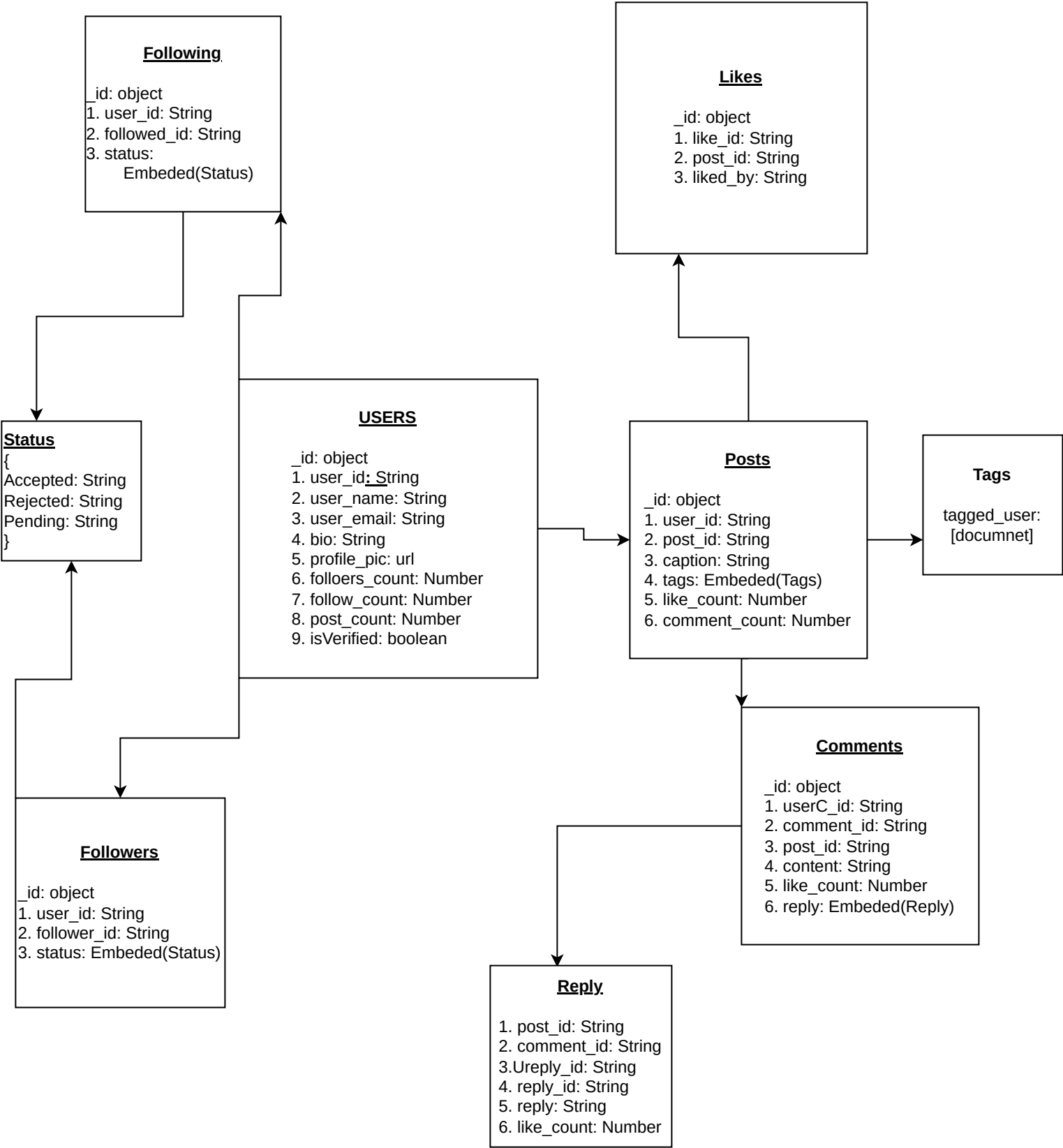


Database Design for Instagram



## Inserting data to database (insta) according to the schema design

### Insert user

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Usres' collection under the 'Insta' database selected. The main editor displays a successful `insertOne` operation on the `db.Usres` collection. The document being inserted contains the following fields:

```
1 db.Usres.insertOne(
2 {
3   user_id: "ankur123",
4   user_name: "Ankur Singh",
5   user_email: "ankur123@gmail.com",
6   bio: "The perfect bio doesn't exist!",
7   profile_pic: "https://profile/pic/image",
8   followers_count: 5,
9   follow_count: 3,
10  post_count: 2,
11  isverfied: false
12 })
```

The result pane shows the inserted document with the following structure:

```
1 {
2   "_id" : ObjectId("64b62807f767f8806c72ad00"),
3   "user_id" : "ankur123",
4   "user_name" : "Ankur Singh",
5   "user_email" : "ankur123@gmail.com",
6   "bio" : "The perfect bio doesn't exist!",
7   "profile_pic" : "https://profile/pic/image",
8   "followers_count" : 5,
9   "follow_count" : 3,
10  "post_count" : 2,
11  "isverfied" : false
12 }
```

### Followers of user

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Followers' collection under the 'Insta' database selected. The main editor displays a successful `insertOne` operation on the `db.Followers` collection. The document being inserted contains the following fields:

```
1 db.Followers.insertOne(
2 {
3   user_id: "ankur123",
4   follower_id: "ayush123",
5   status: ["pending", "accepted", "rejected"]
6 })
```

The result pane shows the inserted document with the following structure:

```
1 {
2   "_id" : ObjectId("64b63eb5f767f8806c72ad02"),
3   "user_id" : "ankur123",
4   "follower_id" : "ayush123",
5   "status" : [ "pending", "accepted", "rejected" ]
6 }
```

## User Following

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Following' collection selected under the 'mongo\_class' database. The main pane displays a query to insert a document into the 'Following' collection:

```
1 db.Following.insertOne(  
2   {  
3     user_id: "ankur123",  
4     followed_id: "ayush123",  
5     status: ["pending", "accepted", "rejected"]  
6   }  
7 )
```

The results pane shows the inserted document:

```
1 {  
2   "_id" : ObjectId("64b629ccf767f8806c72ad01"),  
3   "user_id" : "ankur123",  
4   "followed_id" : "ayush123",  
5   "status" : [ "pending", "accepted", "rejected" ]  
6 }
```

## User post

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Posts' collection selected under the 'mongo\_class' database. The main pane displays a query to insert a document into the 'Posts' collection:

```
1 db.Posts.insertOne(  
2   {  
3     user_id: "ankur123",  
4     post_id: "post1",  
5     caption: "be yourself",  
6     tages: ["ayush123"],  
7     like_count: 5,  
8     comment_count: 2  
9   }  
10 )
```

The results pane shows the inserted document:

```
1 {  
2   "_id" : ObjectId("64b64283f767f8806c72ad03"),  
3   "user_id" : "ankur123",  
4   "post_id" : "post1",  
5   "caption" : "be yourself",  
6   "tages" : [ "ayush123" ],  
7   "like_count" : 5,  
8   "comment_count" : 2  
9 }
```

## Likes on post

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Likes' collection under the 'Insta' database selected. The main editor shows a JavaScript snippet to insert a document into the 'Likes' collection:

```
1 db.Likes.insertOne(  
2 {  
3   like_id: "like1",  
4   post_id: "post1",  
5   liked_by: "ayush123"  
6 })
```

Below the editor, the 'Likes' collection is shown with 1 document. The document details are displayed in JSON format:

```
1 {  
2   "_id" : ObjectId("64b64ea2f767f8806c72ad05"),  
3   "like_id" : "like1",  
4   "post_id" : "post1",  
5   "liked_by" : "ayush123"  
6 }
```

## Comments on post

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, with the 'Comments' collection under the 'Insta' database selected. The main editor shows a JavaScript snippet to insert a document into the 'Comments' collection:

```
1 db.Comments.insertOne(  
2 {  
3   post_id: "post1",  
4   userC_id: "ayush123",  
5   comment_id: "com1",  
6   content: "nice",  
7   like_count: 2,  
8   reply: [  
9     {  
10    post_id: "post1",  
11    comment_id: "com1",  
12    UReply_id: "ankur123",  
13    reply_id: "rep1",  
14    reply: "thanks",  
15    like_count: 1  
16    }  
17  ]  
18 })
```

Below the editor, the 'Comments' collection is shown with 1 document. The document details are displayed in JSON format:

```
1 {  
2   "_id" : ObjectId("64b64e03f767f8806c72ad04"),  
3   "post_id" : "post1",  
4   "userC_id" : "ayush123",  
5   "comment_id" : "com1",  
6   "content" : "nice",  
7   "like_count" : 2,  
8   "reply" : [  
9     {  
10    "post_id" : "post1",  
11    "comment_id" : "com1",  
12    "UReply_id" : "ankur123",  
13    "reply_id" : "rep1",  
14    "reply" : "thanks",  
15    "like_count" : 1  
16    }  
17  ]  
18 }
```

## Error occurred when try to insert with mongoose

Using mongoose try to connect database then throws error, can not connect to database this happens because nosqlbooster can not connect to localhost then for use of nosqlbooster try to connect with atlas. But mongoose can not get the server (cluster) api.

```
1 const mongoose = require('mongoose')
2
3 const main = async () =>{
4   await mongoose.connect("mongodb+srv://ankursingh1:ankur735@cluster0.fmy52sv.mongodb.net/mongo_class");
5   const UsersSchema = new mongoose.Schema({
6     name:String
7   });
8
9   const UsersModel = mongoose.model('Users', UsersSchema);
10  let data = new UsersModel({name:"abhi"});
11  let result = await data.save();
12  console.log(result);
13
14 }
15
16 main()
```

```
admin2@admin2-HP-EliteBook:~/Desktop/notes$ node index.js
admin2@admin2-HP-EliteBook:~/Desktop/notes$ node index.js
/home/admin2/Desktop/notes/node_modules/mongoose/lib/connection.js:791
    err = new ServerSelectionError();
           ^
MongooseServerSelectionError: Could not connect to any servers in your MongoDB Atlas cluster. One common reason is that you're trying to access the database from an IP that isn't whitelisted. Make sure your current IP address is on your Atlas cluster's IP whitelist: https://www.mongodb.com/docs/atlas/security-whitelist/
    at handleConnectionErrors (/home/admin2/Desktop/notes/node_modules/mongoose/lib/connection.js:791:11)
    at NativeConnection.openUri (/home/admin2/Desktop/notes/node_modules/mongoose/lib/connection.js:766:11)
    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)
    at async main (/home/admin2/Desktop/notes/index.js:4:5) {
  reason: TopologyDescription {
```