OAuth : Open Authorisation

To access Information of third Party Websites

Has

① Granular Access Level: Can request specific data from third party Apps

② Read / Read + Write Access: to post on third party App

③ Revoke Access : User can deauthorise from the third party App from where he logged in

Steps: 1) Set up the app
2) Redirect to authenticate
3) User logs in
4) User grants permission for his information to be used
5) Recieve Authorization Code / Token (inside this pieces of information)

Without OAuth we would have required the user to give us his facebook password instead of redirecting him.

* Authorization Code: The User Authentication is successful
Access Token : to access information

Google Auth

* npm install passport-google-oauth20

create Application on google Developers console

* On "oAuth consent Screen" set Credentials

Screen that user sees when they login through google and grant my application access to their data

* Scopes for google API : piece of data which we will recieve from google

To add Scope, enable API

* Create Credentials ⇒ oAuth Client ID

Name: name of Project

Authorized JS origins: (from where will the request to google come from)

http://localHost:3000

Authorized redirect URI's: (after google authorised to return to us to locally authenticate and save everything such as cookies

http://localhost:3000/auth/google/secrets

Now we get Client ID and Client secret

* add CLIENT_ID & CLIENT_SECRET to .env file

const GoogleStrategy = require('passport-google-oauth20').Strategy;

* npm install mongoose-findorcreate
const findOrCreate = require("mongoose-findorcreate");

userSchema.plugin(findOrCreate);

```
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());

passport.use(new GoogleStrategy({
    clientID: process.env.CLIENT_ID,
    clientSecret: process.env.CLIENT_SECRET,
    callbackURL: "http://localhost:3000/auth/google/secrets",
    userProfileURL: "https://www.googleapis.com/oauth2/v3/userinfo" //to avoid depecation Warning
  },                                  result that we get
  function(accessToken, refreshToken, profile, cb) {
    User.findOrCreate({ googleId: profile.id }, function (err, user) {
      return cb(err, user);
    });
  }
));

app.get("/", function(req, res){
  res.render("home");
});
```

So this way we will be able to find or create users in our database

* Problem we will not be able to save information in database after registration from user because in our userSchema there is no googleID: String

```
const userSchema = new mongoose.Schema ({
  email: String,
  password: String,
  googleId: String
});
```

* in register.ejs & in login.ejs

```
<div class="col-sm-4">
  <div class="card">
    <div class="card-body">
      <a class="btn btn-block" href="/auth/google" role="button">
        <i class="fab fa-google"></i>
        Sign Up with Google
      </a>
    </div>
  </div>
</div>
```

whenever user clicks on the button a get request will be triggered to /auth/google

```
app.get('/auth/google',
  passport.authenticate('google', { scope: ["profile"] })
);
```

// to start authentication

"google" → name of strategy which we already created

"profile" → what data we want to extract from google

* After Authentication where to redirect : Authorised redirect URI's

```
app.get("/auth/google/secrets",
  passport.authenticate('google', { failureRedirect: "/login" }),
  function(req, res) {
    // Successful authentication, redirect to secrets.
    res.redirect("/secrets");
  });
```

if failed will redirect

```
app.get("/login", function(req, res){
  res.render("login");
});
```

if successfull will redirect

```
app.get("/secrets", function(req, res){
  if (req.isAuthenticated()){
    res.render("secrets");
  } else {
    res.redirect("/login");
  }
});
```

```
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());
```

can serialize and deserialize only for local authentication

```
passport.serializeUser(function(user, done) {
  done(null, user.id);
});

passport.deserializeUser(function(id, done) {
  User.findById(id, function(err, user) {
    done(err, user);
  });
});
```

can serialize and deserialize for all kinds of authentication

* Social Button for BootStrap download and move from the folder bootstrap-social-CSS file to public< CSS > link it in header.ejs & add class "btn-social" & "btn-google" to button

```
const userSchema = new mongoose.Schema ({
  email: String,
  password: String,
  googleId: String,
  secret: String
});
```

Note : When we login passport saves our details and we can acess everything except Salt and Hash

```
app.get("/submit", function(req, res){
  if (req.isAuthenticated()){
    res.render("submit");
  } else {
    res.redirect("/login");
  }
});
```

// adding secrets to our database

```
app.post("/submit", function(req, res){
  const submittedSecret = req.body.secret;

  console.log(req.user.id);

  User.findById(req.user.id, function(err, foundUser){
    if (err) {
      console.log(err);
    } else {
      if (foundUser) {
        foundUser.secret = submittedSecret;
        foundUser.save(function(){
          res.redirect("/secrets");
        })
      }
    }
  })
});
```

```
app.get("/secrets", function(req, res){
  User.find({"secret": {$ne: null}}, function(err, foundUsers){
    if (err){
      console.log(err);
    } else {
      if (foundUsers) {
        res.render("secrets", {usersWithSecrets: foundUsers});
      }
    }
  });
});
```

```
<%- include('partials/header') %>

<div class="jumbotron text-center">
  <div class="container">
    <i class="fas fa-key fa-6x"></i>
    <h1 class="display-3">You've Discovered My Secret!</h1>

    <% usersWithSecrets.forEach(function(user){ %>
      <p class="secret-text"><%=user.secret%></p>
    <% }) %>

    <hr>
    <a class="btn btn-light btn-lg" href="/logout" role="button">Log Out</a>
    <a class="btn btn-dark btn-lg" href="/submit" role="button">Submit a Secret</a>
  </div>
</div>

<%- include('partials/footer') %>
```