Khimichna St., Lviv, Ukraine
Mechnikova St., Kyiv, Ukraine
(+380) 665 04 44 27
hello@radency.com

RADENCY

Hometask #3 | NodeJS

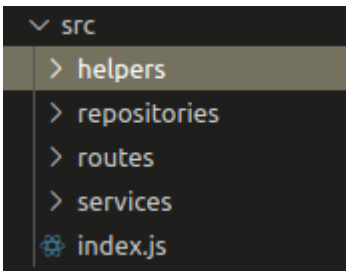
1. Your task is to create a NodeJS application with TypeScript that will have few REST endpoints. No need to wire it with Frontend part that you did in task #2.
2. Endpoints functionality should reflect your work on the frontend side - users can add, edit and remove notes, archive and unarchive them.
3. As an option you can use NestJS instead of NodeJS/Express with TypeScript.

List of endpoints should look like this:

Query type	**Endpoint**	**Action**
POST	/notes	Create a note object.
DELETE	/notes/:id	Remove item.
PATCH	/notes/:id	Edit item.
GET	/notes/:id	Retrieve item.
GET	/notes	Get all notes.
GET	/notes/stats	Get aggregated data statistics. You don't have to mock this data. You need to calculate it based on notes objects you have.

4. Store data in memory as a mocked object. Prepopulate it with 7 notes and use it by default as an initial state so that they are returned when you call an endpoint. You can use the same object structure as in the frontend using following columns: ["Name", "Date", "Category", "Content"] and also additional columns if needed.
5. Use the Postman application to check that your endpoints work correctly.
6. Add validation to each endpoint so that no one can add more properties or miss one. E.g. if you expect { name: <string>, age: <integer> }, there should be no way to send another shape of the object or its data type. You can use Yup for that purpose.

You can use an Express generator that will create a very basic structure for you. We suggest you follow the N-tier architecture pattern. That means you have to add a few more folders to that structure:



Routes - description of your routes and service calls
Services - Essense of business logic. You can call other services here, do validations, call repositories etc.
Repositories - here you work with data itself - add new items, remove, update. However you don’t have to have any business logic here.
Helpers - additional functions that can be used across the codebase.

Since we don’t use a database for the task, we suggest you store data simply in array variables in memory. This is not something you would do within a real project but works just fine for this hometask. However, you can use any library that will allow you to work with a persistence layer of your choice.