



Project Auth

Technigo/project-auth

Replace this readme with your own information about your project. Start by briefly describing the assignment in a sentence or two. Keep it short and to the point. Describe

 <https://github.com/Technigo/project-auth>



For this week's project, you're going to tie all the skills you've learnt so far to build an API with authentication to implement a registration flow, and a frontend with forms to register, sign in, and view some content once you're logged in.

Your project needs two parts; a backend API, and a React frontend. You'll need to create a `User` model using mongoose, with properties for your registered user, and to store a user's access token.

Then, on the frontend side of things, you'll need to build up a registration form that POSTs to your API. You'll need to store the access token you get back in the browser using local storage, and then use that token when making other requests to your API.

Once a user is logged in, you will need to have one last endpoint which returns some content which only logged-in users should be able to access. You can choose what you want this endpoint to return, and if you want to make multiple endpoints, that's fine too. It could return hard-coded data or something from the database - either is fine. Whatever you choose, it should be displayed in the frontend after you've logged in.

To summarise, your API needs:

- Registration endpoint, to create a new user.
- Sign-in endpoint, to authenticate a returning user.
- An authenticated endpoint which only returns content if the `Authorization` header with the user's token was correct.

Your frontend needs:

- A registration form.
- A sign-in form.
- A page to show the authenticated content from the API.
- A 'sign out' button that removes the saved access token and redirects the user to the login form.

What you will learn 

- How to build a registration flow
- How to handle authentication, both in the frontend and in the backend
- How to build a frontend and backend at the same time

How to get started 💪

The [project repository](#) has a 'frontend' folder which contains the react starter project for you to build the frontend with, and a 'backend' folder with the express starter project to build the API with.

To get started, fork [the repo](#), then you will need to use two separate terminal windows to install dependencies, and then run both projects.

For the backend:

1. `cd backend`
2. `npm install`
3. `npm run dev`

For the frontend:

1. `cd frontend`
2. `npm install`
3. `npm start`

How to hand in the code 🎯

- When you're finished with the project, push your code to GitHub with these commands:




```
git add . git commit -m "your commit message" git push origin master
```

- Navigate to your repo and create a Pull Request into the Technigo repo
- Wait for the code review from your teachers

How to get help 🆘

Ask for help and share your knowledge about this project with the 'project-auth-api' tag on [Stack Overflow](#). Talk to your team on Slack and help each other out. Do some research about your problem, you are surely not the first one with this problem, Google is your friend 😊. And you can of course also reach out to your teachers.

Requirements 🖋️

Your project should fulfill the  **Blue Level** and all of the **General Requirements**. Use the  **Red Level** and  **Black Level** to push your knowledge to the next level!

General Requirements

- Contribute by helping others with this project on Stack Overflow.


- If selected; demo your solution for your team.
- Code follows Technigo's code guidelines:

Copy of Guidelines for how to write good code

- Your API should be deployed to Heroku or similar hosting service.
- Your database should be deployed using mongo cloud or similar.
- Your frontend to should be deployed to Netlify or similar.

Blue Level (Minimum Requirements)

- Your API should have routes to register and login, and finally an authenticated endpoint.
- The authenticated endpoint should return a 401 or 403 (see [401 vs. 403 on SO](#)) with an error message if you try to access it without an **Authentication** access token or with an invalid token.
- Your frontend should have a registration form which POSTs to the API to create a new user
- Your passwords in the database should be encrypted with bcrypt
- Your API should validate the user input when creating a new user, and return error messages which could be shown by the frontend (displaying the errors in a nice way in the frontend is a stretch goal - its fine to just show 'Something went wrong' on the frontend if you run out of time)

 Make sure you've committed and pushed a version of your project before starting with the intermediary and advanced goals.



Red Level (Intermediary Goals)

Remember: For any new feature you add to the backend, be mindful of how that will require the frontend to change, and vice-versa.

- Store data in the database for your authenticated data routes.
- When registering, display error messages from the API next to the field which has the error. For example, if the email address is invalid, show an error message next to the email input.

Black Level (Advanced Goals)

- Add more routes, perhaps even a **POST** route to create new objects in your database as a logged-in user.
- Improve validations in the backend to ensure unique email addresses, or validate the email address format using a regular expression.

  Don't forget to add, commit and push the changes to GitHub when you're done. 🏁

