Non-Relational Database

A scenario in which a non-relational database might be the best solution is a social media platform. Here, the company behind the platform has decided to use a non-relational database to store and manage the vast amount of data generated by its users.

Social media platform has millions of users generating large amounts of data every day which includes posts, photos, videos, and other forms of multimedia, as well as information about the users themselves, such as their location, age, gender, and interests.

In terms of the specific form that the non-relational database should take, there are a few different options to consider. Using a document-oriented non-relational database like MongoDB is well-suited to managing unstructured data, and it allows for easy querying and indexing of the data. MongoDB has also built-in support for scaling, which is considered important for this scenario. It also allows for the storage of data in a JSON-like format, which can be easily accessed and modified.

In addition to non-relational databases, there are other options to consider as well. For example, the company could use a distributed file system, such as HDFS, to store and manage the data generated by the platform's users. This would provide a scalable and fault-tolerant solution, but it may not be as efficient for querying and indexing the data as a non-relational database.

In social media scenario, a non-relational database, such as a NoSQL, may be the best solution for a number of reasons. They are highly scalable, which is essential for social media platform as they are designed to distribute data across multiple servers, allowing them to easily scale up or down as needed to handle changes in the volume and complexity of data. Another advantage is that they are more flexible. On social media platform , the structure of data is likely to change over time as new features are added and existing ones are modified. For example it may initially only allow users to post text and images, but later add support for videos, GIFs, and other types of media. In this case, a non-relational database would be better able to accommodate these changes, as it does not require a fixed schema and can easily store and retrieve data of different types and structures.

On social media platforms, users may frequently perform searches and other types of queries that involve looking up data based on multiple criteria, such as searching for posts with a certain keyword or by a specific user. Non-relational databases can store and index data in ways that make it easy to retrieve and sort based on multiple criteria.

Non-relational databases are generally more cost-effective, as they are designed to be highly scalable and can be run on inexpensive commodity hardware. It is well-suited for managing and storing large amounts of unstructured data, which is common in social media applications.

To implement a barebones prototype of this solution, I would create a NoSQL database using a platform such as MongoDB. This database would have several collections to store different types of data, including user information, posts, comments, and likes.

Below are some example records for each of these collections:

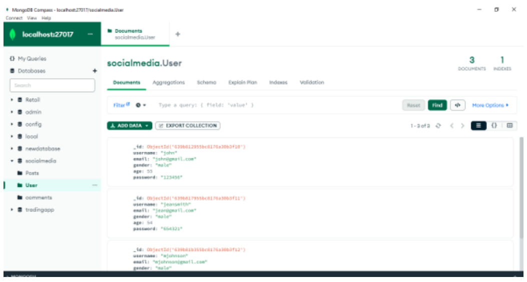
User Collection: John Doe (username: johndoe, email: john@gmail.com, gender: male,age:35, password: 123456)
Jane Smith (username: janesmith,email: jane@gmail.com, gender: male,age:55, email: janesmith@gmail.com, password: 654321)
Michael Johnson (username: mjohnson, email: mjohnson@gmail.com, gender: male,age:45,password: abcdef)

Post Collection: John Doe's post about his weekend trip (text: "Had a great time on my weekend trip! #fun #travel", date: 2021-05-08)
Jane Smith's post about her new job (text: "Excited to start my new job tomorrow! #newjob #career", date: 2021-05-09)
Michael Johnson's post about his favorite sports team (text: "Go Lakers! #basketball #lakers", date: 2021-05-10)

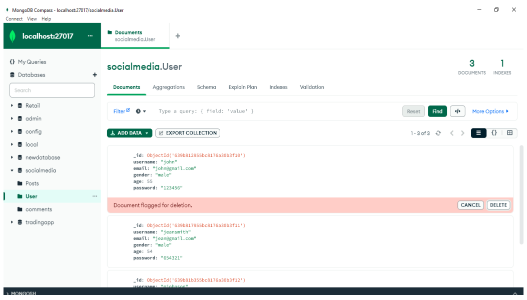
Comment Collection: Comment on John Doe's post by Jane Smith (text: "Sounds like you had a great time! #fun", date: 2021-05-09)
Comment on Jane Smith's post by Michael Johnson (text: "Congrats on the new job! #newjob", date: 2021-05-10)

Comment on Michael Johnson's post by John Doe (text: "Go Lakers! #basketball", date: 2021-05-11)

Like Collection: Like on John Doe's post by Jane Smith (date: 2021-05-09)
Like on Jane Smith's post by Michael Johnson (date: 2021-05-10)
Like on Michael Johnson's post by John Doe (date: 2021-05-11)


MongoDB Compass - localhost:27017/socialmedia.User
localhost:27017
Documents socialmedia.User
socialmedia.User
3 DOCUMENTS
1 INDEXES
Documents Aggregations Schema Explain Plan Indexes Validation
ADD DATA
EXPORT COLLECTION
1 - 3 of 3

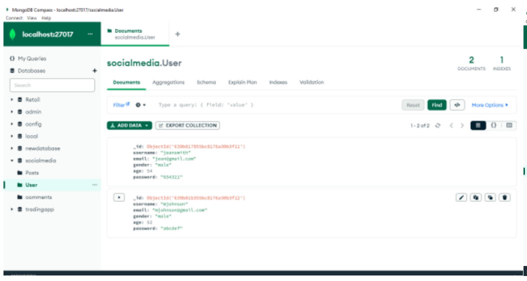
```
_id: ObjectId('639b812955bc8176a30b3f10')
username: "john"
email: "john@gmail.com"
gender: "male"
age: 55
password: "123456"

_id: ObjectId('639b817955bc8176a30b3f11')
username: "jeansmith"
email: "jean@gmail.com"
gender: "male"
age: 54
password: "654321"

_id: ObjectId('639b81b355bc8176a30b3f12')
username: "mjohnson"
email: "mjohnson@gmail.com"
gender: "male"
```



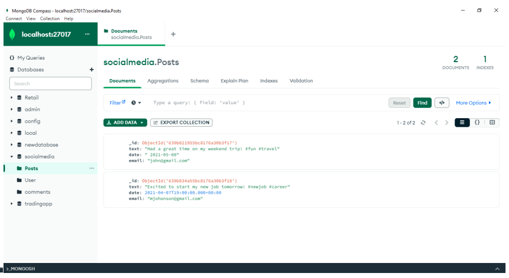
```
_id: ObjectId('639b812955bc8176a30b3f10')
username: "john"
email: "john@gmail.com"
gender: "male"
age: 55
password: "123456"
```

Document flagged for deletion.

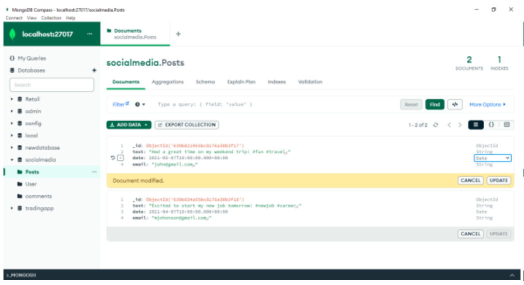
```
_id: ObjectId('639b817955bc8176a30b3f11')
username: "jeansmith"
email: "jean@gmail.com"
gender: "male"
age: 54
password: "654321"

_id: ObjectId('639b81b355bc8176a30b3f12')
username: "mjohnson"
```



```
_id: ObjectId('639b817955bc8176a30b3f11')
username: "jeansmith"
email: "jean@gmail.com"
gender: "male"
age: 54
password: "654321"

_id: ObjectId('639b81b355bc8176a30b3f12')
username: "mjohnson"
email: "mjohnson@gmail.com"
gender: "male"
age: 52
password: "abcdef"
```


MongoDB Compass - localhost:27017/socialmedia.Posts
Documents socialmedia.Posts
socialmedia.Posts
2 DOCUMENTS
1 INDEXES
1 - 2 of 2

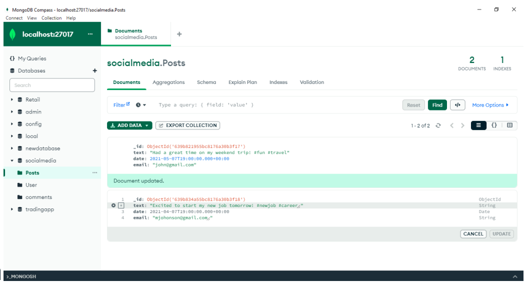
```
_id: ObjectId('639b821955bc8176a30b3f17')
text: "Had a great time on my weekend trip! #fun #travel"
date: " 2021-05-08"
email: "john@gmail.com"

_id: ObjectId('639b834a55bc8176a30b3f18')
text: "Excited to start my new job tomorrow! #newjob #career"
date: 2021-04-07T19:00:00.000+00:00
email: "mjohonson@gmail.com"
```



```
_id: ObjectId('639b821955bc8176a30b3f17')                         ObjectId
text: "Had a great time on my weekend trip! #fun #travel"         String
date: 2021-05-07T19:00:00.000+00:00                               Date
email: "john@gmail.com"                                           String
```

Document modified.

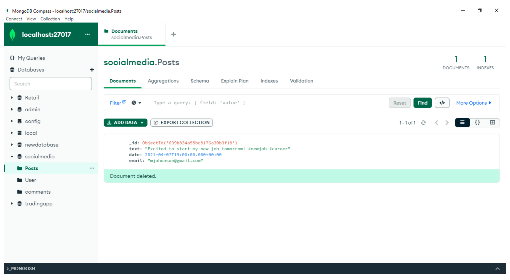
```
_id: ObjectId('639b834a55bc8176a30b3f18')                         ObjectId
text: "Excited to start my new job tomorrow! #newjob #career"     String
date: 2021-04-07T19:00:00.000+00:00                               Date
email: "mjohonson@gmail.com"                                      String
```



```
_id: ObjectId('639b821955bc8176a30b3f17')
text: "Had a great time on my weekend trip! #fun #travel"
date: 2021-05-07T19:00:00.000+00:00
email: "john@gmail.com"
```

Document updated.

```
_id: ObjectId('639b834a55bc8176a30b3f18')                         ObjectId
text: "Excited to start my new job tomorrow! #newjob #career"     String
date: 2021-04-07T19:00:00.000+00:00                               Date
email: "mjohonson@gmail.com"                                      String
```



```
_id: ObjectId('639b834a55bc8176a30b3f18')
text: "Excited to start my new job tomorrow! #newjob #career"
date: 2021-04-07T19:00:00.000+00:00
email: "mjohonson@gmail.com"
```

Document deleted.

Using this non-relational database, the application would be able to support the following functionality:

User registration and login: Users can create accounts using their email and password, and then login to their account to access the social media platform.

Posting: Users can create and share posts on the platform, including text and hashtags.

Commenting: Users can leave comments on other users' posts.
Liking: Users can like other users' posts.

Searching: Users can search for posts using keywords or hashtags.

MongoDB allows for flexible and scalable data storage. It also has built-in support for concurrency and can provide quick access to data through indexing and sharding.

In terms of data organization, I would create a collection for each type of user-generated content, such as photos, videos, and text posts. Each document within these collections would contain the relevant data for that content, such as the user who created it, the date it was created, and any associated tags or metadata.

To demonstrate the functionality of this database, I can show how it would handle common tasks in the hypothetical social media platform. For example, when a user creates a new photo post, the database would quickly store the data in the appropriate collection and index it for efficient retrieval. If a user searches for posts with a specific tag, the database can quickly retrieve and display the relevant documents from the various collections.

To ensure that the data is easily accessible, I would index the user's id in both the posts and comments collections, allowing for quick and efficient lookup of a user's posts and comments. I would also implement a search feature that allows users to search for other users by name or email.

Overall, this non-relational database solution would provide the necessary capabilities to support the requirements of the hypothetical social media platform, allowing for efficient and scalable storage and management of user-generated content.

In terms of functionality, the non-implemented application would be able to handle the requirements of a social media platform, such as allowing users to create and edit their profiles, post and comment on other users' posts, and search for other users. The non-relational database would enable the application to handle large amounts of data and provide fast access to user and post information.

To meet these requirements, the non-relational database would use a distributed architecture, with data stored across multiple nodes. This would allow for efficient scaling and support for high-volume data. The database would also use a flexible data model, allowing for the storage of unstructured data such as images and videos.

In terms of how the non-implemented application would use the designed data, users would be able to upload and share content on the platform. The database would store this content and make it available for other users to view and interact with. The database would also support real-time updates, allowing for a seamless user experience.

Overall, our solution effectively meets the requirements for a social media platform .