1: Web App Tour

Q1. Investigate the Flask app

First, get a sense of the application.

1. Visit the live site: https://ls.up.railway.app/
    a. Create a short link by entering a URL in the box
    b. Enter the shortened url in your browser
    c. Note that you are redirected to the original site

2. Navigate the [source code on Github for the _Flask_ version of the app](https://github.com/kiboschool/link-shortener/tree/main/flask). If you prefer, clone code for the whole repository onto your machine.
    a. Read `README.md` first to get an overview of the code.
    b. Scan `app.py`, which has the application logic
    c. Check out the files in the `static/` and `templates/` folders. (It's CSS, an image, and HTML!)
    d. Peek at `initdb.py` and `schema.sql`. They configure the database for the application.

3. Answer the following questions for the Flask version of the application.

Q1.1. Database

What database does the app use to store the Urls?

Q1.2. Redirect Status Code

What redirect status code does the app use for redirecting the user from a shortened Url to the original?

Q1.3. Redirect line of code

What line of code sets the redirect status code?(when redirecting the user from a shortened URL to the original)

Q1.4. Template file

What template file shows all of the Urls that have been created?

Q1.5. Steps to run locally

What steps would you follow to run the app locally (on your computer)?_Note: You don't have to run this app, but you can if you like!_

Q1.6. Packages

What packages does the app require? (If there are a lot, you can name just 3).

Q1.7. Missing Short Url

When a user tries to access a short url that does not exist...

What lines of code run?What happens? Describe the result in your own words.

Q2. Investigate the Express app

Navigate to the source code (on Github or on your machine) for [the _Express_ version of the app](https://github.com/kiboschool/link-shortener/tree/main/express).

1. Read `README.md` first to get an overview of the code.
2. Scan `app.js`, which has the application logic.
3. Check out the files in the `public/` and `views/` folders. It's (very nearly) the same CSS, image, and HTML as the Flask app.
4. Peek at the files in the `prisma/` folder. They configure the database for the application.

Answer the following questions for the Express version of the app.

Q2.1. Database

What database does the app use to store the Urls?

Q2.2. Redirect line of code

What line of code sets the redirect status code?(when redirecting the user from a shortened URL to the original)

Q2.3. View file

What view file shows all of the Urls that have been created?

Q2.4. Steps to run locally

What steps would you follow to run the app locally (on your computer)?_Note: You don't have to run it, but you can if you like!_

Q2.5. Packages

What packages does the app depend on? (If there are a lot, you can name just 3).

Q2.6. Compare to Flask

What do you notice that is the same between the Express and Flask versions of the app?What's different?

Q3. Investigating another App

In the Github repository, there are lots of versions of the same app, using different languages and frameworks.

Pick one of the [other implementations of the app](https://github.com/kiboschool/link-shortener#languages-and-frameworks). Navigate to the source code and answer the questions for it.

Note: Django and Spring are more challenging to understand. Fastify, Fastapi, and Sinatra are simpler. Feel free to explore all of the others, but choose just one to answer the questions about.

Q3.1. Framework chosen

Which version of the app did you pick?

Q3.2. Application file

What file has the application logic in this version of the app?If there's more than one, pick the one that seems like it has the most logic.

Q3.3. Redirect line of code

What line of code (in what file) sets the redirect status code?

Q3.4. Compare to Flask and Express

What similarities and differences do you notice between this app and the other two versions you've looked at?