6. Social Media Dashboard

Creating a Social Media Dashboard using the MEAN stack (MongoDB, Express.js, Angular, Node.js) to manage multiple social media accounts and post updates is a comprehensive project. Here's a high-level overview and code snippets to guide you.

Project Setup and Structure

Set up a new project folder and structure for your Social Media Dashboard. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application
ng new social-media-dashboard
```

- Backend (Node.js & Express.js)

Create the backend of your Social Media Dashboard using Node.js and Express.js.

Installation of Packages

Install the necessary packages for Express.js, Mongoose, Passport.js for user authentication, and other packages as needed.

npm install express mongoose passport passport-local bcryptjs jsonwebtoken

Setting up Express.js

Create your Express.js server, set up middleware, and handle routes.

- **javascript**

```
// server.js

const express = require('express');
const mongoose = require('mongoose');
const passport = require('passport');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();

// Middleware
app.use(bodyParser.json());
app.use(cors());

// Database connection
mongoose.connect('mongodb://localhost/social-media', {
  useNewUrlParser: true,
```

```
  useUnifiedTopology: true,
  useCreateIndex: true,
});


// Passport.js for user authentication
app.use(passport.initialize());
require('./config/passport')(passport);


// Routes
app.use('/api/users', require('./routes/users'));
app.use('/api/accounts', require('./routes/accounts'));
app.use('/api/posts', require('./routes/posts'));

const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Define Schemas and Models

Create Mongoose schemas and models for users, social media accounts, and posts. These models will define how data is structured and stored in MongoDB.

- **javascript**

```
// models/User.js
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  username: String,
  password: String,
  // Add more fields as needed
});

module.exports = mongoose.model('User', userSchema);

// Create similar models for Account and Post
```

Routes

Define routes for user management, social media accounts, and posting updates. For example, create routes for posting updates to different social media platforms:

- **javascript**

```
// routes/posts.js
const express = require('express');
const router = express.Router();
const Post = require('../models/Post');

// Create a new post
router.post('/', (req, res) => {
  // Validate user authentication and post data
  // Save the post to the appropriate social media account
  // Return success response
});
```

- ## Frontend (Angular)

Create the frontend of your Social Media Dashboard using Angular. Design the user interface and implement social media account management and posting functionality.

Design and UI

Design the user interface for your Social Media Dashboard using Angular components, templates, and styles.

Social Media Account Management

Create components to manage social media accounts, including adding, editing, and deleting accounts.

Posting Updates

Implement components for posting updates to different social media accounts. Use APIs or SDKs provided by the social media platforms for posting.

MongoDB

Create a MongoDB database to store user profiles, social media accounts, and posts.

Authentication

Set up user authentication using Angular components and Passport.js on the server side. Create registration and login forms.

- **typescript**

```
// auth.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
```

```
@Injectable({
  providedIn: 'root',
})
export class AuthService {
  constructor(private http: HttpClient) {}

  register(user) {
    return this.http.post('/api/users/register', user);
  }

  login(user) {
    return this.http.post('/api/users/login', user);
  }
}
```

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you handle user authentication, social media account management, and posting updates properly.

Building a full-fledged Social Media Dashboard is a complex project, and you should consider other aspects like social media API

integration, error handling, and user experience design as you progress with your project.