Project Idea

Problem

If websites/applications are not responsive, then users will NOT want to use them. An example of a really good, responsive website is this one. I couldn't find an example of a slow website (but I certainly know I've visited many of them in my time) BUT you can emulate one by turning on throttling in the Network browser tools. This means that having a quick, responsive application is KEY to keeping users on your website. At some point down the line, I want to build websites/applications/games that are quick and that feel great to use so I want to figure out how to make those things quick and clean.

Goal

The goal of this project will be to gain experience in different caching patterns and their use cases. The reason I want to learn more about the use cases for caching is so that I can recognize opportunities in the future when caching could help improve performance - - and I want to be able to identify which caching pattern should be used in that scenario. The caching patterns are outlined in this short solution brief by Redis:

- *cache-aside*
- *query caching*
- *write-behind caching*
- *write-through caching*
- *cache prefetching*

A secondary goal of this project will be to gain more experience using Redis for caching. I want to learn more about the use cases for caching in general, and how to use Redis to fulfill those use cases.

(Note: Making a nice frontend for this project is NOT a goal - - in fact, I know I could probably waste too much time trying to make CSS work. This means that primarily my goal will be to build the BE for this. I have plans to attempt a basic FE that is just raw HTML and JS with little/no CSS.)

Outline of the project

I can see the project progressing in the following way:

- *week one: planning (5 hours)*
  - review the solution brief by Redis and other information regarding the different caching patterns
    - I would want to see if there is a caching pattern that deals with concurrency - - What if multiple users are likely to modify data at the same exact time? If it is important that the system enforce strict consistency, then how would we need to set it up?
  - *deliverable:* pick THREE (for starting out) caching patterns to tackle; make document describing toy use cases in which I will implement each use case (e.g., a blogging website that has a page for loading user info -> query prefetching)
- *week two: build dummy website (8 hours)*
  - plan the basic pages on the toy site to accommodate each use case

    - NO CSS - - or very little -> should be time-boxed to 2 hours MAX
    - set up the basic DB in the backend for this
    - also research how to do load testing in Postman
    - also set up basic connection to Redis
    - (if time) containerize the BE in Docker
    - *deliverable:* a BASIC frontend with pages that mock the use cases we want; a BASIC backend that mocks the necessary components; a dummy load test in Postman (can be of the server `/health` endpoint); a dummy example using Redis in the server/website - - i.e., BUILD OUT THE DEVELOPMENT ENVIRONMENT THAT WILL BE USED THROUGH THE REST OF THE PROJECT
    - *(if the planning phase doesn't take too long, then this can start part way through week one in order to try to get more done)*
- *week three: use case one (10 hours)*
    - work on the first use case, implementing the caching layer in the website/BE
    - *deliverable:* first use case is implemented; chart detailing the architecture that implements the use case
- *week four: use case two (10 hours)*
    - (same as above)
- *stretch: use case three* (essentially, if there is extra time, then can do this)
    - (same as above - implement another use case)

Technology stack

I would want to use the following technologies:

- basic HTML site served statically via Node (no/little CSS)
- TypeScript/Node BE
- PostgreSQL for cold storage
- Redis for caching
- Docker for containerizing the BE
- Postman for testing the BE, particularly for load testing (5000 reads/5000 writes)