```
// FILE: backend/routes/auth.js

const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { User } = require('../models/userModel');

const router = express.Router();
const JWT_SECRET = process.env.JWT_SECRET;

// Register
router.post('/register', async (req, res) => {
  const { name, email, phone, password } = req.body;

  try {
    const existing = await User.findOne({ where: { email } });
    if (existing) {
      return res.status(400).json({ message: 'Email already exists' });
    }

    const password_hash = await bcrypt.hash(password, 10);

    const user = await User.create({
      name,
      email,
      phone,
      password_hash,
      role: 'investor'
    });

    const token = jwt.sign(
      { id: user.id, role: user.role },
      JWT_SECRET,
      { expiresIn: '7d' }
    );

    res.json({ token, role: user.role });

  } catch (error) {
    res.status(500).json({ message: 'Registration failed' });
  }
});

// Login
```

```
router.post('/login', async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return res.status(400).json({ message: 'User not found' });
    }

    const valid = await bcrypt.compare(password, user.password_hash);
    if (!valid) {
      return res.status(400).json({ message: 'Invalid password' });
    }

    const token = jwt.sign(
      { id: user.id, role: user.role },
      JWT_SECRET,
      { expiresIn: '7d' }
    );

    res.json({ token, role: user.role });

  } catch (error) {
    res.status(500).json({ message: 'Login failed' });
  }
});

// Middleware
const authMiddleware = (roles = []) => {
  return (req, res, next) => {
    const authHeader = req.headers.authorization;
    if (!authHeader) {
      return res.status(401).json({ message: 'No token provided' });
    }

    const token = authHeader.split(' ')[1];
    try {
      const decoded = jwt.verify(token, JWT_SECRET);

      if (roles.length && !roles.includes(decoded.role)) {
        return res.status(403).json({ message: 'Access denied' });
      }

      req.user = decoded;
```

```
      next();

    } catch (error) {
      return res.status(401).json({ message: 'Invalid token' });
    }
  };
};

module.exports = { router, authMiddleware };
```