MINE SOCIAL MEDIA APP :

```
//  Project: Mini Social Media App (Express + SQLite)
// This version separates backend and frontend cleanly for execution.

// ==========================
// File: package.json
// ==========================
{
  "name": "mini-social",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.0",
    "sequelize": "^6.32.1",
    "sqlite3": "^5.1.6"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}

// ==========================
// File: index.js
// ==========================
const express = require('express');
const cors = require('cors');
const path = require('path');
const { sequelize } = require('./models');
const authRoutes = require('./routes/auth');
const userRoutes = require('./routes/users');
const postRoutes = require('./routes/posts');

const app = express();
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

```
// API Routes
app.use('/api/auth', authRoutes);
app.use('/api/users', userRoutes);
app.use('/api/posts', postRoutes);

// Serve Frontend
app.use('/', express.static(path.join(__dirname, 'public')));

const PORT = process.env.PORT || 4000;
(async () => {
  await sequelize.sync();
  app.listen(PORT, () => console.log(` Server running on http://localhost:${PORT}`));
})();

// ==========================
// File: models.js
// ==========================
const { Sequelize, DataTypes } = require('sequelize');
const path = require('path');

const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: path.join(__dirname, 'data.sqlite'),
  logging: false,
});

const User = sequelize.define('User', {
  name: DataTypes.STRING,
  username: { type: DataTypes.STRING, unique: true },
  email: { type: DataTypes.STRING, unique: true },
  passwordHash: DataTypes.STRING,
  bio: DataTypes.TEXT,
});

const Post = sequelize.define('Post', { content: DataTypes.TEXT });
const Comment = sequelize.define('Comment', { content: DataTypes.TEXT });
const Follow = sequelize.define('Follow', {});
const Like = sequelize.define('Like', {});

User.hasMany(Post);
Post.belongsTo(User);
Post.hasMany(Comment);
Comment.belongsTo(Post);
Comment.belongsTo(User);
User.hasMany(Comment);
User.belongsToMany(User, { as: 'Followers', through: Follow, foreignKey: 'followingId' });
User.belongsToMany(User, { as: 'Following', through: Follow, foreignKey: 'followerId' });
User.belongsToMany(Post, { through: Like, as: 'LikedPosts' });
Post.belongsToMany(User, { through: Like, as: 'Likers' });

module.exports = { sequelize, User, Post, Comment, Follow, Like };

// ==========================
// File: middleware/auth.js
// ==========================
const jwt = require('jsonwebtoken');
const { User } = require('../models');
const JWT_SECRET = process.env.JWT_SECRET || 'dev_secret_change_me';

async function authMiddleware(req, res, next) {
  const auth = req.headers.authorization;
  if (!auth) return res.status(401).json({ error: 'Missing auth' });
  const token = auth.split(' ')[1];
  try {
    const payload = jwt.verify(token, JWT_SECRET);
    const user = await User.findByPk(payload.id);
    if (!user) return res.status(401).json({ error: 'Invalid user' });
    req.user = user;
    next();
  } catch {
    res.status(401).json({ error: 'Invalid token' });
  }
}

module.exports = { authMiddleware, JWT_SECRET };

// ==========================
// File: routes/auth.js
// ==========================
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { User } = require('../models');
const { JWT_SECRET } = require('../middleware/auth');

const router = express.Router();

router.post('/register', async (req, res) => {
  const { name, username, email, password } = req.body;
  if (!username || !password) return res.status(400).json({ error: 'username & password required' });
  try {
    const hash = await bcrypt.hash(password, 10);
    const user = await User.create({ name, username, email, passwordHash: hash });
    const token = jwt.sign({ id: user.id }, JWT_SECRET);
    res.json({ token, user: { id: user.id, username: user.username, name: user.name } });
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

router.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ where: { username } });
  if (!user) return res.status(400).json({ error: 'Invalid credentials' });
  const ok = await bcrypt.compare(password, user.passwordHash);
  if (!ok) return res.status(400).json({ error: 'Invalid credentials' });
  const token = jwt.sign({ id: user.id }, JWT_SECRET);
  res.json({ token, user: { id: user.id, username: user.username, name: user.name } });
});

module.exports = router;

// ==========================
// File: routes/users.js
// ==========================
const express = require('express');
const { User, Post } = require('../models');
const { authMiddleware } = require('../middleware/auth');

const router = express.Router();

router.get('/:id', async (req, res) => {
  const user = await User.findByPk(req.params.id, { attributes: ['id', 'name', 'username', 'bio'] });
  if (!user) return res.status(404).json({ error: 'Not found' });
  const posts = await Post.count({ where: { UserId: user.id } });
  const followers = await user.countFollowers();
  const following = await user.countFollowing();
  res.json({ user, counts: { posts, followers, following } });
});

router.post('/:id/follow', authMiddleware, async (req, res) => {
  const target = await User.findByPk(req.params.id);
  if (!target) return res.status(404).json({ error: 'Not found' });
  if (target.id === req.user.id) return res.status(400).json({ error: 'Cannot follow yourself' });
  const exists = await req.user.hasFollowing(target);
  if (exists) {
    await req.user.removeFollowing(target);
    res.json({ following: false });
  } else {
    await req.user.addFollowing(target);
    res.json({ following: true });
  }
});

module.exports = router;

// ==========================
// File: routes/posts.js
// ==========================
const express = require('express');
const { Post, Comment, User } = require('../models');
const { authMiddleware } = require('../middleware/auth');

const router = express.Router();

router.post('/', authMiddleware, async (req, res) => {
  const post = await Post.create({ content: req.body.content, UserId: req.user.id });
  res.json(post);
});

router.get('/', async (req, res) => {
  const posts = await Post.findAll({ order: [['createdAt', 'DESC']], include: [{ model: User, attributes: ['id',
'username', 'name'] }] });
  res.json(posts);
});

router.get('/:id', async (req, res) => {
  const post = await Post.findByPk(req.params.id, { include: [{ model: User, attributes: ['id', 'username', 'name'] },
{ model: Comment, include: [{ model: User, attributes: ['id', 'username'] }] }] });
  if (!post) return res.status(404).json({ error: 'Not found' });
  res.json(post);
});

router.post('/:id/comments', authMiddleware, async (req, res) => {
  const post = await Post.findByPk(req.params.id);
  if (!post) return res.status(404).json({ error: 'Not found' });
  const comment = await Comment.create({ content: req.body.content, PostId: post.id, UserId: req.user.id });
  res.json(comment);
});

module.exports = router;

// ==========================
// File: public/index.html
// ==========================
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Mini Social</title>
  <link rel="stylesheet" href="/styles.css" />
</head>
<body>
  <div class="app">
    <header>
      <h1>Mini Social</h1>
      <div id="auth">
        <button id="btn-show-login">Login</button>
        <button id="btn-show-register">Register</button>
        <button id="btn-logout" style="display:none">Logout</button>
      </div>
    </header>

    <main>
      <section id="auth-forms"></section>
      <section id="compose" style="display:none">
        <textarea id="post-content" placeholder="What's happening?"></textarea>
        <button id="btn-post">Post</button>
      </section>
      <section id="feed"></section>
    </main>
  </div>
  <script src="/app.js"></script>
</body>
</html>

// ==========================
// File: public/styles.css
// ==========================
body { font-family: Arial, sans-serif; background: #f2f4f7; margin:0; }
.app { max-width: 700px; margin: 12px auto; background: white; padding: 12px; border-radius: 8px; }
header { display:flex; justify-content:space-between; align-items:center; }
textarea { width:100%; min-height:70px; }
.post { border-bottom:1px solid #eee; padding:10px 0; }
button { padding:6px 10px; margin:4px; }

// ==========================
// File: public/app.js
// ==========================
const api = (path, opts = {}) => fetch('/api' + path, opts).then(r => r.json());
function token() { return localStorage.getItem('token'); }
function authHeaders() { return token() ? { Authorization: 'Bearer ' + token() } : {}; }

async function loadFeed() {
  const data = await api('/posts');
  const feed = document.getElementById('feed');
  feed.innerHTML = '';
  data.forEach(p => {
    const div = document.createElement('div');
    div.className = 'post';
    div.innerHTML = `<strong>${p.User.username}</strong> <small>${new
Date(p.createdAt).toLocaleString()}</small><p>${p.content}</p><button data-id="${p.id}" class="btn-
like">Like</button><button data-id="${p.id}" class="btn-comment">Comment</button>`;
    feed.appendChild(div);
  });
}

async function init() {
  document.getElementById('btn-show-login').onclick = showLogin;
  document.getElementById('btn-show-register').onclick = showRegister;
  document.getElementById('btn-logout').onclick = () => { localStorage.removeItem('token'); location.reload(); };
  document.getElementById('btn-post').onclick = async () => {
    const content = document.getElementById('post-content').value;
    if (!content) return alert('Write something');
    await fetch('/api/posts', { method: 'POST', headers: { 'Content-Type':'application/json', ...authHeaders() },
body: JSON.stringify({ content }) });
    document.getElementById('post-content').value = '';
    loadFeed();
  };

  if (token()) document.getElementById('compose').style.display = 'block';
  await loadFeed();
}

function showLogin() {
  const forms = document.getElementById('auth-forms');
  forms.innerHTML = `<h3>Login</h3><input id='li-username' placeholder='username' /><input id='li-password'
type='password' placeholder='password' /><button id='li-btn'>Login</button>`;
  document.getElementById('li-btn').onclick = async () => {
    const username = document.getElementById('li-username').value;
    const password = document.getElementById('li-password').value;
    const res = await fetch('/api/auth/login', { method: 'POST', headers: { 'Content-Type':'application/json' }, body:
JSON.stringify({ username, password }) });
    const data = await res.json();
    if (data.token) { localStorage.setItem('token', data.token); location.reload(); }
    else alert(data.error || 'Login failed');
  };
}

function showRegister() {
  const forms = document.getElementById('auth-forms');
  forms.innerHTML = `<h3>Register</h3><input id='re-name' placeholder='name' /><input id='re-username'
placeholder='username' /><input id='re-password' type='password' placeholder='password' /><button id='re-
btn'>Register</button>`;
  document.getElementById('re-btn').onclick = async () => {
    const name = document.getElementById('re-name').value;
    const username = document.getElementById('re-username').value;
    const password = document.getElementById('re-password').value;
    const res = await fetch('/api/auth/register', { method: 'POST', headers: { 'Content-Type':'application/json' },
body: JSON.stringify({ name, username, password }) });
    const data = await res.json();
    if (data.token) { localStorage.setItem('token', data.token); location.reload(); }
    else alert(data.error || 'Register failed');
  };
}

window.addEventListener('DOMContentLoaded', init);
```