# The Application: Social Media App similar to Tweeter

**Goal:** The goal of the project is to implement backend logic which provides the necessary service to the application.

**Features (Section 1):**

- User registration with unique email and password
- Login authentication and session management with JSON Web Tokens and cookies.

**Features (Section 2):**

- Create, Read, Update, and Delete (CRUD) operations for Users while enforcing authentication and authorization
- CRUD operations for Tweets while enforcing authentication and authorization
- Associating Tweet with the Tweeter (user), i.e., relationship between the tweet and the user who created it as well as enforcing through authentication and authorization that only the tweeter can perform CRUD operations on tweets.

**Tech Stack:**

- Express - Node.js web application framework
- Database Layer: MongoDB Atlas
- Version Control with Git/GitHub
- SuperTest- Testing library
- Postman for manual testing

**Setting Up the project:**

In order to run properly, social-media-app project has some installation requirements and environmental dependencies.

1. Project Structure
2. Node Library Dependencies
3. Running the Application
4. Running the Unit Tests

**Project Structure**

**Controllers**: Most of the core code logic are implemented in the controller's directory, which contains all database interfacing functions.

**Doc**: contains a list of endpoints available

**Middleware:** contains function for authentication which is used on many of the routes

**Models**: contains model for interacting with MongoDB

**Routes:** contains exposed endpoints for users, tweet and authentication

**Tests**: contains functions to setup, configure and run tests

**Utils:** contains utility function such as password hashing

**Running the application**

**Application**

1. Get the project: The repo has been made public and available on GitHub. Download or clone the project repo.
    a. git clone https://github.com/TempleOkosun/social-media-app.git
2. Install node library dependencies: Ensure you have node and npm and run from the project root level:
    a. npm i
3. In order for the application to use Atlas, you will need a file called .**env** to contain the connection information. Open the file **dotenv_win** and you may choose to change the PORT, JWT_SECRET or JWT_EXPIRES_TIME. When you've edited the file, rename it to .**env** with the following command:
    a. mv dotenv_unix .env # on Unix
    b. ren dotenv_win .env # on Windows
    c. or simply rename from code editor
4. To start the app, run any of the below:
    a. npm run dev
    b. npm start

**Testing**

5. The test creates it own server and automatically assigns port
6. The test runs with its own database: mongodb-memory-server, and starting, cleaning and closing is taken care of for every test.
7. To run test
    a. npm run test
8. To run manual test, you can load postman manual test collections: **speer-tech.postman_collection.json** into postman and test the endpoints

**End Points**

{

"/api": "api docs",

"/api/register": "register",

"/api/login": "login",

"/api/logout": "logout",

"/api/users": "get all users",

"/api/user/:userId": "get a user/update/delete user" **(Login required),**

"/api/tweets": "get all tweets",

"/api/tweet": "get most recent tweet in the database",

"/api/tweet/new/:userId": "create new tweet" **(Login required),**

"/api/tweets/by/:userId": "get tweets by user" **(Login required),**

"/api/tweet/:tweetId": "update/delete tweet" **(Login required),**

}