Technical exercise

Guidelines

The tasks can be written in Go, JavaScript or Python. Please select the language that you think best represents you. You should implement back-end interfaces as APIs and front-end interfaces as SPAs. We expect you to develop both the back-end and the front-end of a full stack application, packaged as container image(s), deployed in either Google Cloud or AWS, with a chosen infrastructure configuration (e.g. Terraform or CloudFormation templates).

The tasks must not distribute binaries of any form, executable or not. They should only distribute source code through a private repo (please don't put it in a public repository). Instructions for the build and execution of the artefact must come with the project itself in a format of a README file. Please also make sure the code is well commented to guide the reader through the approach of the solution if the code is especially complex.

Please spend about 4 hours on the task and be in touch once the time run out as some code is better than no code.

The tasks are intentionally ambiguous in nature, and assumptions will have to be made to accomplish the tasks. When you need to make these assumptions or a trade-off (due to the limited time, etc.), please make a note of it either as a comment in the code or in the README.

Exercises

Select one task from the list below.

#	**Tasks description**
2	A system that allows a store's employees to view and book shifts
4	A system that given an input string performs entity analysis and returns the entities found (you can use NLP libraries, but please don't use high level ML APIs)
5	A form including image upload and a paginated, responsive display page that renders thumbnail tiles and other data from a database
7	A game application that challenges users with riddles and keep track of leaderboard that keeps track across multiple sessions and multiple users
9	Display a list of 100 items in pages of ten, keeping track of pagination via the URL (extra points for relative pagination, so new items do not affect pagination)
10	An API for tracking number of clicks and displaying them on a graph, over time (extra points for producing a heat-map)
11	Track a long form's input over three pages and output the final result to stdout in json when the whole form is considered complete
12	A service which takes data from an open data source (eg. Kaggle or https://pokeapi.co/) and create a simple SPA to browse or search.

Considerations

Please take the following aspects into the consideration as much as you can while developing the solution:

- Simplicity
- Reusability
- Scalability
- Performance
- Usability
- Testability
- Code quality
- Design patterns and best practices

Consider this a commercial request of work and approach it with the same level of professionalism you would any other task.

Purpose

The purpose of this exercise is to assess the candidates in main areas:

- Programming competency
- Choice of libraries / frameworks
- Structure of the code
- Efficiency / extensibility of the approach
- Automated testing skills
- Knowledge of connecting with backend
- Understanding of code complexity
- Code quality

After the technical assessment, there will be a follow-up face-to-face interview to discuss the approach, so make sure you note down questions you may have asked.