SOCIAL MEDIA DATABASE

A Project Report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Information Technology

by

Aryan Sinha, Arnab Sinha , Abhinav Krishna

230911438 ,230911356, 230911496

Under the guidance of

Dr.Swathi Prabhu
Assistant professor
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Dr. Akshay K C
Assistant professor- Senior scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

SUBMITTED ON - 16TH APRIL 2025 - ARYAN SINHA

ABSTRACT

This project is a social media platform designed to bring people together through simple and meaningful digital interactions. The system allows users to create personal profiles, send and accept friend requests, share posts, like and comment on content, and stay connected in a dynamic online environment.

The goal of this project is to provide an easy-to-use and engaging space where users can interact, build their networks, and stay updated with their friends' activities. Whether it's sharing a moment, reacting to a friend's post, or exploring new connections, the platform makes these experiences smooth and enjoyable.

Behind the scenes, it is powered by a well-organized database system that keeps all user information, posts, and interactions secure and easily accessible. It also includes tools for admins to help monitor activity and ensure a safe and welcoming environment for everyone.

Overall, the project focuses on creating a friendly and responsive social media experience that encourages connection, communication, and community in the digital world.

ACM Classification Keywords:

- [Database Management]: Systems—Relational databases
- [Information Storage and Retrieval]: Online Information Services—Web-based services
- [Group and Organization Interfaces]: Collaborative computing, Web-based interaction
- [Organizational Impacts]: Computer-supported collaborative work
- [Computer-Communication Networks]: Distributed Applications—Social networking

Sustainable Development Goals (SDGs) Addressed:

1. **SDG 9 – Industry, Innovation and Infrastructure:**
   The project supports digital innovation by creating a modern social media platform that enhances online interaction and information sharing. It contributes to building resilient digital infrastructure and encourages the use of advanced web and database technologies.

2. **SDG 11 – Sustainable Cities and Communities:**
   By enabling inclusive online communities and promoting meaningful digital communication, the system encourages social participation and community engagement in a sustainable digital environment.

Table of Contents

List of Tables

1. Users (user_id, email_id, phone_no, password, first_name, last_name, city, pincode, dob, gender)

2. Posts (post_id, posted_user_id, post_date, post_content)

3.Post_Comments (comment_id, post_id, commented_date, comment_content, commented_user_id)

4. Post_Likes (post_id, liked_user_id)

5. Post_Shares (post_id, shared_user_id)

6. Pages (page_id, page_name, page_content)

7. Page_Likes (page_id, page_user_id)

8. Friends (user_id, friend_id)

9. Comments_Like (comment_id, comment_liked_user_id)

List of Figures

ABBREVIATIONS

Abbreviation	Full Form
DBMS	Database Management System
SQL	Structured Query Language
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
ER	Entity-Relationship
DFD	Data Flow Diagram
ID	Identifier
DB	Database
ORM	Object-Relational Mapping
JSON	JavaScript Object Notation
UX	User Experience
UI	User Interface
JWT	JSON Web Token
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

Chapter 1

Introduction

1.1 Section 1

In today’s hyper-connected digital landscape, social media has become an integral part of everyday communication, information sharing, and networking. With the surge of online communities, the demand for secure, dynamic, and scalable platforms is greater than ever.

This project aims to design and implement a comprehensive Social Media Database Management System that handles essential features such as user registration, posts, likes, comments, friend connections, and user interactions. The goal is to create a robust backend infrastructure that ensures data integrity, efficiency, and fast retrieval, while supporting a smooth and responsive frontend user experience.

1.1 Section 2

Key Features of the Social Media Platform:

1. User Registration & Authentication – Allows new users to create accounts and log in securely.

2. Post Management – Users can create, view, like, comment on, and share posts.

3. Friends System – Enables users to connect with others and manage their friend lists.

4. Page Creation and Likes – Allows users to create pages and follow other users’ content.

5. Real-time Activity Recording – Stores and reflects real-time interactions such as likes, comments, and shares.

6. Enquiry & Feedback System – Users can raise queries or send feedback to the admin.

7. Secure Data Transactions – Ensures all user data and activity are safely processed and stored using Oracle DB.

Chapter 2

Literature Survey

2.1 Section 1

To develop the Social Media DBMS platform, the following technical areas and resources were studied:

1. Database Management:

- *"Database System Concepts"* by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan – foundational concepts in RDBMS design.
- *"Learning Oracle SQL and PL/SQL"* by John Garmany – focused on Oracle DB syntax and querying logic.
- Oracle official documentation: for advanced features, transactions, and optimization techniques.

2. Web Development with React & Node.js:

- React Documentation (https://reactjs.org): Core documentation and guides on building responsive UIs.
- Node.js & Express Docs (https://expressjs.com): Server-side programming tutorials and API development best practices.
- Tutorials from platforms like FreeCodeCamp, W3Schools, and GeeksforGeeks were referenced for full-stack integration.

3. Database Connectivity:

- oracledb npm package (https://oracle.github.io/node-oracledb/): Used to connect the Node.js server with Oracle Database securely and efficiently.

Chapter 3

Objectives

3.1 Section 1

The objective of this project is to develop a fully functional social media system that integrates a relational database to manage users, their content, and their interactions. The focus is on creating a scalable backend that supports complex data operations while ensuring an intuitive and responsive frontend.

3.1 Section 2

Role of DBMS in the Social Media Platform

A powerful Database Management System (DBMS) lies at the heart of the platform, handling all operations such as user data storage, posts, comments, likes, and relationships between entities. Oracle DB was chosen for its reliability and ability to manage large-scale relational data efficiently. This setup ensures data integrity, quick access, and effective retrieval of interrelated data.

3.1 Section 3

Why We Need This Social Media DBMS

In conventional systems, data handling may become fragmented and inefficient, especially as the user base grows. By designing a dedicated DBMS with an organized schema for user activities and relationships, this platform overcomes the limitations of loosely integrated solutions.
It improves:

- Query performance for likes, comments, and friends.
- Data consistency and normalization.
- User experience through responsive, data-backed interactions.

This solution aims to become a reference model for scalable, full-stack social media platforms using modern technologies.

CHAPTER 4 - DATA DESIGN

RELATIONAL SCHEMA

Users- Attributes: user_id (Primary Key), username, email, password_hash, full_name, bio, profile_pic_url, date_joined

Posts - Attributes: post_id (Primary Key), user_id (Foreign Key), content, media_url, timestamp

Comments- Attributes: comment_id (Primary Key), post_id (Foreign Key), user_id (Foreign Key), comment_text, timestamp

Likes- Attributes: like_id (Primary Key), post_id (Foreign Key), user_id (Foreign Key), timestamp

Friendships- Attributes: friendship_id (Primary Key), requester_id (Foreign Key), receiver_id (Foreign Key), status, request_date

Pages- Attributes: page_id (Primary Key), owner_id (Foreign Key), page_name, description, creation_date

Page_Likes- Attributes: like_id (Primary Key), page_id (Foreign Key), user_id (Foreign Key), timestamp

Feedback - Attributes: feedback_id (Primary Key), user_id (Foreign Key), message, status, submission_date

Normalization Status of Social Media Database Model

Functional Dependencies

Here are the Functional Dependencies identified in each table:

1. users:

User_ID ->Email_ID, Phone_No, Pass_word, First_name, Last_name, City, PinCode, DOB, Gender

2. posts:

Post_ID ->Posted_User_ID, Post_Date, Post_Content

3. post_comments:

Comment_ID -> textPost_ID, Commented_Date, Comment_Content, Commented_User_ID

4. post_likes:

(Post_ID, Liked_User_ID) is the primary key; no other functional dependencies exist.

5. post_shares:

(Post_ID, Shared_User_ID) is the primary key; no other functional dependencies exist.

6. pages:

Page_ID -> textPage_Name, Page_Content

7. page_likes:

(Page_ID, Page_User_ID) is the primary key; no other functional dependencies exist.

8. friends:

(User_ID, Friend_ID) is the primary key; no other functional dependencies exist.

9. comments_like:

(Comment_ID, Comment_liked_User_ID) is the primary key; no other functional dependencies exist.

Relationships

Below are the relationships between the tables, primarily defined through foreign keys:

1. post_comments:

  - Relationships:

    - Commented_User_ID *references* User_ID in users.

    - Post_ID *references* Post_ID in posts.

2. post_likes:

  - Relationships:

    - Liked_User_ID *references* User_ID in users.

    - Post_ID *references* Post_ID in posts.

3. post_shares:

  - Relationships:

    - Shared_User_ID *references* User_ID in users.

    - Post_ID *references* Post_ID in posts.

4. page_likes:

  - Relationships:

    - Page_User_ID *references* User_ID in users.

    - Page_ID *references* Page_ID in pages.

5. friends:

- Relationships:
    - User_ID *references* User_ID in users.
    - Friend_ID *references* User_ID in users.

6. comments_like:

- Relationships:
    - Comment_ID *references* Comment_ID in post_comments.
    - Comment_liked_User_ID *references* User_ID in users.

Universal Relation (U)

The universal relation (U) can be defined as a set that contains all the attributes from all tables in the database. It can be represented as :

U =

(User_ID, Email_ID, Phone_No, Pass_word, First_name, Last_name, City, PinCode, DOB, Gender,

textPost_ID, Posted_User_ID, Post_Date, Post_Content,

textComment_ID, Commented_Date, Comment_Content, Commented_User_ID,

textLiked_User_ID,

textShared_User_ID,

textPage_ID, Page_Name, Page_Content,

textPage_User_ID,

textFriend_ID, textComment_liked_User_ID)

Check First Normal Form (1NF)

To be in 1NF, each table must meet the following criteria:

- There are no repeating groups or arrays.
- Each column contains atomic values.
- Each column must have a unique name.
- The order in which data is stored does not matter.

Status for 1NF : All tables satisfy the 1NF requirements.

<u>Check Second Normal Form (2NF)</u>

To be in 2NF, the table must:

- Be in 1NF.

- Have all non-key attributes fully functionally dependent on the primary key.

Since most tables have composite primary keys, we must check that each non-key column is fully dependent on the entirety of the primary key.

Example: post_likes and post_shares depend on both Post_ID and Liked_User_ID/Shared_User_ID.

Status for 2NF : All tables satisfy the 2NF requirements.

<u>Check Third Normal Form (3NF)</u>

To be in 3NF, the table must:

- Be in 2NF.

- Have no transitive dependencies.

Example: users table contains City and PinCode. These should be separated into a locations table.

Status for 3NF : All tables satisfy the 3NF requirements.

<u>Check Boyce-Codd Normal Form (BCNF)</u> - *SATISFIED*

Boyce-Codd Normal Form (BCNF) is a stronger version of the Third Normal Form (3NF). A table is in BCNF if:

1.It is in 3NF.

2.For every one of its non-trivial functional dependencies ($X \rightarrow Y$), X is a superkey.

In simpler terms, a table is in BCNF if, for every functional dependency, the left-hand side is a superkey, meaning it can uniquely identify every row in the table.

REDUCED SCHEMA

1.Users
user_id (Primary Key), username, email, password

2.Posts
post_id (Primary Key), user_id (Foreign Key to Users), content, timestamp

3.Comments
comment_id (Primary Key), post_id (Foreign Key to Posts), user_id (Foreign Key to Users), comment_text

4.Likes
post_id (Foreign Key to Posts), user_id (Foreign Key to Users), (Composite Primary Key: post_id, user_id)

5.Friendships
user_id (Foreign Key to Users), friend_id (Foreign Key to Users), (Composite Primary Key: user_id, friend_id)

6.Pages
page_id (Primary Key), owner_id (Foreign Key to Users), page_name

7.Page_Likes
page_id (Foreign Key to Pages), user_id (Foreign Key to Users), (Composite Primary Key: page_id, user_id)

8.Feedback
feedback_id (Primary Key), user_id (Foreign Key to Users), message

Chapter 5

Methodology

The Social Media Database Management System is developed using a clear and structured methodology aimed at delivering a smooth user experience and strong backend performance. The project uses **React.js** for the frontend, **Node.js (without Express)** for the backend, and **SQL+** for relational database management and query execution.

User Registration and Login

1. User Registration:

- New users are required to register by entering details such as full name, email, password, phone number, date of birth, gender, and location.
- These details are stored in the `users` table using SQL+ queries executed from the Node.js backend.

2. User Login:

- Registered users can log in with their email and password.
- The Node.js backend handles manual credential verification using SQL+ queries to check the `users` table.

3. Dashboard Access:

- Upon successful login, users are redirected to the React dashboard, where they can view posts, interact with friends, and explore pages.

Posts and Interactions

1. Creating Posts:

- Users can publish text posts.
- The post content, user ID, and post date are inserted into the `posts` table using SQL+.

2. Commenting on Posts:

- Users can comment on any post.
- Comments are stored in the `post_comments` table along with the commenting user's ID and timestamp.

3. Liking and Sharing Posts:

- Post likes and shares are recorded in `post_likes` and `post_shares` tables respectively.

Friendships and Social Connections

1. Adding Friends:

- Users can send and accept friend requests.
- Accepted friendships are stored in the `friends` table as user ID and friend ID pairs.

2. Viewing Friends:

- Friends list is fetched by querying the `friends` table using SQL+ from the backend.

Pages and Engagement

1. Creating Pages:

- Users can create public pages with titles and descriptions.
- Pages are saved in the `pages` table.

2. Page Likes:

Users can like pages.

- These interactions are stored in the `page_likes` table.

Comment Likes and Feedback

1. Liking Comments:

- Users can like individual comments.
- These likes are tracked in the `comments_like` table using composite keys for `Comment_ID` and `Comment_liked_User_ID`.

Backend Functionality (Node.js Without Express)

- The backend server is developed using the core `http` module in Node.js.
- HTTP request handling is done manually:
  - URLs and methods are parsed using conditional statements.
  - POST data is collected by handling raw data streams.
  - JSON responses are constructed and sent manually.
- The backend uses the `child_process` module to execute **SQL+** commands through shell execution or scripts.
- SQL+ scripts handle data manipulation in the database using standard SQL commands.

Frontend Functionality (React.js)

- React handles all client-side views and components:
  - User login and registration forms

    - Post creation, commenting, liking, and sharing
    - Friend request management and viewing
    - Page creation and liking
- React uses `fetch()` or `Axios` to send requests to the backend Node.js server.

Database Management (SQL+)

- SQL+ (SQL*Plus) is used as the command-line interface for managing the database.
- Tables such as `users`, `posts`, `post_comments`, `post_likes`, `post_shares`, `comments_like`, `friends`, `pages`, and `page_likes` are created and maintained using SQL+ scripts.
- SQL queries for data insertion, retrieval, updates, and deletion are executed through SQL+ from within the Node.js backend.

Growth and Future Development

1. Admin Panel (Planned Feature):

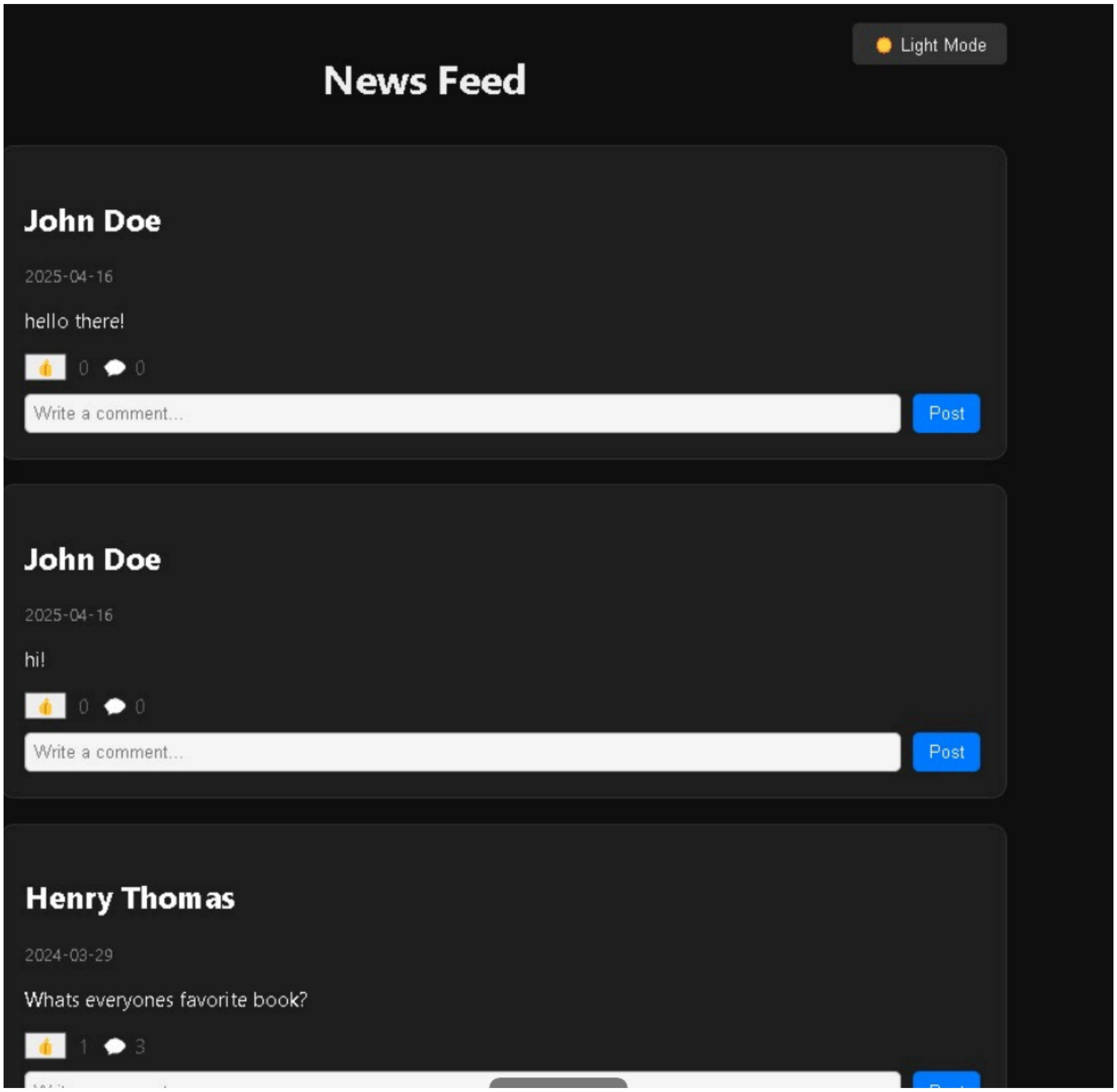
- Admins will be able to monitor flagged content, review user activity, and manage inappropriate content.

2. Feedback and Improvement:

- Feedback features can be added to allow users to report content or provide suggestions.
- Collected feedback will be analyzed to improve the system and enhance user satisfaction.

CHAPTER-6 RESULT

NEWS FEED



News Feed

John Doe

2025-04-16

hello there!

0 0

John Doe

2025-04-16

hi!

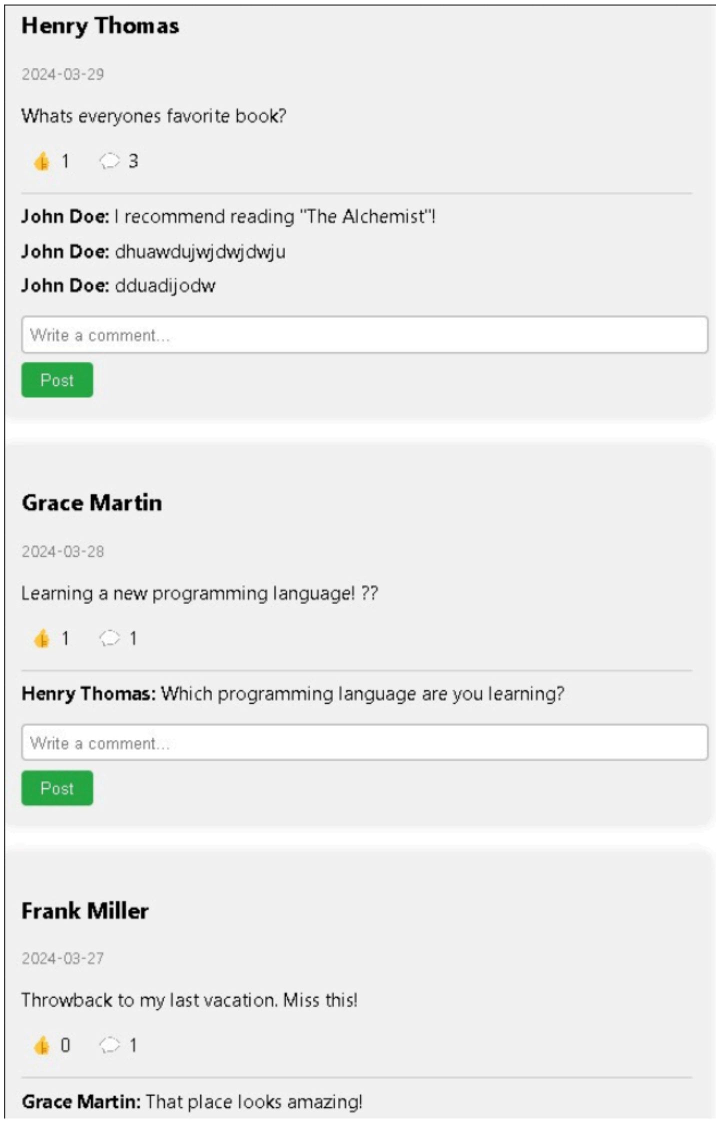
0 0

Henry Thomas

2024-03-29

Whats everyones favorite book?

1 3



Henry Thomas

2024-03-29

Whats everyones favorite book?

👍 1 💬 3

John Doe: I recommend reading "The Alchemist"!

John Doe: dhuawdujwjdwjdwju

John Doe: dduadijodw

Grace Martin

2024-03-28

Learning a new programming language! ??

👍 1 💬 1

Henry Thomas: Which programming language are you learning?

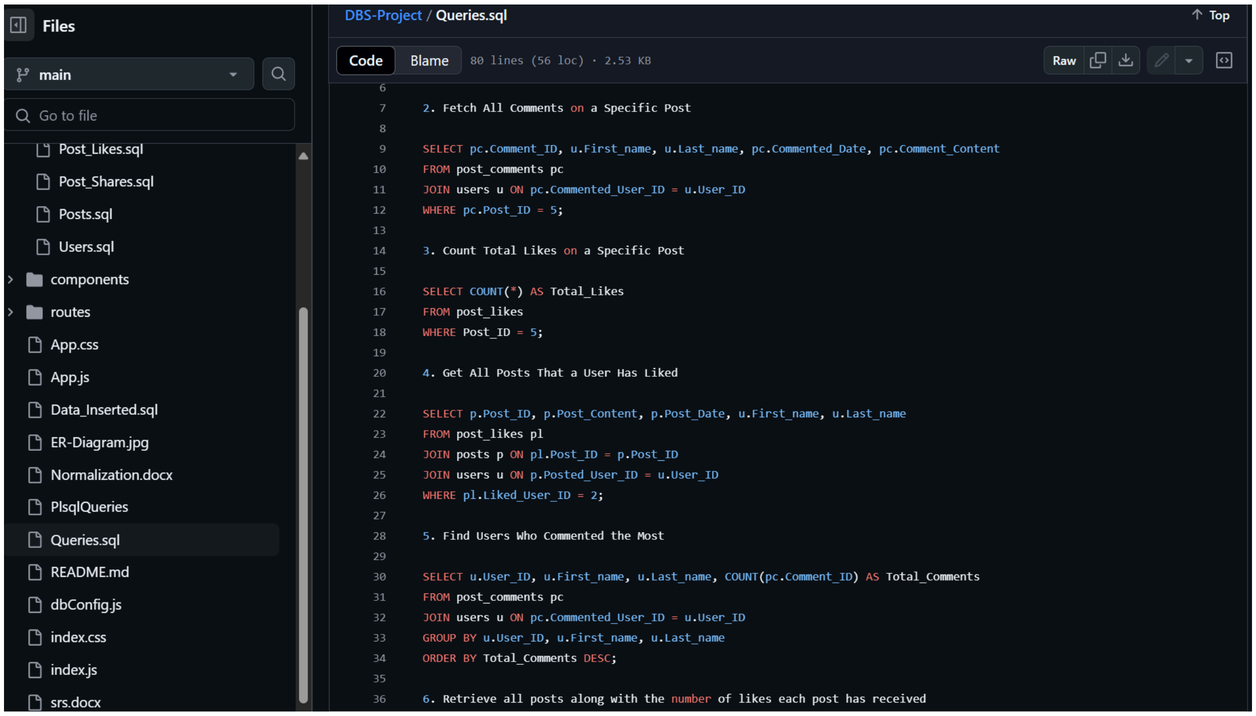
Frank Miller

2024-03-27

Throwback to my last vacation. Miss this!

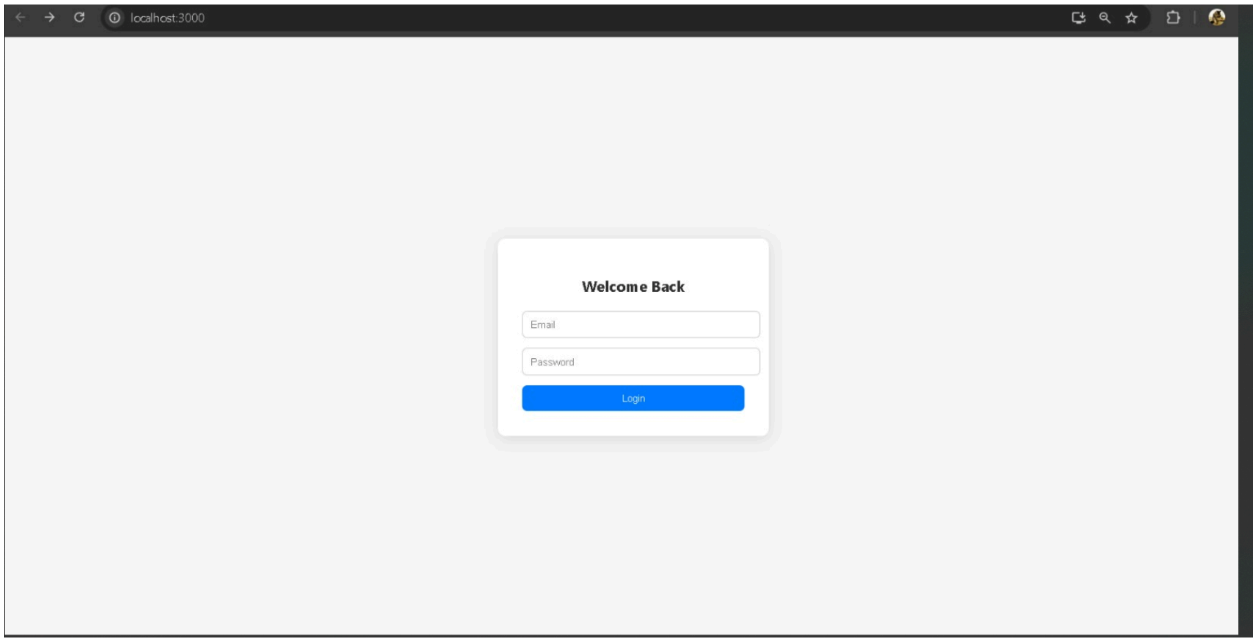
👍 0 💬 1

Grace Martin: That place looks amazing!

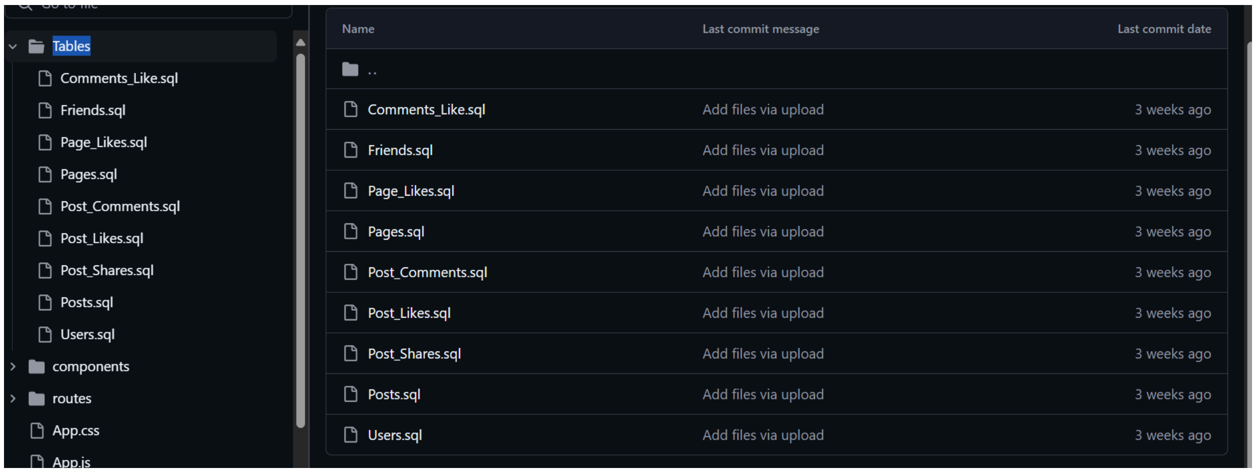


```sql

2. Fetch All Comments on a Specific Post

SELECT pc.Comment_ID, u.First_name, u.Last_name, pc.Commented_Date, pc.Comment_Content
FROM post_comments pc
JOIN users u ON pc.Commented_User_ID = u.User_ID
WHERE pc.Post_ID = 5;

3. Count Total Likes on a Specific Post

SELECT COUNT(*) AS Total_Likes
FROM post_likes
WHERE Post_ID = 5;

4. Get All Posts That a User Has Liked

SELECT p.Post_ID, p.Post_Content, p.Post_Date, u.First_name, u.Last_name
FROM post_likes pl
JOIN posts p ON pl.Post_ID = p.Post_ID
JOIN users u ON p.Posted_User_ID = u.User_ID
WHERE pl.Liked_User_ID = 2;

5. Find Users Who Commented the Most

SELECT u.User_ID, u.First_name, u.Last_name, COUNT(pc.Comment_ID) AS Total_Comments
FROM post_comments pc
JOIN users u ON pc.Commented_User_ID = u.User_ID
GROUP BY u.User_ID, u.First_name, u.Last_name
ORDER BY Total_Comments DESC;

6. Retrieve all posts along with the number of likes each post has received
```

LOGIN PAGE



TABLES



Name	Last commit message	Last commit date
..		
Comments_Like.sql	Add files via upload	3 weeks ago
Friends.sql	Add files via upload	3 weeks ago
Page_Likes.sql	Add files via upload	3 weeks ago
Pages.sql	Add files via upload	3 weeks ago
Post_Comments.sql	Add files via upload	3 weeks ago
Post_Likes.sql	Add files via upload	3 weeks ago
Post_Shares.sql	Add files via upload	3 weeks ago
Posts.sql	Add files via upload	3 weeks ago
Users.sql	Add files via upload	3 weeks ago

Chapter 7

Conclusion

7.1 Section 1

The Social Media Database Management System has been successfully developed with a focus on maintaining structured and efficient user interaction, content engagement, and data handling. By incorporating core modules like user registration, posts, comments, likes, shares, and page management, the system reflects the key functions of a modern social media platform.

The use of SQL+ ensures reliable and consistent data storage, preserving data integrity across all interactions. The frontend, built with React, enhances the user experience through dynamic, responsive components, while the backend, developed using Node.js (without Express), manages database transactions in a lightweight and efficient manner.

This project demonstrates a clear understanding of relational database design, effective CRUD operations, and seamless user communication through posts and reactions. It also emphasizes scalability and modularity for future improvements.

Future Work

7.2 Section 1

While the current version offers essential social media features, several enhancements can be considered to extend functionality and improve user experience:

1. Real-Time Chat Integration – Incorporate WebSocket-based communication (e.g., using `Socket.IO`) to enable real-time messaging between users.

2. Media Upload and Preview Support – Allow users to upload and preview images, videos, and files in posts or comments, enhancing content richness.

3. User Notification System – Implement a system for push or in-app notifications related to likes, comments, friend requests, and page activity.

4. Privacy and Role-Based Access Controls – Add more granular control over who can view, interact with, or manage specific content like posts, pages, or user profiles.

5. Graph-Based Friend Suggestion Algorithm – Use mutual connections and interaction patterns to suggest new friends, improving community engagement.

6. Admin Analytics Dashboard – Provide a dashboard with visualizations of user activity, engagement trends, most liked/shared content, and user growth statistics.

REFERENCES

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media.
-Covers core JavaScript used in frontend and backend development.

Banks, A. & Porcello, E. (2020). *Learning React: Functional Web Development with React and Redux* (2nd ed.). O'Reilly Media.
-Helpful for understanding React.js and component-based frontend architecture.

Node.js Documentation. (n.d.). *About Node.js*. Retrieved from https://nodejs.org/en/docs/
-Official documentation on Node.js, including the `http` module used for creating servers without Express.

Oracle. (n.d.). *SQL*Plus User's Guide and Reference*. Oracle Corporation. Retrieved from https://docs.oracle.com/en/database/oracle/oracle-database/19/sqpug/index.html
-Complete reference for working with SQL+ (SQL*Plus) command-line interface.

W3Schools. (n.d.). *SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/
-A beginner-friendly guide for writing and understanding SQL queries.

Mozilla Developer Network (MDN). (n.d.). *Using Fetch*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
Resource for using the `fetch()` API to interact with backend servers in React.

Oracle. (n.d.). *Oracle Database Concepts*. Retrieved from https://docs.oracle.com/en/database/
-Useful for understanding SQL concepts and database structures compatible with SQL+.

GitHub. (n.d.). *Open-source React Projects*. Retrieved from https://github.com/topics/react
-Explore real-world examples of React-based projects.