One Padmore
Place, off George Padmore Road.
Nairobi, Kenya
P.O. Box 5980-00200.

Velltechgroup - Web Developer
Nairobi, Kenya - Full-time

Frontend Engineer Assessment

The purpose of the assessment is to assess the web frontend skills required for the job. The software written is for **assessment purposes only**. It should be hosted on a code repository service (e.g Github, Gitlab) with public access to make the assessment easy. You will have a maximum of 2 weeks after getting the assignment to have the assessment ready but submitting the assessment earlier is encouraged.

For the Backend Service:

- Option 1: Candidates can build an application that consumes data from https://jsonplaceholder.typicode.com/

- Option 2: Candidates can build a minimal backend application that is able to offer the following Restful APIs.

NB: Code quality for the backend will not be strictly assessed.

- api/users
  - Each user should have the following fields:
    - Name
    - Username
    - Email

- api/albums
  - Each album should have the following fields:
    - Album id
    - User id
    - Album title

- api/photos
  - Each album should have the following fields:
    - Album id
    - Photo title
    - Image URL

For the Frontend Application:

Your client application should have these sets of screens/pages:

1. A landing section accessible to all app visitors with a brief explanation of what the application does.

2. A page to authenticate users (Login page):
   - For the authentication layer, you can use any auth provider you are familiar with (Google, Facebook, Github) and maintain the auth sessions in your application.
   - You have the option to combine the authentication page and the landing page.

3. Logged-in pages that require authentication for access:

   - (Home) Have a page that lists all the users:

i. The page should tell you how many albums a user has

ii. The page must run a GET request for the users

iii. The page must run a GET request for the albums

- (User) Have a page that shows you the user's information:
  - i. This is the page shown when a user is selected
  - ii. The page should list a user's albums
  - iii. The page must run a GET request for the user-selected
  - iv. The page must run a GET request for a user's albums

- (Album) Have a page that shows you an album's information and its photos
  - i. This is the page shown when you select an album
  - ii. The page should list an album's photos
  - iii. The page must run a GET request for the album selected
  - iv. The page must run a GET request for an album's photos

- (Photo) Have a page that displays a photo
  - i. Allow a user to edit the title of the photo
    - A PATCH/PUT request must be sent to the backend server after editing the page
  - ii. The page must run a GET request for the photo

Software Requirements:

Must-Have:

- The application must be responsive on mobile, tablet and desktop
- The application should be able to run with simple commands *(npm install, npm run demo)*
- The code should have a development and production programming environment (branches)
- The screens built should retain the pages after reloading
- Informative commit messages that follow conventional commit messaging formats
- Proper documentation of the code
- Software unit tests that verify the software's correctness.
- A pipeline job to run linting and unit tests automatically
- A pipeline job to automatically deploy the software project once certain checks have been met
- Having the application deployed. Use any deployment services with free tiers e.g. heroku, vercel

Good To Have:

- Linters:
  - Javascript / Typescript
  - Less/SCSS/CSS
  - Commits
- Loaders to show that data is being fetched
- Logging service for application errors
- Use of a UI library for a polished-looking application e.g. Tailwind, Bootstrap, Foundation, Material, Bulma etc