User should be able to follow a hashtag and see the posts with the hashtag used on the user's news feed.
    5. Post Condition: List of hashtags should be inserted to the list of pages followed by the user.

30. Create a page:
    1. Name: Create a page
    2. Actors: page, Linkedin DB
    3. Precondition:Page must have a valid and available page name.
    4. Description: Basic details of the page such as name, username, password , email , phone number , website etc are asked and entered into the database.
    5. Postcondition: New page is created and entered into database.

31. Edit page name:
    1. Name: Change user name.
    2. Actors: page, Linkedin DB
    3. Precondition: Page must have a valid and available new page name.
    4. Description: A page can change its name from one name to another valid page name.
    5. Postcondition: New name of the the page is updated on the database.

32. Edit page photo:
    1. Name: Edit page photo.
    2. Actors Involved: page, Linkedin db.
    3. Precondition: Page admin must be logged into page MyLinkedin profile.
    4. Description:Page admin should be able to edit their page photo.
    5. Post Condition: New page photo should be updated to the database and other users should be able to see the updated page photo.

33. Edit page wall photo:
    1. Name: Edit page wall photo.
    2. Actors Involved: page, Linkedin db.
    3. Precondition:Page admin must be logged into page MyLinkedin profile.
    4. Description: Page admin should be able to edit their page wall photo.
    5. Post Condition: New wall photo should be updated to the database and other users should be able to see the updated page wall photo.

34. Edit page password:
    1. Name: Edit page password.
    2. Actors Involved: Page, Linkedin db.
    3. Precondition: Page admin must be logged into page MyLinkedin profile.
    4. Description:Page admin should be able to change their password.
    5. Post Condition: New page photo should be updated to the database and other users should be able to see the updated page photo.

35. Edit profile intro:
    1. Name: Edit profile intro.
    2. Actors Involved: Page, Linkedin db.
    3. Precondition: Page admin must be logged into page MyLinkedin profile.
    4. Description:Page admin should be able to edit their profile intro of the page.
    5. Post Condition: New page profile intro should be updated to the database and other users should be able to see the updated profile intro

36. Edit contact info:

1. Name: Edit contact info of the page.
2. Actors Involved: Page, Linkedin db.
3. Precondition: Page admin must be logged into page MyLinkedin profile.
4. Description:Page admin should be able to edit their contact info of the page, be it phone number or email id or website.
5. Post Condition: New page contact should be updated to the database and other users should be able to see the updated page contact.

37. Edit mute status of a page:

1. Name: Edit the mute status of a page.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile and should be following the page that is to be muted.
4. Description: User can change the mute status for a page, so no posts are visible to the user of that particular page or if already muted can unmute.
5. Post Condition: Change the mute status in the pages followed list of the user.

38. Add work experience:

1. Name: Add work experience of a user.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: Users should be able to add work experience with the details like date of joining, job title, location, company etc and this should be shown as post to all the user's connections.
5. Postcodition: Work experience section should be updated and the connections should be able to see the new work ex update as a post on their wall.

39. Add license/certification:

1. Name: Add license or certification of the user.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: Users should be able to add license or certification with the details like date of issuing, issuer, title etc and this should be shown as post to all the user's connections.
5. Postcodition: License or certification section should be updated and the connections should be able to see the new license or certification update as a post on their wall.

40. Add volunteer experience:

1. Name: Add volunteer experience of the user.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.

4. Description: Users should be able to add volunteer experience with the details like date, title, organisation etc and this should be shown as post to all the user’s connections.
5. Postcodition: Volunteer experience section should be updated and the connections should be able to see the new volunteer experience update as a post on their wall.

41. Add skill:

1. Name: Add skill of the user.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: Users should be able to add skill with the details like title, organisation etc.
5. Postcodition: Skill section should be updated.

42. Add Course:

1. Name: Add course of the user.
2. Actors Involved: User, Linkedin db.
3. Precondition: User must be logged into his MyLinkedin profile.
4. Description: Users should be able to add course studied by the user with the details like date, title, organisation etc and this should be shown as post to all the user’s connections.
5. Postcodition: Course section should be updated and the connections should be able to see the new volunteer experience update as a post on their wall.

Identified Entities

USER

A. user_id
B. Name (string)
C. Email id(string)
D. Password(string)
E. Connections(array of objects)
    a. User id (unique object id)
    b. Mute status (boolean)
F. Contact info
    a. Email (string)
    b. Url (string)
G. Activity
H. Introduction
I. Pages
J. Profile photo(url - string)
K. Wall photo(url - string)
L. Country(string)
M. Location in country(string)
N. Industry(string)
O. Skills (array of strings)
P. Accomplishments
    a. Publications(array of objects)
        i. Title (string)
        ii. Publisher (string)
        iii. Publication date (date/timestamp)
        iv. Coauthor (object_id (user_id))
        v. Url (string)
        vi. Description (string)
    b. Patent(array of objects)
        i. Patent title (string)
        ii. Patent office (string)
        iii. Patent application number(int)
    c. Courses(array of objects)
        i. Course name(string)
        ii. Course code (string)
        iii. Association(string)
    d. Projects(array of objects)
        i. Title(string)
        ii. Start date(timestamp)

iii. End date (timestamp)
iv. Association(string)
v. Project url(url - string)
vi. Description(string)

e. Honors & Awards(array of objects)
i. Title (string)
ii. Association(string)
iii. Issuer(string)
iv. Issue date(timestamp)
v. Description(string)

f. Test scores(array of objects)
i. Title (string)
ii. Associated with (string)
iii. Scores (int)
iv. Date (timestamp)
v. Description (string)

g. Organisations [array of objects]
i. Organisation name (string)
ii. Position held (string)
iii. Associated with (string)
iv. On going (bool)
v. Start date (timestamp)
vi. End date (timestamp)
vii. Description (string)

h. Languages(array of objects)
i. Language(string)
ii. Proficiency level(int)
iii.

Q. Additional information
a. Recommendations (array of objects_id i.e user_id)

R. Background
a. Work experience
i. Title (string)
ii. Employment Type (Full time/Part time/Self employed/freelance/Internship/trainee)(string)
iii. Location(string)
iv. Company (string)
v. Working status(bool)
vi. Start date(timestamp)
vii. End date(timestamp)
viii. Industry(string)
ix. Description(string)

b. Eduction

i. school_name(string)
ii. degree(string)
iii. field of study(string)
iv. Start year(timestamp)
v. End year(timestamp)
vi. CGPA(float)
vii. Activities and societies(array of strings)
viii. Description(string)

c. License and certification
i. Name(string)
ii. Issuing organization(string)
iii. Issue date(timestamp)
iv. Expiration date(timestamp)
v. Credential id (string)
vi. Credential url(string)

d. Volunteer experience
i. Organization (string)
ii. Role (string)
iii. Cause(string)
iv. Working status(bool)
v. Start date (timestamp)
vi. End date (timestamp)
vii. Description(string)
viii.

S. Featured
a. Posts (array of post_id) (post_id is reference to post entity)
b. Articles
c. Links
d. Media

T. Privacy Settings
a. Profile visibility (bool)
b. Headline visibility (bool)
c. Articles and activities visibility (bool)
d. Education visibility (bool)
e. Details visibility (bool)

U. Saved Posts (array of post_id) (post_id is reference to post entity)
V. Notifications (array of strings)
W. Saved Jobs (array of jobs_id)(jobs_id is reference to the jobs entity)

POSTS

A. User(user_id)
B. Hashtags(string)
C. Tagged users(array of user_ids)

D. Time(timestamp)
E. Likes(array of objects)
    a. User ID
    b. Timestamp
F. Comments
    a. User ID
    b. Text(string)
    c. Likes(int)
    d. Timestamp
G. Media(url - string)
H. Description (Caption)
I. Visibility (anyone/connections only/no one)

PAGE

A. Page name (string)
B. Profile photo(url - string)
C. Wall photo(url - string)
D. Description(string)
E. Contact info
    a. Email(string)
    b. Url (string)
    c. Phone number(string)
    d. Address(string)
F. Followers(array of user_ids)
G. About
    a. Industry(string)
    b. Website(url - string)
    c. Company size (integer)
    d. Head quarters(string)
    e. Type(string)
    f. Specialities(string)
    g. Locations(string)
H. Posts (id)
I. Jobs(array of jobs_id)(jobs_id is reference to the jobs entity)
J. Affiliated pages

JOBS

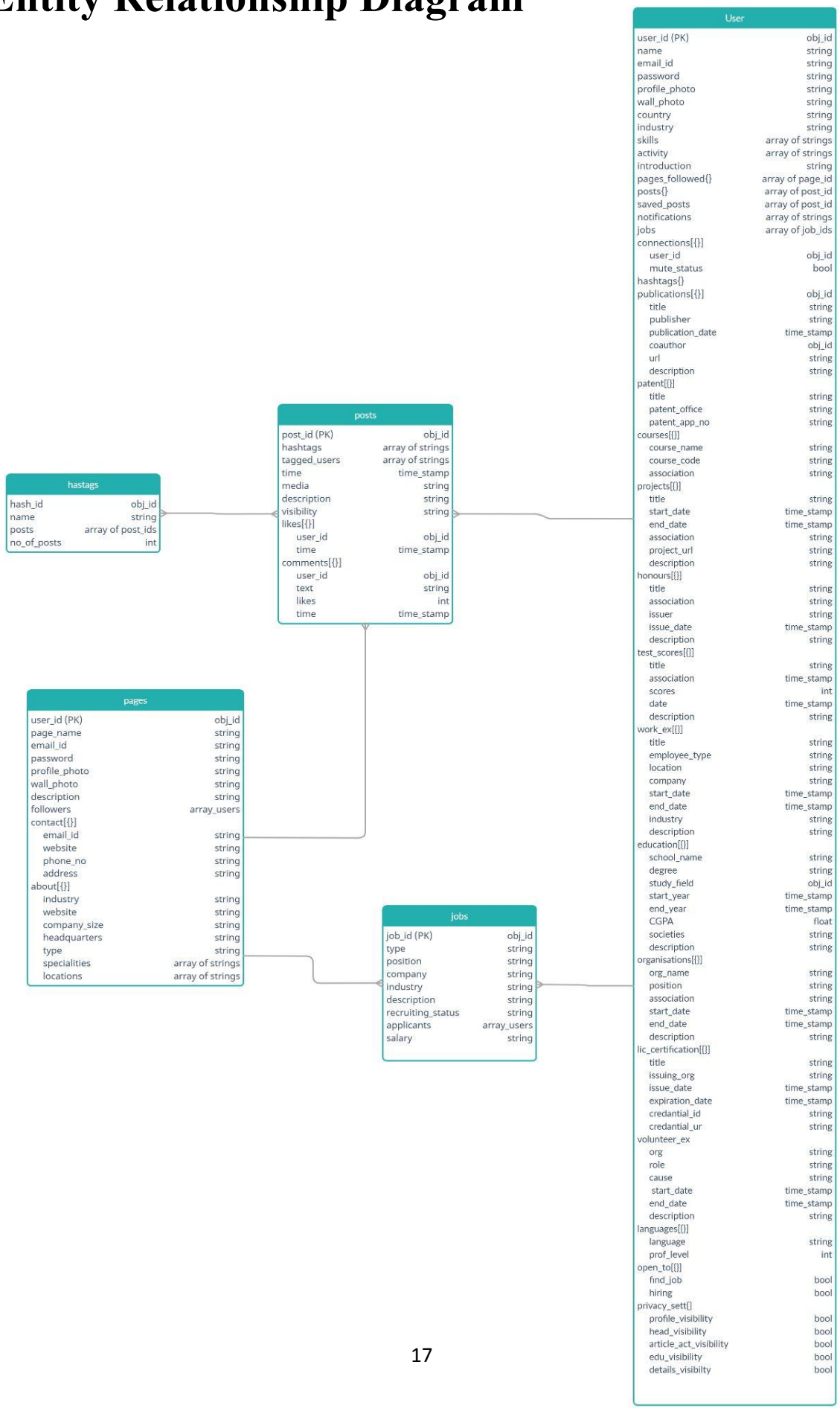
A. Jobs id
B. Type(full time/part time/trainee/intern)(string)
C. Position (string)
D. Company(string)
E. Industry(string)
F. Description(string)

G. Recruiting status (bool)
H. Applicants (array of user_ids)
I. Salary(integer)

Hashtags

A. Name
B. posts(array of post_ids)
C. followers(array of user_ids)
D. Number of followers(integer)

Entity Relationship Diagram



MongoDB schema

Jobs:

```
db.createCollection("jobs",
{
   validator:
   {
      $jsonSchema:
      {
         bsonType : "object",
         title: "jobs",
         required: ["position","company","industry","recruiting_status"],
         description: "This document is a collection  of all the jobs",
         properties:
         {
            position:
            {
               description: "name of the job position",
               bsonType: "string",
               minLength:1
            },
            company:
            {
               description: "name of the company",
               bsonType: "string",
               minLength:1
            },
            industry:
            {
               description: "type of industry of the company",
               bsonType: "string",
               minLength:1
            },
            job_description:
            {
               description: "Description of the job profile",
               bsonType: "string",
```

```
            minLength:1
          },
          job_requirement:
          {
            bsonType: "string",
            minLength:1
          },
          recruiting_status:
          {
            bsonType:"bool",
          },
          salary:
          {
            bsonType: "string",
            minLength:1
          },
          applicants:
          {
            bsonType: "array",
            items:
            {
              description: "List of applicants",
              bsonType: "objectId"
            },
            uniqueItems: true
          },
        },
      },
    },
    validationLevel: "strict",
    validationAction: "error",
})
```

Hashtags:

```
db.createCollection("hashtags",
{
   validator:
   {
      $jsonSchema:
      {
         bsonType : "object",
         title: "hashtags",
         required: ["name","posts"],
         description: "This document is a collection  of the hashtags used in posts",
         properties:
         {
            name:
            {
               description: "name of the hashtag used",
               bsonType: "string",
               minLength:1
            },
            posts:
            {
               bsonType: "array",
               items:
               {
                  description: "List of post_ids with this hashtag used",
                  bsonType: "objectId"
               },
               minItems: 1,
               uniqueItems: true
            },
            no_of_posts:
            {
               bsonType: "int"
            },
         },
         followers:
            {
               bsonType: "array",
               items:
               {
                  description: "List of followers",
```

```
                bsonType: "objectId"
            },
            uniqueItems: true
        },
    },
},
validationLevel: "strict",
validationAction: "error",
})
```

Pages:

```
db.createCollection("pages",
{
   validator:
   {
      $jsonSchema:
      {
         bsonType : "object",
         title: "pages",
         required: ["name"],
         description: "This document is a collection all pages on Linkedin",
         properties:
         {
            name:
            {
               description: "name of the page",
               bsonType: "string",
               minLength:1,
               uniqueItems: true
            },
            profile_photo:
            {

               bsonType: "string",
               minLength:1

            },
            wall_photo:
            {
```

```
    bsonType: "string",
    minLength:1
},
description:
{
    bsonType: "string",
    minLength:1
},
contact_info:
{
    bsonType:"object",
    properties:
    {
        email:
        {
            bsonType: "string",
            minLength:1
        },
        phone_no:
        {
            bsonType: "array",
            items:
            {
                description: "List of phone numbers",
                bsonType: "string"
            },
            minItems: 1,
            uniqueItems: true
        },
        url:
        {
            bsonType: "string",
            minLength:1

        },
        address:
        {
            bsonType: "string",
            minLength:1
        },
```

```
    },
},
followers:
{
    bsonType: "array",
    items:
    {
        description: "List of followers",
        bsonType: "objectId"
    },

    uniqueItems: true
},
posts:
{
    bsonType: "array",
    items:
    {
        bsonType: "objectId"
    },
    uniqueItems: true
},
jobs:
{
    bsonType: "array",
    items:
    {
        bsonType: "objectId"
    },

    uniqueItems: true
},
affiliated_pages:
{
    bsonType: "array",
    items:
    {
        bsonType: "objectId"
    },
    uniqueItems: true
```

```
},
about:
{
   bsonType:"object",
   properties:
   {
      industry:
      {
         bsonType: "string",
         minLength:1
      },
      locations:
      {
         bsonType: "array",
         items:
         {
            description: "List of locations",
            bsonType: "string"
         },
         minItems: 1,
         uniqueItems: true
      },
      website:
      {
         bsonType: "string",
         minLength:1

      },
      sector_type:
      {
         bsonType: "string",
         minLength:1
      },
      company_size:
      {
         bsonType: "string",
         minLength:1
      },
      specialities:
      {
```

```
                    bsonType: "string",
                    minLength:1

                },
            },
        },
    },
},
validationLevel: "strict",
validationAction: "error",
})
```

Users:

```
db.createCollection("users",
{
   validator:
   {
      $jsonSchema:
      {
         title: "user",
         description:"This is the collection of all the users of MyLinkedin",
         bsonType: "object",
         properties:
         {
            user_id:
            {
               description: "unique username of the user over myLinkedIn",
               bsonType: "string",
               minLength: 1,
               uniqueItems:true,
            },
            name:
            {
               description: "name of the user",
               bsonType: "string",
               minLength: 1,
            },
            email_id:
```

```
{
   description: "email-id of the user",
   bsonType: "string",
},
password:
{
   description: "Security password of the user",
   bsonType: "string",
},
connections:
{
   bsonType: "array",
   items:
   {
      bsonType: "object",
      required: ["user_id","mute_status"],
      properties:
      {
         user_id:
         {
            bsonType: "objectId",
         },
         mute_status:
         {
            bsonType: "bool",
         },

      },
   },
},
contact_info:
{
   bsonType: "object",
   properties:
   {
      email:
      {
         bsonType: "string",
      },
      url:
```

```
            {
                bsonType: "string",
            },
        },
    },
    introduction:
    {
        description: "description  of the page",
        bsonType: "string",
        minLength:1,
    },
    pages_followed:
    {
        bsonType: "array",
        items:
        {
            description: "List of page_ids of pages which are followed by the
user",
            bsonType: "objectId",
        },
        minItems: 1,
        uniqueItems: true,
    },
    profile_photo:
    {
        description: "URL of the profile photo",
        bsonType: "string",
    },
    wall_photo:
    {
        description: "URL of the wall photo",
        bsonType: "string",
    },
    country:
    {
        description:"Country in which user is residing",
        bsonType: "string",
    },
    skills:
    {
```

```
      description: "A list of skills of the user",
      bsonType: "array",
      items:
      {
         bsonType: "string",
         uniqueItems: true,
      },
   },
   industry:
   {
      description:"industry in which user is working ",
      bsonType:"string",
   },
   accomplishments:
   {
      bsonType: "object",
      properties:
      {
         projects:
         {
            bsonType: "array",
            items:
            {
               bsonType: "object",
               properties:
               {
                  title:
                  {
                     bsonType: "string",
                  },
                  start_date:
                  {
                     bsonType: "date",
                  },
                  end_date:
                  {
                     bsonType: "date",
                  },
                  association:
                  {
```

```
            bsonType: "string",
        },
        description:
        {
            description: "description  of the page",
            bsonType: "string",
            minLength:1,
        },
        project_url:
        {
            bsonType: "string",
        },
    },
},
courses:
{
    bsonType: "array",
    items:
    {
        bsonType: "object",
        required:["course_name","course_code"],
        properties:
        {
            course_name:
            {
                bsonType: "string",
            },
            association:
            {
                bsonType: "string",
            },
            course_code:
            {
                bsonType: "string",
            },

        },
    },
},
```

```
patents:
{
   bsonType: "array",
   items:
   {
      bsonType: "object",
      required:["patent_title","patent_no","patent_office"],
      properties:
      {
         patent_title:
         {
            bsonType: "string",
         },
         patent_no:
         {
            bsonType: "int",
         },
         patent_office:
         {
            bsonType: "string",
         },

      },
   },
},
publications:
{
   bsonType: "array",
   items:
   {
      bsonType: "object",
      properties:
      {
         title:
         {
            bsonType: "string",
         },
         publication_date:
         {
            bsonType: "date",
```

```
            },
            coauthor:
            {
               bsonType: "string",
            },
            description:
            {
               description: "description  of the page",
               bsonType: "string",
               minLength:1,
            },
            publication_url:
            {
               bsonType: "string",
            },
            publisher:
            {
               bsonType: "string",
            },
         },
      },
   },
   test_scores:
   {
      bsonType: "array",
      items:
      {
         bsonType: "object",
         properties:
         {
            title:
            {
               bsonType: "string",
            },
            date:
            {
               bsonType: "date",
            },
            association:
            {
```

```
                bsonType: "string",
            },
            description:
            {
                description: "description  of the score",
                bsonType: "string",
                minLength:1,
            },
            score:
            {
                bsonType: "int",
            },
            max_score:
            {
                bsonType: "int",
            },
        },
    },
},
languages:
{
    bsonType: "array",
    items:
    {
        bsonType: "object",
        properties:
        {
            language:
            {
                bsonType: "string",
            },
            profeciency_level:
            {
                bsonType: "int",
            },
        },
    },
},
},
```

```
background:
{
   bsonType: "object",
   properties:
   {
      work_exp:
      {
         bsonType: "array",
         items:
         {
            bsonType: "object",
            properties:
            {
               title:
               {
                  bsonType: "string",
               },
               start_date:
               {
                  bsonType: "date",
               },
               end_date:
               {
                  bsonType: "date",
               },
               company:
               {
                  bsonType: "string",
               },
               description:
               {
                  description: "description  of the job",
                  bsonType: "string",
                  minLength:1,
               },
               working_status:
               {
                  bsonType: "bool",
               },
               industry:
```

```
            {
               bsonType: "string",
            },
            location:
            {
               bsonType: "string",
            },
            emp_type:
            {
               bsonType: "string",
            },
         },
      },
   },
   education:
   {
      bsonType: "array",
      items:
      {
         bsonType: "object",
         properties:
         {
            degree:
            {
               bsonType: "string",
            },
            start_date:
            {
               bsonType: "date",
            },
            end_date:
            {
               bsonType: "date",
            },
            school_name:
            {
               bsonType: "string",
            },
            description:
            {
```

```
                        description: "description  of the degree",
                        bsonType: "string",
                        minLength:1
                    },
                    cgpa:
                    {
                        bsonType: "double",
                    },
                    field_of_study:
                    {
                        bsonType: "string"
                    },
                },
            },
        },
    },
},
validationLevel: "strict",
validationAction: "error",
})
```

Posts:

```
db.createCollection("posts",
   {
      validator:
      {
         $jsonSchema:
         {
            title: "posts",
            description:"This is the collection of all the posts created by the users",
            bsonType: "object",
            required: ["user","media","description"],
            properties:
            {
               user:
               {
                  description: "user_id of the user who created the post",
```

```
    bsonType: "string"
},
hashtags:
{
    description: "name of the company with job vacancy",
    bsonType: "array",
    items:
    {
        bsonType: "string"
    },
},
tagged_users:
{
    description: "A list of users tagged in the post",
    bsonType: "array",
    items:
    {
        bsonType: "string"
    },
},
time:
{
    bsonType: "date"
},
likes:
{
    bsonType: "array",
    items:
    {
        bsonType: "string"
    },
},
comments:
{
    bsonType: "array",
    items:
    {
        bsonType: "object",
        properties:
        {
```

```
            user_id:
            {
               bsonType:"string",
            },
            likes:
            {
               bsonType: "array",
               items:
               {
                  bsonType: "string"
               },
            },
            time:
            {
               bsonType: "date"
            },
            text:
            {
               bsonType:"string",
               minLength:1
            },
         },
      },
   },
   media:
   {
      description: "list of URL of media",
      bsonType: "array",
      items:
      {
         bsonType: "string"
      },
   },
   description:
   {
      description: "Caption for the post",
      bsonType: "string"
   },
   visibility:
   {
```

```
                description:"decide the visibility of this post",
                bsonType:"bool"
            },
        },
    },
},
validationLevel: "strict",
validationAction: "error",
})
```

Cassandra Schema:

```
CREATE TABLE user_tab(
  user_id uuid,
  name text,
  email_id text,
  pass_word text,
  connections map<uuid, boolean>, //bool for mute status
  url text,
  profile_photo text,
  wall_photo text,
  introduction text,
  pages list<uuid>,
  country text,
  location text,
  industry text,
  skills list<text>,
  activity list<text>,

  saved_posts list<uuid>,
  notifications list<text>,
  saved_jobs list<uuid>,

  profile_visibility boolean,
  headline_visibility boolean,
  education_visibility boolean,
  details_visibility boolean,
  activity_visibility boolean,

  user_posts list<uuid>,
  projects list<map<text,text>>, //
  awards list<map<text,text>>,
  projects list<map<text,text>>,
  test_scores list<map<text,text>>,
  languages map<text, int>,
  organisations list<text>,
  courses list<map<text, text>>,
  patent list<map<text,text>>,
  publications list<map<text,text>>,

  work_ex list<map<text,text>>,
  education list<map<text,text>>,
```

```
  license_certi list<map<text,text>>,
  volunteer_ex list<map<text,text>>,
  PRIMARY KEY (user_id)
);

CREATE TABLE posts(
  user_id uuid,
  hashtags list<text>,
  tagged_users list<uuid>,
  creation_time timestamp,
  likes int,
  liked_by list<uuid>,
  comments list<uuid>,
  media list<text>,
  description text,
  visibility map<text,boolean>
);

CREATE TABLE comments (
  post_id uuid,
        user_id uuid,
        likes int,
        likedBy list<uuid>,
        time timestamp,
        commentText text,
        PRIMARY KEY (post_id, user_id, time)
);

CREATE TABLE page(
        page_name text,
  profile_photo text,
  wall_photo text,
  description text,
  followers list<uuid>,
  posts list<uuid>,
  affiliated_pages list<uuid>,
  jobs list<uuid>,
  email text,
  url text,
  phone_no text,
  address text,
  industry text,
  website text,
```

```
 company_size int,
 head_quarters text,
 locations list<text>,
 sector_type map<text, boolean>,
 PRIMARY KEY (page_name)
);

CREATE TABLE jobs(
        position text,
 company text,
 industry text,
 description text,
 recruiting_status boolean,
 applicants list<uuid>,
 salary text,
 job_type map<text, int>,
 PRIMARY KEY (company, position)
);

CREATE TABLE hashTag (
        name text,
        no_of_posts int,
        posts list<uuid>,
        followers list<uuid>
 PRIMARY KEY (name)
);
```

Queries to be run on database for use cases :

1. ***Name:*** Signup
   Description:
   1. Search for the entered email_id in the user table.
   2. If email_id already exists the show error message that the user already registered.
   3. Else INSERT new user in the user document/table.

2. ***Name:*** Login
   Description:
   1. Search for the entered email_id in the user document/table.

2. If email_id || user_id not present then show error message that user doesn't exist.
3. Else if entered credentials = = database credentials =>start a new session and turn logged_in=true

3. ***Name:*** Like a post
   Description:
   a. INSERT the *user_id* to the *likers* array of the post .
   b. likes +=1
   c. INSERT this in the activity array of the user.
   d. INSERT a new notification in the *notifications* array of the *user_id* associated with the post.?

4. ***Name:*** Unlike a post
   Description:
   a. REMOVE the *user_id* to the *likers* array of the post .
   b. likes -=1

5. ***Name:*** Comment
   Description:
   a. INSERT the *user_id* and the *comment_text* to the *comments* array of the post .
   b. comments_count +=1
   c. INSERT this in the activity array of the user.
   d. INSERT a new notification in the *notifications* array of the *user_id* associated with the post.

6. ***Name:*** add post
   Description:
   a. INSERT new post into the post document/table and generate a new post_id .

b. SEARCH for the hashtags included in this post into the hashtags , ***if*** the hashtag already exists => INSERT this post_id into an array of posts in the hashtag document ,else INSERT new object in the hashtag document .
c. *no_of_posts* +=1 (for each # included in the post)
d. INSERT notification into each user_id's notification array which are tagged into the post.
e. INSERT this in the activity array of the user.
f. INSERT this post's *post_id* in the *posts* array of the user.

7. ***Name:*** edit post
   Description:
   a. SEARCH for the post_id and UPDATE the attributes of the post.
   b. SAVE changes in the database.
   c. RETURN updated post.

8. ***Name:*** Connection request
   Description:
   a. SEARCH for the *user_id* (of the user whom you want to connect)
   b. If user_id already present in sent_invitations array , then you cannot resend the invitation
   c. Else INSERT sender's *user_id* to the *connection_requests* array of the receiver's user.
   d. INSERT this request in the *notifications* array of the receiver's user.
9. ***Name:*** Respond to connection request
   Description:
   a. If response = = "accept" => INSERT user_id of sender into receivers connections array && INSERT user_id of receiver into senders connections array .
   b. INSERT this in the *activity* array of the user.
   c. INSERT this request in the *notifications* array of the receiver user and the sender user.
   d. Else if response = = "decline" => DELETE user_id of senders from the *connection_requests* array of the receiver.
10. ***Name:*** Save post

Description:

a. INSERT the *post_id* in the *saved_post* array of users.

11. ***Name:*** View Profile

Description:

a. SEARCH for the user_id in the database
b. RETURN the profile page of user with *user_id*

12. ***Name:*** Edit Profile Intro

Description:

a. UPDATE attributes of *intro* in user profile.
b. SAVE changes into profile changes
c. Return the updated profile

13. ***Name:*** Edit Profile Photo

Description:

a. UPDATE url of *profile_photo* in user profile .
b. SAVE changes into profile in database.
c. Return the updated profile

14. ***Name:*** Wall Photo

Description:

a. UPDATE url of *wall_photo* in user profile .
b. SAVE changes into profile in database.
c. Return the updated profile

15. ***Name:*** Like a comment on a post

Description:

a. INSERT this like in the *notifications* array of the user whose comment is liked.
b. INSERT *user_id* of the user who liked the post into the likes attribute of the *comments* of the post.

16. ***Name:*** Edit post visibility

Description:

a. UPDATE the visibility setting [private/connections/public]
b. Save value in the database

17. ***Name:*** Edit open_to setting
    Description:
    a. UPDATE open_to setting [hiring/looking for job/none]
    b. SAVE status in the database .
18. ***Name:*** Cancel connection request
    Description:
    a. DELETE request from the recipient *user_id*'s *connection_requests* list.
19. ***Name:*** View my connections
    Description:
    a. DISPLAY list of all the user profiles connected to you.
20. ***Name:*** Mute a user
    Description:
    a. SEARCH for the *user_id* in the connections.
    b. UPDATE *mute_status* in the *connections* attribute = true
21. ***Name:*** Follow a page
    Description:
    a. SEARCH for the *page_id* in the connections.
    b. INSERT *user_id* of the user into the *followers* list of the page with *page_id*
    c. INSERT *page_id* in the *pages_followed* by the user with *user_id.*
    d. INSERT this in the activity list of the user.

22. ***Name:*** Follow a hashtag
    Description:
    a. SEARCH for the hashtag in the hashtags table.
    b. INSERT *hash_id* of that hashtag in the *hashtags* list of the user.
    c. INSERT this in the activity list of the user.

23. ***Name:*** Create a page
    Description:
    a. a.Search for the entered page_id in the pages table.
    b. If page_id already exists the show error message that the user has already registered.
    c. Else INSERT new page in the user document/table.

24. ***Name:*** Edit page_name

Description:

d. SEARCH for the new page_name in the page document
e. If page_name already exists show error message
f. Else UPDATE the page_name and SAVE the document.

25. ***Name:*** Edit Page Photo

Description:

a. UPDATE url of *page_photo* in pages document .
b. SAVE changes into page in database.
c. Return the updated page.

26. ***Name:*** Edit Page wall Photo

Description:

a. UPDATE url of *page_wall_photo* in pages document .
b. SAVE changes into page in database.
c. Return the updated page.

27. ***Name:*** Edit page password

Description:

a. Enter current password, if current password != *password* show error message that invalid password.
b. Else if entered password = = *password* => ask for new password.
c. UPDATE *password*.

28. ***Name:*** Edit Page Description

Description:

a. UPDATE *description* of the page and SAVE changes into the database
b. RETURN updated page

29. ***Name:*** Edit Contact information.

Description:

a. UPDATE *phone* || *email* || *website* || *address*.
b. SAVE changes into page in database.
c. Return the updated page.

30. ***Name:*** Mute a page
Description:
a. SEARCH for the *page_id* in the connections.
b. UPDATE *mute_status* in the *connections* attribute = true

31. ***Name:*** Change Privacy setting
Description:
a. UPDATE *privacy_sett* in the *user* attribute,
b. Choose profile_visibility = false or true.
c. Choose head_visibility = false or true.
d. Choose article_visibility = false or true.
e. Choose edu_visibility = false or true.
f. Choose details_visibility = false or true.
g. SAVE changes into the user in the database.

32. ***Name: Add education***
Description:
a. INSERT new education object in user's *education* field .
b. Create a *post* for a new education update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

33. ***Name: Add work experience***
Description:
a. INSERT new work-experience object in user's *work_experience* array.
b. Create a *post* for a new work_position update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

34. ***Name: Add Licence/Certificate***
Description:
a. INSERT new object in user's *licence/certificates* array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .

d. INSERT this information into the *activity* array of the user.

35. ***Name: Add Volunteer Experience***

Description:

a. INSERT new object in user's volunteer_experience field .
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

36. ***Name: Add Skill***

Description:

a. INSERT the new skill into the *skills* array of the user document.

37. ***Name: Add Course***

Description:

a. INSERT new object in user's courses array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

38. ***Name: Add Publication***

Description:

a. INSERT new object in user's *publications* array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

39. ***Name: Add Patent***

Description:

a. INSERT new object in user's *patent array*.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

40. ***Name: Add Project***

Description:

a. INSERT new object in user's *projects* array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

41. ***Name: Add Awards***

Description:

a. INSERT new object in user's *awards* array.

b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

42. ***Name: Add Test_Scores***

Description:

a. INSERT new object in user's *test_scores* array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

43. ***Name: Name: Add Language***

Description:

a. INSERT new object in user's *language* array.
b. Create a *post* for a new update for the user.
c. INSERT this into *my_items* .
d. INSERT this information into the *activity* array of the user.

44. ***Name: Save Post***

Description:

a. $push ObjectId of the post into *saved_posts* array of the *user*

45. ***Name:Save Jobs***

Description:

a. $push ObjectId of the job into the saved_jobs array of the *user.*

46. ***Name:View Notifications***

Description:

a. FIND all the notifications in the *notifications* array of the user.

47. ***Name:View my_items***

Description:

a. FIND all the objects in the *accomplishments* and *background* fields of the user.

48. ***Name:Change open to settings***

Description:

a. UPDATE *open_to* boolean settings of the user document.

49. ***Name:View activity***

Description:

a. FIND all the activities in the *activity* array of the user.

50. ***Name:Request Recommendation***

Description:

a. $push ObjectID of user_1 in the *recommendation_requests* array of the user_2

51. ***Name:Respond to a Recommendation request***

Description:

a. If accepted $push ObjectID of user_1 into user_2's recommendation array.
b. $pull the ObjectId from the *recommendation_requests* array of the user.

52. ***Name:Apply for jobs***

Description:

a. $push ObjectId of the user into *applicants* array of the user
b. $push ObjectId of jo into *my_jobs* array of the user

Shell Scripts:

1. Task: Add a user

```
db.users.insertOne(
  {
    user_id: "aayush_patel",
    name: "aayush",
    email_id: "aayush_patel@gmail.com",

    password: "123456",
    contact_info:
    {
        email: "aayush_patel@gmail.com"
    }
  })
```

1. Task:Create new post

```
db.posts.insert(
  {
    user: ,
    description: ,
    time: new Date()
  })
```

2. Task:Push a hashtag associated with the post:

```
db.posts.update({ _id: ObjectId("60f073a526065893f1cb4e03") },{ $push: { hashtags: "" } })
```

3. Task:Find Hashtag document :

```
db.hashtags.find({name:""}).pretty()
```

4. Task:Insert new hashtag

```
db.hashtags.insertOne(
   {
      name:"",
      posts:[],
      followers:[]
   }
)
```

5. Task : Add post to an existing hashtag

```
db.hashtags.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: {
posts:ObjectId("60f064e3ad1074b645a883eb")  } })
```

6. Task: Add update to the activity array

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:""  } })
```

7. Task:Edit a post

```
db.posts.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set :{field:""}})
```

9. Search for a profile of particular user_id

```
db.users.find({user_id:""}).pretty()
```

10. Add user to the likes array when he likes a post

```
db.posts.update({ name: "60f064e3ad1074b645a883eb"},{ $push: { likes:
ObjectId("60f064e3ad1074b645a883eb") } })
```

11. Remove user from the likes array when he unlikes a post

```
db.posts.update({ name: "60f064e3ad1074b645a883eb"},{ $pull: { likes:
ObjectId("60f064e3ad1074b645a883eb") } })
```

12. Search for a profile of particular user_id

```
db.users.find({user_id:""}).pretty()
```

13. Insert user id in connection request array

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{ $push: {
connection_requests:ObjectId("60f064e3ad1074b645a883eb") } })
```

14. Task: Insert connection request notification in acitvity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"There is a new
connection request" } })
```

15. Task: Accept the connection request.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: {
connections.user_id:ObjectId("60f064e3ad1074b645a883eb") } })
```

16. Task: Remove the request from connections requests

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{ $pull: {
connection_requests:ObjectId("60f064e3ad1074b645a883eb") } })
```

17. Task: Insert connection confirmed notification in activity array of both sender and receiver.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"The request you
sent has been accepted" } })
```

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"You are connected
to the user xyz by accepting the request"} })
```

18. Task: Insert a comment in a post.

```
db.posts.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: {
comments.user_id:ObjectId("60f064e3ad1074b645a883eb"), comments.likes:0,
comments.time:new Date() } })
```

19. Task: Insert comment notification in activity array of the user posting.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"New comment on a
post" } })
```

20. Task: Save a post.

```
db.posts.update({ _id: ObjectId("60f073a526065893f1cb4e03") },{ $push: {
saved_posts:ObjectId("60f064e3ad1074b645a883eb") } })
```

21. Edit profile introduction:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{introduction:"Hey there i am using linkedin"}})
```

22. Modify profile photo:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{profile_photo:"LINK_of_photo"}})
```

23. Modify wall photo:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{wall_photo:"LINK"}})
```

24. Like a comment:

```
db.posts.update({_id: ObjectId("60f073a526065893f1cb4e03")},{$push:{"comments.likes":"user_id"}})
```

25. Change visibility settings:

```
db.posts.update({_id: ObjectId("60f073a526065893f1cb4e03")},{$set:{visibility:true}})
```

26. Change open _to setting:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{open_to:"Hiring"}})
```

27. Mute a user:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe"),"connections.user_id":"aayushh11"},
        {$set:{"connections.mute_status":true}})
```

28.Insert page id into users followed pages:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{"pages_followed":ObjectId("60f064e3ad1074b645a883eb")}})
```

29. Insert user_id into page's follower list

```
db.pages.update({_id:ObjectId("60f073a526065893f1cb4e03")},{$push:{"followers":ObjectId("60f05df226065893f1cb4dfe")}})
```

30. Insert a hashtag id into users followed hashtag array:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{"pages_followed":ObjectId("60f064e3ad1074b645a883eb")}})
```

31.Insert user_id into hashtag's follower list

```
db.pages.update({_id:ObjectId("60f073a526065893f1cb4e03")},{$push:{"followers":ObjectId("60f05df226065893f1cb4dfe")}})
```

32.View a given page:

```
db.pages.find({_id:ObjectId("60f08b3a26065893f1cb4e04")}).pretty()
```

33.Insert a new page

```
db.pages.insert({name:"Microsoft"})
```

34.Edit pages name:

```
db.pages.update({_id: ObjectId("61052fb7870af319ae1f1003")},{$set:{name:"Amazon"}})
```

35. Edit profile photo:

```
db.pages.update({_id: ObjectId("61052fb7870af319ae1f1003")},{$set:{profile_photo:"LINK"}})
```

36. Edit wall photo:

```
db.pages.update({_id: ObjectId("61052fb7870af319ae1f1003")},{$set:{wall_photo:"LINK"}})
```

37. edit page password

```
db.pages.update({_id: ObjectId("61052fb7870af319ae1f1003")},{$set:{password:"123456789"}})
```

38. Edit page description:

```
db.pages.update({_id: ObjectId("61052fb7870af319ae1f1003")},{$set:{about:"Hello this is Amazon COrporation"}})
```

39. Edit page contact info:

```
db.pages.update({_id:ObjectId("61052fb7870af319ae1f1003")},{$set:{phone:"9824088779",email:"contactus@amazon.in",website:"www.amazon.in"}})
```

40. Change mute settings of a page:

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe"),"pages_followed.name":"Amazon"},{$set:{visibility:true}})
```

41 Task: Change privacy setting for profile visibility to on or off.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{profile_visibility:"True"}})
```

42. Task: Change privacy setting for head visibility to on or off.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{head_visibility:"True"}})
```

43. Task: Change privacy setting for article visibility to on or off.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{article_visibility:"True"}})
```

44. Task: Change privacy setting for education visibility to on or off.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{edu_visibility:"True"}})
```

45. Task: Change privacy setting for detials visibility to on or off.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$set:{details_visibility:"True"}})
```

46.Task: Add education of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{education.degree:"",
education.start_date:"", education.end_date:"", education.school:"", education.description:"",
education.cgpa:"", education.field_of_study:""}})
```

47.Task: Create a post for a new education update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have added new education field in  my profile,
go check it out.",time: new Date()})
```

48.Task: Education field addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new education
field has been added."  } })
```

49.Task: Add work experience of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{background.work_exp.title:"",
background.work_exp.start_date:"", background.work_exp.end_date:"",
background.work_exp.company:"", background.work_exp.description:"",
background.work_exp.industry:"", background.work_exp.working_status:"",
background.work_exp.emp_type:""}})
```

50.Task: Create a post for a new work experience update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have added new work experience in  my
profile, go check it out.",time: new Date()})
```

51.Task: Education field addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new work
experience has been added."  } })
```

52.Task: Add license and certification of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{license.title:"",
license.validTill_date:"", license.authority:"", license.description:""}})
```

53.Task: Create a post for a new education update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have added new license/certificate field in  my
profile, go check it out.",time: new Date()})
```

54.Task: License/certification field addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new licence or
certificate has been added."  } })
```

55. Task: Add volunteer experience of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{background.vol_exp.title:"",
background.vol_exp.start_date:"", background.vol_exp.end_date:"",
background.vol_exp.organisation:"",
background.vol_exp.description:"",background.work_exp.working_status:""}})
```

56. Task: Create a post for a new work experience update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have added new volunteer experience in  my
profile, go check it out.",time: new Date()})
```

57. Task: Volunteer experience addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new volunteer
experience has been added."  } })
```

58. Task: Add a skill of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{skill:""}})
```

59. Task: Add course of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.courses.cours
e_name:"", accomplishments.courses.course_code"", accomplishments.courses.association:""}})
```

60. Task: Create a post for a new work experience update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have completed a course, go check it
out.",time: new Date()})
```

61. Task: Volunteer experience addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new course
accompplishment has been added."  } })
```

62. Task: Add publication of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.publications.title:"", accomplishments.publications.publication_date:"", accomplishments.publications.coauthor:"", accomplishments.publications.description:"", accomplishments.publications.url:"", accomplishments.publications.publisher:""}})
```

63. Task: Create a post for a new publication update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have published a paper, go check it out.",time: new Date()})
```

64. Task: Volunteer experience addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new publication has been added." } })
```

65. Task: Add patent of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.patents.patent_title:"", accomplishments.patents.patent_no:"", accomplishments.patents.patent_office:""}})
```

66. Task: Create a post for a new patent update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have owned a patent on my name, go check it out.",time: new Date()})
```

67. Task: Patent addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new patent has been added." } })
```

68. Task: Add a project of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.projects.title:"", accomplishments.projects.start_date:"", accomplishments.projects.end_date:"", accomplishments.publications.description:"", accomplishments.projects.association:""}})
```

69. Task: Create a post for a new project update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have accomplished a project, go check it out.",time: new Date()})
```

70. Task: Project addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new project has been added." } })
```

71.Task: Add a award of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{awards.title:"",
awards.date:"", awards.authority:""}})
```

72.Task: Create a post for a new award update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have been awarded, go check it out.",time:
new Date()})
```

73.Task: Award addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new award has
been added." } })
```

74.Task: Add a test score of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.test_scores.titl
e:"", accomplishments.test_scores.date:"", accomplishments.publications.description:"",
accomplishments.test_scores.association:"", accomplishments.test_scores.score:"",
accomplishments.test_scores.maxscore:""}})
```

75.Task: Create a post for test update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have achieved a good score, go check it
out.",time: new Date()})
```

76.Task: Test score addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new test score
has been added." } })
```

77.Task: Add a language of a user.

```
db.users.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{accomplishments.languages.lan
guage:"", accomplishments.languages.profiency_level:""}})
```

78.Task: Create a post for test update.

```
db.posts.insert({ user:"madhav_choksi" , description:"I have learned a language, go check it
out.",time: new Date()})
```

79.Task: Language addition to be added in activity array.

```
db.users.update({ _id: ObjectId("60f05df226065893f1cb4df5") },{ $push: { activity:"A new language
has been added." } })
```

80. Task: Save a post.

```
db.posts.update({ _id: ObjectId("60f073a526065893f1cb4e03") },{ $push: {
saved_posts:ObjectId("60f064e3ad1074b645a883eb") } })
```

81. Task: Save a job.

```
db.posts.update({ _id: ObjectId("60f073a526065893f1cb4e03") },{ $push: {
saved_jobs:ObjectId("60f064e3ad1074b645a883fk") } })
```

82. View Notifications:

```
db.users.find({_id:ObjectId("60f05df226065893f1cb4dfe")},{notifications:1})
```

83. View My_items:

```
db.users.find({_id:ObjectId("60f05df226065893f1cb4dfe")},{my_items:1})
```

84. View Activity:

```
db.users.find({_id:ObjectId("60f05df226065893f1cb4dfe")},{activity:1})
```

85. Push user_id to applicants array:

```
db.jobs.update({_id:ObjectId("61052fb7870af319ae1f1003")},{$push:{applicants:ObjectId("60f05df22
6065893f1cb4dfe")}})
```

86. Push job_id to my_jobs array of the user:

```
db.jobs.update({_id:ObjectId("60f05df226065893f1cb4dfe")},{$push:{my_jobs:ObjectId("61052fb7870
af319ae1f1003")}})
```