# Una Health Backend Tech Challenge (v2)

Dear candidate,

We're glad you're considering joining Una on the mission to change the course of metabolic disease!

The main objective of this tech challenge is to build an API to store glucose levels.

The blood glucose level is a key indicator for the human metabolism so it's important for our users to be able to store and access the data.

## Meta

- Please schedule 4 hours for the tech challenge, we'd expect this to be sufficient for the outlined tasks. If you require, or you want to spend more time on the tech challenge please do let us know and we can extend the time to up to 8 hours.
- Once you're finished send us information on how to access your solution (e.g. a URL where it's deployed). We prefer you to use a GitHub repository for your code, but we'll also accept BitBucket and GitLab or a similar solution, this way we can see your commit history and you can easily share the link with us.
- We require solutions in Django, or other Python frameworks, e.g. FastAPI, Flask.
- Please organize, design, test and document your code as if it were going into production.
- You're free to choose additional libraries as you see fit.
- Be sure to quote sources properly. Please refrain from cheating as we will check for fraud / plagiarism.

- If you're using LLMs, such as GPT-3.5, to generate/augment parts of your response please make sure to explain your usage.

# Tasks

### API

- The provided `zip` file (please download here: `https://s3-de-central.profitbricks.com/una-health-full-stack-tech-challenge/sample-data.zip`) contains three sample `csv` files which contain glucose values for users. The naming pattern for the files is: `user_id.csv`. Create a suitable model in the backend to store the glucose values. Also supply means of loading the sample data into the model / database.
- Implement the following API endpoints which use the model / database:
  - `/api/v1/levels/` : Retrieve (`GET`) a list of glucose levels for a given `user_id`, filter by `start` and `stop` timestamps (optional). This endpoint should support pagination, sorting, and a way to limit the number of glucose levels returned.
  - `/api/v1/levels/<id>/` : Retrieve (`GET`) a particular glucose level by `id`.

# Bonus

- Implement a `POST` endpoint to fill / pre-populate the model / database via an API endpoint.
- Implement an export feature, e.g. to `JSON`, `CSV`, or `Excel` in the API.
- Create means to run your solution locally, e.g. via Docker Compose or Minikube.
- Deploy and make your solution available online.

## Evaluation Criteria

- API: created model; implemented endpoints; added error checks for passed parameters; used suitable URL routing; implemented loading of sample data
- General: selected and explained chosen software architecture, used clean code structure, followed and explained coding conventions
- Documentation: added clear inline & high-level documentation, added documentation on how to start & run the submitted solution
- Tests: written unit tests
- We evaluate your communication with us
- *Bonus*: implemented `POST` endpoint
- *Bonus*: implemented export functionality to one of the mentioned target file formats
- *Bonus*: created a local deployment
- *Bonus*: created a remote deployment

All the best and happy coding,

**The Una Health Team**