zetaton-task

Node.js

Node.js Task

1. Create a RESTful API using Node.js and Express to interact with Firebase Firestore.
2. Design endpoints for CRUD operations (Create, Read, Update, Delete) on the Firestore Images collection.
3. Implement another API endpoint to shorten image links using axios.
   Use https://www.shareaholic.com/api/shortener for shortening links and save the new link on Firebase.
4. Ensure the project runs locally. No need to deploy it.

⚠ Monorepo vs polyrepo

- Monorepo vs polyrepo

✓ Node best practices

- nodebestpractices

⚠ security

- securing-node-js-in-production-expert-practices-for-every-developer

Firestore

- create firestore in test mode
- REST API with Firebase Cloud Functions, TypeScript, and Firestore
- 5-tips-firestore
- quickstart#node.js
- add-data

✓ Validators

- https://ajv.js.org/guide/getting-started.html

- https://zod.dev/
- https://express-validator.github.io/docs/

CRUD

- CRUD-Firestore

problem with axios

- html instead of json

  ✓ works fine with `fetch`

⚠ date-time with firestore

- change date format

React.js

React.js Task

1. Implement sign-up and sign-in functionality using Firebase authentication.
2. Fetch photos from the Pexels API (https://www.pexels.com/api/) and display them on the homepage in a grid view.
3. Provide a button to add photos to favorites. When clicked, the photo URL should be stored in the user document in Firebase.
4. Create a "Favorites" screen to display all the favorite images stored by the user.

- Note: Please use `Material UI` for styling the React components.

Project structure

- bulletproof-react

✓ React best practices

- react-best-practices

Material UI

- Material UI — Usage Guidelines
- styling-a-react-app-with-material-ui

✓ React Forms

- A Better Guide To Forms in React

⚠ Firebase Auth

- manage-users