TenantBase Full-stack Technical Assignment

For this exercise you will be creating a key-value storage server that speaks a small subset of the memcached protocol and persists data in SQLite. It should also have an HTTP server running out of the same process for monitoring. The monitoring page should include some basic Javascript interactivity.

Why this project?
We aim to have our assignment demonstrate your knowledge and coding style, both backend and frontend. We want to respect your time, so this should be mostly implementable by wiring together off-the-shelf libraries. Also, the memcached interface is just cool!

How will it be evaluated?
Your code will be evaluated on a few basic dimensions: does it work? Is it simple and clear? Does it follow prevailing Python and Javascript style conventions?

Please deliver a git repository so we can see the commit history, and how you went about implementing the assignment!

Including extra features or beautification beyond the spec below can earn you bonus points. Feel free to show off, but having a simple and correct core is more important!

The interface to run this server should be:

```
python main.py database.sqlite
```

The server process should start, using `database.sqlite` as its store and reading or initializing the database file as appropriate. Log and print whatever you like. The process should begin accepting connections on two ports:

- On port 11211, implementing a subset of memcached's TCP protocol.
- On port 8000, implementing a HTTP server that serves a simple status page, with a listing of keys and their associated values.

Persistence
Keys and values should be persisted in the SQLite database, so that your server has some level of durability and can shut down and restart without losing values.

Memcached Interface
Your server will speak a tiny subset of the memcached protocol. It should support three commands:

- `set` - You should accept but ignore the `exptime` field. Stored values will never expire in our system.
- `get`
- `delete`

The memcached protocol reference[1] will be extremely helpful! You can easily test your implementation via telnet[2] — there are `set` and `get` examples at the link in the footnote.

Web Server Interface
Upon visiting port 8000, your server should display a simple monitoring interface:

- It should show a list of keys stored in the server, with the values hidden.
- You should be able to "toggle open" a key using Javascript, and see the associated value stored for that key.

Again, the purpose of this section is to display a comfortability with HTML and modern Javascript. With that in mind, a few clarifications:

- Feel free to use a CSS/HTML framework of your choice to make a more beautiful end result. Though not required, it will definitely win you points!
- Though overkill, please use a "modern" JS framework for your interactivity. React, Vue, Angular 2+ are all fine, just not jQuery, no matter how right it may be for the job 🙂.
- This project is meant to show your coding style and skill across the stack, not make production ready software. As such, feel free to make simplifications. Two examples that should save you a ton of time:
    - Dump the full list of KV pairs in the monitoring HTML template — don't worry about fetching values dynamically via AJAX.
    - Don't worry about webpack/Rollup or fancy JS bundling if you don't want to. Including React and Babel via script tags[3] should get you rolling with React + JSX with just your HTML template and a JS file.

Last Thoughts
If you have any additional questions or thoughts, please reach out for clarification! This is software development, and if something seems complicated, we can talk it out and find a simple solution.

---

[1] https://github.com/memcached/memcached/blob/master/doc/protocol.txt
[2] https://wincent.com/wiki/Testing_memcached_with_telnet
[3] https://reactjs.org/docs/add-react-to-a-website.html