Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Microblogging App</title>
   <link rel="stylesheet" href="style.css">

</head>
<body>
   <div class="container">

      <h1>Microblogging Application</h1>
      <div id="auth">
         <h2>Register</h2>
         <input type="text" id="regUsername" placeholder="Username">
         <input type="password" id="regPassword" placeholder="Password">
         <button id="registerBtn" onclick="alert('Registered successfully')">Register</button>
      </div>
         <input type="text" id="loginUsername" placeholder="Username">
         <input type="password" id="loginPassword" placeholder="Password">
      <a id="loginLink" href="post.html">
         <button id="loginBtn">Login</button>
      </a>
      </div>

   </div>

   <script>
```

```
let token = '';
document.getElementById('registerBtn').addEventListener('click', async () => {
  const username = document.getElementById('regUsername').value;
  const password = document.getElementById('regPassword').value;
  const response = await fetch('/register', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ username, password })
  });
  const result = await response.json();
  alert(result.message);
});

document.getElementById('loginBtn').addEventListener('click', async () => {
  const username = document.getElementById('loginUsername').value;
  const password = document.getElementById('loginPassword').value;
  const response = await fetch('/login', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ username, password })
  });
  const result = await response.json();
  token = result.token;
  document.getElementById('auth').style.display = 'none';
  document.getElementById('app').style.display = 'block';
  fetchFeed();
});

document.getElementById('postBtn').addEventListener('click', async () => {
  const content = document.getElementById('postContent').value;
  await fetch('/post', {
```

```
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ content, token })
        });
        document.getElementById('postContent').value = '';
        fetchFeed();
    });

    async function fetchFeed() {
        const response = await fetch('/feed', {
            headers: { Authorization: `Bearer ${token}` }
        });
        const posts = await response.json();
        const feedDiv = document.getElementById('feed');
        feedDiv.innerHTML = '';
        posts.forEach(post => {
            const postDiv = document.createElement('div');
            postDiv.textContent = post.content;
            feedDiv.appendChild(postDiv);
        });
    }
  </script>
</body>
</html>
```

Post.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Document</title>
    <link rel="stylesheet" href="style1.css">
</head>
<body>
    <h1>Microblogging Application</h1>
        <h2>Create a Post</h2>
        <input type="text" id="postContent" placeholder="What's on your mind?">
        <button id="postBtn" onclick="alert('Post Successfull')">Post</button>
        <h2>Your Feed</h2>
        <p></p>
        <div id="feed"></div>

</body>
</html>
```

Express.js

```
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const app = express();
app.use(express.json());
app.use(express.static('public'));

const jwtSecret = "supersecretkey";

// Connect to MongoDB
mongoose.connect('mongodb://localhost/microblog', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('Connected to MongoDB...'))
```

```
  .catch(err => console.error('Could not connect to MongoDB...', err));

// User and Post Schemas
const userSchema = new mongoose.Schema({
    username: { type: String, unique: true },
    password: String,
    followers: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
    following: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }]
});
const User = mongoose.model('User', userSchema);

const postSchema = new mongoose.Schema({
    content: String,
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    date: { type: Date, default: Date.now }
});
const Post = mongoose.model('Post', postSchema);

// Helper to verify token and get user ID
const verifyToken = (token) => jwt.verify(token, jwtSecret).userId;

// Register user
app.post('/register', async (req, res) => {
    const { username, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);
    await new User({ username, password: hashedPassword }).save();
    res.json({ message: 'User registered successfully!' });
});

// Login user
app.post('/login', async (req, res) => {
```

```
    const { username, password } = req.body;
    const user = await User.findOne({ username });
    if (user && await bcrypt.compare(password, user.password)) {
        const token = jwt.sign({ userId: user._id }, jwtSecret);
        return res.json({ token, userId: user._id });
    }
    res.status(400).json({ message: 'Invalid username or password' });
});

// Post content
app.post('/post', async (req, res) => {
    const { content, token } = req.body;
    await new Post({ content, user: verifyToken(token) }).save();
    res.json({ message: 'Post created successfully!' });
});

// Follow a user
app.post('/follow', async (req, res) => {
    const { token, followUserId } = req.body;
    const userId = verifyToken(token);
    const user = await User.findById(userId);
    const followUser = await User.findById(followUserId);

    if (!user.following.includes(followUserId)) {
        user.following.push(followUserId);
        followUser.followers.push(userId);
        await user.save();
        await followUser.save();
    }
    res.json({ message: `You are now following ${followUser.username}` });
});
```

```
// Fetch posts from people the user follows
app.get('/feed', async (req, res) => {
   const userId = verifyToken(req.headers.authorization.split(' ')[1]);
   const user = await User.findById(userId).populate('following');
   const posts = await Post.find({ user: { $in: user.following } })
                        .populate('user')
                        .sort({ date: -1 });
   res.json(posts);
});

app.listen(3000, () => console.log('Server started on port 3000'));
```