Task guide

Thank you for expressing your interest in joining the Whoppah team! In front of you is a short task based on which we would like to gain insights into your technical skills. Even though the task is short, try to follow your usual practices and build it as best as you can. If you have any questions regarding the task, please feel free to reach out to us at any time. Estimated date of submission should be at most one week from the moment you receive the task. Good luck!

Task should be implemented in **Node.js** using the **Koa** framework with **Typescript**. The backend application should expose a **GraphQL** API that is built using the **Apollo Server**. You have a **free choice** of **ORM** that should be connected to the **PostgreSQL** database. If you want to cache anything, use **Redis**. The task should be done in a **GIT** repository which should be shared later for a review. We also want to see how you collaborate with others, so organize your GIT as if you were working in a team.

Requirements:

- Build a GraphQL API based on the task description
  Note: Frontend or any kind of UI is not needed for the backend task.

Expectations:

- Clean code and quality architecture
- Tidy git versioning

Additional:

- Dockerize the application for easier startup and testing whoppah®

Backend task

Whoopah is a marketplace for secondhand design and art, and the task will resemble a small section of it. The end goal of this task is to create a small backend application that will allow basic interactions between the platform's **products** and **categories**.

Category is identified by its title and slug. Slug should be unique, automatically created and is to be used for better SEO. Each category can have multiple products in it. For the sake of simplicity, model the system so that there is only one level of categories (no subcategories).

Products consist of title, slug, price and belong to a category. Slug should be unique, automatically created and is to be used for better SEO. For the sake of simplicity, model the system so that a product can only belong to one category.

The marketplace should be rich with content, therefore an API for **creating categories** and **products** should be implemented.

What would a marketplace be without a good overview of what it has to offer?

Let's highlight the **most popular categories** of the system by developing an API for fetching the categories. The API should support simple **paging pagination** and should **sort** categories **alphabetically** by default. It should also accept an optional parameter to **sort** the categories **by** their **popularity** instead. A category's popularity can be calculated by the number of products in it. The higher the number of products, the higher the category's popularity.

The system should also expose an API to **paginate** the **products** ordered by their creation date. The mentioned API should also support **filtering** the products **by** their **category**. For the sake of simplicity, don't implement any external search services. Implement a simple paging pagination API, with default sort by creation date and an optional parameter to filter the products by their category.

Lastly, let's boost the UX of our customers by implementing a mechanism to instantly **push the newly created products** to them. Server should have a way of communicating with the client applications and pushing data to them. Whenever a new product is created, active clients should receive it.