

```

frontend > <> index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Login - JWT Demo</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
8  </head>
9  <body class="bg-light">
10
11  <div class="container mt-5">
12      <div class="row justify-content-center">
13          <div class="col-md-5 bg-white p-4 rounded shadow">
14              <h2 class="text-center mb-4">Login</h2>
15
16              <div id="login-msg" class="text-danger mb-2"></div>
17
18              <div class="mb-3">
19                  <label>Email</label>
20                  <input type="email" id="login-email" class="form-control" placeholder="Enter email">
21              </div>
22              <div class="mb-3">
23                  <label>Password</label>
24                  <input type="password" id="login-password" class="form-control" placeholder="Enter password">
25              </div>
26              <button onclick="login()" class="btn btn-primary w-100">Login</button>
27
28              <p class="mt-3 text-center">
29                  Don't have an account? <a href="signup.html">Signup here</a>
30              </p>
31          </div>
32      </div>
33  </div>
34
35  <script src="script.js"></script>
36 </body>
37 </html>

```

```

frontend > <> dashboard.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Dashboard - JWT Demo</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
8  </head>
9  <body class="bg-light">
10
11  <div class="container mt-5">
12      <div class="row justify-content-center">
13          <div class="col-md-6 bg-white p-4 rounded shadow">
14              <h2 class="text-center mb-4">Dashboard</h2>
15
16              <div id="dashboard-msg" class="text-success mb-2"></div>
17
18              <button onclick="getUserProfile()" class="btn btn-primary mb-2 w-100">User Profile</button>
19              <button onclick="getAdminOnly()" class="btn btn-warning mb-2 w-100">Admin Only</button>
20              <button onclick="getAllUsers()" class="btn btn-info mb-2 w-100">Show All Users (Admin)</button>
21              <button onclick="logout()" class="btn btn-danger w-100">Logout</button>
22
23              <pre id="output" class="mt-3 bg-light p-3 rounded"></pre>
24          </div>
25      </div>
26  </div>
27
28  <script src="script.js"></script>
29 </body>
30 </html>

```

frontend > <> signup.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Signup - JWT Demo</title>
7    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
8  </head>
9  <body class="bg-light">
10
11  <div class="container mt-5">
12    <div class="row justify-content-center">
13      <div class="col-md-5 bg-white p-4 rounded shadow">
14        <h2 class="text-center mb-4">Signup</h2>
15
16        <div id="signup-msg" class="text-danger mb-2"></div>
17
18        <div class="mb-3">
19          <label>Username</label>
20          <input type="text" id="signup-username" class="form-control" placeholder="Enter username">
21        </div>
22        <div class="mb-3">
23          <label>Email</label>
24          <input type="email" id="signup-email" class="form-control" placeholder="Enter email">
25        </div>
26        <div class="mb-3">
27          <label>Password</label>
28          <input type="password" id="signup-password" class="form-control" placeholder="Enter password">
29        </div>
30        <div class="mb-3">
31          <label>Role</label>
32          <select id="signup-role" class="form-control">
33            <option value="user">User</option>
34            <option value="admin">Admin</option>
35          </select>
36        </div>
37
38        <button onclick="signup()" class="btn btn-success w-100">Signup</button>
39
40        <p class="mt-3 text-center">
41          Already have an account? <a href="index.html">Login here</a>
42        </p>
43      </div>
44    </div>
45  </div>
46
47  <script src="script.js"></script>
48 </body>
49 </html>
50
```

```

frontend > JS scriptjs > ...
1 // Get token from localStorage if available
2 let token = localStorage.getItem('token') || '';
3
4 // Backend URL
5 const BASE_URL = 'http://localhost:5000/api/auth';
6
7 // ===== Signup =====
8 async function signup() {
9     const username = document.getElementById('signup-username').value.trim();
10    const email = document.getElementById('signup-email').value.trim();
11    const password = document.getElementById('signup-password').value;
12    const role = document.getElementById('signup-role').value;
13
14    if (!username || !email || !password) {
15        document.getElementById('signup-msg').innerText = 'All fields are required!';
16        return;
17    }
18    if (password.length < 6) {
19        document.getElementById('signup-msg').innerText = 'Password must be at least 6 characters';
20        return;
21    }
22
23    try {
24        const res = await fetch(`${BASE_URL}/signup`, {
25            method: 'POST',
26            headers: { 'Content-Type': 'application/json' },
27            body: JSON.stringify({ username, email, password, role })
28        });
29        const data = await res.json();
30
31        if (data._id) {
32            alert('Signup successful! Please login.');
```

frontend > JS script.js > ...

```
75  async function getUserProfile() {
82  try {
83    const res = await fetch(`${BASE_URL}/user-profile`, {
84      headers: { Authorization: `Bearer ${token}` }
85    });
86    const data = await res.json();
87
88    if (!res.ok) {
89      alert(data.message || 'Unauthorized! Please login again.');
```

```
90      if (res.status === 401) logout();
91    } else {
92      document.getElementById('output').innerText = JSON.stringify(data, null, 2);
93    }
94  } catch (err) {
95    document.getElementById('output').innerText = 'Error fetching data.';
96    console.error(err);
97  }
98  }
99
100 async function getAdminOnly() {
101   if (!token) {
102     alert('Please login first!');
103     window.location.href = 'index.html';
104     return;
105   }
106
107   try {
108     const res = await fetch(`${BASE_URL}/admin-only`, {
109       headers: { Authorization: `Bearer ${token}` }
110     });
111     const data = await res.json();
112
113     if (!res.ok) {
114       alert(data.message || 'You are not authorized!');
```

```
115     } else {
116       document.getElementById('output').innerText = JSON.stringify(data, null, 2);
117     }
118   } catch (err) {
119     document.getElementById('output').innerText = 'Error fetching data.';
120     console.error(err);
121   }
122 }
123
124 // ===== Admin: Get All Users =====
125 async function getAllUsers() {
126   if (!token) {
127     alert('Please login first!');
128     window.location.href = 'index.html';
129     return;
130   }
131
132   try {
133     const res = await fetch(`${BASE_URL}/users`, {
134       headers: { Authorization: `Bearer ${token}` }
135     });
136     const data = await res.json();
137
138     if (!res.ok) {
139       alert(data.message || 'You are not authorized!');
```

```
140     } else {
141       // FIX: Backend returns array directly, not { users: [...] }
142       document.getElementById('output').innerText = JSON.stringify(data, null, 2);
143     }
144   } catch (err) {
145     document.getElementById('output').innerText = 'Error fetching users.';
146     console.error(err);
147   }
148 }
149
150 // ===== Logout =====
151 function logout() {
152   token = '';
153   localStorage.removeItem('token');
154   window.location.href = 'index.html';
155 }
156
157 // ===== Auto redirect to login if token missing =====
158 if (window.location.pathname.includes('dashboard.html') && !token) {
159   alert('Please login first!');
160   window.location.href = 'index.html';
}
```


backend > src > controllers > JS authController.js > ...

```
1  const User = require('../models/User');
2  const jwt = require('jsonwebtoken');
3
4  // Function to generate JWT
5  const generateToken = (user) => {
6    return jwt.sign(
7      { id: user._id, role: user.role },
8      process.env.JWT_SECRET,
9      { expiresIn: process.env.ACCESS_TOKEN_EXPIRY }
10   );
11 };
12
13 // Signup
14 exports.signup = async (req, res) => {
15   const { username, email, password, role } = req.body;
16   try {
17     const existingUser = await User.findOne({ email });
18     if (existingUser) return res.status(400).json({ message: 'User already exists' });
19
20     const user = await User.create({ username, email, password, role });
21     const token = generateToken(user);
22
23     res.status(201).json({
24       message: 'User created successfully',
25       token,
26       user: { id: user._id, username: user.username, email: user.email, role: user.role }
27     });
28   } catch (err) {
29     res.status(500).json({ message: 'Server error', error: err.message });
30   }
31 };
32
33 // Login
34 exports.login = async (req, res) => {
35   const { email, password } = req.body;
36   try {
37     const user = await User.findOne({ email });
38     if (!user) return res.status(400).json({ message: 'Invalid credentials' });
39
40     const isMatch = await user.matchPassword(password);
41     if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });
42
43     const token = generateToken(user);
44     res.status(200).json({
45       message: 'Login successful',
46       token,
47       user: { id: user._id, username: user.username, email: user.email, role: user.role }
48     });
49   } catch (err) {
50     res.status(500).json({ message: 'Server error', error: err.message });
51   }
52 };
53
```

backend > src > middlewares > JS authMiddleware.js > ...

```
1  const jwt = require('jsonwebtoken');
2  const User = require('../models/User');
3
4  // Middleware to verify JWT token
5  exports.protect = async (req, res, next) => {
6    let token;
7
8    // Check if token is in headers
9    if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
10     token = req.headers.authorization.split(' ')[1];
11   }
12
13   if (!token) {
14     return res.status(401).json({ message: 'Not authorized, token missing' });
15   }
16
17   try {
18     // Verify token
19     const decoded = jwt.verify(token, process.env.JWT_SECRET);
20
21     // Attach user to request (without password)
22     req.user = await User.findById(decoded.id).select('-password');
23     next();
24   } catch (err) {
25     return res.status(401).json({ message: 'Not authorized, token invalid' });
26   }
27 };
28
29 // Middleware to check user role
30 exports.authorize = (...roles) => {
31   return (req, res, next) => {
32     if (!roles.includes(req.user.role)) {
33       return res.status(403).json({ message: `Role ${req.user.role} is not allowed to access this route` });
34     }
35     next();
36   };
37 };
38
```

backend > src > config > JS db.js > ...

```
1  const mongoose = require('mongoose');
2
3  const connectDB = async () => {
4    try {
5      await mongoose.connect(process.env.MONGODB_URI, {
6        useNewUrlParser: true,
7        useUnifiedTopology: true
8      });
9      console.log('MongoDB connected');
10     } catch (err) {
11       console.error('MongoDB connection failed:', err);
12       process.exit(1);
13     }
14   };
15
16   module.exports = connectDB;
17
```

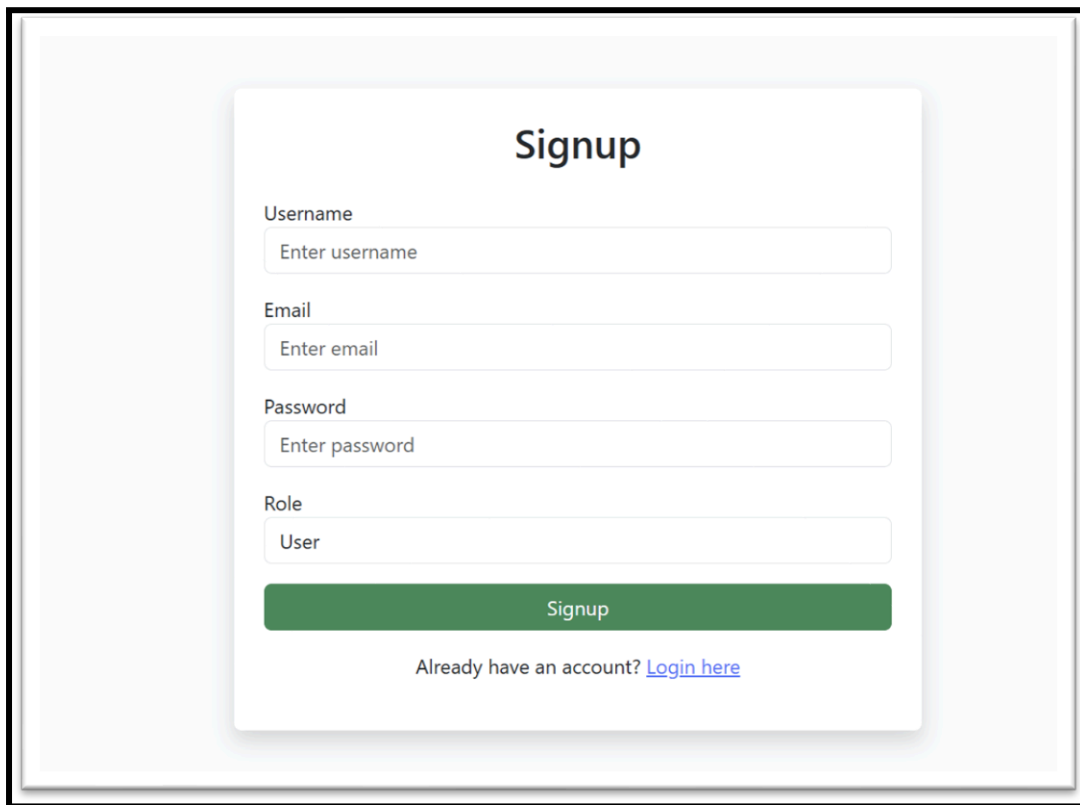
backend > src > routes > JS authRoutes.js > ...

```
1  const express = require('express');
2  const router = express.Router();
3  const { signup, login } = require('../controllers/authController');
4  const { protect, authorize } = require('../middlewares/authMiddleware');
5  const User = require('../models/User'); // Import User model
6
7  // Signup route
8  router.post('/signup', signup);
9
10 // Login route
11 router.post('/login', login);
12
13 // Protected route: accessible by any logged-in user
14 router.get('/user-profile', protect, (req, res) => {
15   res.json({ user: req.user });
16 });
17
18 // Protected route: accessible only by admin
19 router.get('/admin-only', protect, authorize('admin'), (req, res) => {
20   res.json({ message: 'Welcome admin!', user: req.user });
21 });
22
23 // Admin can see all registered users
24 router.get('/users', protect, authorize('admin'), async (req, res) => {
25   try {
26     const users = await User.find({}, '-password'); // Exclude password field
27     res.json(users);
28   } catch (err) {
29     res.status(500).json({ message: 'Error fetching users' });
30   }
31 });
32
33 module.exports = router;
```

```
backend > src > JS index.js > ...
1  require('dotenv').config();
2  const express = require('express');
3  const cors = require('cors');
4  const connectDB = require('./config/db'); // Import MongoDB connection
5
6  const app = express();
7
8  // Middleware
9  app.use(cors());
10 app.use(express.json());
11
12 // Routes
13 const authRoutes = require('./routes/authRoutes');
14 app.use('/api/auth', authRoutes);
15
16 // Test route
17 app.get('/', (req, res) => {
18   res.send('JWT Blog API running');
19 });
20
21 // Connect to MongoDB and start server
22 const PORT = process.env.PORT || 5000;
23
24 connectDB().then(() => {
25   app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
26 }).catch((err) => {
27   console.error('Failed to connect to MongoDB:', err);
28   process.exit(1);
29 });
```

```
backend > .env
1  PORT=5000
2  MONGODB_URI=mongodb://127.0.0.1:27017/jwt_blog
3  JWT_SECRET=fullstackdevelopment@6
4  ACCESS_TOKEN_EXPIRY=15m
5  REFRESH_TOKEN_EXPIRY=7d
6  SALT_ROUNDS=10
7
```


OUTPUT:



The image shows a 'Signup' form within a light gray container. The form itself is white with a subtle drop shadow. It features four input fields: 'Username' with placeholder 'Enter username', 'Email' with placeholder 'Enter email', 'Password' with placeholder 'Enter password', and 'Role' with a dropdown menu currently showing 'User'. Below these fields is a green 'Signup' button. At the bottom of the form, there is a link: 'Already have an account? [Login here](#)'.

Signup

Username
Enter username

Email
Enter email

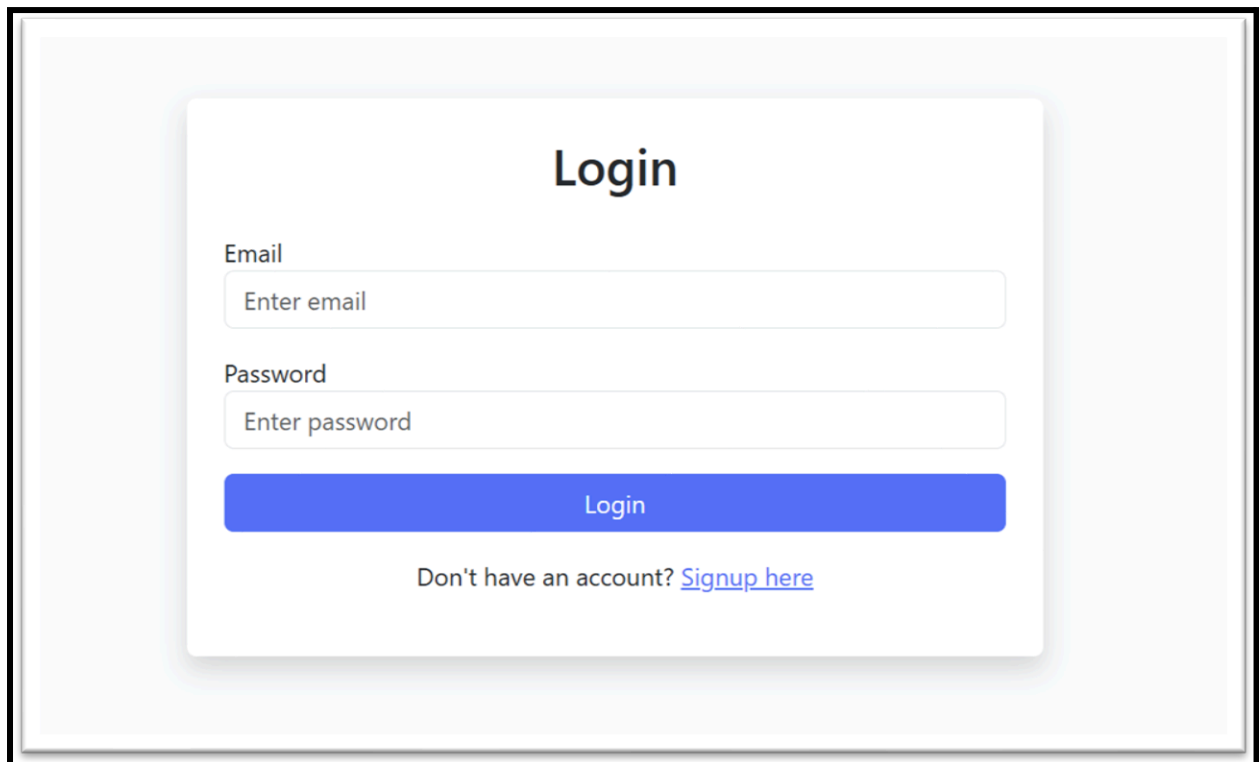
Password
Enter password

Role
User

Signup

Already have an account? [Login here](#)

Fig 1



The image shows a 'Login' form within a light gray container. The form is white with a subtle drop shadow. It has two input fields: 'Email' with placeholder 'Enter email' and 'Password' with placeholder 'Enter password'. Below these is a blue 'Login' button. At the bottom, there is a link: 'Don't have an account? [Signup here](#)'.

Login

Email
Enter email

Password
Enter password

Login

Don't have an account? [Signup here](#)

Fig 2

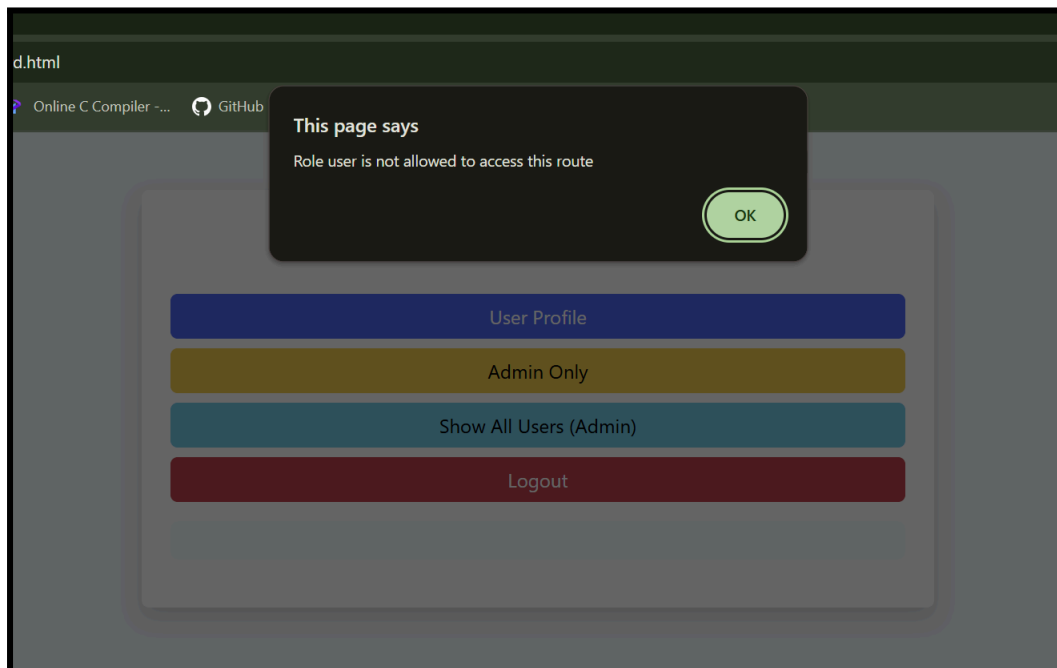
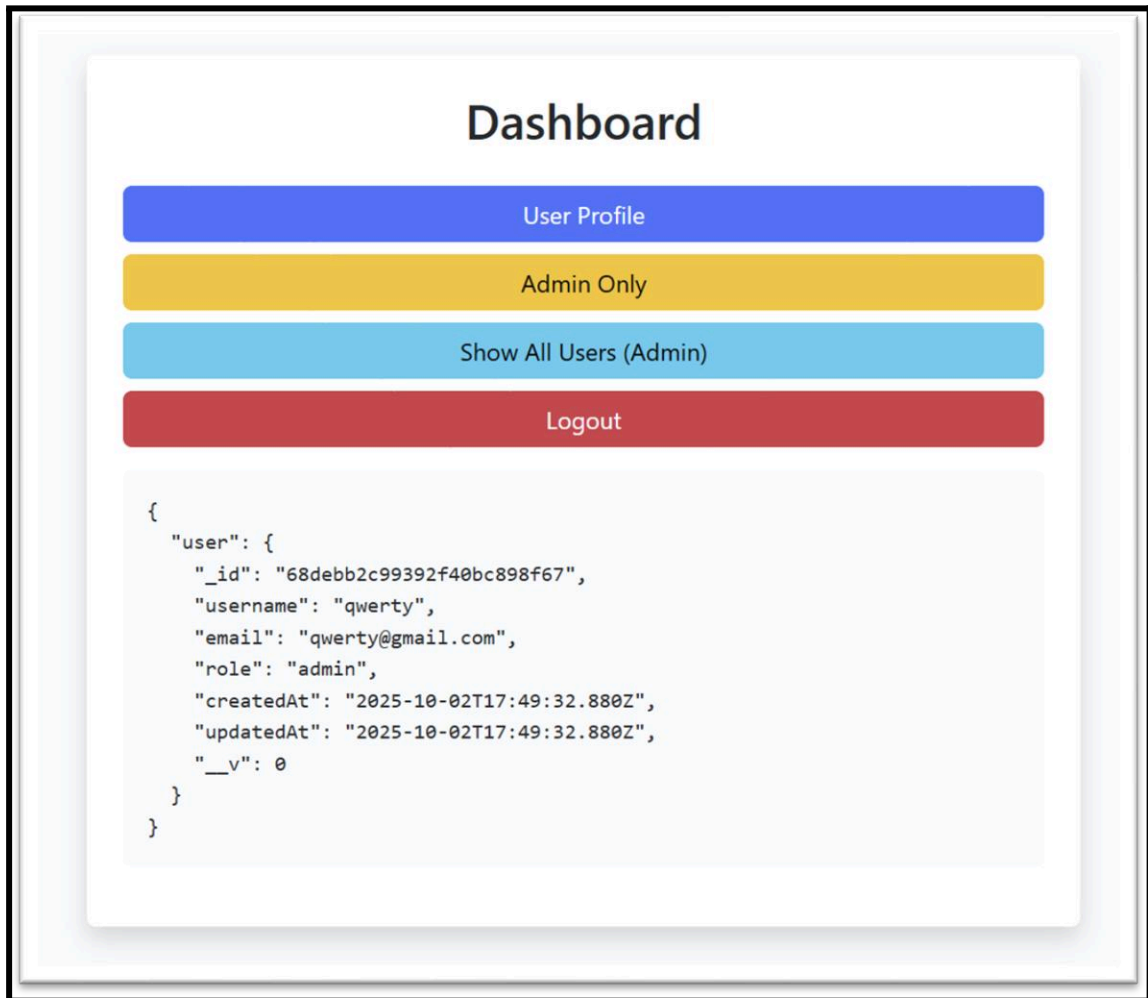


Fig 3

Fig 4

Dashboard

User Profile

Admin Only

Show All Users (Admin)

Logout

```
[
  {
    "_id": "68de7baa3c5cb28f3d2a7c4f",
    "username": "Sameer",
    "email": "jagiasisameer@gmail.com",
    "role": "admin",
    "createdAt": "2025-10-02T13:18:34.420Z",
    "updatedAt": "2025-10-02T13:18:34.420Z",
    "__v": 0
  },
  {
    "_id": "68de9a523e0345977e193462",
    "username": "atharva",
    "email": "d2024.abc@gmail.com",
    "role": "user",
    "createdAt": "2025-10-02T15:29:22.819Z",
    "updatedAt": "2025-10-02T15:29:22.819Z",
    "__v": 0
  },
  {
    "_id": "68dea9013e0345977e193471",
    "username": "serty",
    "email": "abcd@gmailcom",
    "role": "user",
    "createdAt": "2025-10-02T16:32:01.630Z",
    "updatedAt": "2025-10-02T16:32:01.630Z",
    "__v": 0
  },
  {
    "_id": "68debb2c99392f40bc898f67",
    "username": "qwerty",
    "email": "qwerty@gmail.com",
```

Fig 5