// @route   POST

// @desc    Login user

// @access  Public

router.post("/login-user", (req, res) => {

  const { email, password } = req.body;

  if (!email || !password) {

    return res.status(400).json({ error: "Please provide email and password" });

  }

  User.findOne({ email: email }).then((user) => {

    if (!user) {

      return res.status(401).json({ error: "User not found" });

    }

    bcrypt.compare(password, user.password).then((isMatch) => {

      if (!isMatch) {

        return res.status(401).json({ error: "Incorrect password" });

      }

      // Ajoutez ici le rôle de l'utilisateur à la réponse

      jwt.sign(

        { id: user.id, role: user.role }, // Incluez le rôle dans le payload du token si nécessaire

        config.get("jwtSecret"),

        { expiresIn: config.get("tokenExpire") },

        (err, token) => {

          if (err) {

            console.error(err);

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

          }

          // Retournez le token et le rôle dans la réponse

          return res.status(200).json({ token, role: user.role });

        }

      );

    });

  }).catch((err) => {

    console.error(err);

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

  });

});