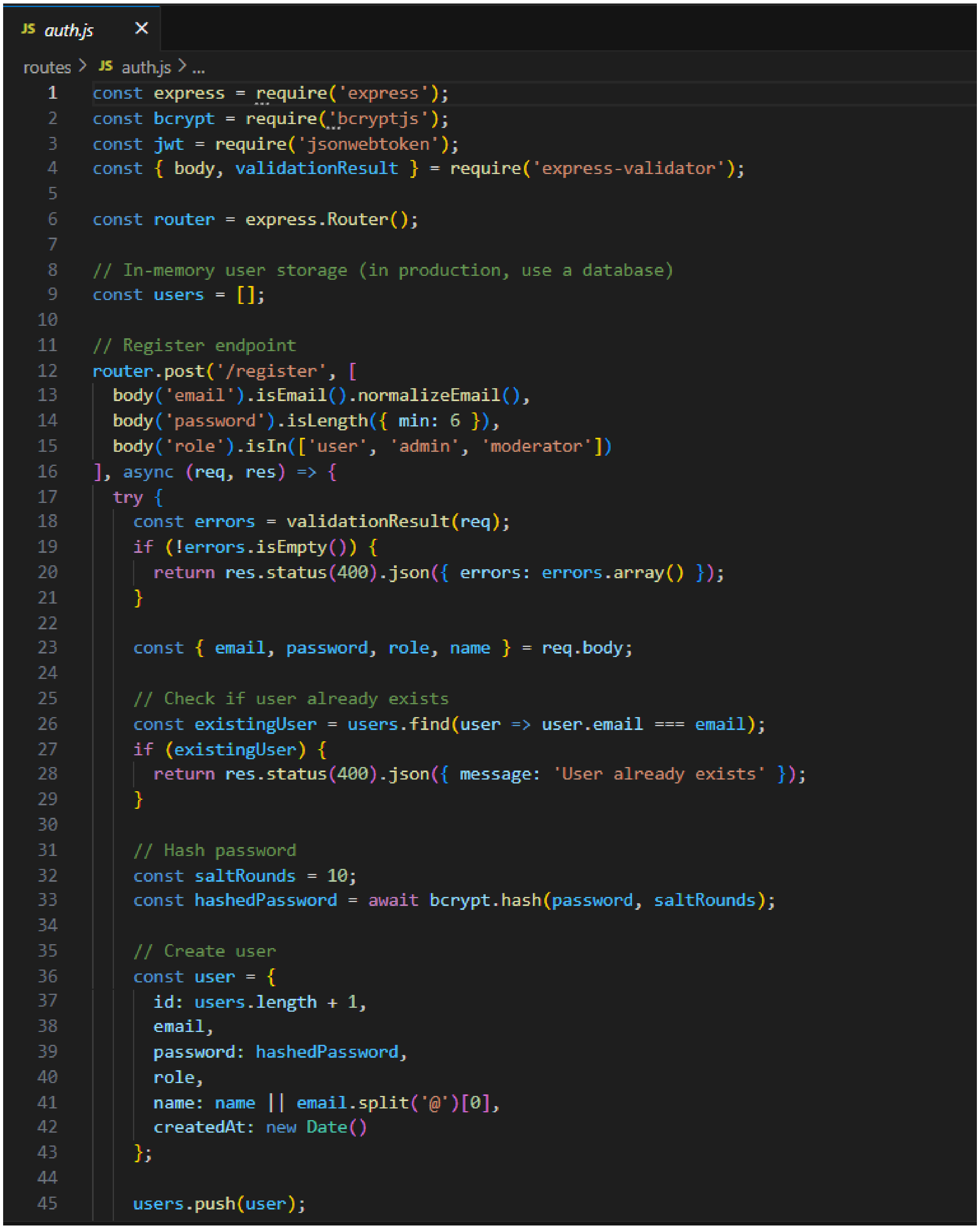
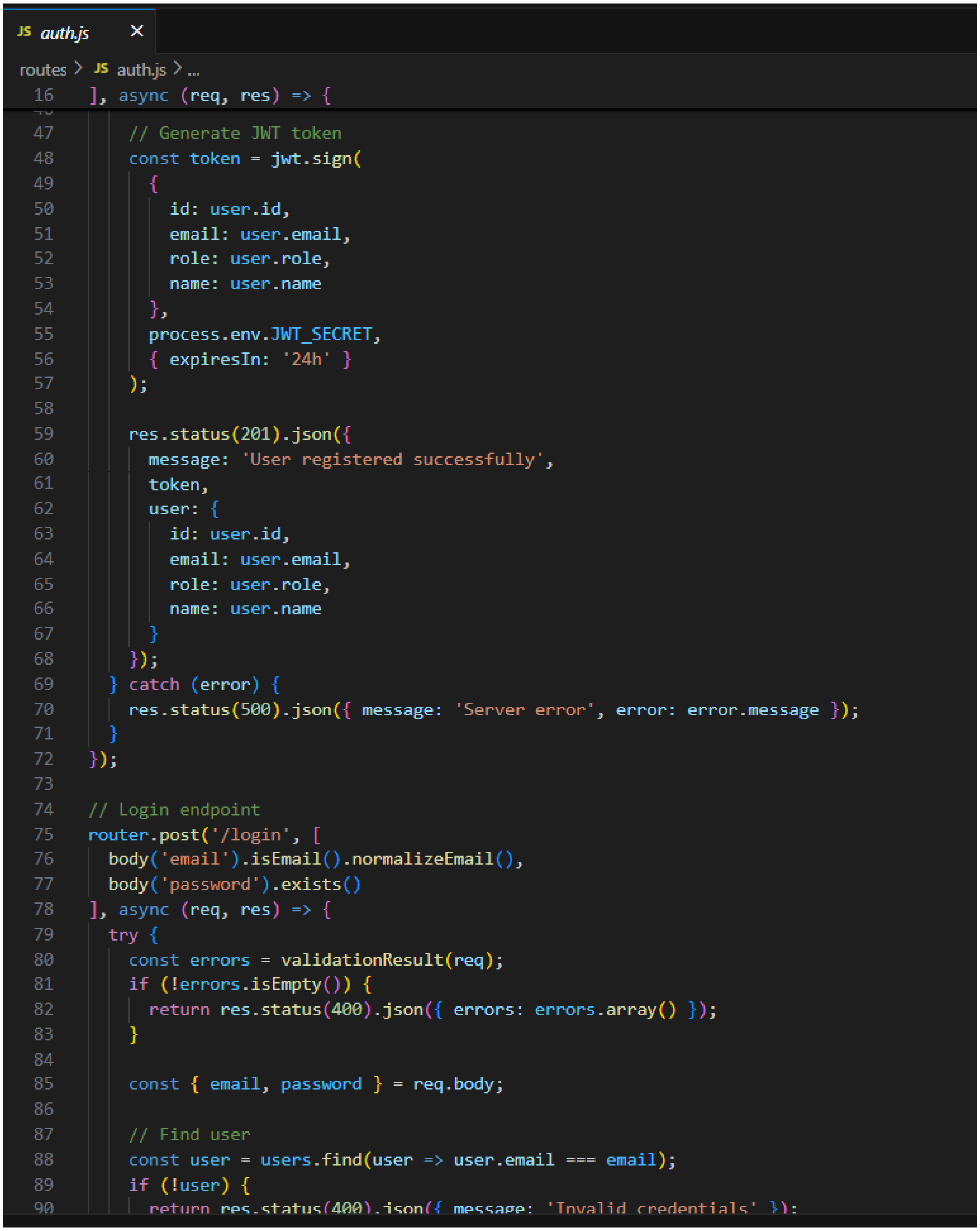


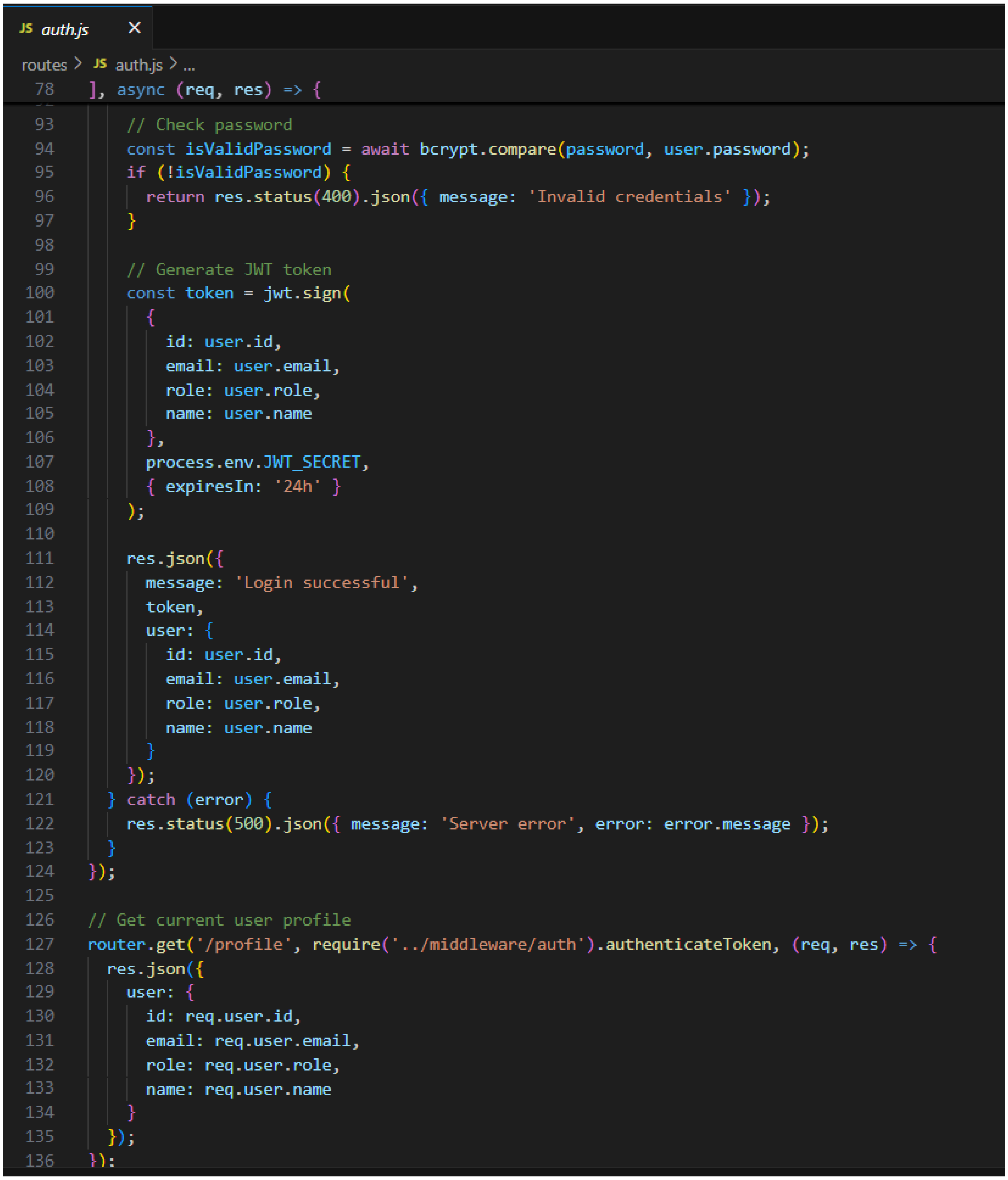
```
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { body, validationResult } = require('express-validator');

const router = express.Router();

// In-memory user storage (in production, use a database)
const users = [];

// Register endpoint
router.post('/register', [
  body('email').isEmail().normalizeEmail(),
  body('password').isLength({ min: 6 }),
  body('role').isIn(['user', 'admin', 'moderator'])
], async (req, res) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { email, password, role, name } = req.body;

    // Check if user already exists
    const existingUser = users.find(user => user.email === email);
    if (existingUser) {
      return res.status(400).json({ message: 'User already exists' });
    }

    // Hash password
    const saltRounds = 10;
    const hashedPassword = await bcrypt.hash(password, saltRounds);

    // Create user
    const user = {
      id: users.length + 1,
      email,
      password: hashedPassword,
      role,
      name: name || email.split('@')[0],
      createdAt: new Date()
    };

    users.push(user);
```



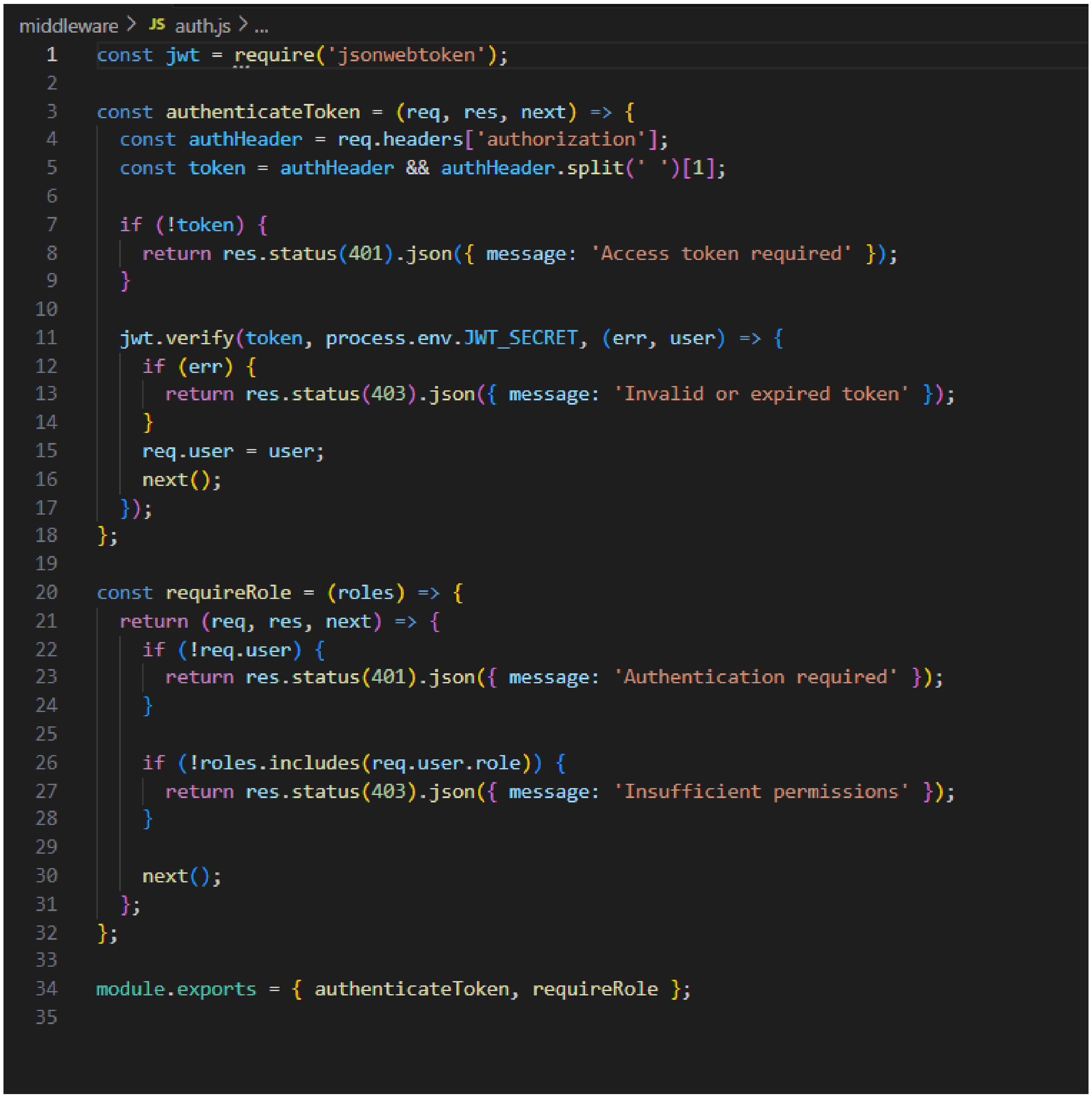
auth.js ×

routes > auth.js > ...

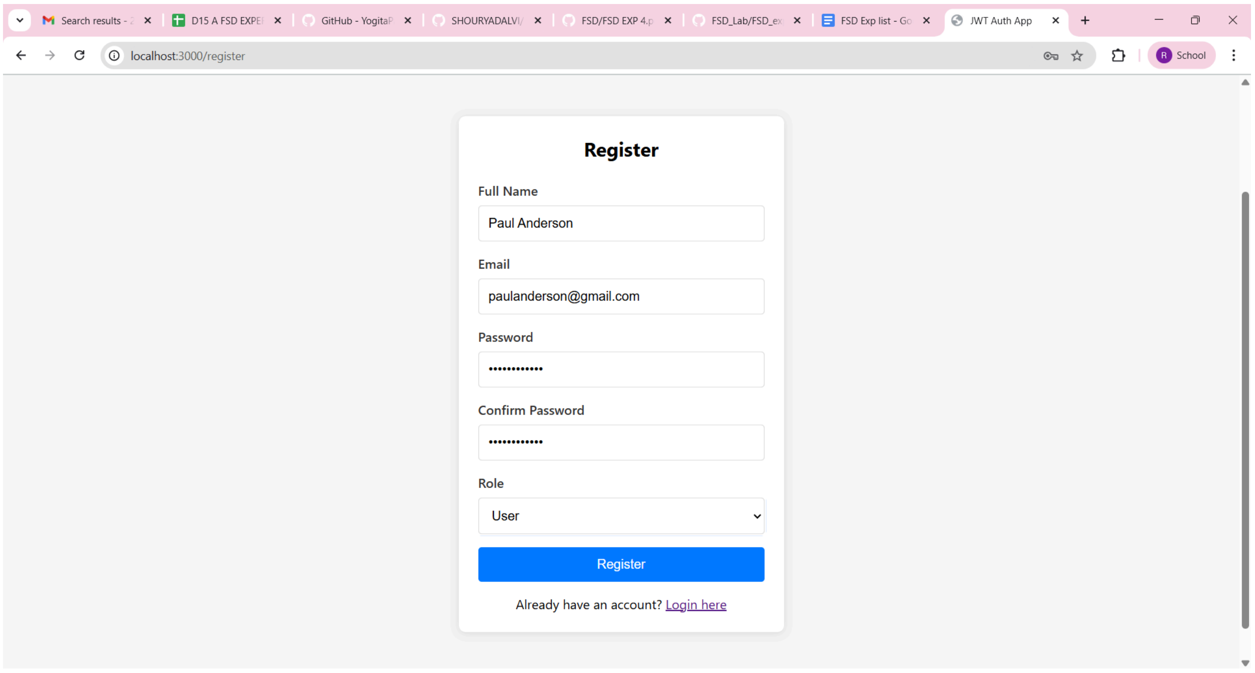
```
], async (req, res) => {
    // Generate JWT token
    const token = jwt.sign(
      {
        id: user.id,
        email: user.email,
        role: user.role,
        name: user.name
      },
      process.env.JWT_SECRET,
      { expiresIn: '24h' }
    );

    res.status(201).json({
      message: 'User registered successfully',
      token,
      user: {
        id: user.id,
        email: user.email,
        role: user.role,
        name: user.name
      }
    });
  } catch (error) {
    res.status(500).json({ message: 'Server error', error: error.message });
  }
});

// Login endpoint
router.post('/login', [
  body('email').isEmail().normalizeEmail(),
  body('password').exists()
], async (req, res) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { email, password } = req.body;

    // Find user
    const user = users.find(user => user.email === email);
    if (!user) {
      return res.status(400).json({ message: 'Invalid credentials' });
```



```javascript
], async (req, res) => {
    // Check password
    const isValidPassword = await bcrypt.compare(password, user.password);
    if (!isValidPassword) {
      return res.status(400).json({ message: 'Invalid credentials' });
    }

    // Generate JWT token
    const token = jwt.sign(
      {
        id: user.id,
        email: user.email,
        role: user.role,
        name: user.name
      },
      process.env.JWT_SECRET,
      { expiresIn: '24h' }
    );

    res.json({
      message: 'Login successful',
      token,
      user: {
        id: user.id,
        email: user.email,
        role: user.role,
        name: user.name
      }
    });
  } catch (error) {
    res.status(500).json({ message: 'Server error', error: error.message });
  }
});

// Get current user profile
router.get('/profile', require('../middleware/auth').authenticateToken, (req, res) => {
  res.json({
    user: {
      id: req.user.id,
      email: req.user.email,
      role: req.user.role,
      name: req.user.name
    }
  });
});
```


middleware > JS auth.js > ...

```javascript
const jwt = require('jsonwebtoken');

const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'Access token required' });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) {
      return res.status(403).json({ message: 'Invalid or expired token' });
    }
    req.user = user;
    next();
  });
};

const requireRole = (roles) => {
  return (req, res, next) => {
    if (!req.user) {
      return res.status(401).json({ message: 'Authentication required' });
    }

    if (!roles.includes(req.user.role)) {
      return res.status(403).json({ message: 'Insufficient permissions' });
    }

    next();
  };
};

module.exports = { authenticateToken, requireRole };
```



localhost:3000/register

Register

Full Name
Paul Anderson

Email
paulanderson@gmail.com

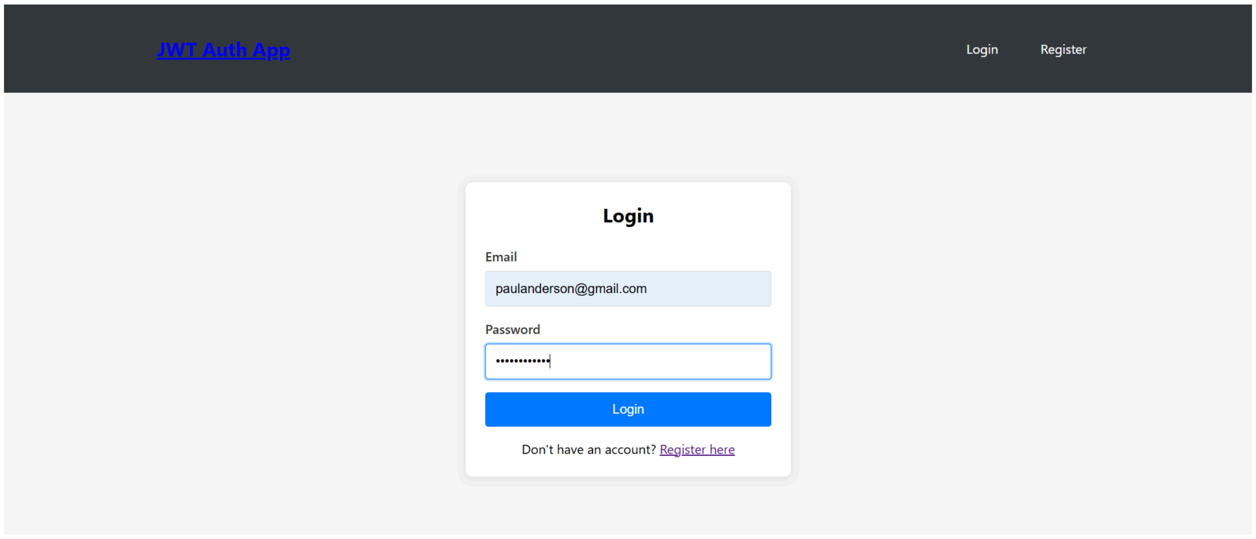
Password
••••••••••••

Confirm Password
••••••••••••

Role
User

Register

Already have an account? Login here



Login

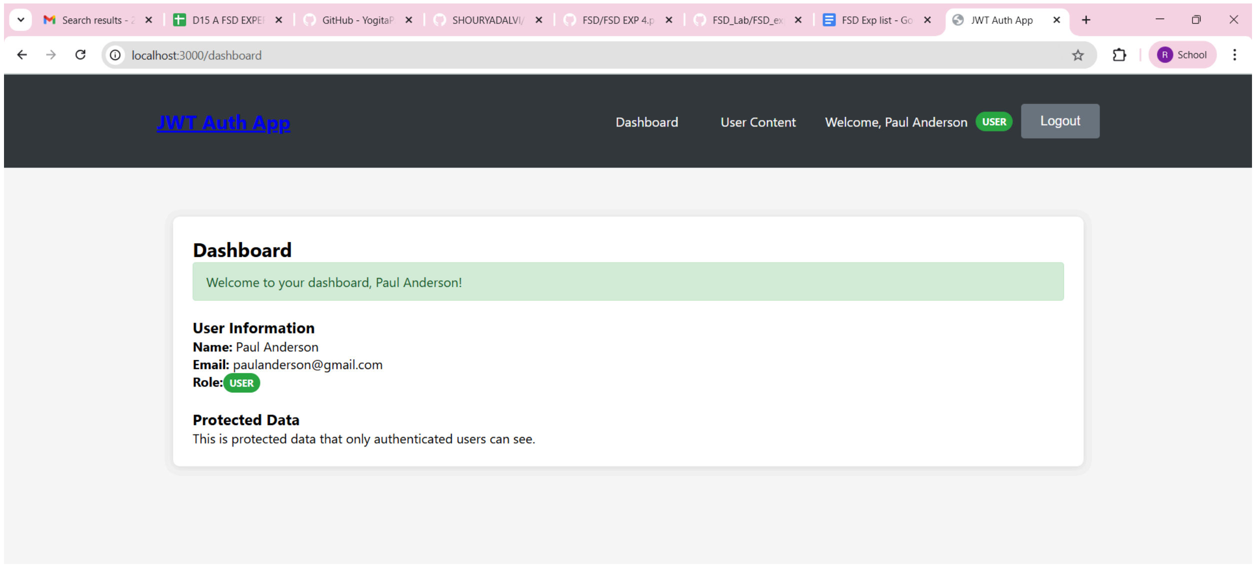
Email

paulanderson@gmail.com

Password

••••••••••••

Don't have an account? Register here



Dashboard

Welcome to your dashboard, Paul Anderson!

User Information
Name: Paul Anderson
Email: paulanderson@gmail.com
Role: USER

Protected Data
This is protected data that only authenticated users can see.