

[Repository](#)

[Live Link](#)

Social Media Platform Development

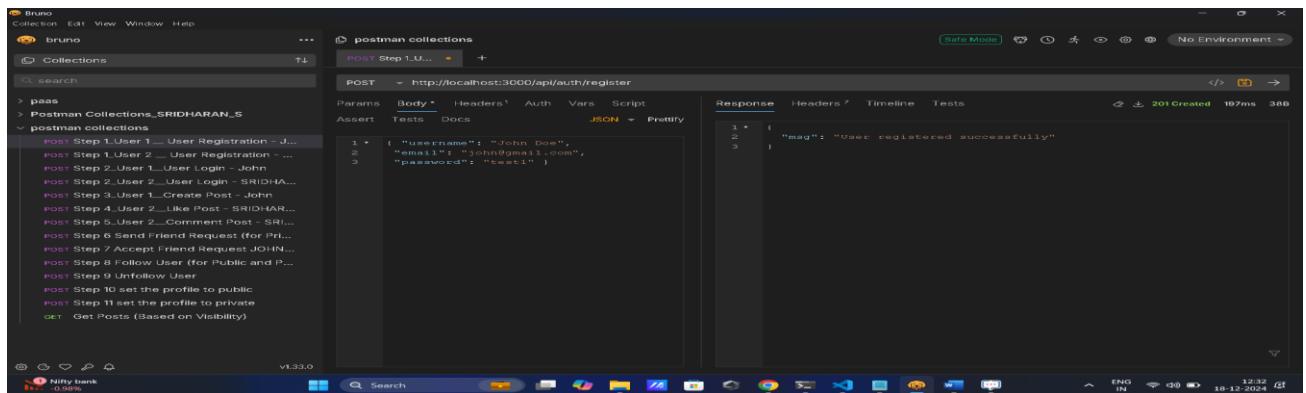
Project Overview

The purpose of this project is to create a social media platform inspired by Facebook's functionality. The application will allow users to register, interact, and manage their relationships and posts. It is built using **Node.js**, **Express**, and **MongoDB** for database management.

Features

1. User Registration

- Users can register and create profiles with basic details.
- Required fields: Username, Email, Password.
- Passwords are securely hashed using **bcrypt**.
- Example:



test.users

STORAGE SIZE: 84KB LOGICAL DATA SIZE: 506B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 70KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass®

Filter Type a query: { field: 'value' } **Reset** **Apply** Options *

QUERY RESULTS: 1-2 OF 2

```

_id: ObjectId("67618e65239fcfa2386229e6b")
username: "John Doe"
email: "john.doe11@gmail.com"
password: "$2a$10$O9yfRhcc5etonRd1X0gZnfhx0H6KpAL02Y5SHVy0U2XRQ1TzY"
profileVisibility: "private"
followers: Array (empty)
following: Array (empty)
posts: Array (empty)
updatedAt: 2024-12-18T05:13:13.671+00:00

_id: ObjectId("67618e65239fcfa2386229e6c")
username: "SRIDHARAN S."
email: "srishanthreddy11.com"
password: "$2a$10$O9yfRhcc5etonRd1X0gZnfhx0H6KpAL02Y5SHVy0U2XRQ1TzY"
profileVisibility: "private"
  
```

System Status: All Good

©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

30°C Haze 12:33 18-12-2024 ENG IN

Login

postman

Collection Edit View Window Help

bruno

Collections

search

peas

Postman Collections_SRIDHARAN_S

postman collections

POST Step 10 s... POST Step 2.U... GET Get Posts ... POST Step 2.U... POST Step 11 s... +

POST http://localhost:3000/api/auth/login

Params Body * Headers * Auth Vars Script Assert Tests Docs JSON Pretty

```

1   {
2     "email": "john@gmail.com",
3     "password": "test1"
  
```

Response Headers Timeline Tests 200 OK 242ms 188B

```

1   {
2     "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
3     .eyJpcz42VjSWQj01ZlZNYXOGUIMzMSZjYTisODYyMjllNml
4     iLCJpYXQiOjE3MzQ1MTQ1NTMsImVAcCIGMtcsNDUxODE1Mz0
5     .fbbYu8ViHF1uTykoOle2iBXoJ26P9uVyeFoayXF5vFg"
  
```

NIFTY -0.56% 15:05 18-12-2024 ENG IN

2. Follow System

- Users can follow or unfollow other registered users.
- A followers and following mechanism is maintained in the database.
- Follow

- Example: Private Account Means
- **profileVisibility : "private"**
- Error: "msg": "Cannot follow a private account without an accepted friend request."

- **Screenshot:**

The screenshot shows the MongoDB Compass interface with the 'test' database selected. The 'test.users' collection is open, displaying two documents. Both documents have the following structure:

```

{
  "_id": "ObjectId('6761edb393fca2386229e6e')",
  "username": "John Doe",
  "email": "john@gmail.com",
  "password": "SecurePassword1234567890!@#%^&*",
  "profileVisibility": "private",
  "followers": [],
  "posts": []
}
{
  "_id": "ObjectId('6761edb393fca2386229e6f')",
  "username": "SRIDHARAN S",
  "email": "sridharan.s@gmail.com",
  "password": "SecurePassword1234567890!@#%^&*",
  "profileVisibility": "private"
}

```

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3000/api/friends/follow`. The response status is 403 Forbidden, and the response body is:

```

{
  "msg": "Cannot follow a private account without an accepted friend request."
}

```

- Example: Public Account Means
- Check Users table
profileVisibility : "public"

Screenshot:

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar includes sections for Overview, DATABASE (with a dropdown for social_media_p...), Clusters (with a dropdown for test), SERVICES (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), SECURITY (Quickstart, Backup, Database Access, Network Access, Advanced), and a Goto link. The main area displays the 'test' database and the 'users' collection. The 'test.users' collection has 2 documents with logical sizes of 606B and total size of 72KB. A query is being run to find a user with ID 67618edb393fca2386229e6e. The results show the user's details including username, email, password, profile visibility, following, and posts.

The screenshot shows the Postman application interface. The left sidebar lists collections: paaS, Postman Collections_SRIDHARAN_S, and postman collections. Under postman collections, several steps are listed: POST Step 1_User Registration - John, POST Step 1_User 2_User Registration - ... , POST Step 2_User 1_User Login - John, POST Step 2_User 2_User Login - SRIDHARAN, POST Step 3_User 1_Create Post - John, POST Step 4_User 2_Like Post - SRIDHARAN, POST Step 5_User 2_Comment Post - SRIDHARAN, POST Step 6_Send Friend Request (for Public and Private), POST Step 7_Accept Friend Request JOHN..., POST Step 8_Follow User (for Public and Private), POST Step 9_Unfollow User, POST Step 10_Set Profile to Public, POST Step 11_Set Profile to Private, and GET_Get Posts (Based on Visibility). The main panel shows a POST request to http://localhost:3000/api/friends/follow with the ID 67618edb393fca2386229e6e. The response tab shows a 201 Created status with a JSON response body: { "msg": "Follow successful.", "following": { "_id": "67627630e26d0aab22c43ad1", "_v": 0 } }

Following means which you follow that user id

DB:

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various service icons and a 'Clusters' section. Under 'Clusters', there's a 'test' cluster with a 'follows' collection selected. The main panel shows the 'test.follows' collection details, including storage size (24KB), logical data size (64B), total documents (1), and index details. A 'Find' button is highlighted. Below it, a query result table shows one document with the following data:

_id	following
{ObjectId('67627630e26d0aab22c43ad1')}	{ObjectId('67628ed3993fc2a386229e6e')}

- Unfollow
- Example: Screenshot

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections and environments. The main panel shows a POST request to 'http://localhost:3000/api/friends/unfollow/67618edb393fc2a386229e6e'. The 'Headers' tab is selected, showing 'Authorization' and 'Content-Type' headers. The 'Body' tab is also visible. To the right, the response pane shows a JSON object with a single key 'msg' and the value 'Unfollow successful.'.

The screenshot shows the MongoDB Atlas interface again. The 'test.follows' collection now has 0 documents, as indicated by the 'QUERY RESULTS' count at the top of the results table.

3. Posting & Interactions

- Users can:
 - Create posts with images (handled via **Multer** for file uploads).
 - Like or comment on posts.
- Posts contain fields for:
 - userId: Creator of the post.
 - content: The text of the post.
 - image: Uploaded image.
 - likes: List of users who liked the post.
 - comments: Array of comments.

- Example:

Screenshot:

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3000/api/posts`. The request body is set to **Multipart Form** with the following fields:

Key	Value
content	"This is a post!"
image	
userId	6761ee53393fca2386229e6b

The response from the server is a JSON object:

```
1  {
2   "id": "676279t3e26d0aab22c43ad7",
3   "user": {
4     "_id": "6761ee53393fca2386229e6b",
5     "username": "John Doe",
6     "email": "john@gmail.com"
7   },
8   "content": "This is a post!",
9   "image": "uploads\\1734506995721.jpg",
10  "likes": [],
11  "comments": [],
12  "createdAt": "2024-12-18T07:29:55.788Z",
13  "updatedAt": "2024-12-18T07:29:55.788Z",
14  "v": 0
15 }
```

Image upload - Uploads folder

- Comment & Likes
 - Example Screenshot Comments

The screenshot shows the MongoDB Atlas interface for a cluster named 'Self'. The left sidebar lists various services and security features. The main area displays the 'test.users' collection with two documents. Each document includes fields like '_id', 'username', 'email', 'password', 'profileVisibility', and arrays for 'followers', 'following', and 'posts'. The interface also features a search bar, filters, and query results.

System Status: All Good

©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like 'paas' and 'Postman Collections_SRIDHARAN_S'. The main area displays a POST request to 'http://localhost:3000/api/posts/comment/676279f3e26d0aab22c43ad7'. The response tab shows a JSON object indicating the comment was added successfully:

```
1  {
2    "msg": "Comment added successfully",
3    "newComment": {
4      "post": "676279f3e26d0aab22c43ad7",
5      "comment": "Nice post John Doe!",
6      "_id": "67627d74cb252e38b50a3688",
7      "createdAt": "2024-12-18T07:44:52.805Z",
8      "_v": 0
9    }
10 }
```

● Example Screenshot Likes

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like 'paas' and 'Postman Collections_SRIDHARAN_S'. The main area displays a POST request to 'http://localhost:3000/api/posts/like/676279f3e26d0aab22c43ad7'. The response tab shows a JSON object indicating the post was liked successfully:

```
1  {
2    "msg": "Post liked successfully",
3    "post": {
4      "_id": "676279f3e26d0aab22c43ad7",
5      "user": "67618e53393fca2386229e6b",
6      "content": "\"This is a post!\"",
7      "image": "uploads\\1734506995721.jpg",
8      "likes": [
9        "67618edb393fca2386229e6e"
10      ],
11      "comments": [
12        "67627d74cb252e38b50a3688"
13      ],
14      "createdAt": "2024-12-18T07:29:55.788Z",
15      "updatedAt": "2024-12-18T08:06:04.742Z",
16      "_v": 4
17    }
18 }
```

● Check Posts Schema

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases (social_media_platform) and collections (posts). The main panel displays the 'posts' collection with one document. The document structure is as follows:

```

_id: ObjectId("676279f7e260aa02c43ad7")
user: ObjectId("6761a4d5399fc2a23862294eb")
content: "This is a post!"
image: "uploads/17345969695721.jpg"
likes: Array (1)
  0: ObjectId("6761a8ed93fca23862294eb")
comments: Array (1)
  0: ObjectId("676279f74c9252ac8b49a9688")
createdAt: 2024-12-18T07:29:58.788+00:00
updatedAt: 2024-12-18T08:06:04.742+00:00
__v: 4

```

- Likes : Array (1)
- Comments : Array(1)

- Check Comments Schema

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases (social_media_platform) and collections (comments). The main panel displays the 'comments' collection with one document. The document structure is as follows:

```

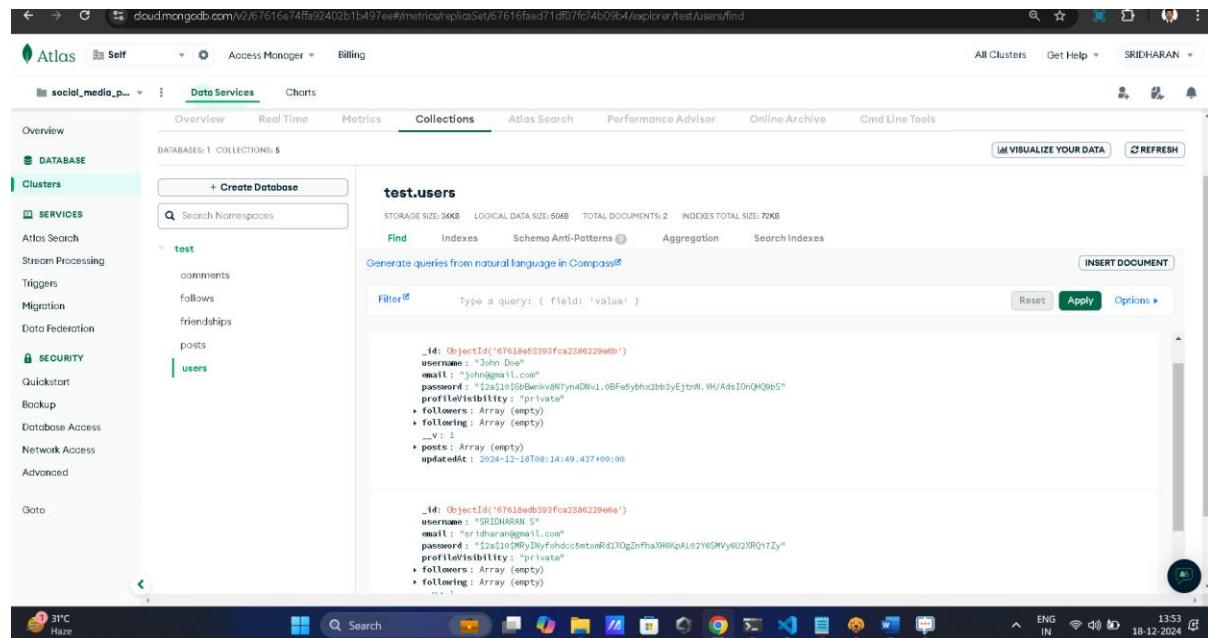
_id: ObjectId("676279f74c9252ac8b49a9688")
post: ObjectId("676279f7e260aa02c43ad7")
comment: "Nice post John Doe!"
createdAt: 2024-12-18T07:44:52.895+00:00
__v: 0

```

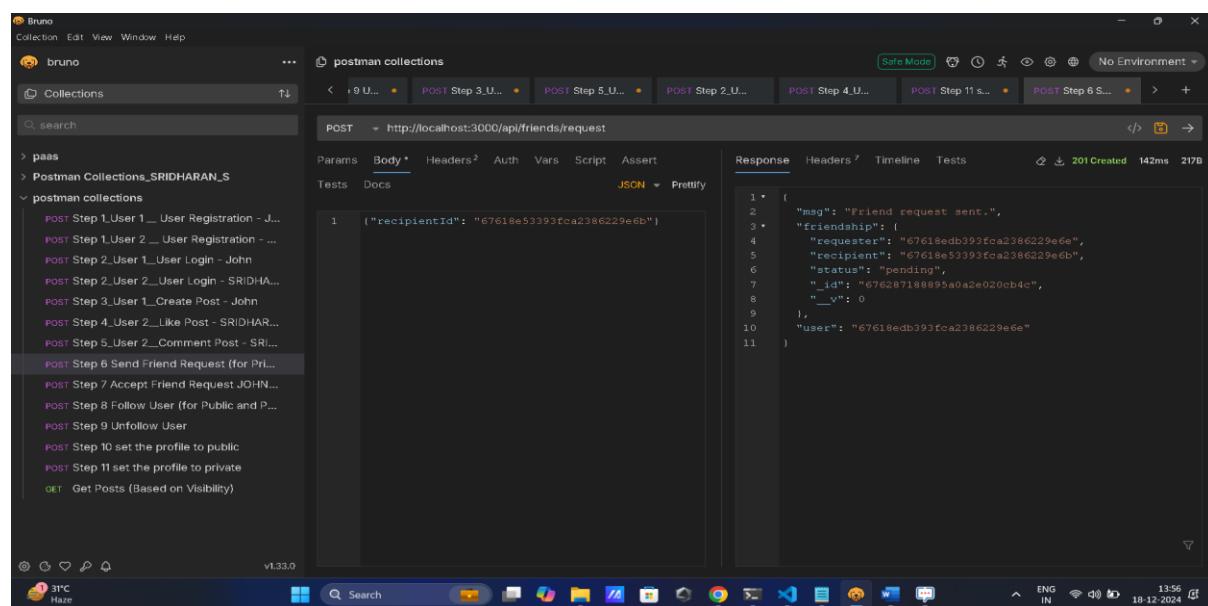
4. Friendship Mechanism

- Users can send, accept friend requests.
 - Example:

Screenshot:



- Send Request



The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various tabs like Overview, Database, Clusters, Services, and Security. The main area shows a database named 'test' with a collection named 'friendships'. The collection details indicate 1 document, 24KB storage, and 97B logical data size. A query result table shows one document with the following fields:

```

_id: ObjectId('676287188895a0a2e020cb4c')
requester: ObjectId('67616edb393fc2386229e6e')
recipient: ObjectId('67616ed53393fc2386229e6b')
status: "pending"
__v: 0

```

- Status : "pending"

- Accept Request

The screenshot shows the Postman application interface. The left sidebar lists collections and environments. The main window shows a POST request to 'http://localhost:3000/api/friends/accept'. The response tab displays the following JSON data:

```

1  {
2    "msg": "Friend request accepted.",
3    "friendship": {
4      "_id": "676287188895a0a2e020cb4c",
5      "requester": "67616edb393fc2386229e6e",
6      "recipient": "67616ed53393fc2386229e6b",
7      "status": "accepted",
8      "__v": 0
9    },
10   "userId": "67616ed53393fc2386229e6b"
11 }

```

The screenshot shows the MongoDB Atlas Data Services interface. The left sidebar is titled 'Clusters' and lists various services like Stream Processing, Migration, and Security. The main area shows a database named 'test' containing a collection named 'friendships'. The collection details indicate a storage size of 96KB, logical data size of 98B, and one document. The document content is shown as:

```
_id: ObjectId('67618e74f592402b1b497ee')
requester: ObjectId('67618e6db393fc2238829e6e')
recipient: ObjectId('67618e53393fc2238829e6b')
status: "accepted"
__v: 0
```

- Status : "accepted"

5. Profile Visibility

- Two visibility options for user profiles:
- Public

The screenshot shows the Postman application interface. On the left, the sidebar displays a collection named 'Bruno' containing several API steps: Step 1_1, Step 1_2, Step 2_1, Step 2_2, Step 3, Step 4, Step 5, Step 6, Step 7, Step 8, Step 9, Step 10, Step 11, and a GET step for 'Get Posts'. The main panel shows a POST request to 'http://localhost:3000/api/users/profile/public' with a status of 200 OK. The Headers tab shows an 'Authorization' header with the value 'N0gltgS5gN43t2NvN7PY'. The Response tab displays a JSON object with a single key-value pair: 'msg': 'Profile visibility set to public'.

The screenshot shows the MongoDB Atlas Data Services interface. The left sidebar lists various services and clusters. The main area shows the 'test' database and the 'users' collection. A single document is listed with the following fields and values:

```
_id: ObjectId('67618e03393fca2386229e6b')
username: "John Doe"
email: "john@gmail.com"
password: "1234567890"
profileVisibility: "public"
followers: []
following: []
posts: []
updatedAt: 2024-12-18T09:06:34.204+00:00
```

- Private

The screenshot shows the Postman application interface. In the left sidebar, there are several collections listed under 'paas' and 'postman collections'. The main workspace displays a POST request to 'http://localhost:3000/api/users/profile/private/67618e53393fc2386229e6b'. The 'Headers' tab shows an 'Authorization' header with the value 'Bearer eyJhbGciOiJUzI1NiisIn...' and a checked checkbox. The 'Response' tab shows a JSON response with three lines of code: 1. `{"msg": "Profile visibility set to private"}`. The status bar at the bottom indicates 'v1.33.0' and the system tray shows the date and time as '18-12-2024 14:37'.

The screenshot shows the MongoDB Atlas Data Explorer interface. The left sidebar has sections for 'Atlas', 'Self', 'Access Manager', 'Data Services', and 'Clusters'. Under 'Clusters', 'Cluster0' is selected. The main area shows the 'test' database with the 'users' collection. A search bar at the top right says 'VISUALIZE YOUR DATA' and 'REFRESH'. Below it, a 'Find' button is highlighted. A query input field contains 'Type a query: { field: 'value' }'. The 'QUERY RESULTS' section shows '1-2 OF 2' results, each being a document from the 'users' collection. One document is partially visible:

```

_id: ObjectId('67618e53393fc2386229e6b')
username: "John Doe"
email: "john@gmail.com"
password: "$2a$10$GbmekvN7yndDnV1.0BFa5ybx2bb3yEjtnN.WhAdzQhQ9BS"
profileVisibility: "private"
followers: Array (empty)
following: Array (empty)
_v: 1
posts: Array (empty)
updatedAt: 2024-12-18T09:07:46.131+00:00
  
```

The system tray at the bottom shows the date and time as '18-12-2024 14:39'.

- **Public:** Allows all followers to view posts.

The screenshot shows the MongoDB Atlas Data Services interface. On the left, the sidebar lists the database (social_media_posts) and collection (test). The main area shows the 'Find' results for the collection. Two documents are listed:

```

{
  "_id": "ObjectId('67618e53393fca2386229e6b')",
  "username": "John Doe",
  "email": "john@gmail.com",
  "password": "1234567890!QWERTY",
  "profileVisibility": "public",
  "followers": [],
  "following": [],
  "posts": []
}

{
  "_id": "ObjectId('67618e53393fca2386229e6b')",
  "username": "SRIDHARAN S",
  "email": "srividharan@gmail.com",
  "password": "1234567890!QWERTY",
  "profileVisibility": "public",
  "followers": [],
  "following": [],
  "posts": []
}

```

The interface includes a search bar, filter options, and an 'INSERT DOCUMENT' button. The bottom status bar shows system information like temperature (31°C), battery level (Haze), and system date (18-12-2024).

John Already Post Pictures Sridharan To View John Posts, Comments, Likes

The screenshot shows the Postman application interface. The left sidebar shows a collection named 'bruno' with various steps. The current step is 'GET Step 10 s...', which is a GET request to `http://localhost:3000/api/posts/posts/67618e53393fca2386229e6b/67618edb393fca2386229e6b`. The response pane displays the JSON data for a post:

```

{
  "_id": "676279f3e26d0aab22c43ad7",
  "user": {
    "_id": "67618e53393fca2386229e6b",
    "username": "John Doe",
    "email": "john@gmail.com"
  },
  "content": "This is a post!",
  "image": "uploads\\1734506959721.jpg",
  "likes": [
    {
      "_id": "67618edb393fca2386229e6b",
      "username": "SRIDHARAN S"
    }
  ],
  "comments": [
    {
      "_id": "67627d74cb252e38b50a3688",
      "comment": "Nice post John Doe!"
    }
  ],
  "createdAt": "2024-12-18T07:29:55.788Z",
  "updatedAt": "2024-12-18T08:06:04.742Z",
  "__v": 4
}

```

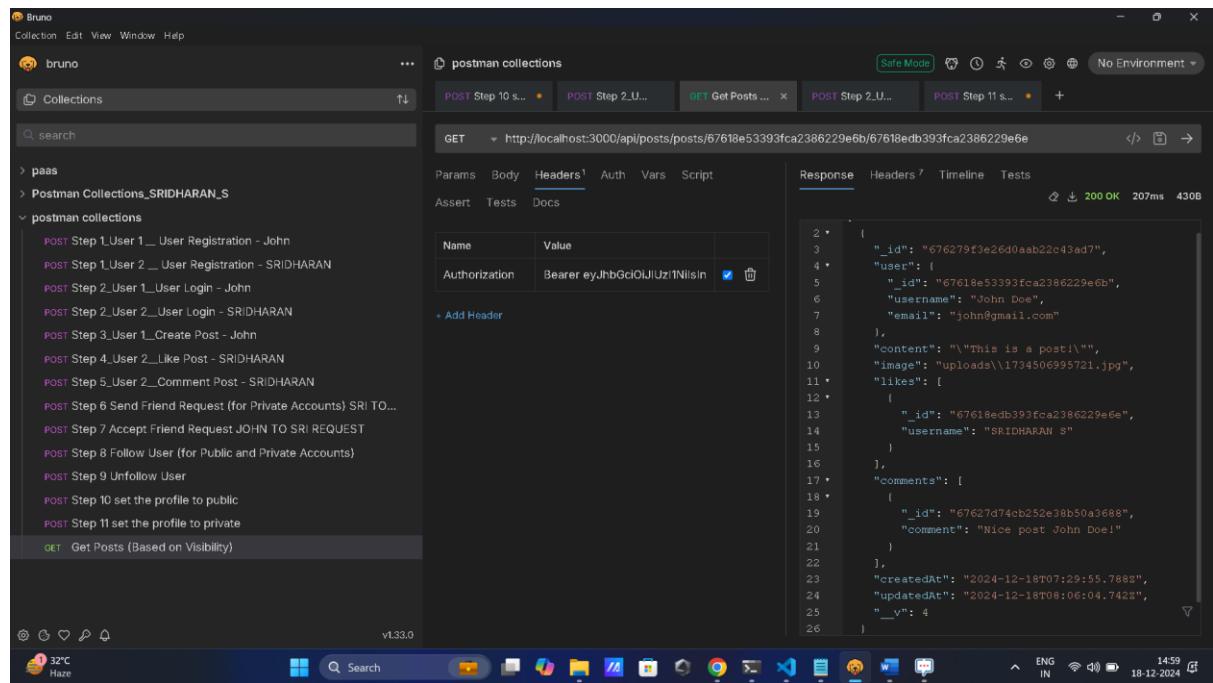
The bottom status bar shows system information like temperature (31°C), battery level (Haze), and system date (18-12-2024).

- **Private:** Only friends can view posts.

The screenshot shows the MongoDB Atlas Data Services interface. The left sidebar shows the 'social_media_p...' cluster, with the 'test' database selected. Under 'test', the 'users' collection is highlighted. The main panel displays the schema for the 'test.users' collection, which includes fields like '_id', 'username', 'email', 'password', 'profileVisibility', 'followers', 'following', 'posts', and 'updatedAt'. Below the schema, there is a 'Find' section with a query builder and a preview of the results. The results show two documents: one for 'John Doe' and one for 'SRIDHARAN S'. The desktop taskbar at the bottom shows various application icons.

This screenshot is similar to the previous one but shows the 'friendships' collection instead of the 'users' collection. The main panel displays the schema for the 'test.friendships' collection, which includes fields like '_id', 'requester', 'recipient', 'status', and 'v'. Below the schema, there is a 'Find' section with a query builder and a preview of the results. The results show one document with status 'accepted'. The desktop taskbar at the bottom shows various application icons.

"status" : "accepted"



Vs code Console Log:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a set of system icons. The title bar displays "social-media-platform".

The left sidebar contains the Explorer view, showing a tree structure of files and folders:

- OPEN EDITORS:
 - JS server.js
 - JS friendshipRoutes.js
 - JS postRoutes.js
 - X JS postController.js
- SOCIAL-MEDIA-PLATFORM:
 - > config
 - > controllers
 - JS authController.js
 - JS friendshipController.js
 - JS postController.js
 - JS userController.js
 - > middleware
 - JS authMiddleware.js
 - > models
 - JS Comment.js
 - JS Follow.js
 - JS Friendship.js
 - JS Post.js
 - JS User.js
 - > node_modules
 - > postman collections
 - > routes
 - JS authRoutes.js
 - JS friendshipRoutes.js
 - JS postRoutes.js
 - JS userRoutes.js
 - > uploads
 - > utils
 - > views
 - env
- > OUTLINE
- > TIMELINE

The main editor area has four tabs open:

- JS server.js
- JS friendshipRoutes.js
- JS postRoutes.js
- JS postController.js (active tab)

The code in the active tab (postController.js) is as follows:

```
controllers > JS postController.js @ getPostsByUserId
193 exports.getPostsByUserId = async (req, res) => {
  194   const posts = await Post.find({ userId: req.params.userId })
  195   .populate('likes', 'username')
  196   .populate('comments', 'comment user')
  197   .exec();
  198
  199   if (!posts) {
  200     console.log(`Profile is private. Checking friendship status...`);
  201     // If profile is private, only friends can view posts
  202     const friendship = await Friendship.findOne({
  203       requester: req.user._id,
  204       recipient: req.params.userId,
  205       status: "accepted"
  206     });
  207
  208     if (!friendship) {
  209       console.log(`No friendship found between user ${req.user._id} and viewer ${req.params.userId}`);
  210       return res.status(403).json({ msg: "You are not friends with this user!" });
  211     }
  212   }
  213
  214   res.json(posts);
  215 }
```

The Problems, Output, Debug Console, Terminal, Ports, and Digma tabs are visible at the bottom.

The bottom status bar shows the following information:

- Dims: Δ 0 \circ 0 \otimes 14 Δ 0 \otimes 0 (.) Scanning.
- 32°C Haze
- 1500 18-12-2024
- ENG IN
- Wi-Fi signal icon
- Bluetooth icon
- USB icon
- 1500 18-12-2024

The bottom navigation bar includes icons for File, Save, Undo, Redo, Find, Replace, Go To, and others.

Technology Stack

- **Backend:** Node.js, Express.js
- **Database:** MongoDB (or PostgreSQL)
- **Authentication:** JSON Web Token (JWT)
- **File Uploads:** Multer
- **Password Hashing:** Bcrypt

Database Schema (MongoDB Example)

The screenshot shows the MongoDB Compass interface connected to a cluster named 'Cluster0'. The left sidebar shows various services like Stream Processing, Triggers, Migration, Data Federation, and Security. The main area displays the 'test' database and its 'comments' collection. The collection has 1 document with the following data:

```
_id: ObjectId('67637d74cb253e38b69a2688')
post : ObjectId('676279f3e06d0eab22c43ad7')
comment : "Nice post John Doe!"
createdAt : 2024-12-30T07:44:52.805+00:00
__v : 0
```

User Table

Field	Type	Description
<code>_id</code>	<code>ObjectId</code>	Unique identifier for the user.
<code>username</code>	<code>String</code>	User's username.
<code>email</code>	<code>String</code>	User's email.
<code>password</code>	<code>String</code>	Hashed password.
<code>profileVisibility</code>	<code>Enum (public/private)</code>	Visibility of the profile.
<code>followers</code>	<code>Array of ObjectId</code>	List of users following this user.
<code>following</code>	<code>Array of ObjectId</code>	List of users this user is following.
<code>posts</code>	<code>Array of ObjectId</code>	List of posts created by the user.

test.users

STORAGE SIZE: 34KB LOGICAL DATA SIZE: 50B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 7KB

Find **Indexes** **Schema Anti-Patterns** **Aggregation** **Search Indexes**

Generate queries from natural language in Compose

QUERY RESULTS: 1-2 OF 2

```

-1: _id: ObjectId('07618e63393fc2386229e6b')
username: "John Doe"
email: "john@gmail.com"
password: "$2a$10$5bdbe1v8N7yndDhv1.0BFc5ybhs2bb3yEjtrN.W/AdzIOnQHqbsS"
profileVisibility: "private"
followers: Array (empty)
following: Array (empty)
--v: 1
posts: Array (empty)
updatedAt: 2024-12-18T09:26:52.893+00:00

-1: _id: ObjectId('07618e63393fc2386229e6b')
username: "SRIDHARAN S"
email: "sridharan@gmail.com"
password: "$2a$10$5bdbe1v8N7yndDhv1.0BFc5ybhs2bb3yEjtrN.W/AdzIOnQHqbsS"
profileVisibility: "private"
followers: Array (empty)
following: Array (empty)
--v: 1
posts: Array (empty)
updatedAt: 2024-12-18T09:26:52.893+00:00
  
```

System Status: All Dead

©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Post Table

Field	Type	Description
<code>_id</code>	ObjectId	Unique identifier for the post.
<code>userId</code>	ObjectId	Reference to the user who created it.
<code>content</code>	String	Content of the post.
<code>image</code>	String	Path to the uploaded image.
<code>likes</code>	Array of ObjectId	List of users who liked the post.
<code>comments</code>	Array of Comments	Comments on the post.

test.posts

STORAGE SIZE: 34KB LOGICAL DATA SIZE: 21B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 24KB

Find **Indexes** **Schema Anti-Patterns** **Aggregation** **Search Indexes**

Generate queries from natural language in Compose

QUERY RESULTS: 1-1 OF 1

```

-1: _id: ObjectId('076279f3e568aae2254bad7')
user: ObjectId('07618e63393fc2386229e6b')
content: "Write a post"
image: "uploads\175450696985721.jpg"
likes: []
comments: Array (1)
createdAt: 2024-12-18T07:29:55.788+00:00
updatedAt: 2024-12-18T08:06:64.742+00:00
--v: 1
  
```

System Status: All Dead

©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Friendship Table

Field	Type	Description
_id	ObjectId	Unique identifier for the friendship.
requester	ObjectId	User who sent the friend request.
recipient	ObjectId	User who received the friend request.
status	Enum (pending/accepted)	Status of the friend request.

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar includes sections for Overview, Clusters, Services (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), Security (Quickstart, Backup, Database Access, Network Access, Advanced), and Goto. The main area displays the 'test.friendships' collection. It shows a single document with the following data:

```
_id: ObjectId('676287188895a0a2e020cb4c')
requester: ObjectId('6761a0db39fcac2386229ebe')
recipient: ObjectId('6761a0e5339fcac23862279ebe')
status: "accepted"
```

The interface also features a 'Find' button, a search bar, and a 'QUERY RESULTS' section indicating 1-1 OF 1 result.

Security Measures

- JWT Authentication:** Secures API endpoints to ensure only authenticated users can access resources.
- Password Hashing:** All passwords are hashed before storage.
- Authorization Middleware:** Ensures proper access control to posts and profiles based on visibility settings.

Installation & Setup

1. Clone the repository:

```
git clone https://github.com/SRIDHARAN1819/social-media-platform.git  
cd social-media-platform
```

2. Install dependencies:

```
npm install
```

3. Configure environment variables in a .env file:

```
PORT=3000
```

```
MONGO_URI=<your-mongodb-connection-string>
```

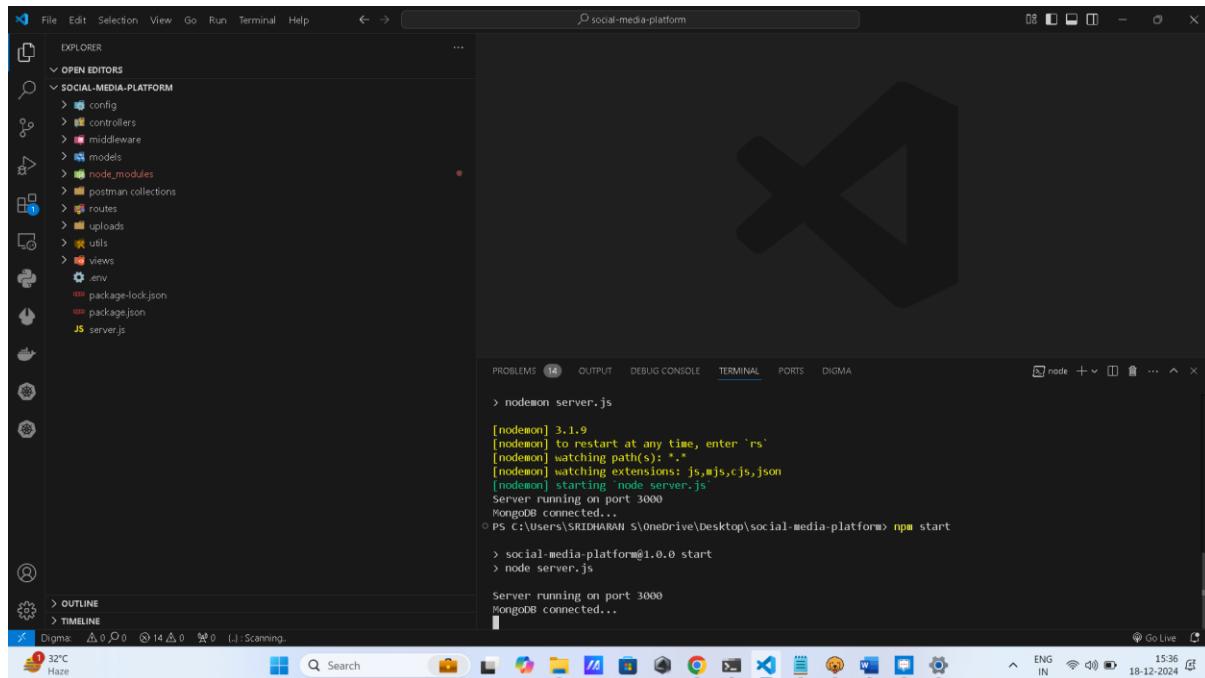
```
JWT_SECRET=<your-secret-key>
```

4. Start the server:

```
npm start
```

5. Access the application at http://localhost:3000.

Project Structure



Conclusion

This documentation outlines the development of a scalable and secure social media platform with key features like user registration, post creation, interaction, and profile management. The project structure and best practices ensure that the application is robust and maintainable.