



## General Assembly - Software Engineering Immersive

### Poppins 📖 - A Python/ Flask/ Postgres full-stack app

#### Overview

Project 4 in the General Assembly Software Engineering Immersive course built up the knowledge I had gained in the previous three weeks. The project required my team to use a Python Flask API, using a Flask REST framework to serve our data from a Postgres database. We built a separate frontend using React, and deployed the website online.

A group of three of us worked on the website for eight days and then I presented it to the rest of the class. You can see the files for the completed app on GitHub pages [here](#).

#### Brief

We had to:

- Work in a team of three, using Git to code collaboratively
- Build a full-stack application by making our own backend and our own frontend
- Use a Python Flask API to serve our data from a Postgres database
- Consume our API with a separate frontend built with React
- Create a complete product, which meant multiple relationships and CRUD functionality
- Consider user stories/wireframes and identifying the features that were core to MVP
- Ensure we developed a visually impressive design
- Deploy our app online

#### Technologies Used

- Python
- React
- JavaScript (ES6)
- CSS
- HTML
- APIs
- Insomnia
- Heroku
- VS Code
- Git and GitHub

#### Approach

We initially spent some time working out what project we wanted to do, and we had a few different ideas. We decided on a website called Poppins, named after Mary Poppins, which would allow parents to rate books that their children have read, and would allow them to choose what book their children might like to read next, based on the comments of other parents.

We next looked at the functionality that we would need in order to reach a minimum viable project. This included users signing up and logging in so that they can comment on other books, and we realised the user would need to be able to add their own books to the library well.

We also decided the user would likely want to sort the books by age category and genre for the website to have maximum use. This functionality means that the user will be able to genuinely use the site and find the perfect book for their children.

A neat feature was to include a link to Google Books, which displays the first few pages of each book. This was included through an API, with a button appearing under each book that invites the user to 'read more'.

We looked at some stretch goals as well, and this included introducing an external API so that we would have a large number of books in the library. And we considered a rating system where books could be marked out of five or ten.

## Screenshots

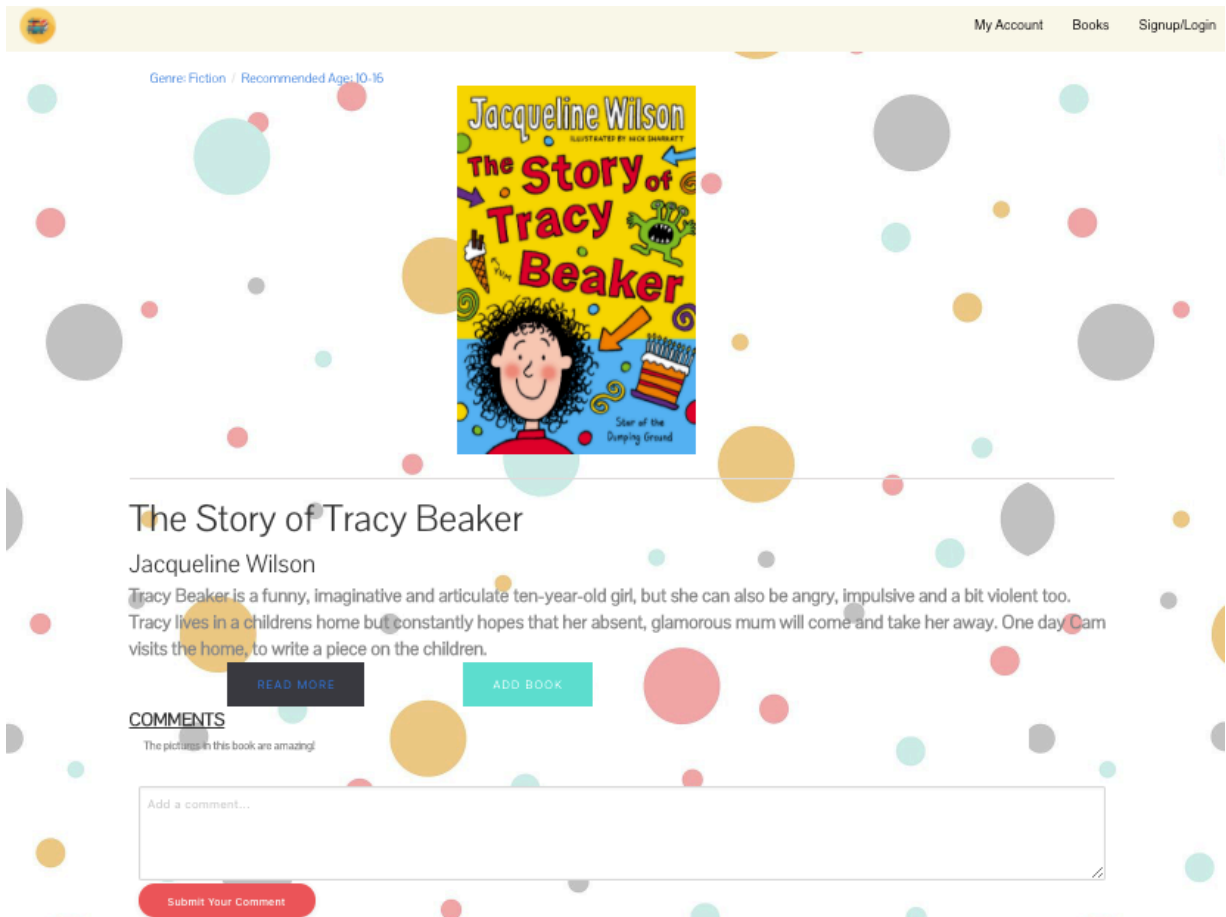
Below is an image of the Home page. The image in the background gently moves from left to right, which looks great, but the styling could be sharpened up a little more.



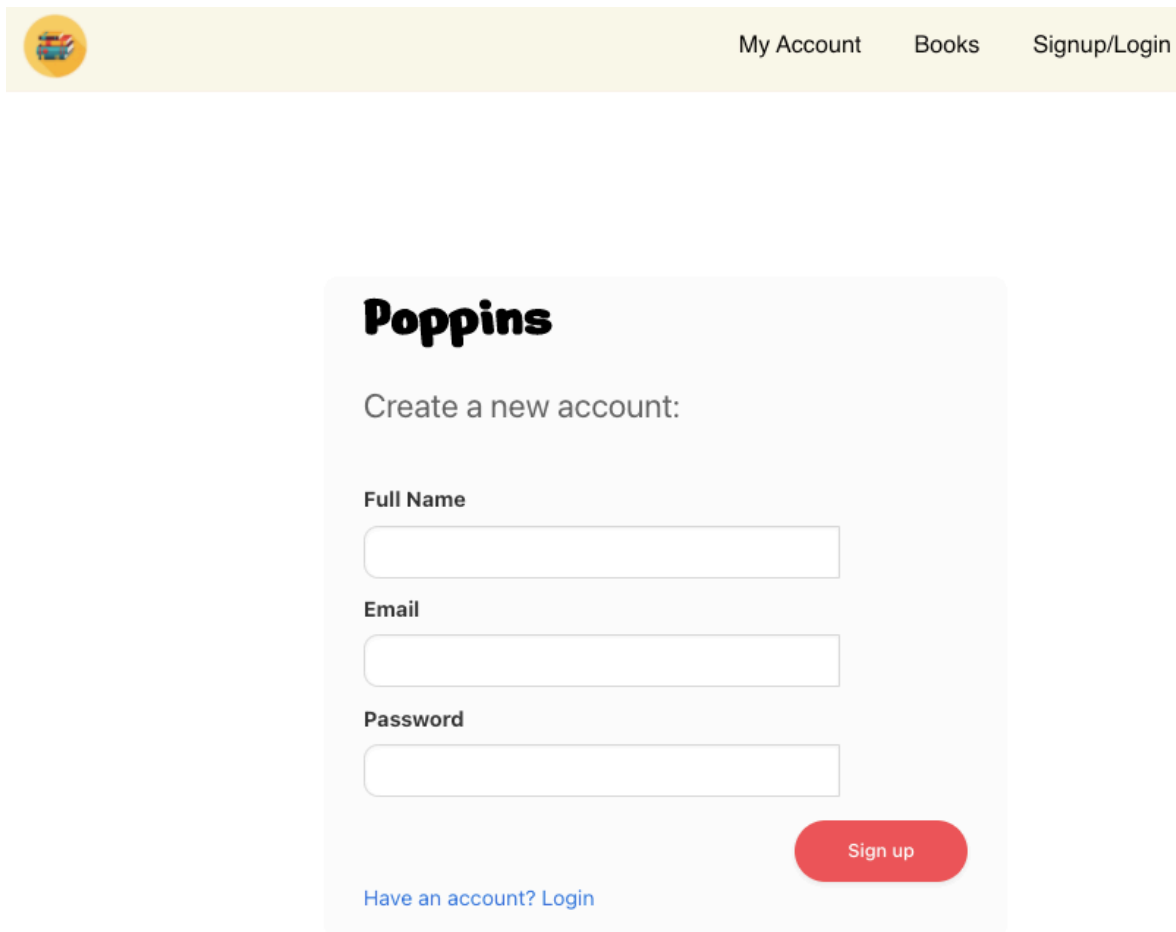
On the next page shows the list of all the books. It's a colourful page because children's books are inherently bright by design.



The screenshot below shows the page that appears when the user clicks on a single book. Note the comments section, the design of which needs further fine-tuning, the comment box, and the ‘Read More’ button.



The screenshot below shows the Signup/Login page. The user can create a new account, as shown, or they can click the link below to jump to the login page if they've already created their account.



The screenshot shows a web page with a yellow header bar. On the left is a circular logo with a colorful abstract design. To the right of the logo are three links: "My Account", "Books", and "Signup/Login". Below the header is a white card with the "Poppins" logo at the top. Underneath is the text "Create a new account:". There are three input fields labeled "Full Name", "Email", and "Password". To the right of the "Password" field is a red "Sign up" button. Below the input fields is a link that says "Have an account? Login".

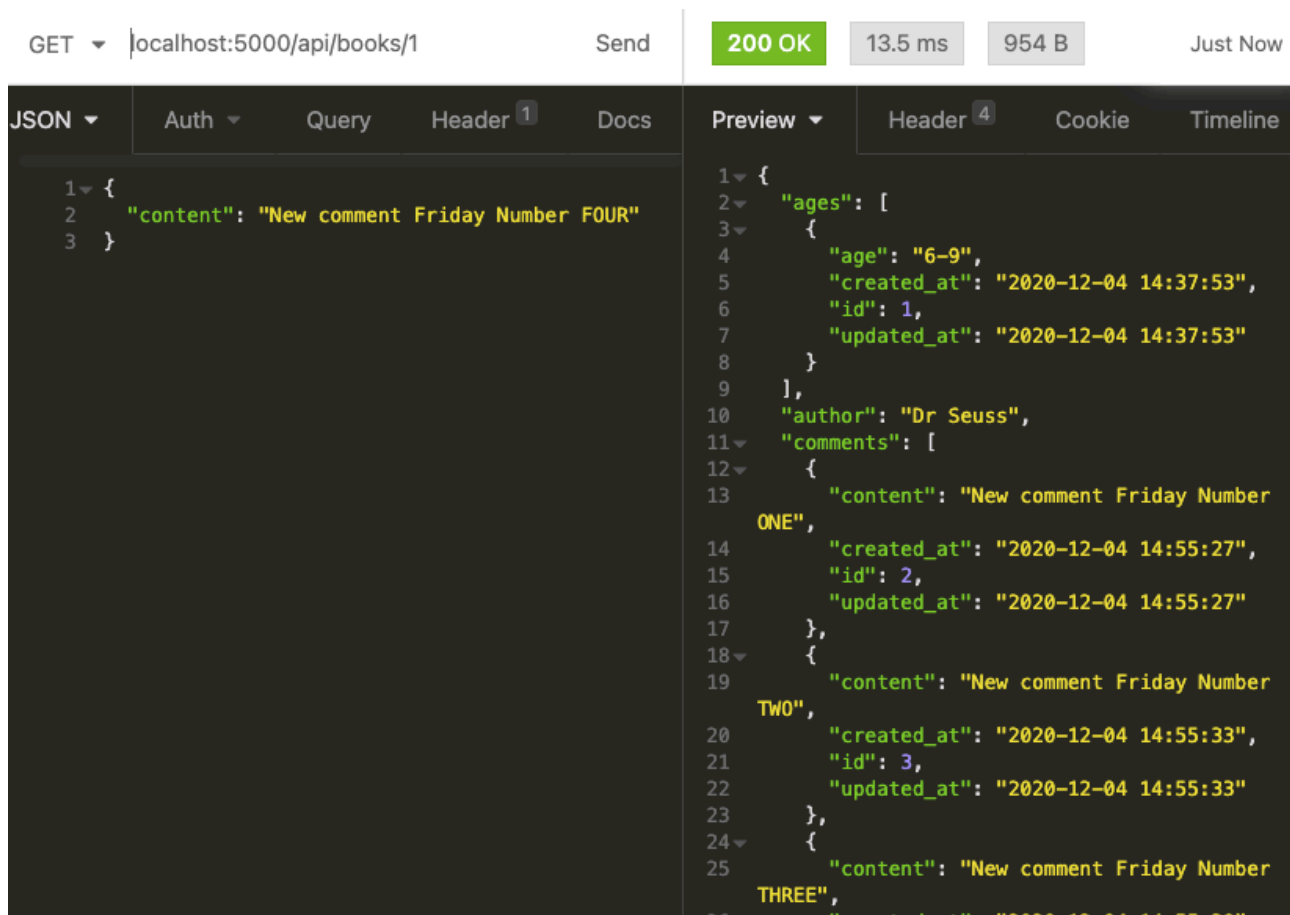
## Lessons Learned

We learnt a lot more about working together to produce a finished product. Breaking the work down into sections that each of us could tackle wasn't too difficult, but by its nature this project meant we would often be working in the same files and this created a few conflicts that we had to think about before we could choose which option we wanted to keep. This was clearly down to experience, and was partly down to writing code that was properly laid out (particularly around indentations) and was easy to read. My experience as a journalist and a writer means I understand the importance of making sure what we right in code is clear and concise.

We worked hard to ensure the functionality was perfect, but that meant we didn't leave much time for the styling. It can be difficult to get the balance right because ultimately the styling is what people see when they visit the site, but at least we can relax in the knowledge that it all works properly!

I was already aware that taking a step-by-step process was critical, but this project really underlined the idea that I had to test every part of the code as it was written. Going step by step was much better than writing a lot of code and later spending time trying to find out where there was a problem.

I used Insomnia to work out exactly how the code for writing and submitting comments was working, and I've included a screenshot over the page that shows a highlight of the project for me - when the comments would appear on the write book with the write user ID.



## Bugs

The styling is perhaps the biggest bug, it needs a day or two spent on it to sharpen it up. The user's profile page is also inactive because we ran out of time. This is a big frustration actually because it would have brought in key functionality such as allowing the user to upload an image and allowing the user to see all of the comments that they've submitted in one place, as well as the books that they've created and added to the overall list.

## Potential Future Features

We would like to incorporate an external API so that we can dramatically expand the number of books available. One challenge to this was the lack of available APIs that focus on books, and this is something that we should have considered in more detail when we began the project.

We have also considered how we could set up a 'book swap' function, which would allow users to swap books with each other. This would be relatively easy to do because you could simply install a chat feature, or you could do it a more complex way that involved the user establishing a wish list of books they'd like to borrow.

## Credits

Thanks to the rest of the Poppins team: Rebecca Acio-Adea and Sagal Osman.