Authentication Workshop

Aaron Sipser
Where are we at currently?
Let’s add some tools to our toolbelt
MyAuthApp will receive:
your public profile.

- Review the info you provide

Continue as Thierry

Cancel

🔒 This doesn’t let the app post to Facebook
How does Passport interact with Facebook?

GET - “/auth/facebook”

GET - “/auth/facebook/callback”
Let’s Start!
Step 1 A: Making a User Model

1. Clone the repository at go.6148.io/workshop4

2. Create a model for a user just like you did for story and comment.
   a. Remember that we need to store the user’s last post!

```javascript
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
    name: String,
    fbid: String,
    last_post: String,
});
module.exports = mongoose.model('User', UserModelSchema);
```
Now for a leap of faith
(more copy paste configurations)
Step 1B: Understanding Passport.JS

Generally what it does:

1. Configuration settings so that Facebook can communicate with your application.

   ```javascript
   clientID: 'FILL THIS OUT LATER!',
   clientSecret: 'FILL THIS OUT LATER!',
   callbackURL: '/auth/facebook/callback'
   ```

2. Look up the user by the “fbid” supplied by Facebook.

   ```javascript
   User.findOne({'fbid': profile.id }, function(err, user) {
   ...}
   ```

   a. If no user exists create them and save them to the database
   b. If they exist call the `done(err, user)` function, which logs you in!

3. Don’t be confused by what the `done` function does, Passport is magical!
Step 1C: Let’s create a facebook developer account

1. Go to [https://developers.facebook.com/apps/](https://developers.facebook.com/apps/)
2. Click **+ Add a New App**
3. Fill in information and create an App ID.

![Create a New App ID](image)

4. Choose ‘Facebook Login’ by hovering over it and clicking ‘Set up’
5. Ignore the Quickstart menu and click on ‘settings’ on the left navigation bar.
6. Type "http://localhost:3000" under redirect URIs and press the enter key. ‘Use Strict Mode for Redirect URIs’ should be set to ‘No’
7. Go to dashboard and copy down your App ID, and App Secret into passport.js. Put your App ID where it says “clientID” and your secret in “clientSecret”
Example passport.js

```javascript
// set up passport configs
passport.use(new fbp.Strategy({
  clientId: '719255378280457', Facebook App ID here
  clientSecret: '8ca7913aec0e56012adb5d8bf417546f', Facebook App Secret Here
  callbackURL: '/auth/facebook/callback'
})
```
Step 1D: Understanding app.js changes

1. Import our new passport.js file under local dependencies
2. Hook up passport with the magical express `app.use`
3. Set up facebook specific authentication routes that passport uses.
Step 1E : Adding a login button in navbar.js

1. Go to navbar.js
2. Add a ‘Login’ navbar item with `newNavbarItem(text, url)` which links you to ‘/auth/facebook’

Goal →

```
<a class="nav-item nav-link" href="/">Home</a>
<a class="nav-item nav-link" href="/u/profile?anoid">Profile</a>
<a class="nav-item nav-link" href="/auth/facebook">Login</a> == $0
```
Let’s see our changes.

npm start
go to localhost:3000
After clicking login, hopefully you see something like this pop up!

After clicking **Continue**, you should now have something a user saved in your MLab Database :)

**CoolTeamName**! will receive:
- your public profile.
- Review the info you provide

[Continue as Aaron]

[Lock] This doesn't let the app post to Facebook
Common Error

You have not fixed your redirect URI in Facebook Dev settings.

Detailed guide to fix here
Any questions?

If you got lost let’s catch back up

git reset --hard
git checkout step2

Note if you do this you have to re-enter your mongoDB URI and facebook AppID + Secret!

If you cannot get Facebook Dev working use

clientID: '1783717201933159',
clientSecret: '9ed4fe148172cd10d034bf8008d2fe23'
Step 2A: Making your server remember you! (app.js)

1. If you log in and refresh the page, you should stay logged in!
2. Use cookies through a new module called ‘express-sessions’
3. Go to app.js
4. Import express-session
5. Set up sessions configuration

```javascript
const session = require('express-session');

// set up sessions
app.use(session({
  secret: 'session-secret',
  resave: 'false',
  saveUninitialized: 'true'
}));

// hook up passport
app.use(passport.initialize());
app.use(passport.session());
```
Step 2B: What is req.user (more passport magic!)

1. Passport has a way of sending the client the User object.
   a. This is the same User object that exists in our database.

```javascript
{
    "_id": {
        "$oid": "5a5876c995cc7c3a8c7bb492"
    },
    "name": "Aaron Sipser",
    "fbid": "1816889578343526",
    "__v": 0
}
```

= req.user
Step 2C : Enhance api.js with req.user

1. Make the /whoami route return req.user!
2. Make index.js make an API call to the ‘/api/whoami’
   a. Inside the callback function console.log the user!
Let’s see our changes.
npm start
go to localhost:3000
The option to post a story should now appear!! Why??
Any questions?

If you got lost let’s catch back up

git reset --hard
git checkout step3

Note if you do this you have to re-enter your MongoDB URI and facebook AppID + Secret!
Step 3A: POST A STORY! (api.js)

1. Together let’s write the code needed to post a story
2. Go to api.js and go to the `post` route for a story
3. Make use of the magical `req.user` object!
4. Being able to post a story with the correct name requires only a two line change!
5. Try posting a story!
Step 3B: Let’s make the post route more robust

1. This code is not very secure. We want to make sure only logged in users can post.

2. Add the following dependency

3. Change the post route by adding the following line

   ```javascript
   const connect = require('connect-ensure-login');
   ```

   ```javascript
   router.post(
   '/story',
   connect.ensureLoggedIn(),
   function(req, res) {
     const newStory = new Story(
       { 'creator_id': req.user_id, 'creator_name': req.user.name, 'content': req.body.content, });

     newStory.save(function(err,story) {
       if (err) console.log(err);
     });

     res.send([]);
   });
   ```
Step 3C: Write the code for posting a comment!

1. Go to the post route for a comment in api.js and change it so you can post a comment!
   a. Hint this follows the exact same process as the post for a story
Any questions?

If you got lost let’s catch back up

git reset --hard

git checkout step4

Note if you do this you have to re-enter your mongoDB URI and facebook AppID + Secret!
Step 4A: Updating the navbar!

1. Use the fact that user._id !== undefined if the user is logged in!
2. Similar as to how the Login URL links to ‘/login’ make the Logout URL link to ‘/logout’
3. Don’t worry about creating a route for logging out, we’ll do that together.
   More passport magic!
Any questions?

If you got lost let’s catch back up

git reset --hard

git checkout step5

Note if you do this you have to re-enter your mongoDB URI and facebook AppID + Secret!
Step 5A: Fixing the profile!
Step 5A: Updating the navbar!

1. Go to profile.js
2. What is it doing currently?
3. Fix the navbar using the ‘/whoami’ route (hint: similar to index.js)
Step 5B: Fixing the ‘/api/user’ route in api.js

1. Go to api.js in the server code
2. What do GET requests at ‘/api/user’ currently return?
3. Make ‘/api/user’ do a lookup in the database (hint: you will have to import the User collection just like you import Story and Comment above)

If you get the above error, you haven’t imported the User collection!
Step 5C: Making the last post update

1. Go to api.js in the server code
2. Final challenge try it out on your own! (Hint: all the code you have to write is in the POST for `/api/story` in api.js.