ZenRows Technical Assessment (Software Engineer)

Welcome to the ZenRows technical assessment! Your task is to build a maintainable MVP for a REST API that manages “device profiles” for a web scraping service. We want to see how you approach real-world problems, prioritizing **code that is clean, readable, and logically structured**.

You’ll design an API that lets users create, modify, list, and delete device profiles. Each profile should store options such as device type (desktop or mobile), window size (Width and Height in pixels), user agent string, country code (for proxy), and custom headers (an optional list of key-value pairs to send as custom headers with the request). Users should be able to build profiles from scratch or select from predefined templates. Your database schema should support future changes gracefully - without breaking existing profiles. Choose any database you prefer and explain your reasoning in your README.

As you build, focus on maintainability. Organize your code with clear structure, modular functions or classes, and sensible naming. Use proven frameworks and tools, and briefly explain your choices in comments, documentation or README.

Implement basic authentication to secure your endpoints, ensuring users can only access their own device profiles. There’s no need for full user management - just a simple mechanism to restrict access. Write tests for your core logic and at least one endpoint, using any testing framework you like. Be sure to mention why you picked it.

Prepare a README that explains how to set up, run, and test your project, along with a summary of key design decisions. Containerize your project with a Dockerfile so it can be run easily; the API should be accessible on port 8080.

If you have ideas for optional enhancements - such as validating profiles before saving - feel free to add them. However, don’t worry about advanced scalability concerns like sharding, distributed systems, or complex caching. Avoid over-engineering and focus on delivering a solid, maintainable MVP.

If you run out of time, add comments about what you'd do next. Use Go as a programming language, any frameworks and resources you prefer, and make sure you can explain your choices.

We value autonomy, maintainability, and clear reasoning. Good luck!