Blog Application (MERN)

Full-Stack Software Engineer Assessment

Name: Arava Leela Naga Venkata Sai Kumar

Mail: saikumararava23@gmail.com

Phn num: 9000133390 / 9989563390

Linkedin: https://www.linkedin.com/in/sai-kumar-arava-870505201/

Resume Drive Link: https://drive.google.com/file/d/1iGkUwBuilRQisJe-KXy5c-C3cR16u1es/view?usp=sharing

Github repo: https: //github.com/Saikumararava/BlogApp

PostMan invite link: https://web.postman.co/workspace/392c837f-c84e-4f93-88c7-9877ad797fd1

Versal Deployment Link: https://blog-bkufxcbc1-saikumararavas-projects.vercel.app/

Render Deployment Link: https://blogapp-mfku.onrender.com

Introduction:

This document describes the design, setup, and testing of the **Blog Application** developed as part of the Full-Stack Software Engineer Assessment (Intern to Full-time Track).
The app implements authentication, user roles, publishing workflow, comments, and paginated list endpoints, with both backend APIs and a frontend UI

Objectives:

-Build a full-stack blog app (MERN stack).

-authentication (Sign Up / Sign In).

-Enforce roles (APP_ADMIN, AUTHOR, USER) in the backend.

-Provide APIs and frontend UI for posts, comments, and role management.

-Ensure pagination is implemented on list endpoints.

-Provide documentation, seed data, and API tests.

Tech Stack:

Frontend: React + Material-UI

Backend: Node.js + Express

Database: MongoDB Atlas

Authentication: JWT (JSON Web Tokens with bcrypt password hashing)

Testing: Postman collection

Deployment: Vercel for frontend, Render for backend

Setup Instructions:

For Backend

Navigate to Backend/

npm init -y

npm install express mongoose bcryptjs jsonwebtoken dotenv cors express-validator

npm install -D nodemon

To start Backend--> npm run dev

(listens on port 5000 by default)

For Frontend

cd Frontend

npx create-react-app .

npm install @mui/material @emotion/react @emotion/styled @mui/icons-material axios react-router-dom

To start Frontend App--> npm start

(runs on port 3000)

.env.example

For backend:

PORT=5000

MONGO_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/<dbname>?retryWrites=true&w=majority

JWT_SECRET=supersecret123

JWT_EXPIRES_IN=7d

CLIENT_URL=http://localhost:3000

For frontend:

REACT_APP_API_URL=http://localhost:5000/api

Authentication & Roles:

- **Approach:** Custom JWT auth (email + password).
- **Reason:** Self-contained and transparent for assessment; no external provider setup needed.
- **How it works:**
    - User signs up with email + password.
    - Password is hashed with bcrypt.
    - A JWT is issued and stored in localStorage.

- Token is sent with Authorization: Bearer <token> for protected APIs.

Roles:

APP_ADMIN → can manage any post, grant/revoke AUTHOR roles.

AUTHOR → can manage only their own posts.

USER (default) → can only view published posts and comment.

Role enforcement is done in backend middleware (401 for unauthenticated, 403 for insufficient role, 404 if resource not visible).

Features Implemented:

- **Authentication:** Sign Up & Sign In.
- **Role Permissions:** APP_ADMIN, AUTHOR, USER.
- **Blog Posts:** Draft/Published states, visibility rules per role.
- **Comments:** Only on published posts, only by signed-in users.
- **Pagination:** Applied to posts, comments, and users.
- **Frontend UI:**
  - Paginated posts list.
  - Post detail with comments + comment form.
  - Post editor (for Author/Admin).
  - Admin dashboard (role management).

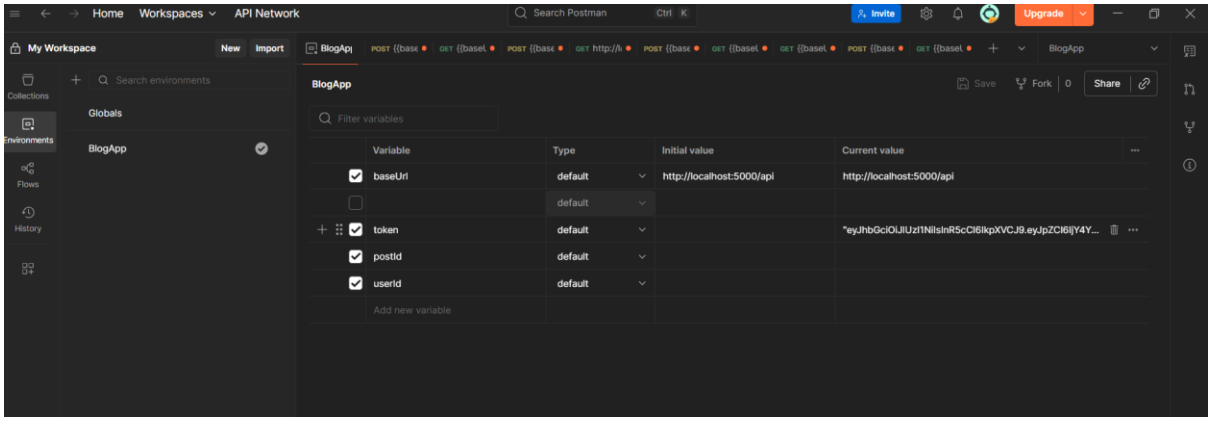
Seed Data / Test Accounts:

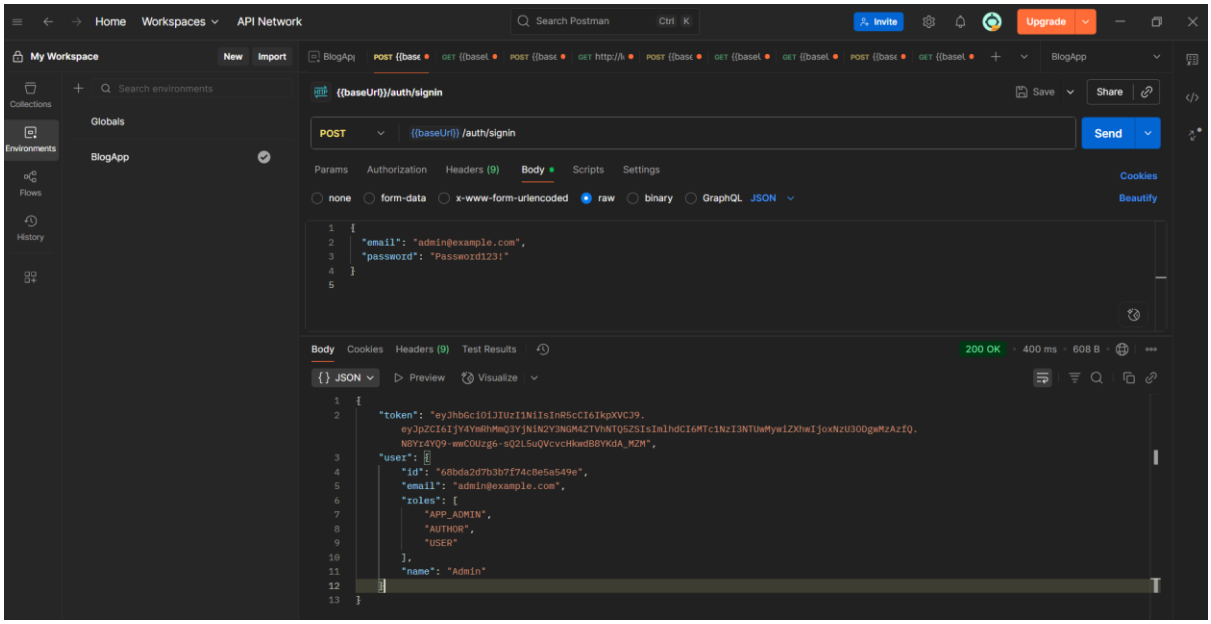
Seed script (npm run seed) creates:

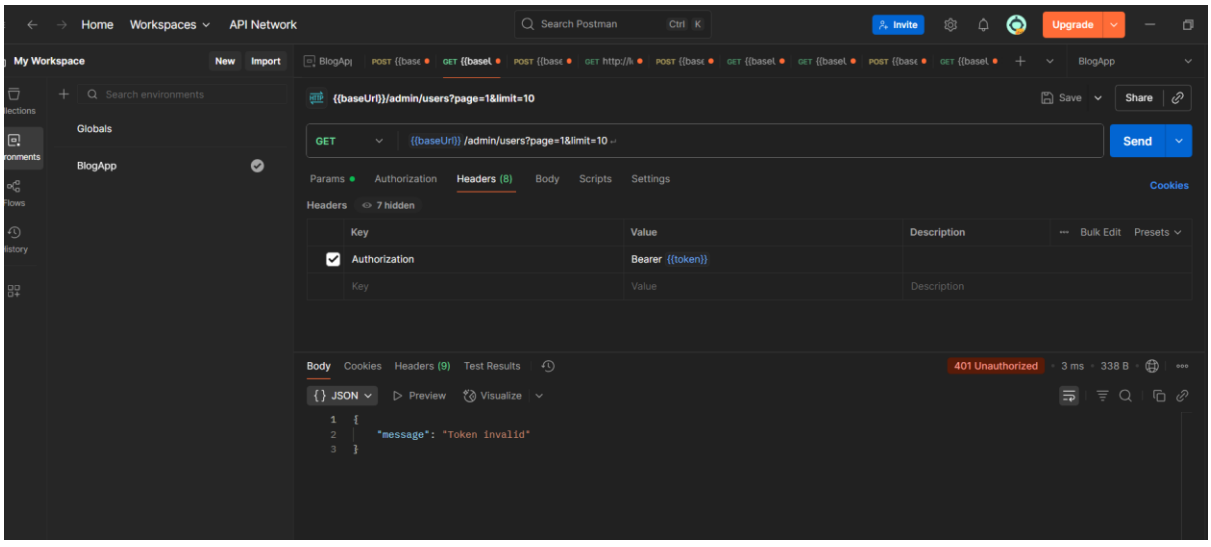
- **Admin:**
  - Email: admin@example.com
  - Password: Password123!
  - Roles: [APP_ADMIN]
- **Author:**
  - Email: author@example.com
  - Password: Password123!
  - Roles: [AUTHOR]
- **User:**
  - Email: user@example.com
  - Password: Password123!
  - Roles: [USER]

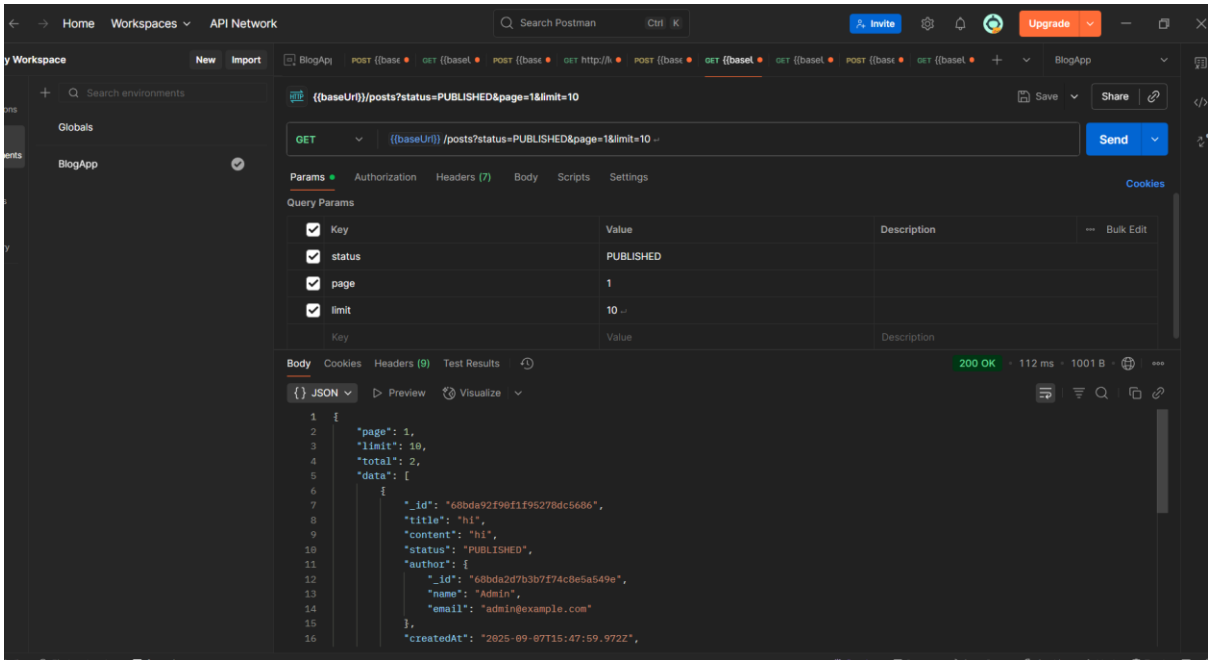
API Testing (Postman):

I tested the API using **Postman**.
A Word file with all prompts, responses, and screenshots is attached below


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
Flows
History
Search environments
Globals
BlogApp
BlogApp
Save
Fork 0
Share
Filter variables
Variable
Type
Initial value
Current value
baseUrl
default
http://localhost:5000/api
http://localhost:5000/api
token
postId
userId
Add new variable

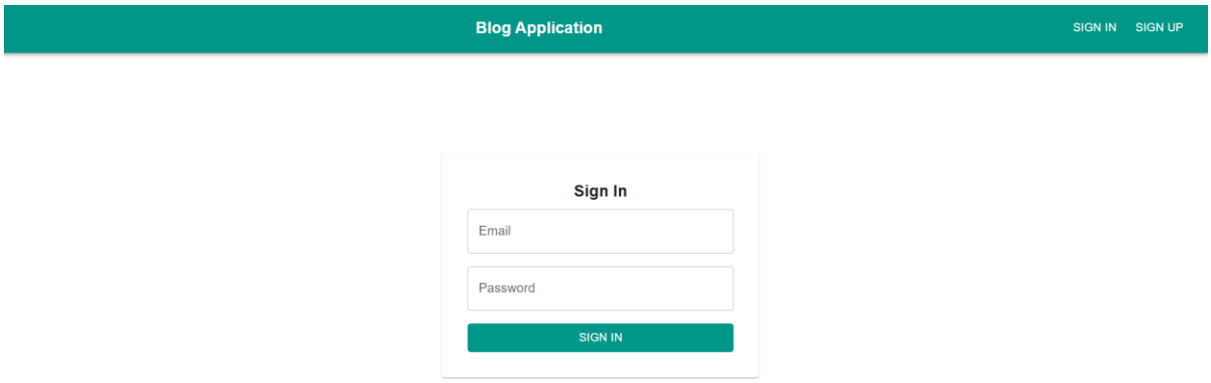

Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
{{baseUrl}}/auth/signin
Save
Share
POST
{{baseUrl}} /auth/signin
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"email": "admin@example.com",
"password": "Password123!"
}
Body
Cookies
Headers (9)
Test Results
200 OK
400 ms
608 B
JSON
Preview
Visualize
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6IjY4YmRhMmQ3YjNiN2Y3NGM4ZTVhNTQ5ZSIsImlhdCI6MTc1NzI3NTUwMywiZXhwIjoxNzU3ODgwMzAzfQ.
N8Yr4YQ9-wwCOUzg6-sQ2L5uQVcvcHKwdB8YKdA_MZM",
"user": {
"id": "68bda2d7b3b7f74c8e5a549e",
"email": "admin@example.com",
"roles": [
"APP_ADMIN",
"AUTHOR",
"USER"
],
"name": "Admin"
}
}


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
{{baseUrl}}/admin/users?page=1&limit=10
Save
Share
GET
{{baseUrl}} /admin/users?page=1&limit=10
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Headers
7 hidden
Key
Value
Description
Bulk Edit
Presets
Authorization
Bearer {{token}}
Body
Cookies
Headers (9)
Test Results
401 Unauthorized
3 ms
338 B
JSON
Preview
Visualize
{
"message": "Token invalid"
}

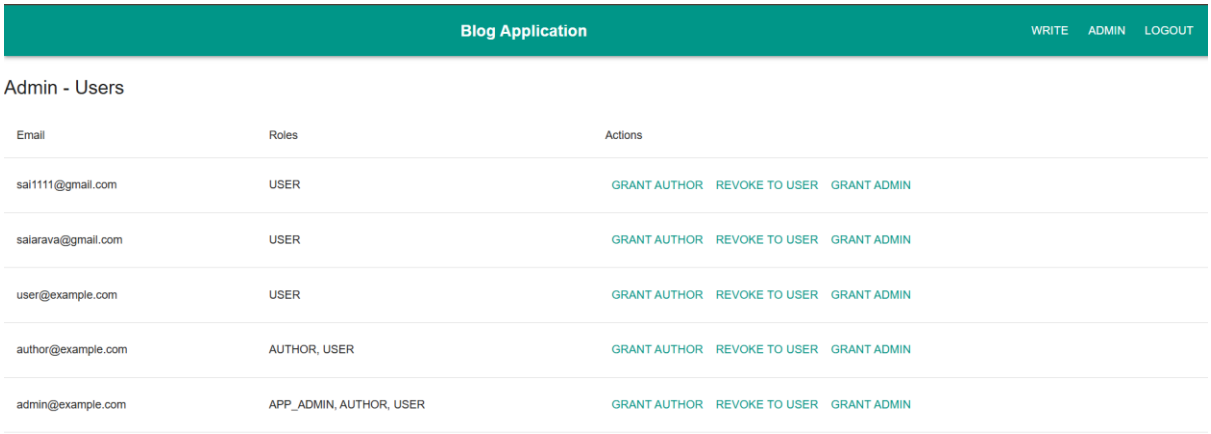


ScreenSnaps of Results of frontend and Mongodb Atlas and Compass

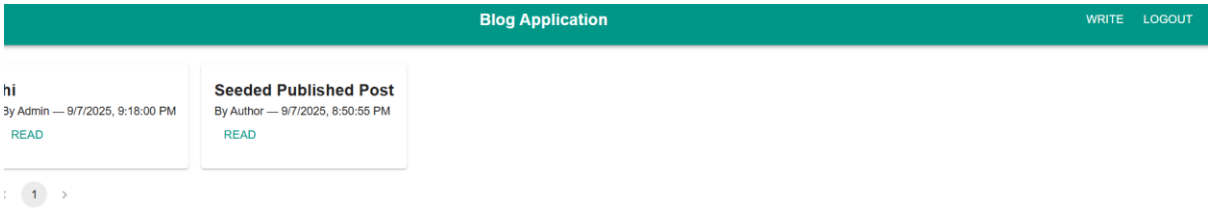
Signup and Signin page



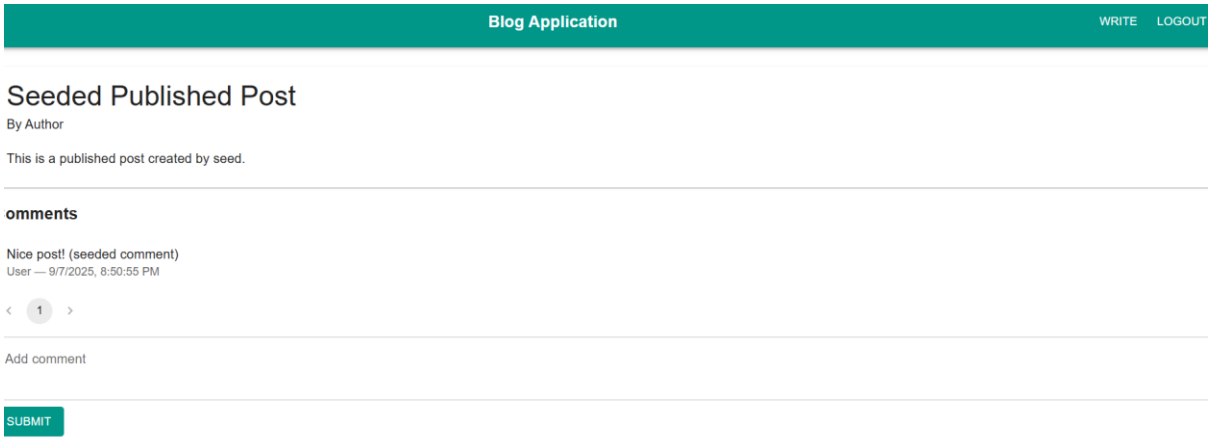
Admin Dashboard



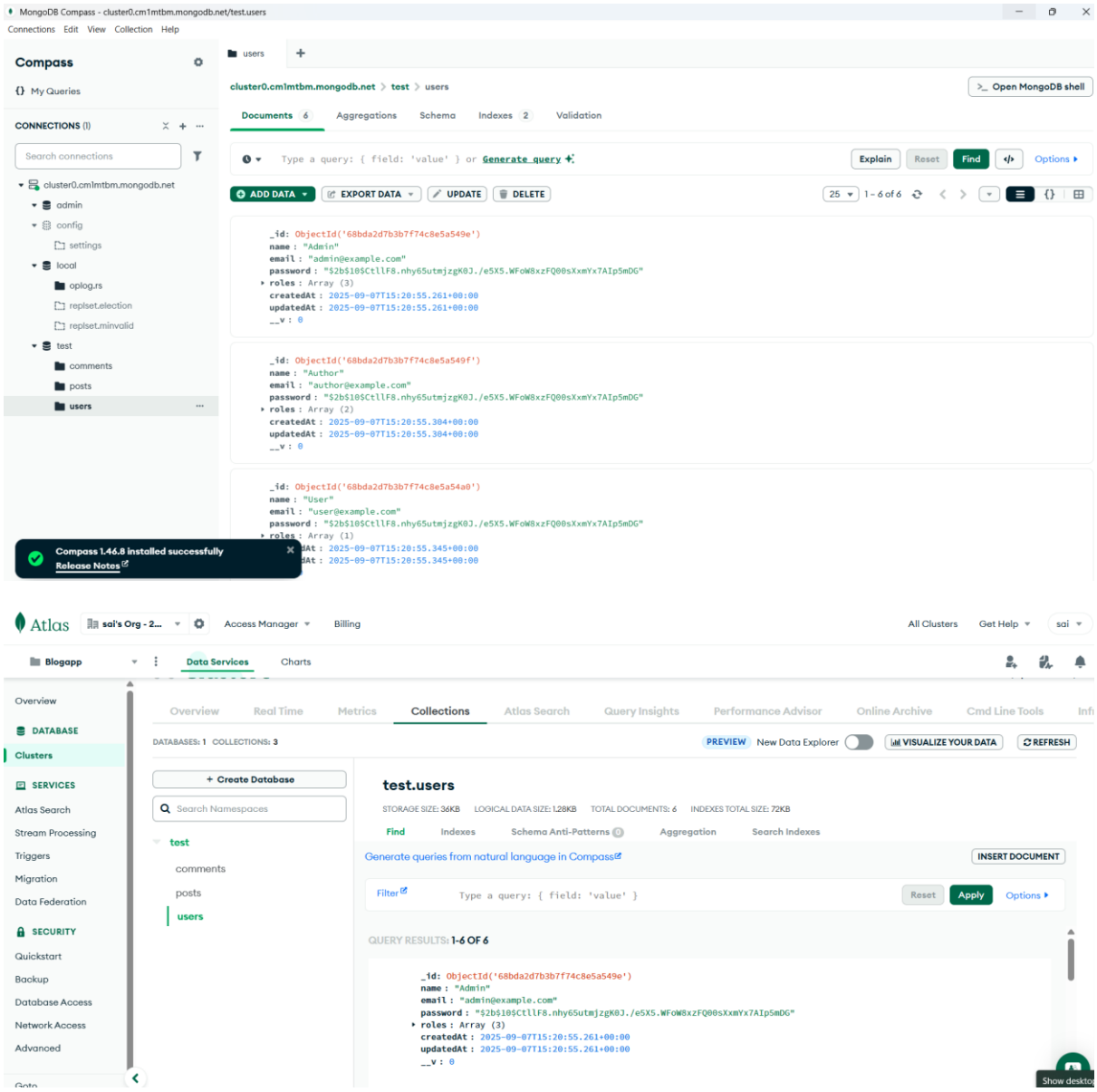
Author Dashboard



Comments and Posts that are seeding:



MongoDB Atlas and Compass Results:



Conclusion:

The project meets the functional requirements of the assessment:

- Authentication and role-based permissions work correctly.
- Blog posts and comments are restricted by roles.
- All list endpoints are paginated.
- Admin can promote users to AUTHOR and revoke.
- Postman testing confirms expected results.
- Documentation, seed data, and test accounts are provided.