authMiddleware

const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {

  try {

    // Log the Authorization header (only in development mode)

    if (process.env.NODE\_ENV === "development") {

      console.log("Authorization Header:", req.headers.authorization);

    }

    // Ensure the Authorization header exists and follows "Bearer <token>" format

    if (!req.headers.authorization || !req.headers.authorization.startsWith("Bearer ")) {

      return res.status(401).json({ message: "Authorization token is missing or malformed" });

    }

    // Extract the token from the "Bearer <token>" format

    const token = req.headers.authorization.split(" ")[1];

    console.log("Token received:", token); // Debugging: Log the token

    // Ensure JWT\_SECRET is set

    if (!process.env.JWT\_SECRET) {

      console.error("JWT\_SECRET is missing in environment variables.");

      return res.status(500).json({ message: "Server configuration error" });

    }

    // Verify token

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

    console.log("Token decoded:", decoded); // Debugging: Log the decoded token

    // Attach user data to request object

    req.user = decoded;

    next();

  } catch (error) {

    console.error("Token Verification Error:", error);

    if (error.name === "TokenExpiredError") {

      return res.status(401).json({ message: "Token has expired, please login again" });

    } else if (error.name === "JsonWebTokenError") {

      return res.status(403).json({ message: "Invalid token" });

    }

    return res.status(500).json({ message: "Internal server error" });

  }

};

module.exports = authMiddleware;