**Blogging Tool Project Report**

**1. Introduction**

The Blogging Tool project aims to provide a simple yet robust platform for authors to manage blog posts and for readers to view published content. Built using Express.js, SQLite, and EJS templates, the application incorporates essential features such as authentication, session management, CRUD operations for articles, and customizable blog settings. write few more lines.The application's architecture ensures seamless integration of front-end and back-end functionalities, providing a user-friendly interface for both authors and readers. By leveraging Express.js for server-side logic and SQLite for efficient data management, the Blogging Tool prioritizes reliability and performance. The implementation of CRUD operations enables authors to create, edit, publish, and delete articles with ease, while customizable blog settings empower them to tailor their blog's presentation to their preferences. Overall, the project aims to balance simplicity with functionality, offering a robust blogging solution adaptable to diverse user needs and preferences.

**2. Objectives**

### Create a Secure Authentication System:

Implementing a password-based authentication system involves securely storing passwords using best practices such as hashing and salting. It ensures that only authorized authors can access and manage their blog posts, maintaining the integrity and confidentiality of user credentials.

### Enable CRUD Operations for Articles:

Enabling CRUD operations (Create, Read, Update, Delete) for articles empowers authors to efficiently manage their content lifecycle. It allows them to draft new posts, edit existing ones, publish when ready, and remove outdated or irrelevant articles, thereby maintaining a dynamic and relevant blog.

### Implement Blog Settings Management:

Implementing blog settings management provides authors with the flexibility to personalize their blog according to their branding and content strategy. Features such as modifying the blog title, author name, and potentially other settings like themes or categories, allow for a customized user experience tailored to the audience's expectations.

### Ensure Smooth Reader Experience:

A smooth reader experience is critical for engaging and retaining audience interest. This includes designing an intuitive and visually appealing interface for readers to easily navigate, search, and access published articles. Features like categorization, search functionalities, and clear presentation of article metadata (such as publication date, author, and tags) contribute to a positive user experience, encouraging return visits and interactions with the blog content.

**3. Features Implemented**

**Authentication and Session Management**

**Login System:** Authors can log in using a password stored securely in environment variables.

**Session Handling**: Utilizes `express-session` for managing user sessions securely.

**Article Management**

**Create and Edit Articles**: Authors can create new articles and edit existing ones.

**Publish and Delete Articles**: Functionality to publish draft articles and delete unwanted articles.

**Draft Management**: Authors can save articles as drafts for later editing and publishing.

**Blog Settings**

**Customizable Settings**: Authors can modify blog title and author name through a settings page.

**User Interfaces**

**Author Interface:** Includes pages for managing articles, settings, and authentication.

**Reader Interface:** Provides a clean interface for readers to view published articles.

**Extension Description**

Implemented Extension: Password Access for Author Pages

**Description**

- Secure the author pages with password authentication.

- Middleware to prevent unauthorized access to author endpoints.

- Create an author login page that authenticates the author against a server-side password stored in an environment variable.

- Use `express-session` to create secure sessions.

**4. Implementation Details**

**Technology Stack**

**Backend**: Node.js with Express.js for server-side development.

**Database**: SQLite for storing article data and blog settings.

**Frontend**: EJS templates for dynamic HTML rendering and minimal CSS for styling.

**High Level Schematic Diagram**

**Website Architecture**

The architecture of our blogging tool is divided into three main tiers:

1. **Client-Side (Frontend)**

- HTML, CSS, and JavaScript

- Uses EJS templates for server-side rendering

- Communicates with the server via HTTP requests

2. **Server-Side (Backend)**

- Express.js for handling HTTP requests and responses

- Routes for different functionalities (e.g., authentication, author actions, reader actions)

- Session management using `express-session`

3. **Database**

- SQLite3 for storing data

- Tables for authors, articles, and comments

**High Level Data Model**

**ER Diagram**

The Entity-Relationship (ER) diagram shows the tables and their relationships:

1. **Authors**

- `id` (Primary Key)

- `name`

- `password`

2. **Articles**

- `id` (Primary Key)

- `title`

- `content`

- `created\_at`

- `updated\_at`

- `published\_at`

- `views`

- `likes`

- `author\_id` (Foreign Key referencing `Authors`)

3. **Comments**

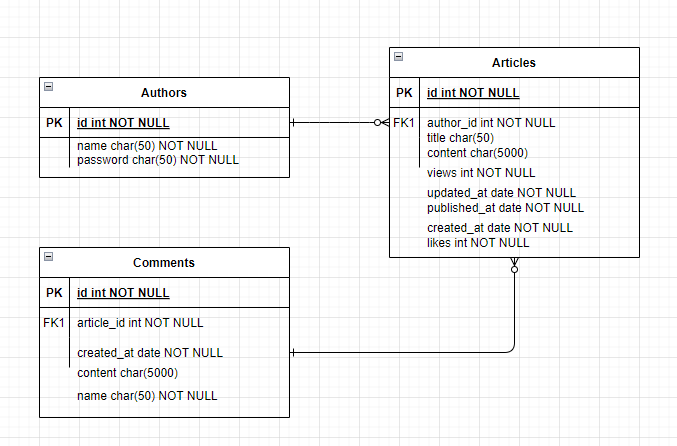
- `id` (Primary Key)

- `article\_id` (Foreign Key referencing `Articles`)

- `name`

- `content`

- `created\_at`



**Project Structure**

**Routing:** Organized into separate routes for authentication (`/auth`), author operations (`/author`), and reader views (`/reader`).

**Middleware**: Includes `authMiddleware` to protect author routes from unauthorized access.

**Views:** EJS templates used for rendering HTML pages dynamically.

**Database Management**

**Initialization**: Database schema and initial data setup managed through SQL scripts (`init.sql`).

**Queries:** CRUD operations handled in `queries.js` for interacting with the SQLite database.

**Future Considerations**

**Enhanced Security**: Implement additional security measures such as CSRF protection and input validation.

**Improved User Interface**: Enhance user experience with more responsive design and interactive features.

**Scalability:** Consider migrating to a more robust database solution for handling larger datasets and increased traffic.

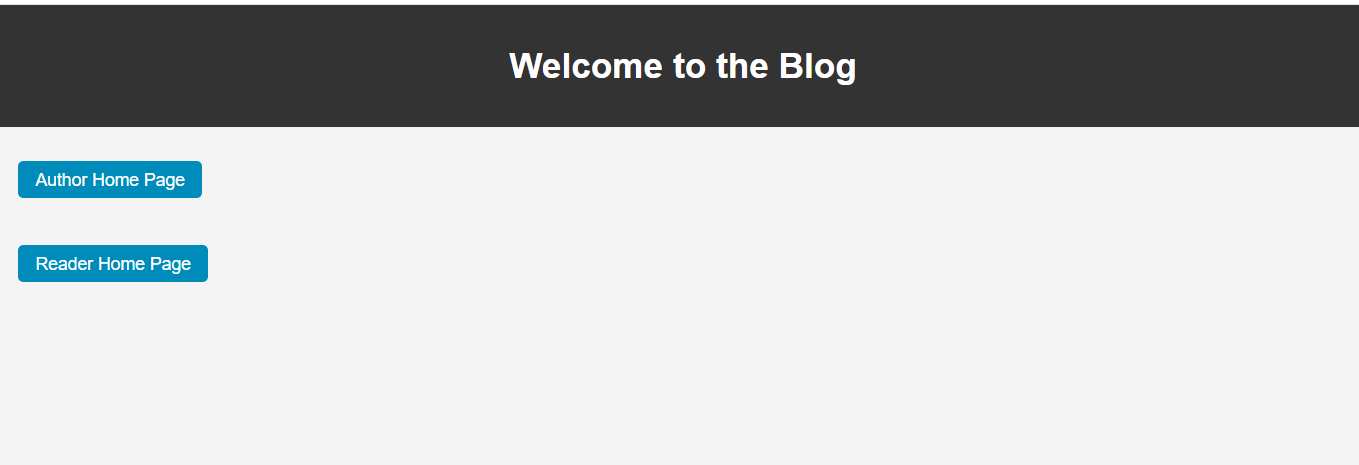
**Analytics and SEO**: Integrate analytics tools and optimize for search engines to enhance blog visibility.

**6. Conclusion**

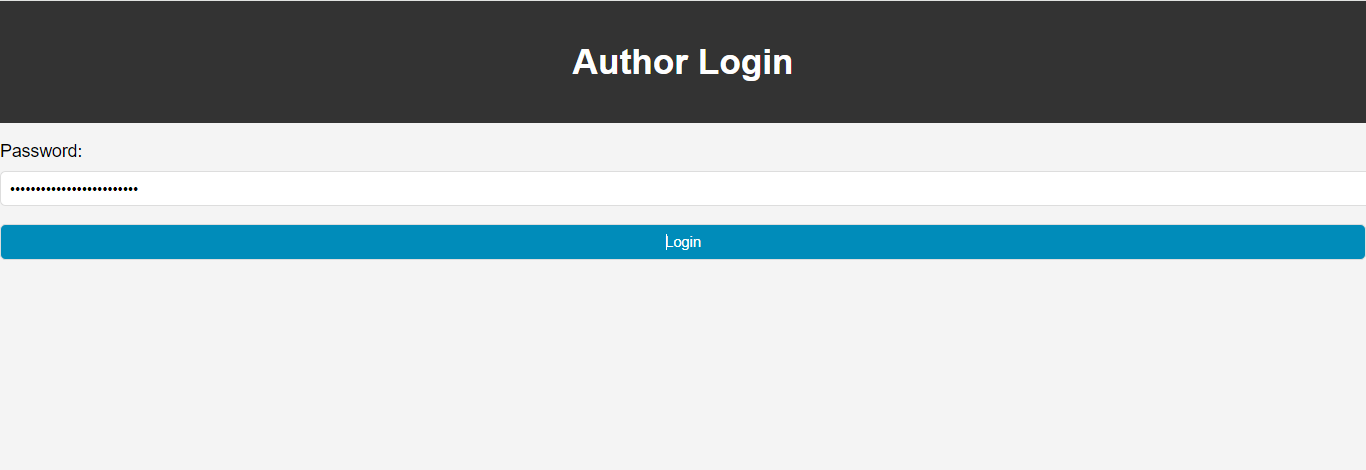
The Blogging Tool project successfully meets its objectives of providing a functional blogging platform for authors and readers alike. By leveraging modern web technologies and focusing on essential features, the application ensures a seamless experience for managing and consuming blog content. Future iterations will aim to further enhance security, usability, and scalability to meet evolving user needs.Future iterations of the Blogging Tool project will focus on enhancing the platform's security through additional measures like implementing CSRF protection and input validation to safeguard user data. Usability improvements will include refining the user interface for greater intuitiveness and responsiveness, ensuring a more enjoyable experience for both authors and readers. Scalability enhancements will involve optimizing database performance and considering migration to more robust database solutions to accommodate growing content and user traffic. By addressing these areas, the project aims to evolve into a more versatile and reliable tool that continues to meet the dynamic demands of blogging in the digital age.

**FrontEnd Screenshort**

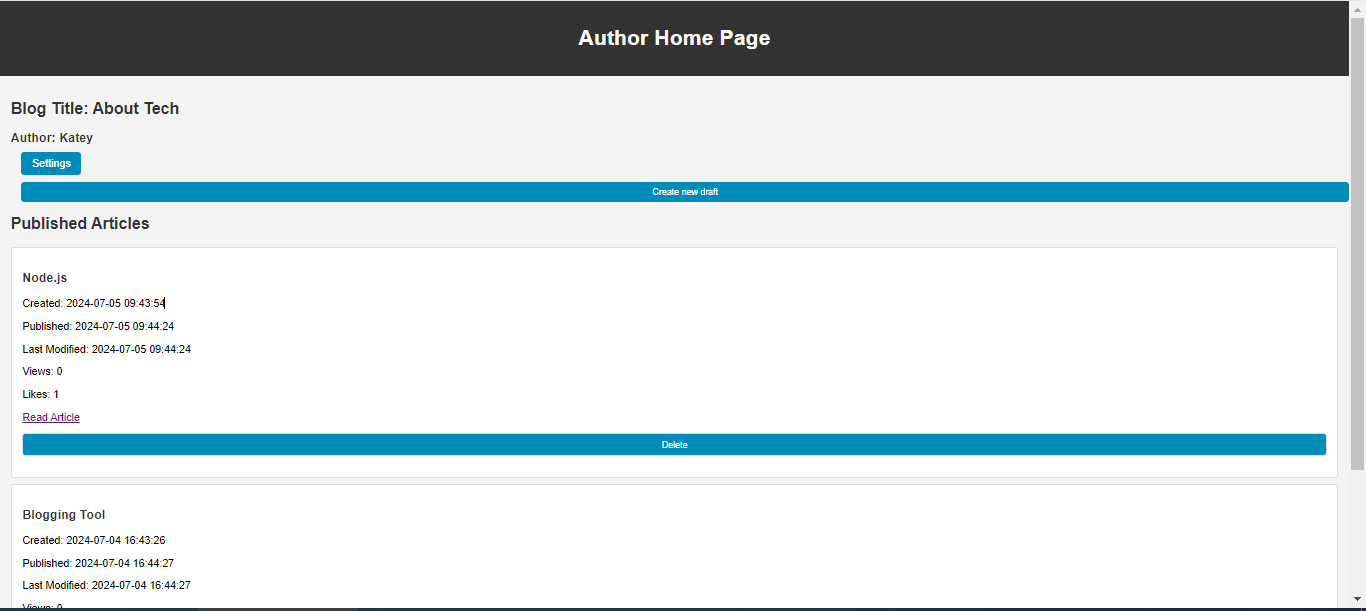
**Home Page**



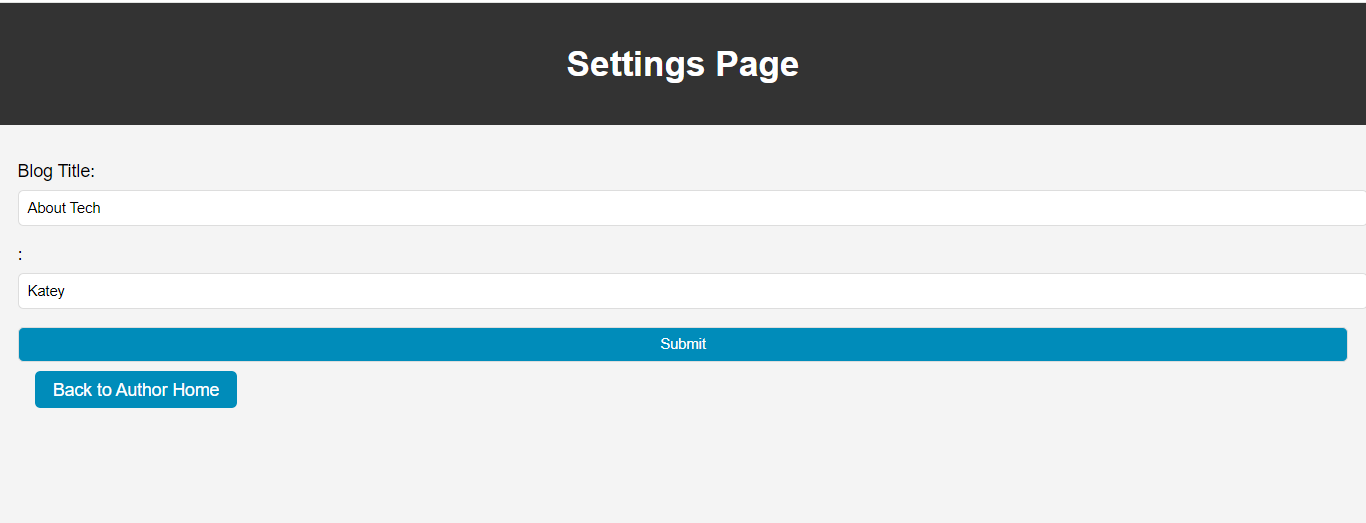
**Login page**



**Author Home Page**



**Author Setting Page**

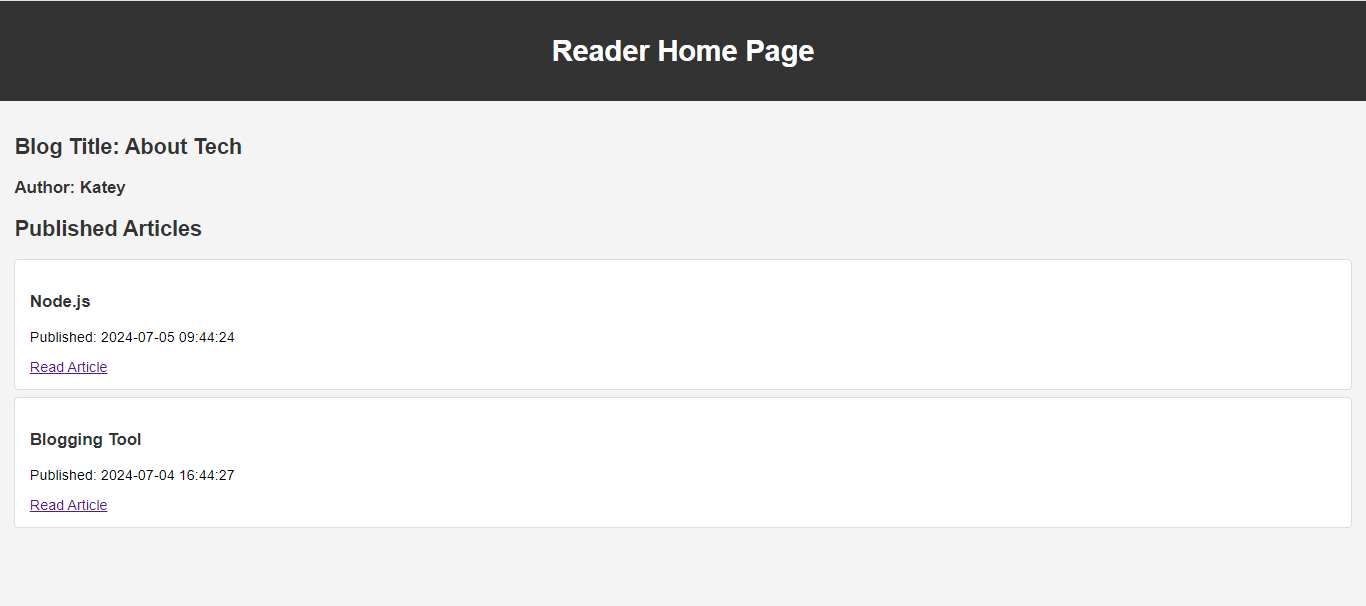




**Author Reading Page**



**Reader home page**



**Reader Page Add Comment**

