STORD URL Shortener Exercise

The goal of this exercise is to create a URL shortener web application in the same vein as bitly, TinyURL, or the now defunct Google URL Shortener. It is intentionally open-ended and you are welcome to implement your solution using the language and tech stack of your choice (if you know React and/or Elixir well, then please use them for your submission). The core functionality of the application should be expressed through your own original code.

Although this project is small in scope, it is a great opportunity for you to show off your fullstack skills in: attention to detail, testing, CI/CD, design and development. Show us what you got!

Application Requirements

- When navigating to the root path (e.g. `http://localhost:8080/`) of the app in a browser a user should be presented with a form that allows them to paste in a (presumably long) URL (e.g. `https://www.google.com/search?q=url+shortener&oq=google+u&aqs=chrome.0.69i59j69i60l3j0j69i57.1069j0j7&sourceid=chrome&ie=UTF-8`).
- When a user submits the form they should be presented with a simplified URL of the form `http://localhost:8080/{slug}` (e.g. `http://localhost:8080/h40Xg2`). The format and method of generation of the slug is up to your discretion.
- When a user navigates to a shortened URL that they have been provided by the app (e.g. `http://localhost:8080/h40Xg2`) they should be redirected to the original URL that yielded that short URL (e.g `https://www.google.com/search?q=url+shortener&oq=google+u&aqs=chrome.0.69i59j69i60l3j0j69i57.1069j0j7&sourceid=chrome&ie=UTF-8`).
- Only allow valid URLs (e.g., start with `http(s)://{domain}/`)

Deliverables

- Implement your solution, including test cases for your application code.
- We will execute your code using either:
    - (a) the `make` targets specified in `Makefile` (download this file).
        - Edit the contents of `Makefile` to provide an interface for running and testing your application

        - Include any other notes for our engineering team that you would like regarding your approach, assumptions you have made, how to run your code, how to use your application, etc in a file named `notes.txt`.
    - (b) **Or you could use other tools** (like Docker/Docker-Compose, npm, or Yarn) **to make it easy for the Evaluator to quickly and easily setup your submission, run the tests, and run the application**.
        - Please put your instructions in `notes.txt` how to setup, run, and test your application if you are not using the Makefile
- E-mail the point of contact that sent you this exercise and include a link to a public GitHub (or GitLab) repository.
    - Note: if your github account is monitored or the like, then simply create a new github account for this submission.

Evaluation

To ensure consistency in the evaluation, our Engineering team uses a rubric to score your submission, which will be evaluated along the following five criteria by the Reviewer:

1. **Completeness** - Does your submission meet the **Application Requirements** and **Deliverables** specified above?
2. **Testing** - evaluate your use of test coverage to allow for iterative development
3. **Ease of setup** - how easy was it for the Reviewer to setup and run your app?
    a. hint: Try to emulate what it would be like to run a fresh install of your app following your own instructions in `notes.txt`. If you are not sure how to do that, then simply have someone that you know try the setup --following your instructions-- on their machine to see if the install, tests, and application all run smoothly
4. **Front end design** - demonstrate your familiarity with html, css, and front-end javascript frameworks. Consider the User Experience for this application.
    a. Note: we have a UI Architect for expert front-end help, but all our developers at STORD are expected to be full stack developers.
5. **Technical design** - Separation of concerns, adherence to certain 12 factor App principles, knowledge of backend frameworks, security concerns, etc. -- Note on the Database- We would like you to use a persistent datastore.

a. Please be ready to speak to your technical choices during the face-to-face (video) interview, as well as how your design might need to change if the requirements change.
b. Curious about the "performance requirements" for this exercise? Your application should be able to handle at least 5 requests per second. During the on-site interview, you can talk about how you might change your design if the system had to scale beyond that

Notes:

- "How long should this take?" Best rule of thumb, "turn this in when you are proud of your work"
    - There's no time limit (e.g., within four days) on when you need to turn this exercise into us.
    - You have a job, family and life that has many demands on your time, and we get that.
    - Just see us as your accountability partner here. You tell us when you think you can get it done, and we'll follow up with you around that time if we haven't heard from you yet. And we can set a new goal on when you can get it done if need be.
- The good news is that we will not subject you to a code exercise on a whiteboard in any subsequent interviews! You will discuss your code submission with the same engineer that reviewed it, which I think is pretty cool.
- You can use either Elixir and/or React (for SRE candidates, you can use any language). If you know one of these languages, then we'll teach you the other one!
- Please show us your strengths for **CI/CD (ease of setup), coding, testing, user experience, technical design, and attention to detail! Please show that you are taking this exercise as seriously as we do. For example, your Reviewer will spend 1.0 to 1.5 hours reviewing your submission.**

Sincerely, I want to thank you for the time that you are spending with us as a candidate with STORD. It is very much appreciated and we know it is a big ask. We hope to have you join our great team members that have made it through the same diligent process.

- David Hardwick