const express = require('express');

const app = express();

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

// Authentication middleware

const authenticate = async (req, res, next) => {

  try {

    const token = req.header('Authorization').replace('Bearer ', '');

    const decoded = jwt.verify(token, process.env.JWT\_SECRET);

    req.user = decoded;

    next();

  } catch (error) {

    res.status(401).json({ message: 'Please authenticate' });

  }

};

// Password hashing

const hashPassword = async (password) => {

  const salt = await bcrypt.genSalt(10);

  return await bcrypt.hash(password, salt);

};

// Compare password

const comparePassword = async (password, hashedPassword) => {

  return await bcrypt.compare(password, hashedPassword);

};

// Secure password storage

[app.post](http://app.post/)('/register', async (req, res) => {

  try {

    const hashedPassword = await hashPassword(req.body.password);

    // Store the hashed password in the database

  } catch (error) {

    res.status(500).json({ message: 'Failed to register user' });

  }

});

// Authentication

[app.post](http://app.post/)('/login', async (req, res) => {

  try {

    const user = await User.findOne({ email: req.body.email });

    if (!user) {

      return res.status(401).json({ message: 'Invalid credentials' });

    }

    const isValidPassword = await comparePassword(req.body.password, user.password);

    if (!isValidPassword) {

      return res.status(401).json({ message: 'Invalid credentials' });

    }

    const token = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1h' });

    res.json({ token });

  } catch (error) {

    res.status(500).json({ message: 'Failed to authenticate user' });

  }

});