**Full Stack Project Report**

# Title Page

Project Title: BlogNest- A Markdown-Based Blogging Platform For Developers

Institution / Organization: IBM Summer Internship Program

Mentor/Supervisor: Mr. Jaivik Panchal

Submission Date: 21 July 2025

# Certificate

This is to certify that the project titled 'BlogNest - A Markdown-Based Blogging Platform For Developers' is an original work completed by the Sakshi Yadav as part of the Full Stack Development Internship under the guidance of Mr. Jaivik Panchal.

# Acknowledgment

I would like to express my sincere gratitude to my mentor, peers, and the IBM Internship Program for their support and guidance throughout the development of the BlogNest project.

# Table of Contents

1. Abstract
2. Objective
3. System Architecture
4. Technology Stack
5. Modules/Features
6. Frontend Development
7. Backend Development
8. Database Design
9. Data Flow Diagrams
10. Testing
11. Security Measures
12. Limitations
13. Future Enhancements
14. Screenshots
15. Annexures
16. References

# Abstract

Blog Nest is a developer-centric full stack blogging platform designed for Markdown-based content creation. It was developed to provide a streamlined, modern experience for tech writers to create, manage, and share their posts, complete with tags, comments, author profiles, and admin oversight. Key features include a Markdown editor, authentication system, comment section, search and tag filtering, and admin dashboard. It allows users to write, preview, and publish markdown-based blog posts with real-time rendering. Built using modern web technologies like Node.js, Express, MongoDB, and HTML/CSS/JS on the frontend, it aims to provide a robust, developer-centric blogging experience.

# Objective

The goal of Blog Nest is to empower developers and technical writers with a clean, fast, and user-friendly platform for publishing and managing Markdown-based content, while also offering features like user profiles, search, and administrative control.

- Build a secure and scalable blog platform.  
- Support markdown-based post creation and editing.  
- Implement user authentication with role-based access (admin/user).  
- Enable comment sections for feedback.  
- Provide admin tools for managing users, posts, and analytics.

# System Architecture

* The system follows a client-server architecture where the frontend communicates with the backend API using HTTP requests.
* The backend validates requests, accesses the MongoDB database, and sends responses back to the client.
* Authentication is handled using JWT, and user roles are checked via middleware.
* The system follows a three-tier architecture:
  + Frontend (HTML/CSS/JS)
  + Backend (Node.js with Express)
  + Database (MongoDB with Mongoose ORM)
* The architecture ensures separation of concerns between UI rendering, API routing, and data storage.

# Technology Stack

* Frontend: HTML5, CSS3, JavaScript
* Backend: Node.js, Express.js
* Database: MongoDB (with Compass for GUI management)
* Authentication: JWT (JSON Web Token)
* Tools: Postman(API), GitHub, VSCode, MongoDB Compass
* Hosting: (localhost for now, can be deployed on Render/Vercel)

# Modules/Features

1. Authentication Module – Sign up, login, logout with JWT token support and password hashing.

2. User Dashboard – Displays recent posts, allows tag search, and supports user interactions, Tagging System (Assign tags to posts and filter/search by tags) and Tag-based Search.

3. Markdown Editor – Real-time editor with live HTML preview, download as .md/.html, and publish functionality, Create, Edit, Delete Blog Posts.

4. Author Profile Page – Manages user-specific posts with comment integration and edit/delete options.

5. Comment System – Users can read and post comments to enhance discussion.

6. Admin Panel – Admin can view statistics (total users/posts/comments/tags), manage users, promote users to admin, and delete posts/users.

7. API Routes – Organized route handling for authentication (`/auth`), posts (`/posts`), users (`/admin`).

# Frontend Development

* The frontend is built using HTML, CSS, and vanilla JavaScript.
* Key UI screens include:

- Home page  
- Login/Signup  
- Profile and Author Dashboard  
- Markdown Editor  
- Admin Panel (Posts, Users)

* The UI uses a modern dark theme and Google Fonts for an aesthetic developer-centric look.

# Backend Development

* The backend is implemented using Node.js and Express.js.
* It exposes RESTful API routes to manage:  
  - Authentication (`/api/auth`)  
  - Posts and Comments (`/api/posts`)  
  - Admin Actions (`/api/admin`)
* Middleware handles authentication and admin-only access control.

# Database Design

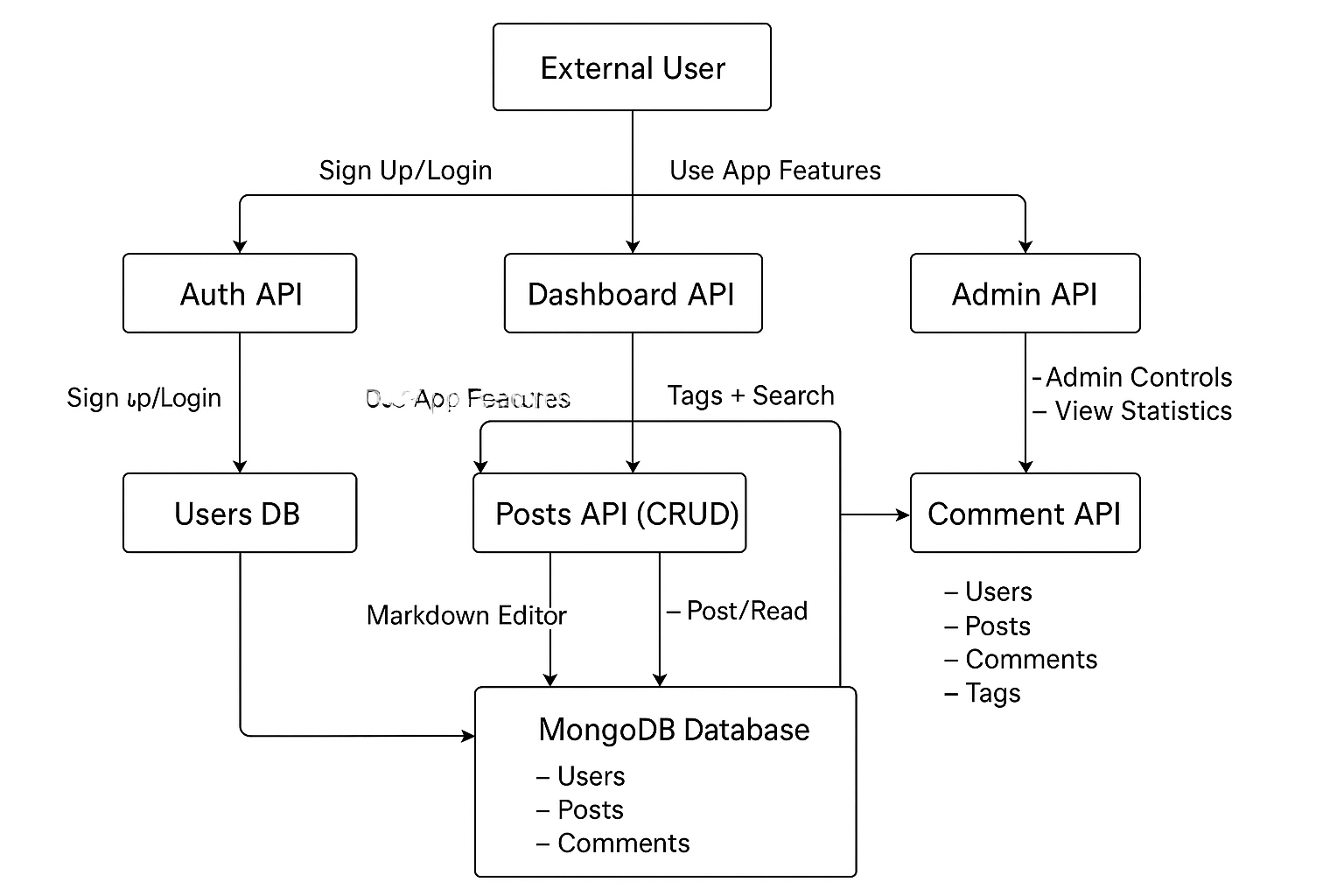
* MongoDB is used for storing:

- Users: name, email, password (hashed), isAdmin  
- Posts: title, markdown, html, tags, authorId, authorName  
- Comments: postId, authorId, authorName, text  
- Reset Tokens (if implemented)

* Data relationships are maintained using MongoDB references.
* Database Schema
* User Collection: name, email, password (hashed), isAdmin
* Post Collection: title, markdown, html, tags, authorId, authorName, createdAt
* Comment Collection: postId, authorId, authorName, text, createdAt

# Data Flow Diagrams

1. User submits login → API verifies credentials → JWT token returned → stored in localStorage.  
2. User creates a post → Form sends POST request → Backend stores to MongoDB → Post appears on profile.  
3. Comments are retrieved dynamically per post via GET requests.  
4. Admin Panel fetches all data using protected admin routes.



# Testing

* Manual testing was done using Postman for backend APIs, and browser testing for all UI flows.
* Validation includes:  
  - Signup/Login with correct and incorrect data  
  - Creating, editing, deleting posts  
  - Comment posting  
  - Admin actions with protected routes

# Security Measures

* JWT-based Authentication
* Password hashing using bcrypt
* Role-based access control (admin middleware)
* API route protection via middleware
* Input validation and error handling on server side

# Limitations

* No image/file uploads in blog posts
* Markdown editor supports limited styling
* Admin actions lack pagination
* Minimal analytics or user activity tracking

# Future Enhancements

* Add rich media uploads (images/videos).
* Email notifications for comments.
* Full-text search across posts.
* Analytics dashboard for admins.
* Deploy to a cloud platform (Render/Vercel/Mongo Atlas).
* Add likes/bookmarks to posts.
* Real-time notifications and email support.
* Responsive mobile UI improvements.
* Theme toggle (dark/light) for user accessibility.

# Screenshots

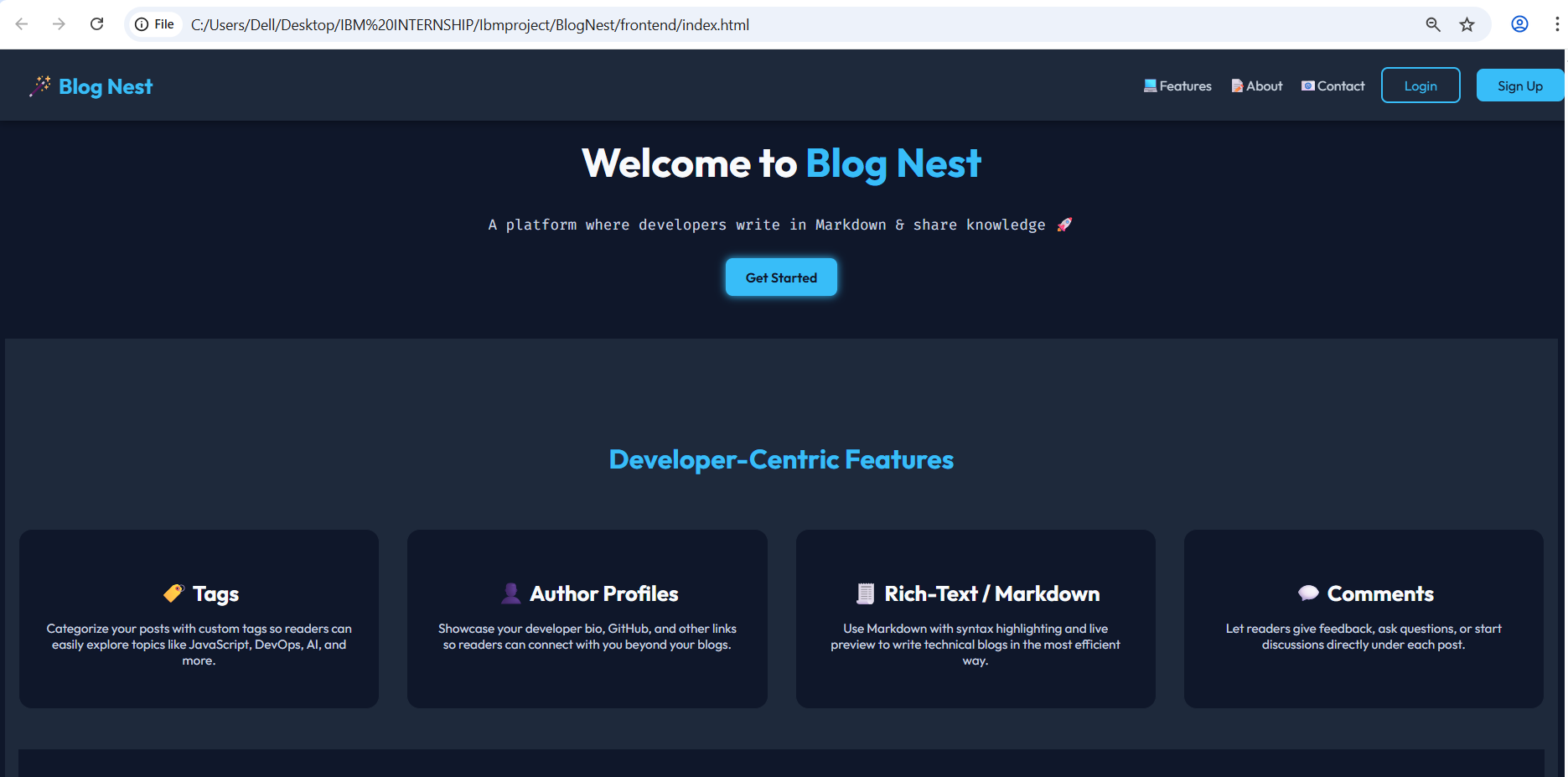


Fig.1 Home page

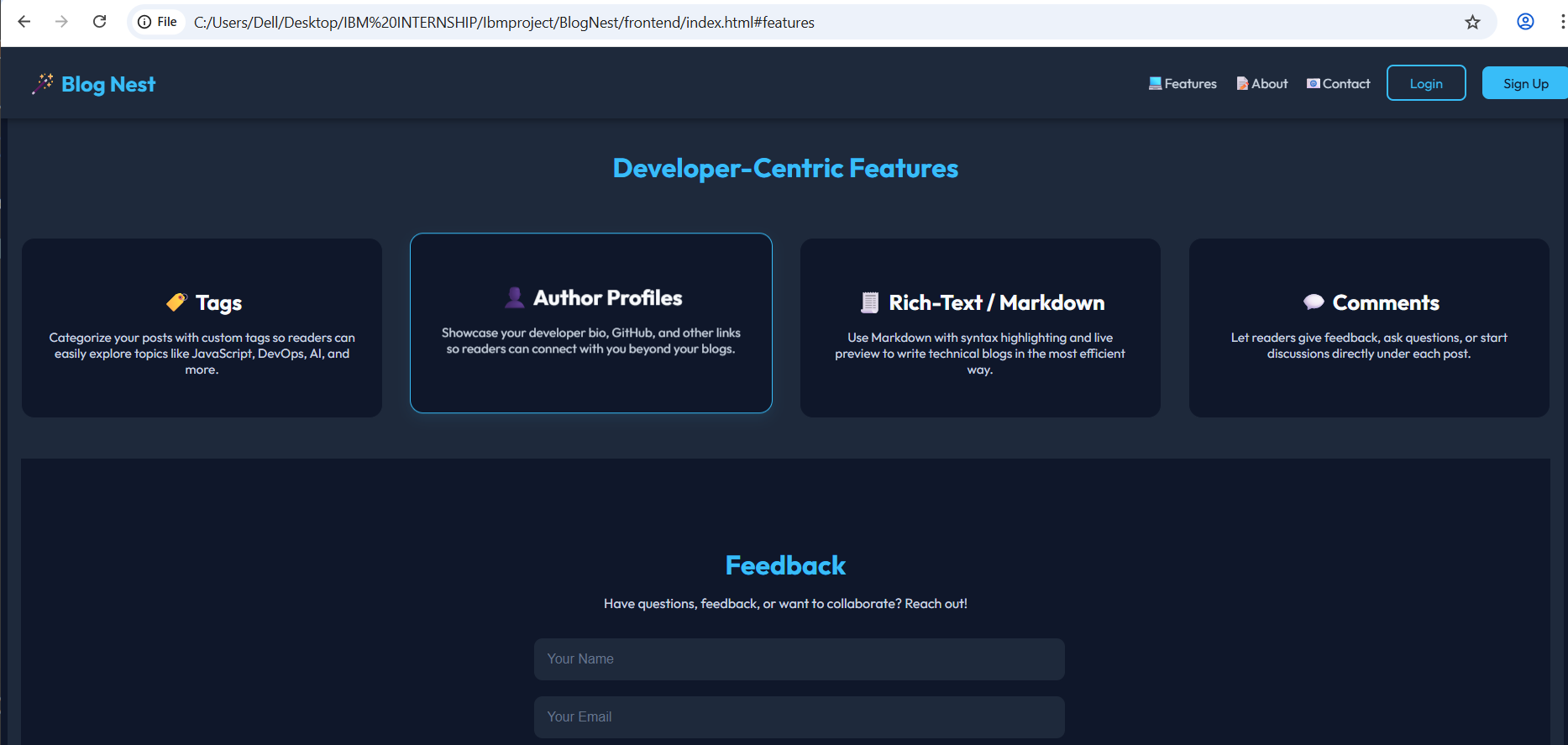
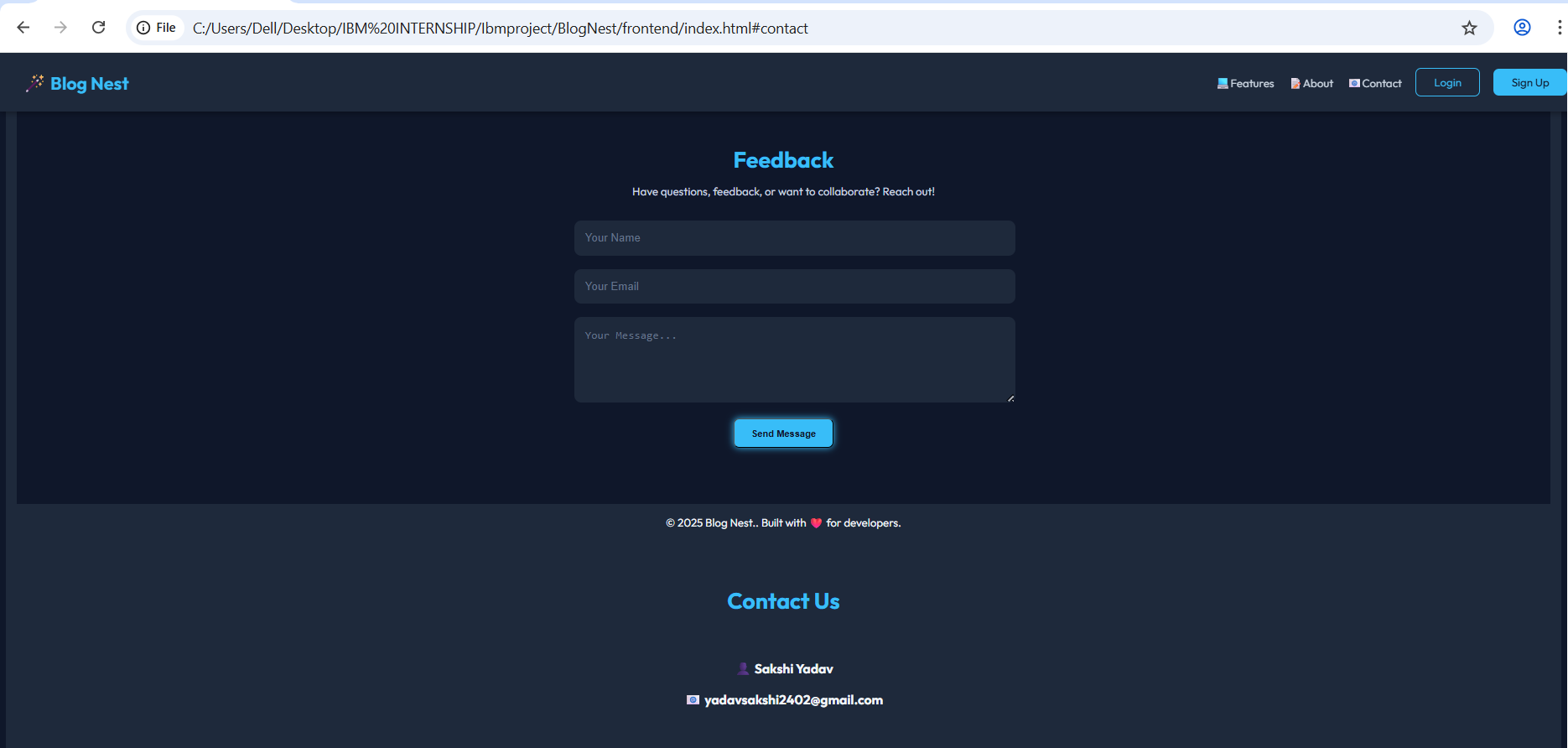


Fig.2 Features

Fig.3 Contact Us

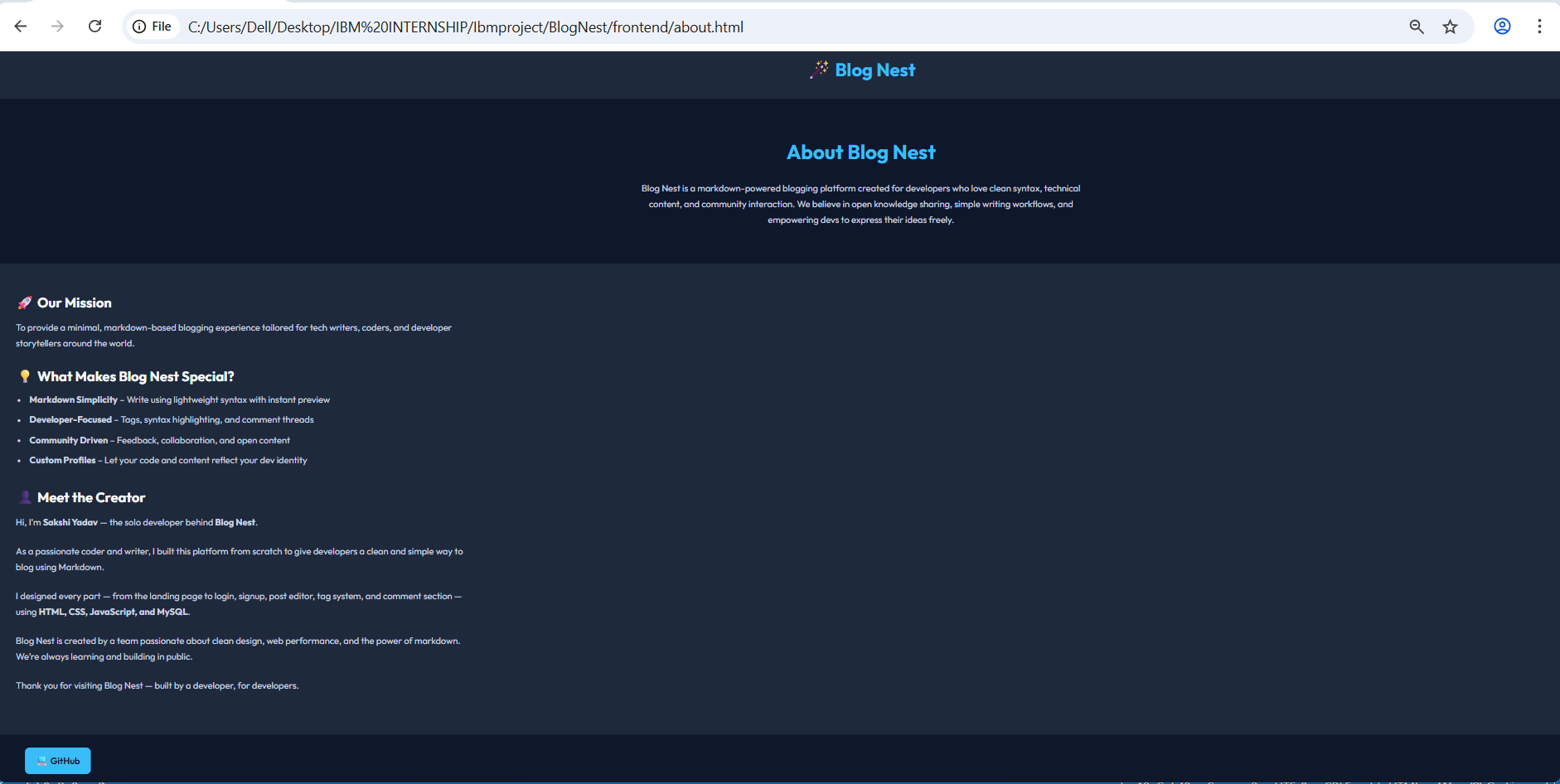
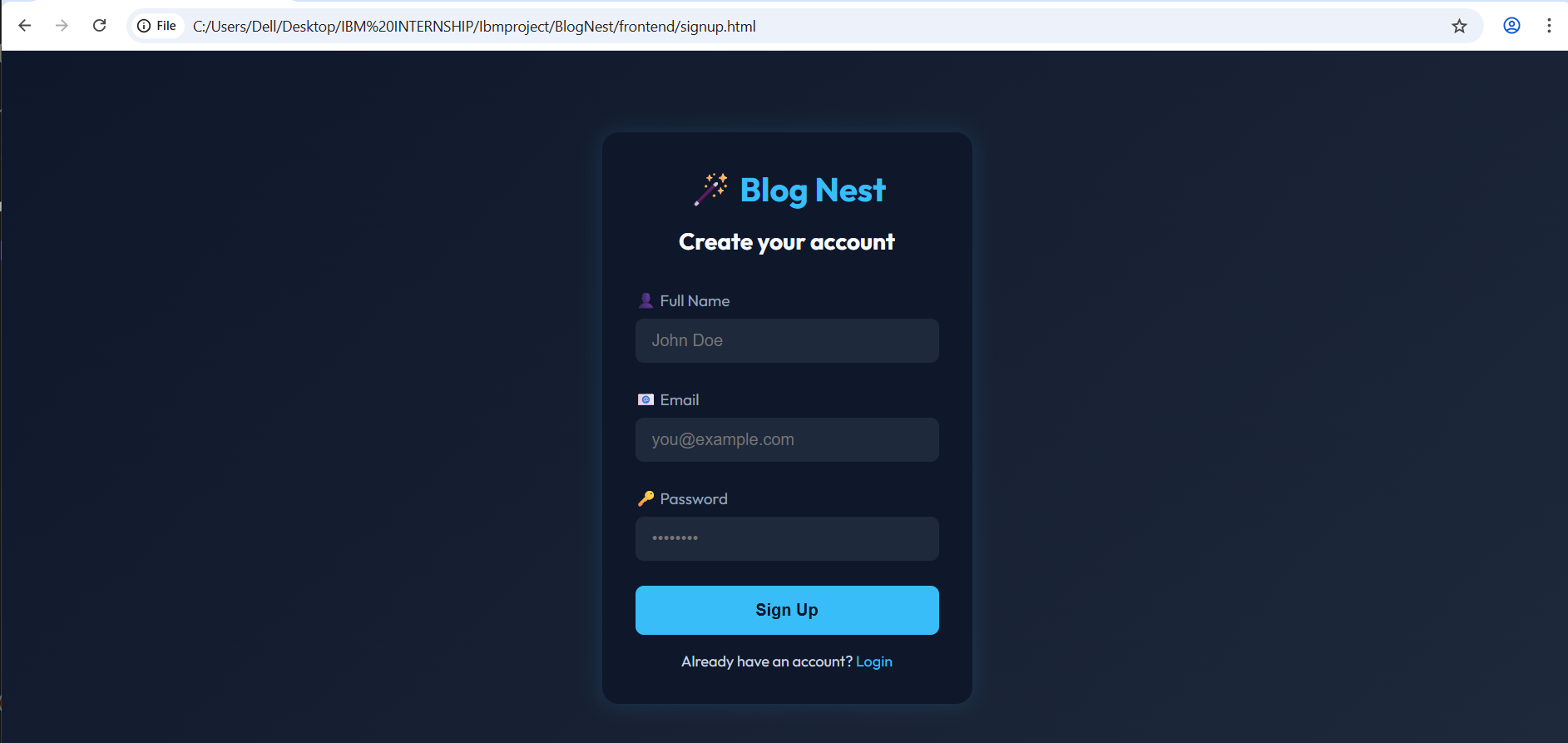


Fig.4 About Us

Fig.5 Signup Page

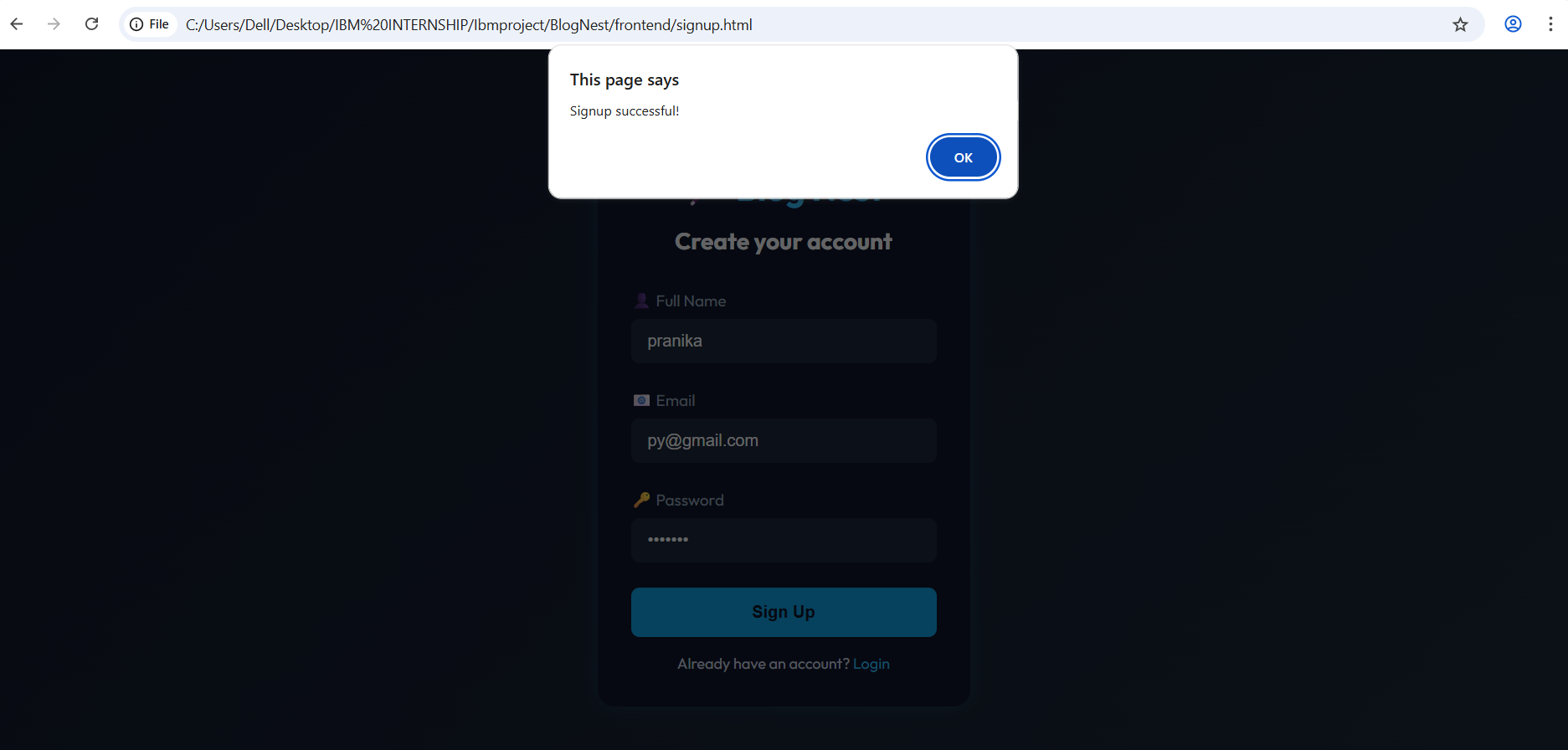
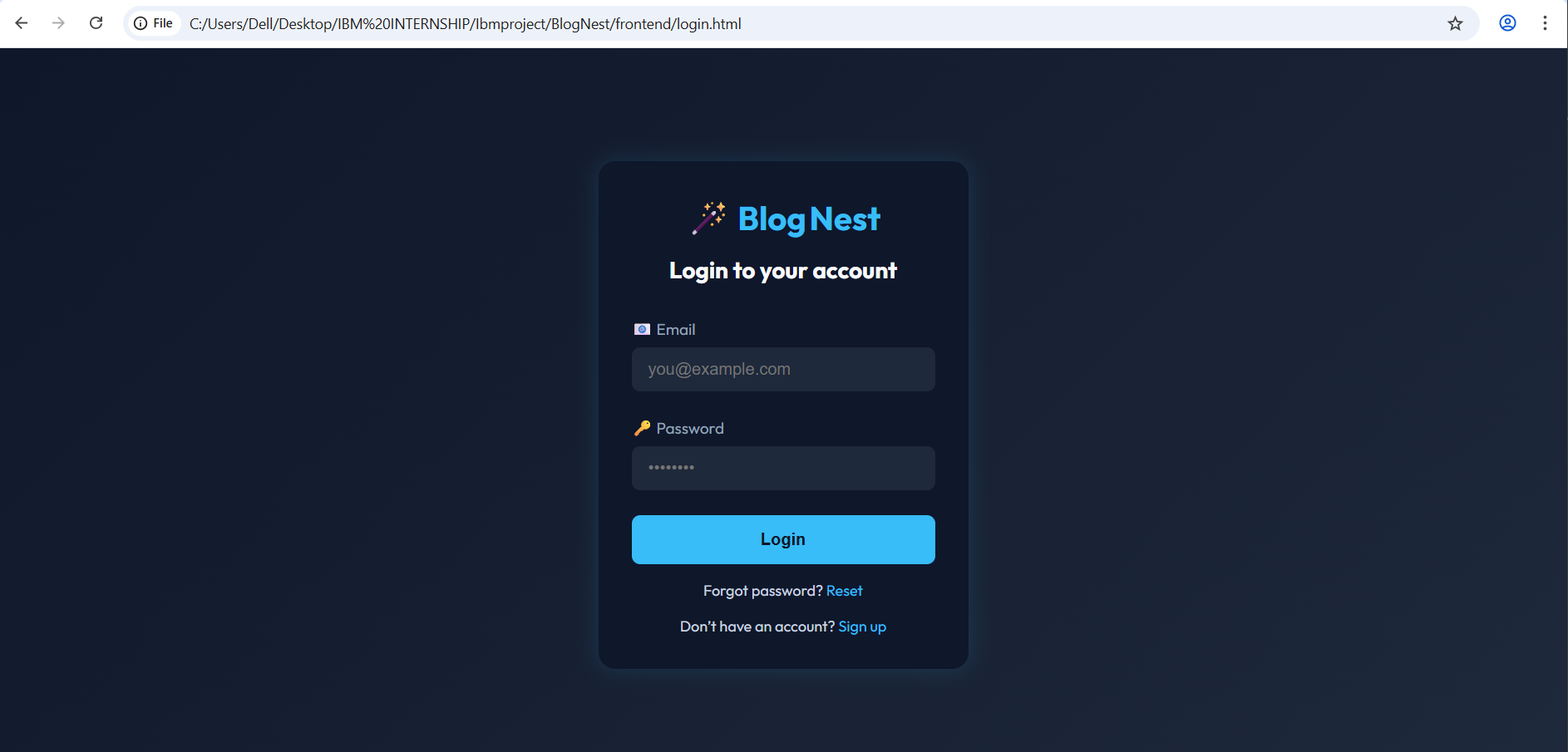


Fig.6 Signup Successfully

Fig.7 Login Page

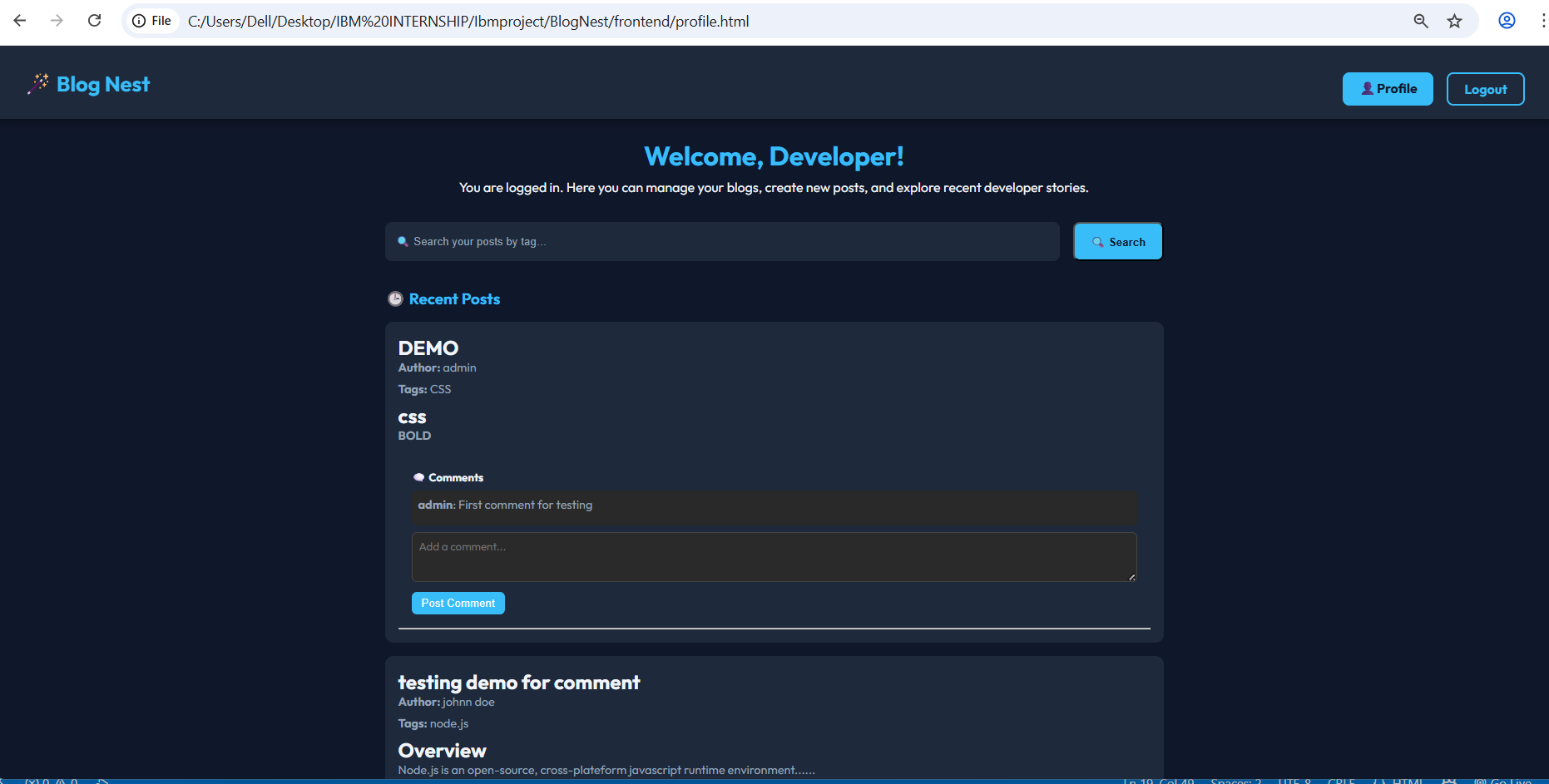
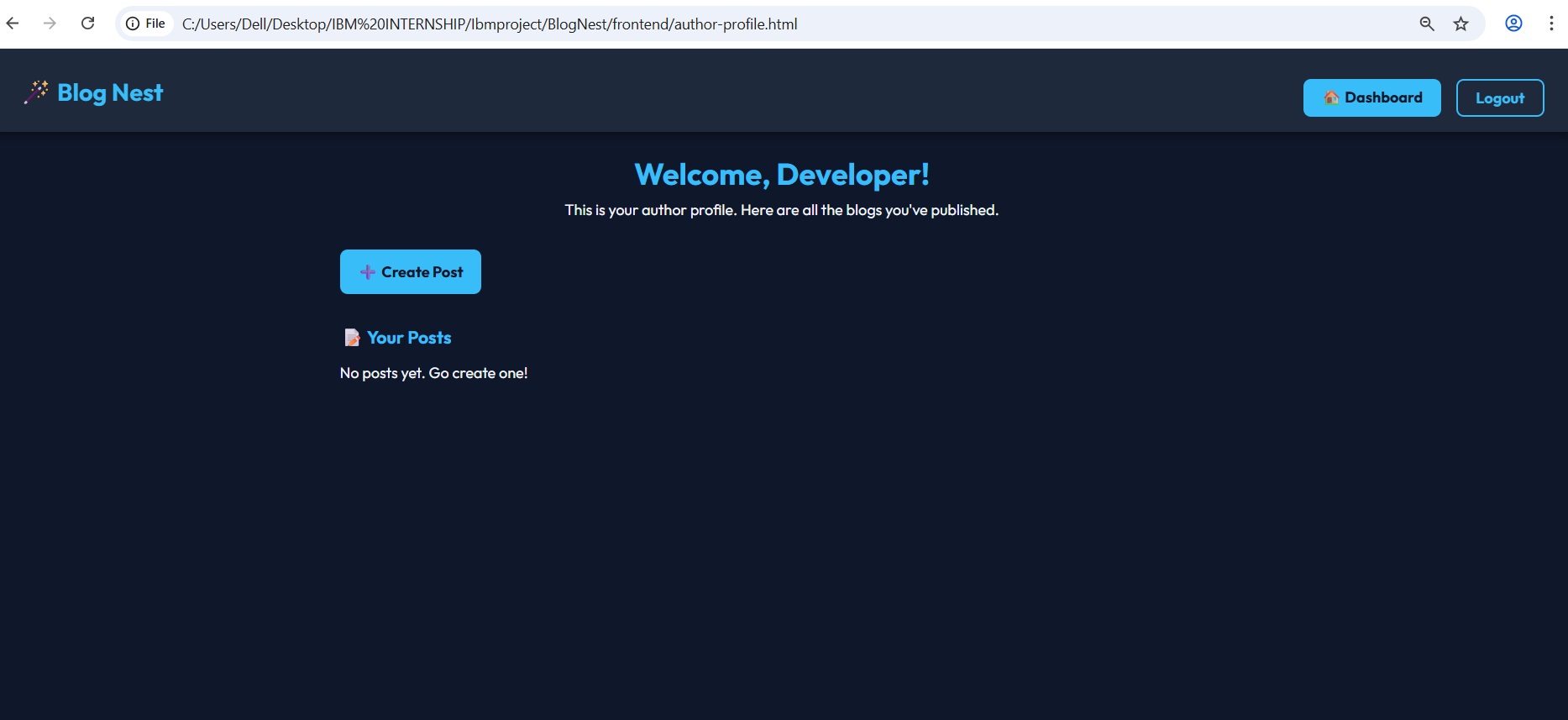


Fig.8 User Profile

Fig.9 Author Profile

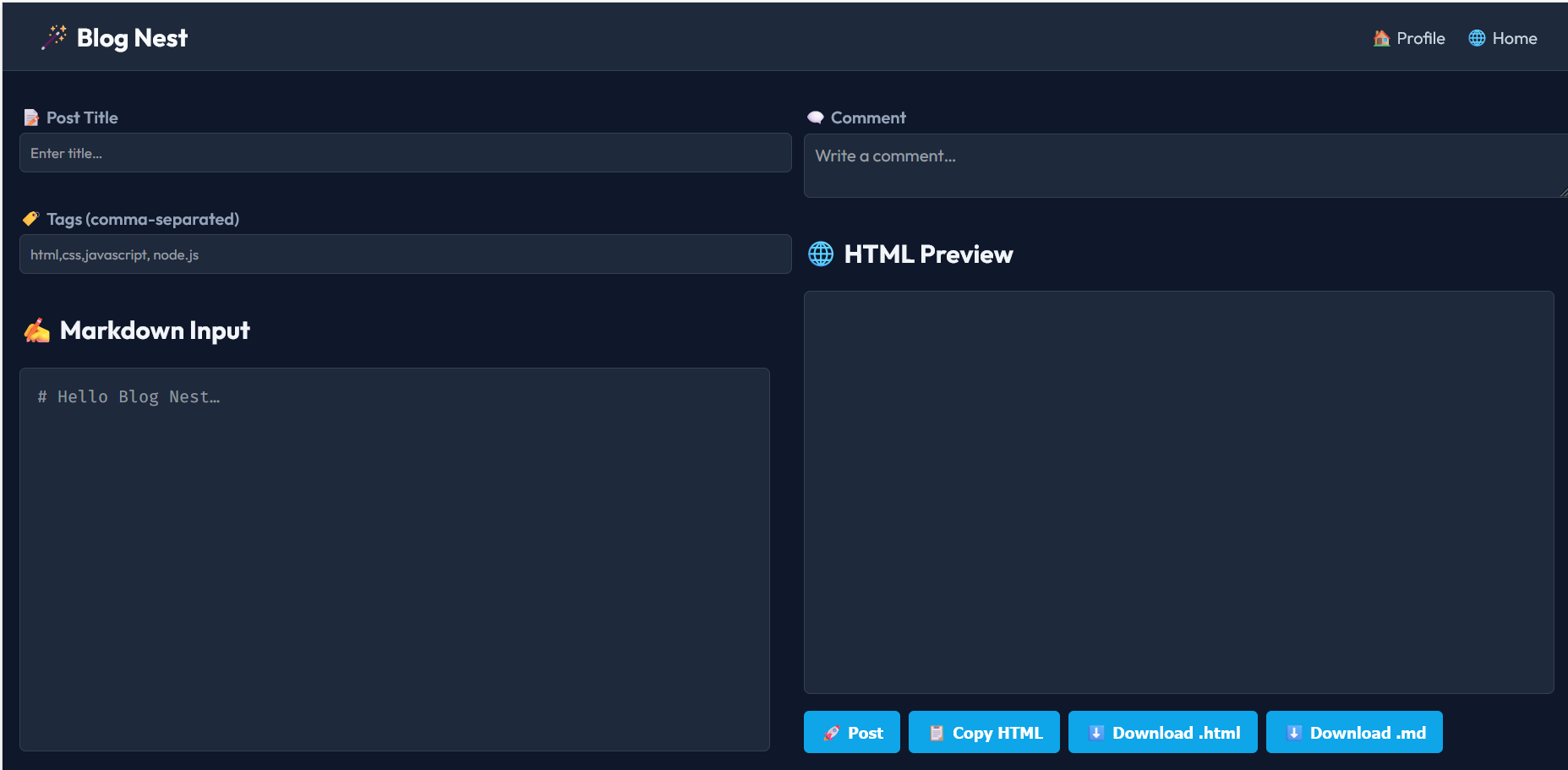
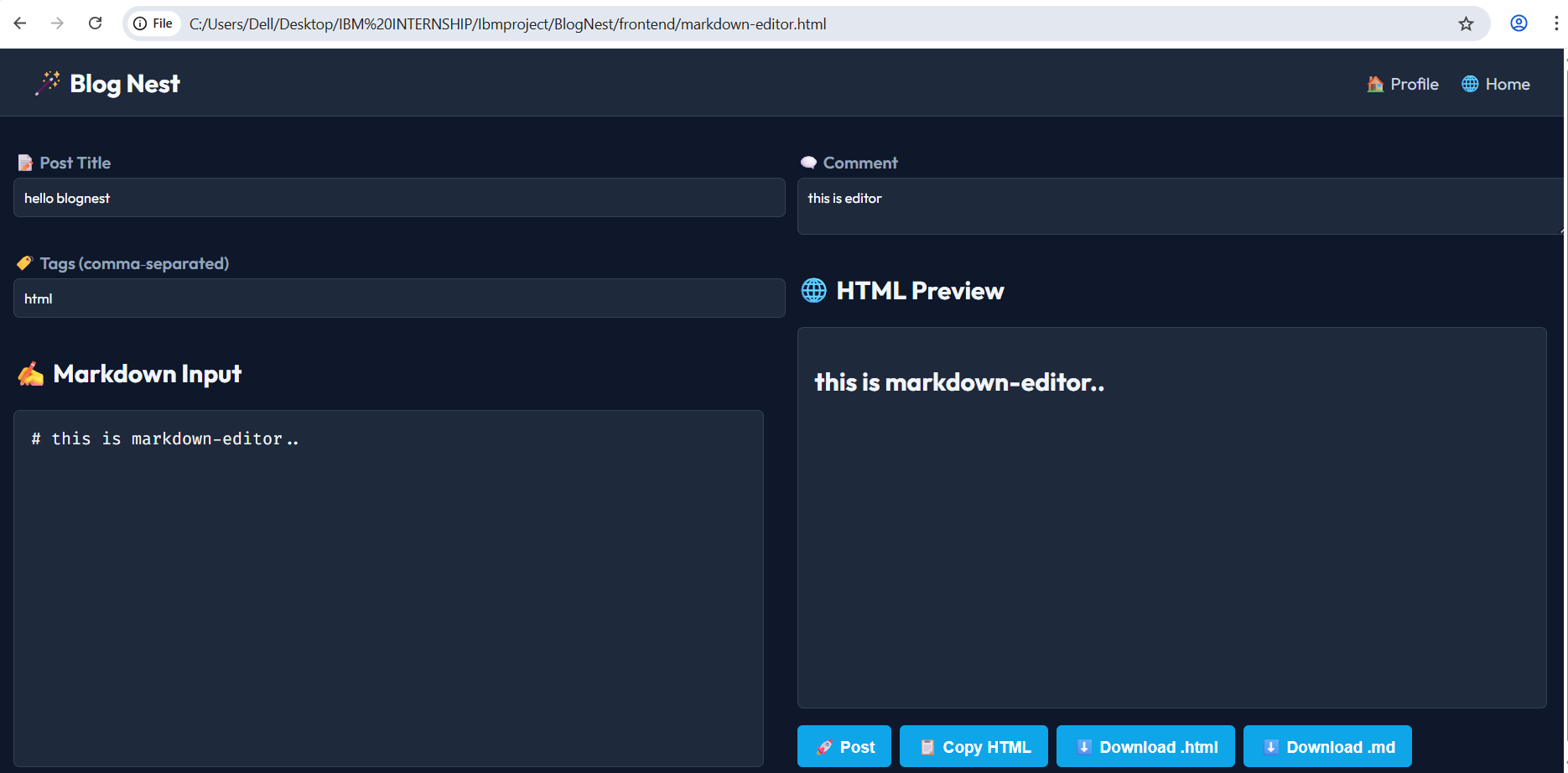


Fig.10 Create Post

Fig.11 Markdown Editor

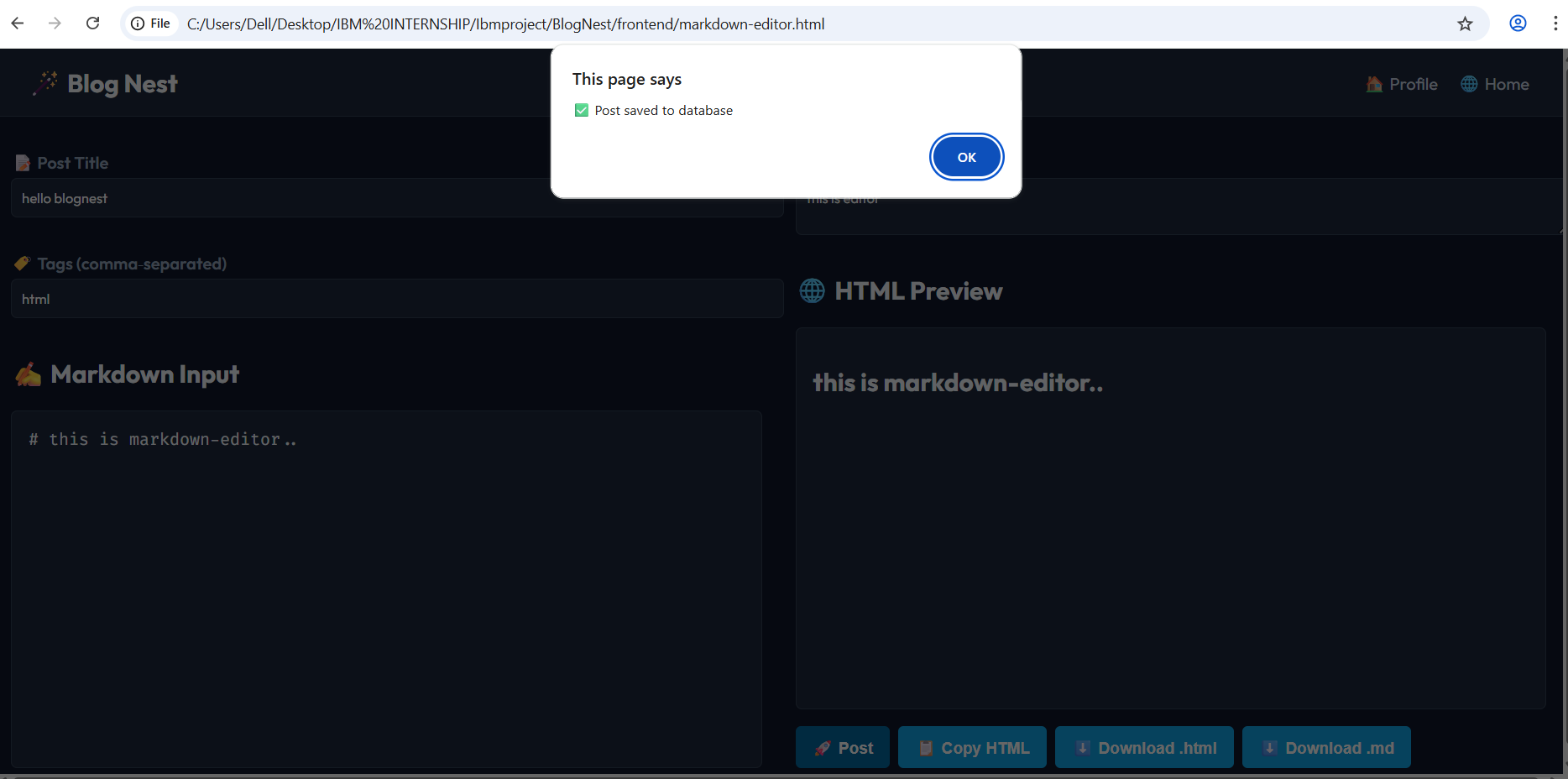
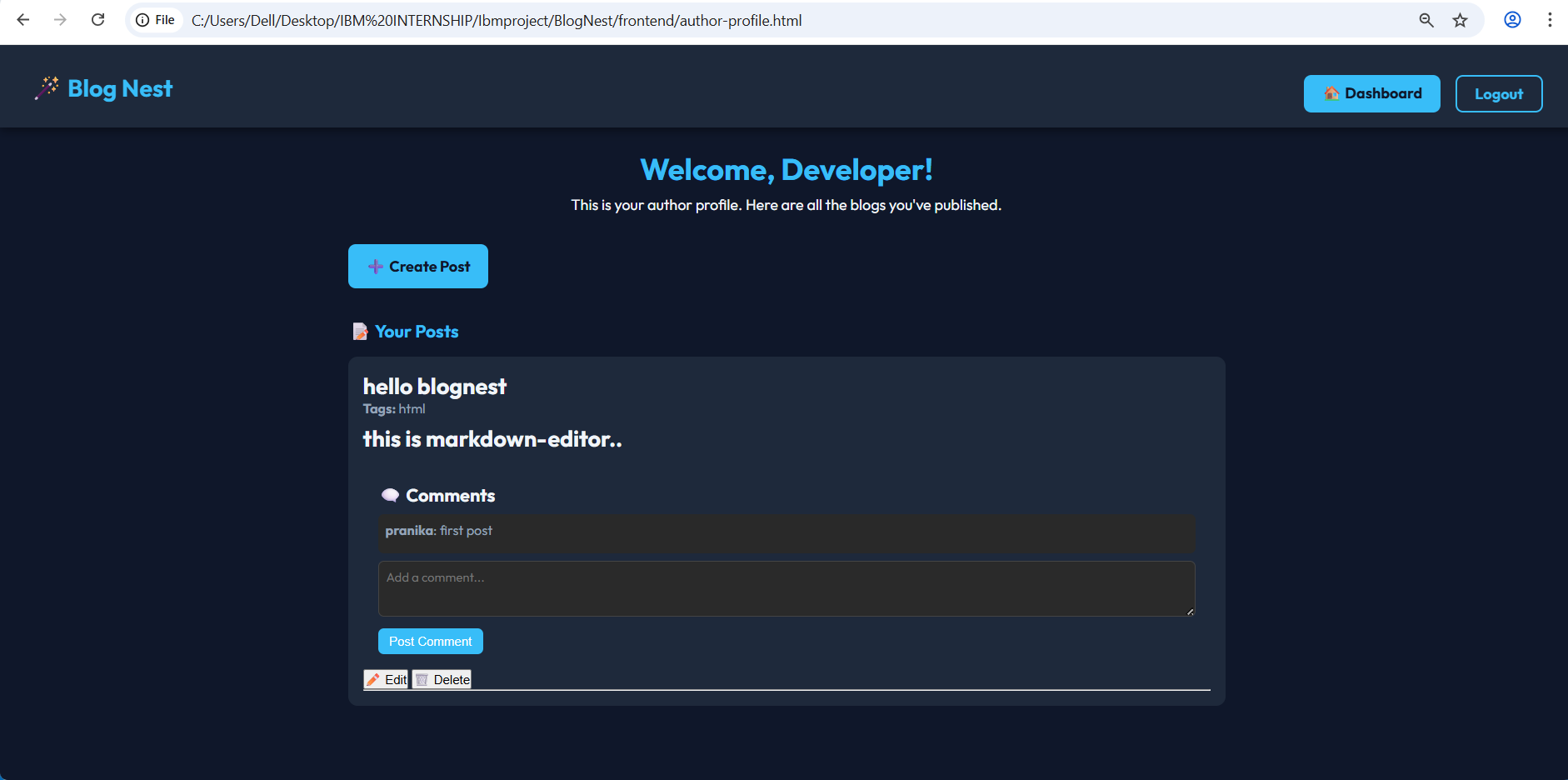


Fig.12 New Post

Fig.13 Updated Author Profile

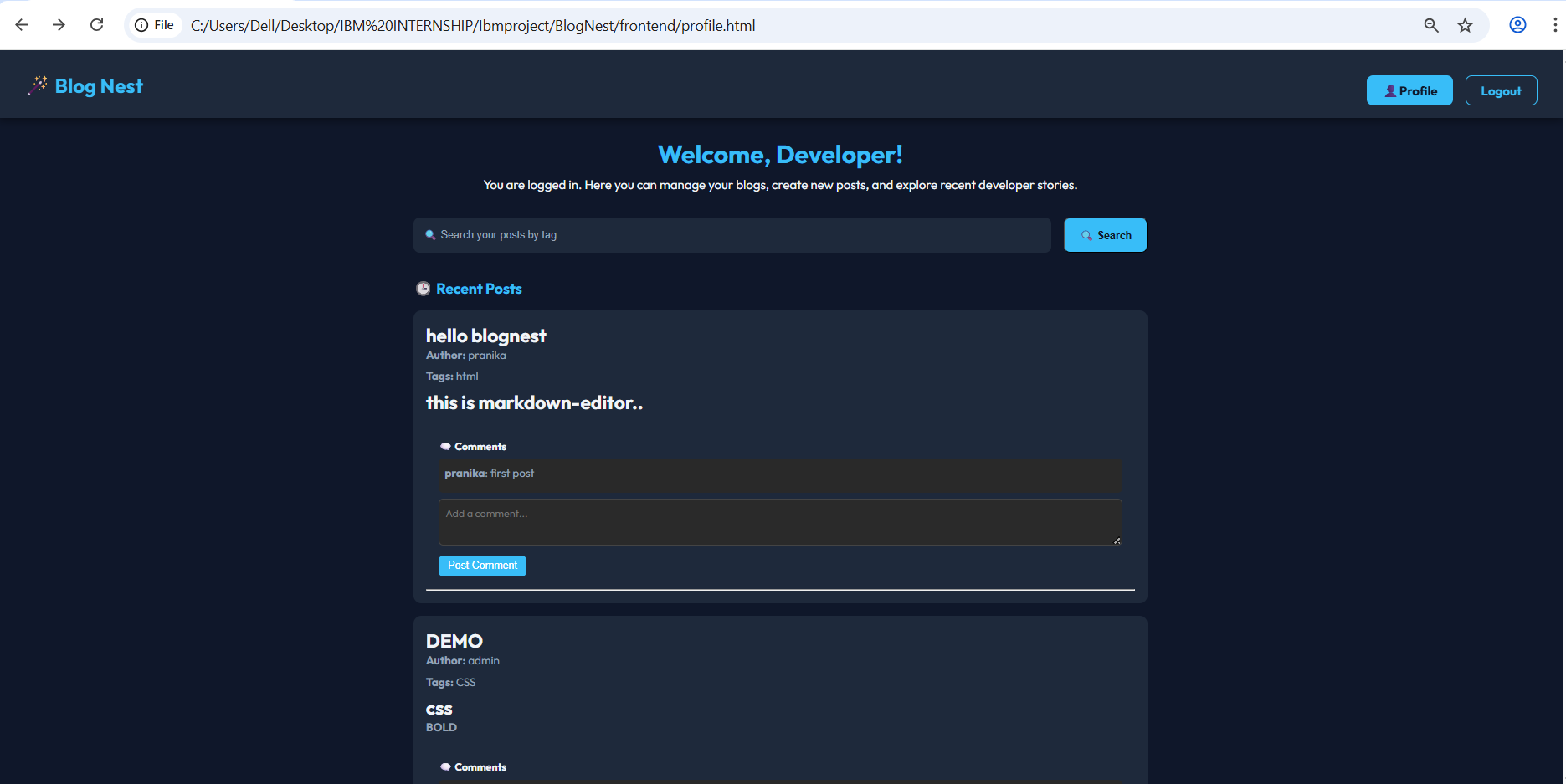
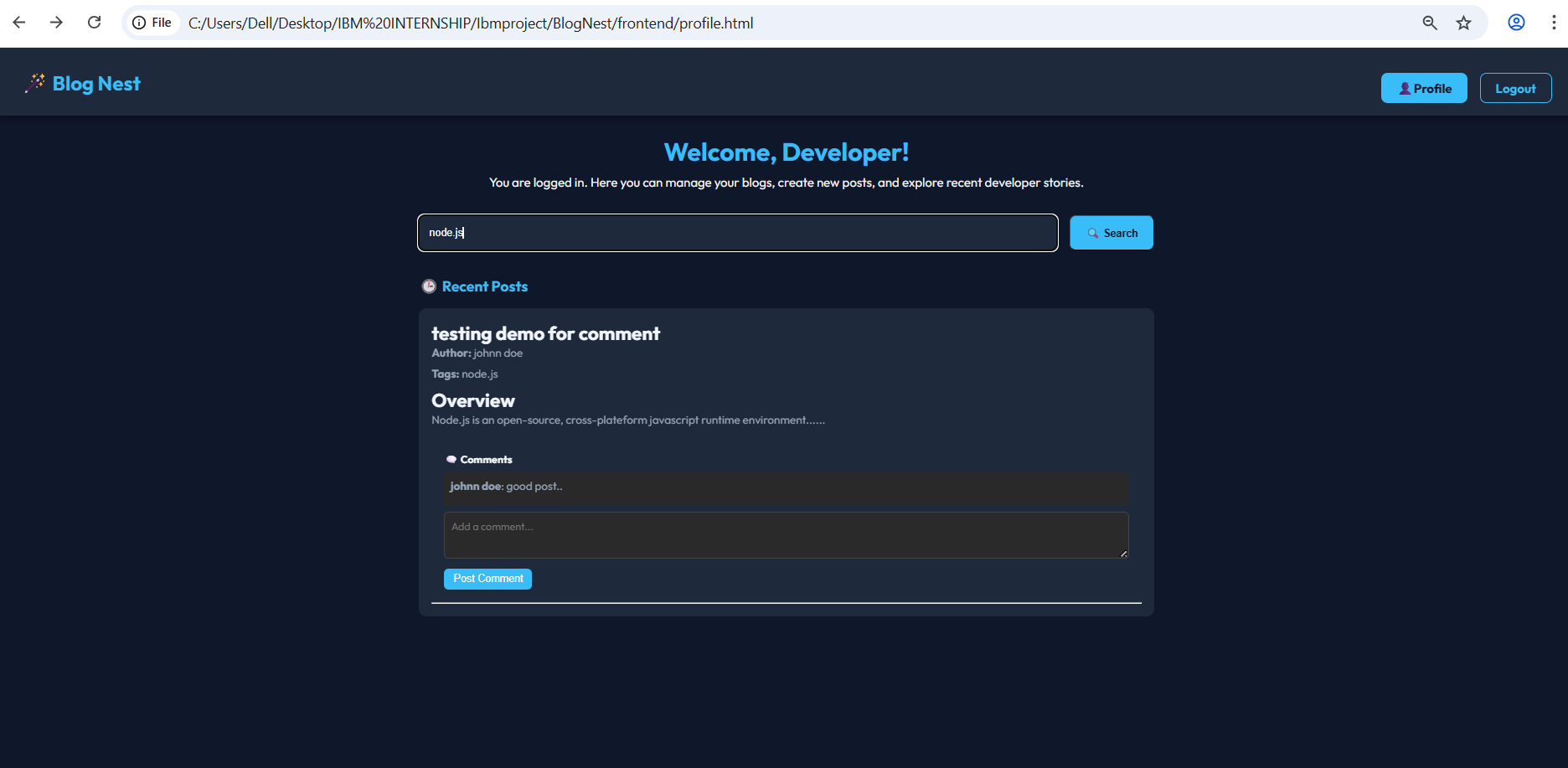


Fig.14 Updated User Profile

Fig.15 Searching Post By Tag

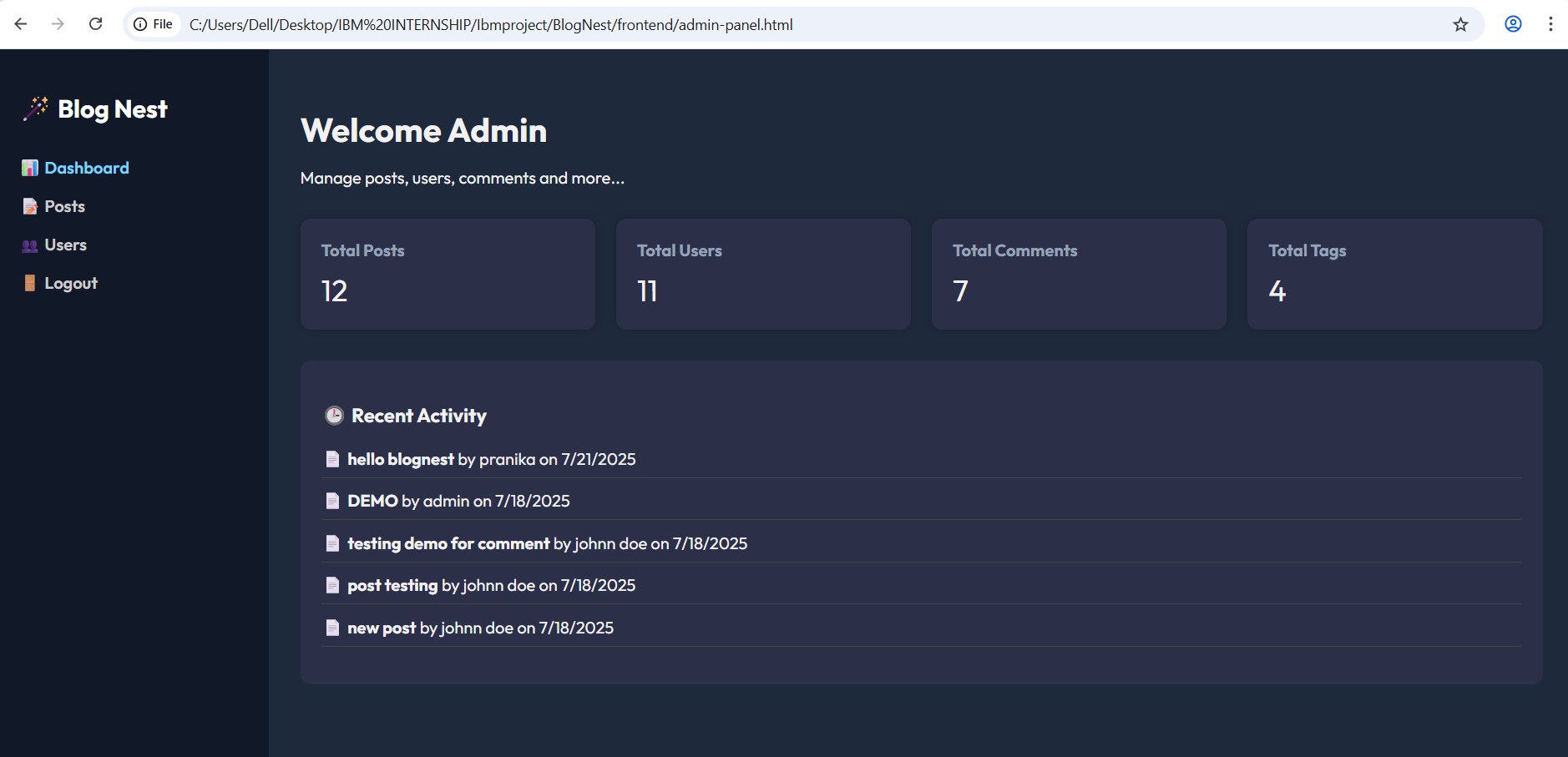
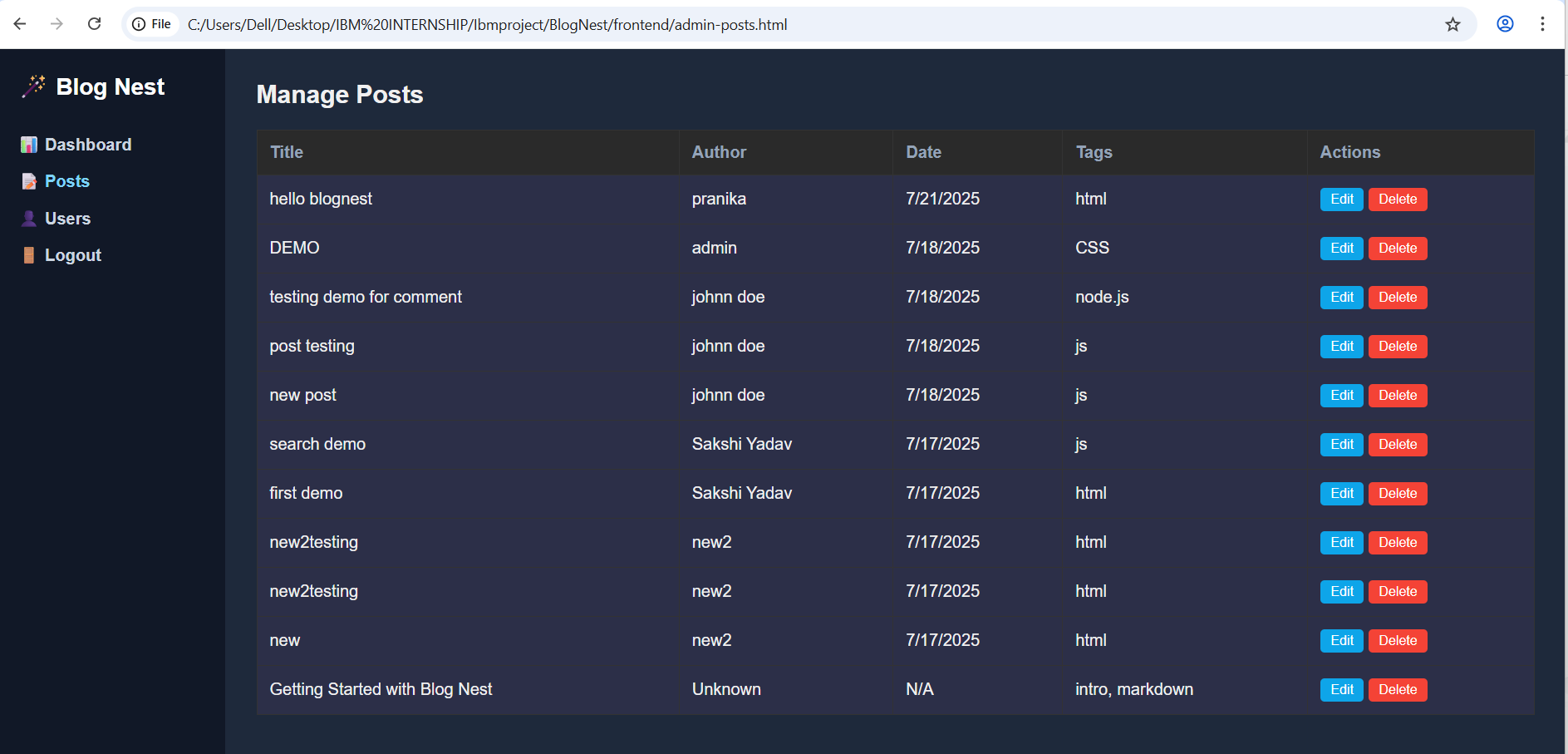


Fig.16 Admin Panel

Fig.17 Manage Posts

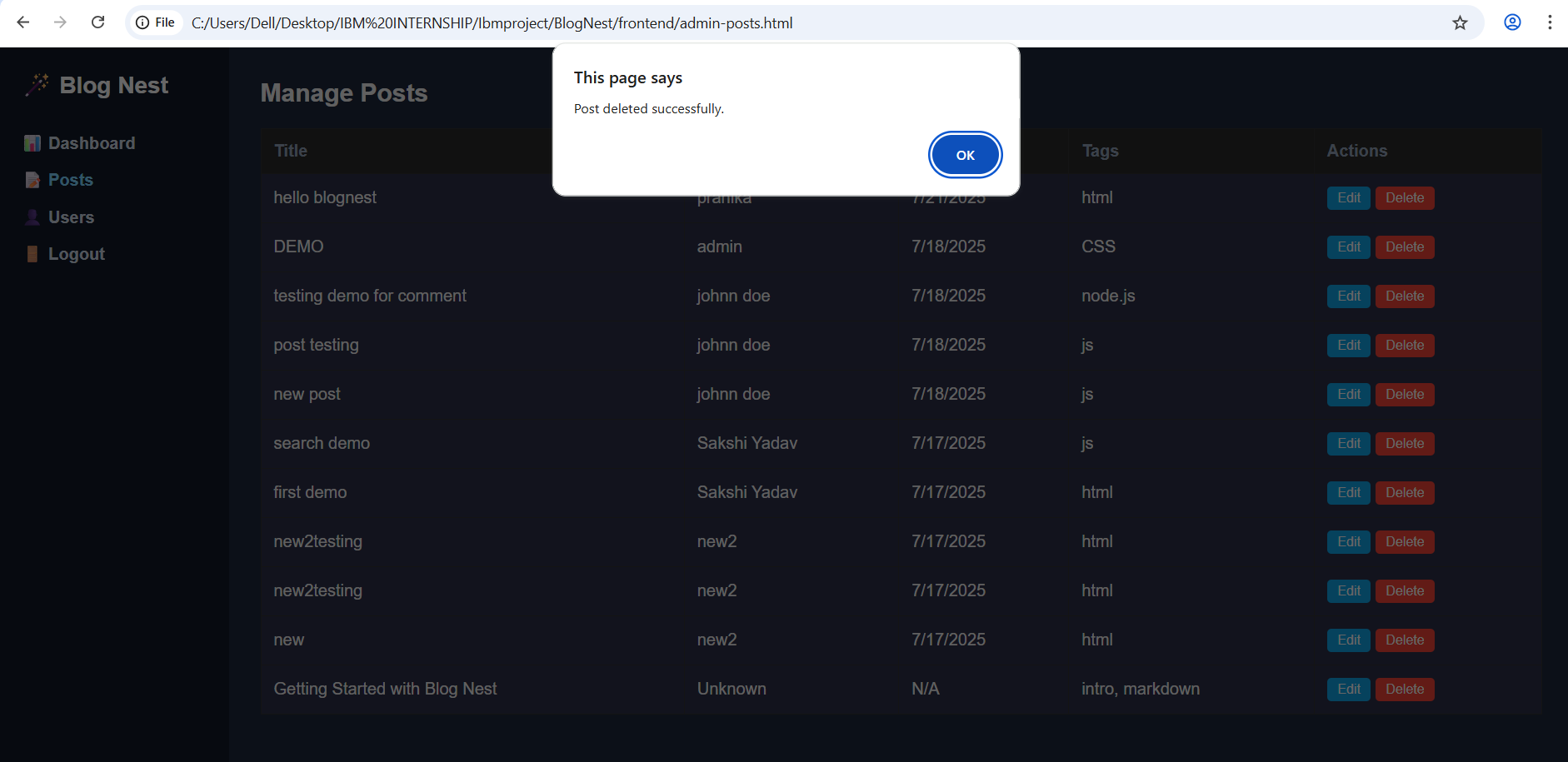
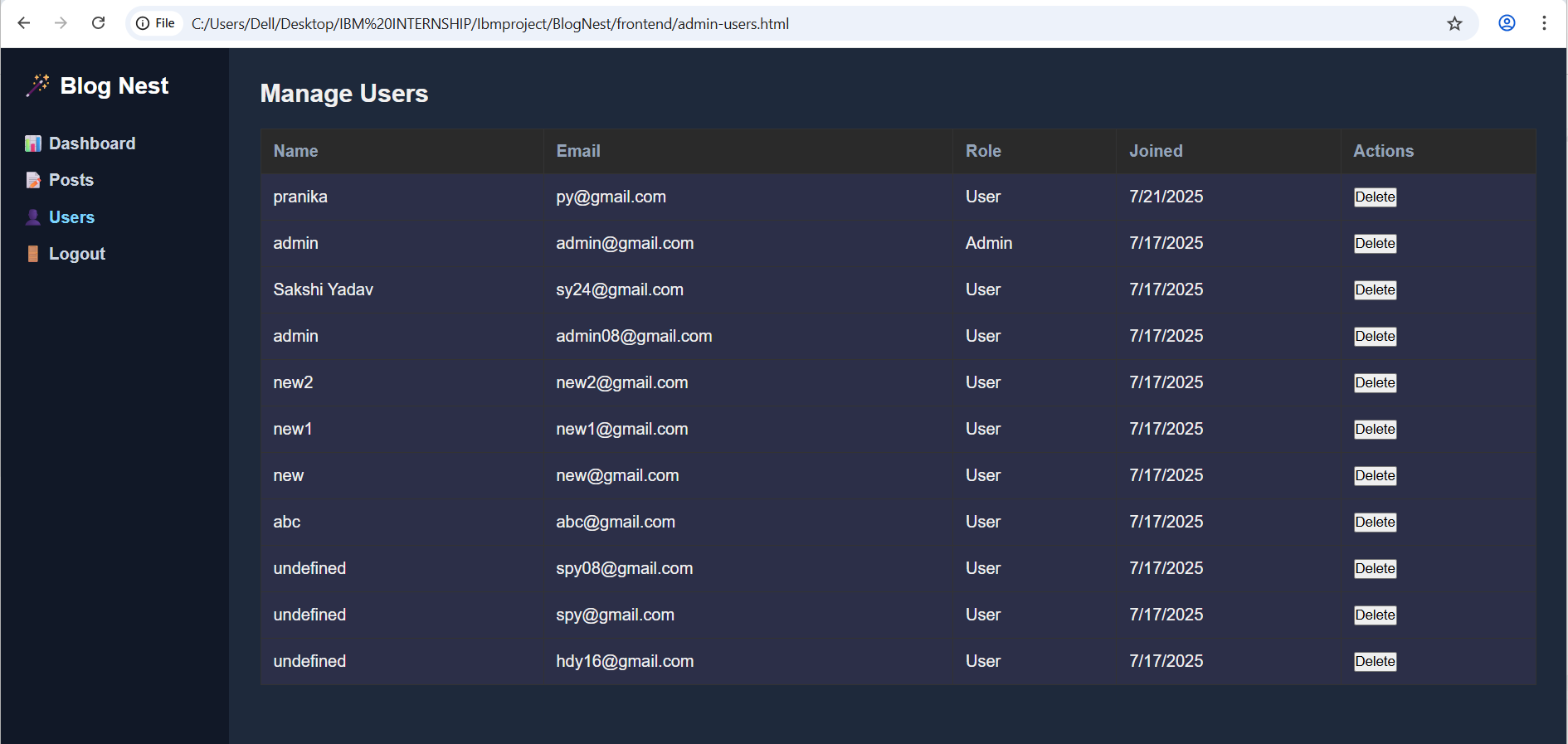


Fig.18 Delete Post

Fig.19 Manage Users

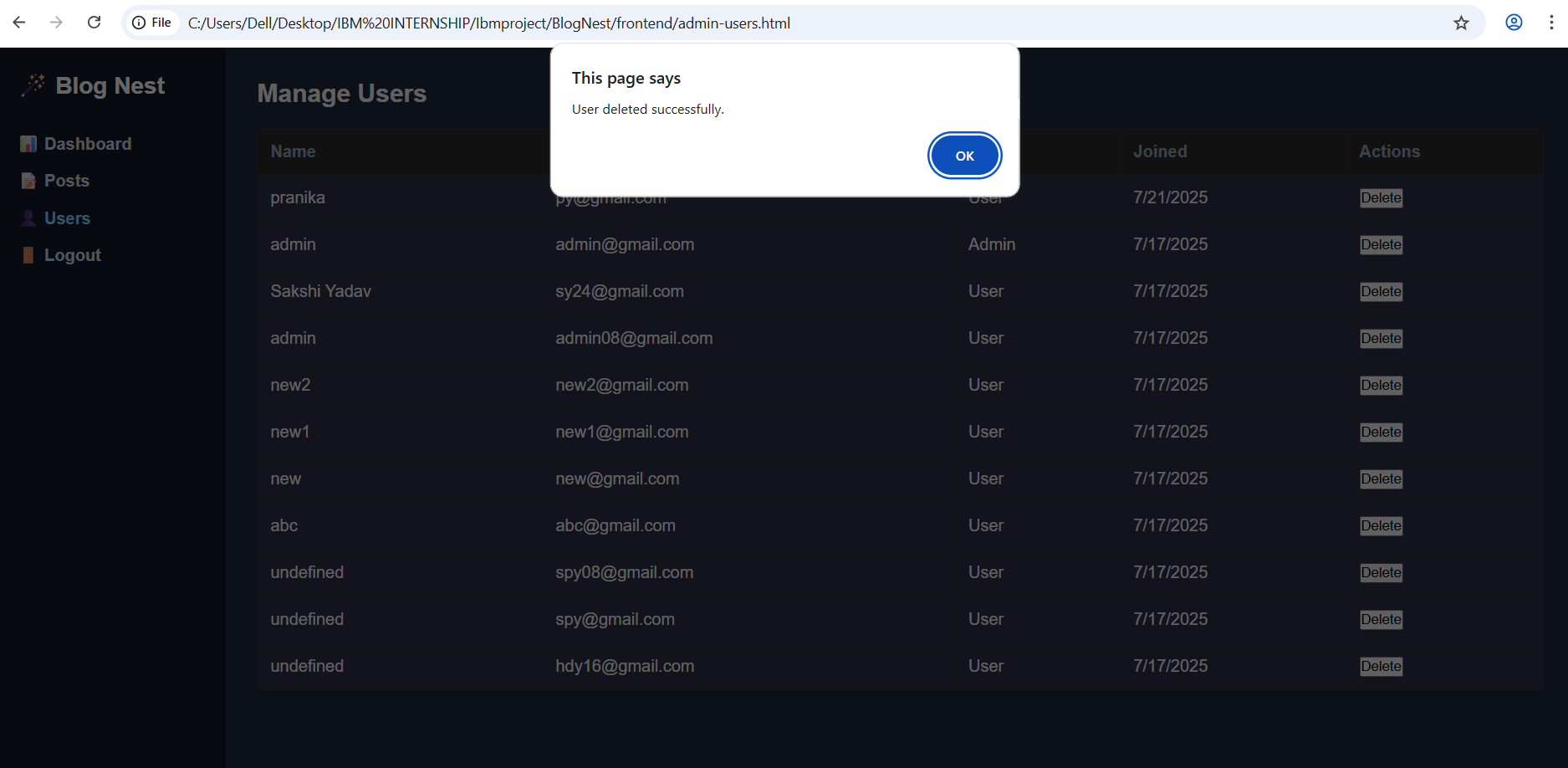


Fig.20 Delete Use

# Annexures

<https://github.com/sakshiyadav2408/BlogNest/tree/main>

# References

<https://expressjs.com>,

<https://mongoosejs.com>,

<https://jwt.io>,

<https://developer.mozilla.org>