Purple JS Team Case Study

Overview

Your goal is to make a simple currency conversion web app with Node.js backend. User will input amount, currency and destination currency. Your app will show the converted amount.

You can find detailed requirements below. To be successful you don’t have to follow them strictly. Invest reasonable time depending on your free time and don’t worry if you can’t finish it all. Focus on quality of your code not on quantity of tasks. Make it readable, expandable and production ready. Make sure you deal with possible errors etc.

Feel free to ask any questions. Write down any significant problems you encountered. Let us know how you solved them or how would you tackle them if you had more time.

Requirements

- For backend use Node.js with any common framework (express, hapi, koa and serverless are all recommended)
- Your API can be based on REST or GraphQL
- For frontend we prefer React but you can also use other modern frameworks
- In the backend, use an external API to get the currency rates.
- The frontend has to communicate only with your custom API.
- The app should also display the following stats:
  - Most popular destination currency
  - Total amount converted (in USD)
  - Total number of conversion requests made
- Make sure the stats are not cleared on restart and are aggregate for all visitors. This means they have to be calculated and stored in the backend.
- Do not use locally installed database. You can use local file, free version of some cloud database or any other method.
- Your app should not require any pre-requisites. It should run on any machine with latest LTS version of Node. Alternatively you

can create a Docker image.

Additional tips

- Currency rates APIs:
  - https://openexchangerates.org
  - https://currencylayer.com
  - http://fixer.io
- Try to use ES6+ and/or typescript
- Split the code into several modules. Don’t use just one file for the Node server.
- When building the API, consider it public. Other developers might use it so make sure it's understandable, validate inputs, etc.
- Submit your work as a Git repo. Commit often as you work.
- Make your work production-ready
- Expect the app will be later developed further by someone else. Write your code with that in mind.
- Take a look at the stack we use in our GitHub. You can use it or take some inspiration from it.