❤️

Project Happy Thoughts API

Technigo/project-happy-thoughts-api

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

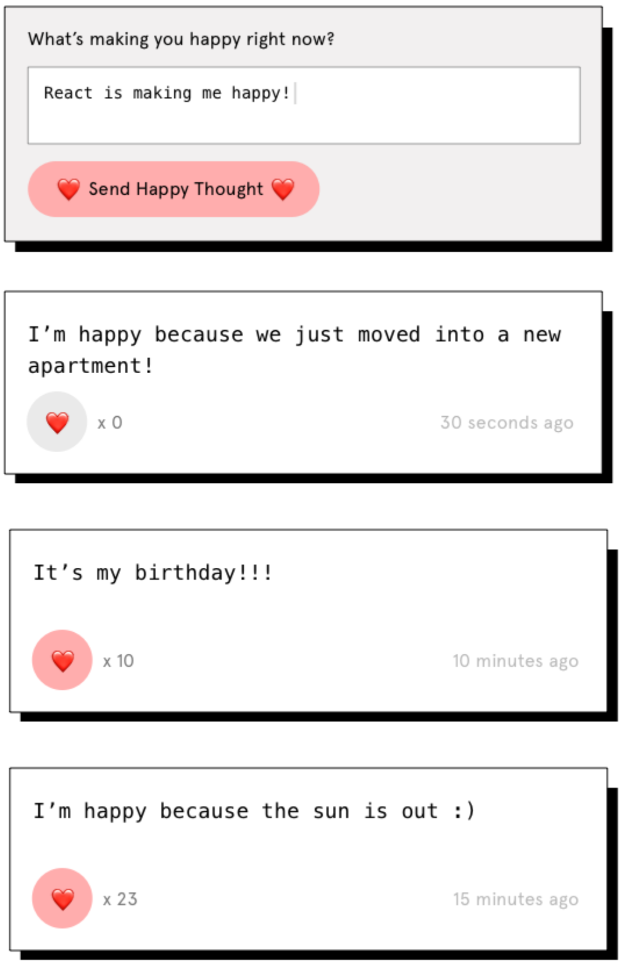
https://github.com/Technigo/project-happy-thoughts-api

> This week's project is to use your new skills with Express and Mongodb to build an API which includes both GET request endpoints to return data and POST endpoints to create data.

If you followed the 'Happy Thoughts' React assignment, you should remember this one - we're going to build our own version of Twitter, but focusing on positivity and friendliness rather than, well, Twitter. 😉

In the Happy Thoughts project, you built a frontend in React which uses an API we created to store thoughts. For this project, we want you to build your own API which works in the same way and should become a drop-in replacement for the API you used in the React frontend.

We built a frontend that has a form to write a new 'happy thought', lists recent thoughts, and shows a count of 'hearts' on each thought. Users could then click the heart to like a thought. It looked like this:



I'm happy because we just moved into a new apartment!

It's my birthday!!!

I'm happy because the sun is out :)

In order to replace the API we built, you're going to need to build a `Thought` Mongoose model which has properties for the `message` string, a `heart` property for tracking the number of likes, and a `createdAt` property to store when the thought was added.

Then, you'll need to add 3 endpoints:

`GET /thoughts`

This endpoint should return a maximum of 20 thoughts, sorted by `createdAt` to show the most recent thoughts first.

`POST /thoughts`

This endpoint expects a JSON body with the thought `message`, like this: `{ "message": "Express is great!" }`. If the input is valid (more on that below), the thought should be saved, and the response should include the saved thought object, including its `_id`.

`POST thoughts/:thoughtId/like`

This endpoint doesn't require a JSON body. Given a valid thought id in the URL, the API should find that thought, and update its `hearts` property to add one heart.

The thought model

We mentioned the `Thought` model and its properties a bit earlier. Each of these properties has some special rules or validations which you must implement to make a good API for the frontend:

- `message` - the text of the thought

  → Required

  → Min length of 5 characters

  → Max length of 140 characters
- `hearts` - the number of heart clicks this thought has received

  → Defaults to `0`

  → Should not be assignable when creating a new thought. For example, if I send a POST request to `/` to create a new thought with this JSON body; `{ "message": "Hello", "hearts": 9000 }`, then the `hearts` property should be ignored, and the object we store in mongo should have 0 hearts.
- `createdAt` - the time the Thought was added to the database

  → Defaults to the current time

  → Should not be assignable when creating a new thought

Using your API

Once you've created your API, you should deploy it, and update your frontend project to use your own API instead of the old Technigo one. The idea is that if you build this API correctly, **the only thing you should need to change in the frontend code is the URL to the API,** to change it from the Technigo one to the one you deploy.

What you will learn 🧠

- How to build a full API which includes handling of user input
- How to include error handling to return good validation errors
- How to build an API which works well with an existing frontend

How to get started 💪🏼

1. Fork the repo
2. Clone the repo into your projects folder on your computer
3. Open up VS Code and start coding!
4. Install dependencies: `npm install`
5. Start the development server: `npm run dev`

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

How to get help 🆘

Ask for help and share your knowledge about this project with the 'project-thoughts-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.

🔵 Blue Level (Minimum Requirements)

- Your API should implement the routes exactly as documented above.
- Your `GET /thoughts` endpoint should only return 20 results, ordered by `createdAt` in descending order.
- Your API should validate user input and return appropriate errors if the input is invalid.
- In the `POST /thoughts` endpoint to create a new thought, if the input was invalid and the API is returning errors, it should set the response status to `400` (bad request).
- The endpoint to add hearts to a thought should return an appropriate error if the thought was not found.

💡 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.

- Give thoughts a category or tags. So you could organize them. For example 'Food thoughts', 'Project thoughts', 'Home thoughts', etc.

- Allow users to enter their name in a new property on the thought model, or remain anonymous.

⚫ **Black Level (Advanced Goals)**

- Add filtering and sorting options to the endpoint which returns all thoughts. So you could choose to sort by oldest first, or only show thoughts which have a lot of hearts.
- Implement pagination in your backend & frontend so you can click through pages of thoughts. The frontend could request a specific page, and show only that page. The backend would take the request for that page and return only the thoughts for that page. Rather than only showing the most recent 20 thoughts.
- You could also experiment with implementing infinite scrolling on the frontend rather than having a list of page numbers. This idea is similar to paging and involves frontend & backend changes.
- Feel free to add other features that pop into your mind to exercise creating and fulfilling a virtual "contract" between the frontend and backend. This is a very valuable exercise in understanding both parts.

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