```
import { Request } from "express";
import jsonwebtoken from "jsonwebtoken";
import passport from "passport";
import { Profile } from "passport";
import nodemailer from "nodemailer";
import { Strategy as GoogleStrategy, VerifyCallback} from "passport-google-oauth2";
import dotenv from "dotenv";
dotenv.config();

import User from "../models/userModel.js";
import { userType } from "../types/authTypes.js";

const emailConfiguration = {
  host: process.env.Brevo_smtpServer,
  port: Number(process.env.Brevo_port),  // TLS (use 465 for SSL)
  secure: false,                // Set to true for port 465
  auth: {
    user: process.env.Brevo_login,
    pass: process.env.Brevo_pwd
  }
}

const sendMail = (email: string | undefined, name: string | undefined)=>{

  const transporter = nodemailer.createTransport(emailConfiguration);

  const info = transporter.sendMail({
    to: email,
    from: 'alexandre.the.conquerer@gmail.com',
    subject: 'SignUp successful',
    html: `<h1>Welcome to Open Aura</h1><p> We welcome you ${name} to our
platform </p>`
  }, (err) => {
    if (err) {
      console.log("Error in sending mail", err);
    }
  });
}

const generateTokens = (user: userType) => {
  const accessToken = jsonwebtoken.sign(
    {
      id: user._id,
      name: user.name,
      email: user.email
    },
    process.env.JWT_Access_Secret!,
    { expiresIn: Number(process.env.JWT_Access_Token_Expiry) }

  );
```

```
    const refreshToken = jsonwebtoken.sign(
        {
            id: user._id
        },
        process.env.JWT_Refresh_Secret!,
        { expiresIn: Number(process.env.JWT_Access_Token_Expiry) }
    )
    return { accessToken, refreshToken };
}

passport.use(new GoogleStrategy({
    clientID:     process.env.Google_Client_Id!,
    clientSecret: process.env.Google_Client_Secret!,
    callbackURL: process.env.Google_Auth_Callback_Url!,
    passReqToCallback   : true,
  },
  async (req: Request, accessToken: string, refreshToken: string | undefined, profile: Profile,
done: VerifyCallback) =>{

    try{

        const user = await User.findOne({googleId: profile.id}) as userType;

        if(user){

            const { accessToken, refreshToken} = generateTokens(user);
            return done(null , { accessToken, refreshToken, user});
        }

        const localAuthUser = await User.findOne({email: profile.emails?.[0].value}) as
userType;

        if(localAuthUser){

            localAuthUser.googleId =  profile.id;
            localAuthUser.auth =  "google";

            await localAuthUser.save();

            const { accessToken, refreshToken} = generateTokens(localAuthUser);


            return done(null, { accessToken, refreshToken, localAuthUser});
        }
        else{

            const newUser = new User({

                googleId: profile.id,
                email: profile.emails?.[0].value,
                name: profile.displayName,
                auth: 'google'
            }) as userType;

            await newUser.save();

            const { accessToken, refreshToken} = generateTokens(newUser);

            sendMail(profile.emails?.[0].value, profile.displayName);

            return done(null, { newUser, accessToken: accessToken, refreshToken:
refreshToken});
        }

    }
    catch(err){
        return done(err);
    }

}));

export default passport;
```

Event Model:

```
import mongoose from "mongoose";
const Schema = mongoose.Schema;
const eventSchema = new Schema({
    eventHostId:{
        type: String,
        required: true
    },
    eventId:{
        type: String,
        required: true
    },
    eventHost:{
        type: String,
        required: true
    },
    title:{
        type: String,
        required: true
    },
    description:{
        type: String,
        required: true
    },
    thumbnail:{
        type: String,
        required: true
    },
    eventDateTime:{
        type: Date,
        required: true
    },
});

export default mongoose.model('Events', eventSchema);
```

User Model:

```
import mongoose from "mongoose";
const Schema = mongoose.Schema;
const user = new Schema({
    googleId:{
        type: String
    },
    auth:{
        type: String
    },
    name:{
        type: String,
        required: true
    },
    email:{
        type: String,
        required: true
    },
    password:{
        type: String,

    },
    refreshToken:{
        type: String
    }
});

export default mongoose.model('Users', user);
```