Every big project out in the internet, is generally, a single github repository. It will contain backend and frontend in the same repository.

**How things happened before monorepos:**

Introduced after npm 7.5 .

Before people would have made a single folder for FE and BE and push that onto git/npm.

Or upload them in separate repos.

**Why packages/modules:**

1. Re-usable code
2. Separation of concern
3. Teams can work on modules independently.

**Explore cal.com repo for modules and packages. There is a video of harkirat on his youtube channel.**

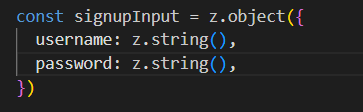
**Monorepos:**

Ex: Lerna, turborepo

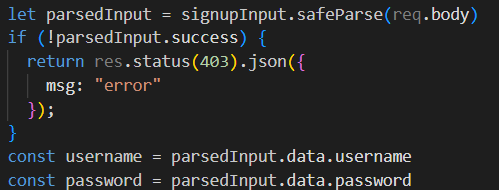
**Zod inference:**makes type safe and adds input validation. <https://zod.dev/>

How to add ‘zod’ in our project:

1. Install ‘zod’ library – npm install zod
2. Import {z} from “zod”, in your input authentication file
3. Describe a schema



1. Validate request



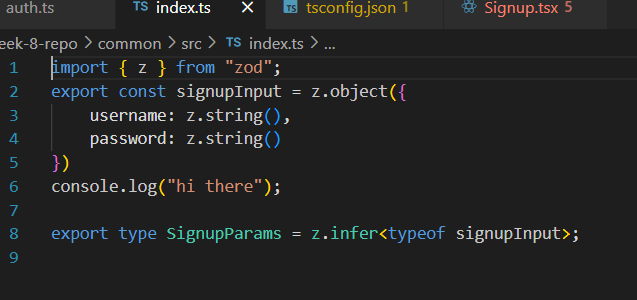
Here, we cannot know what is the type of input given to this signUp endpoint.

We can know the types from the backend, and write the same things again in frontend, which takes lot of redundancy.

Instead of that, is there any way to reuse the backend types only. Yess.

**How can we do it:**

1. To import this type into frontend and backend, create a common folder ‘common’
2. This common folder contains small modules, which are used by both frontend and backend.
3. Import zod in the common folder and add this code in it’s index.ts file.



**How to publish common packages, In pre-monorepo eras:**

1. If there is a module, that is used by both frontend and backend, that module is published onto npm.
2. Npm is also the manager of few such modules, that we import like zod
3. We will publish this common repo onto ‘npmjs.com’. React and express are also published here
4. Step 1: go to npmjs.com and create an account
5. Step 2: sign into your account in browser
6. Step 3: run ‘npm login’ for the folder you want to put in npm.
7. Step 4: login in browser with the otp you receive in your email
8. We cannot directly import ‘common’ because the we cannot import the folders that are outside the source folder directly.
9. Step 5: give your package a unique name. In package.json “name”:”@kdr\_10/comon”. Prefix it with your username.
10. Step 6: give the route of main js folder in package.json file. “main”:”dist/index.js”
11. Only publish your javascript files when pushing onto npm. Because that is what they use.
12. Step 7: npm publish –access=public
13. We can see the published packages in our browser account under packages in our profile
14. Run ‘npm pack’ to see the published packages in that directory.
15. There is a compressed zip file in the same folder. That is the folder that gets published on the npm.
16. If u try to republish, by changing any code, it will give error, saying package with the same version already exists.
17. So, change the version number in package.json and publish
18. Step 8: create a .npmignore file. Add all the folders and files that needs to be ignored in this file
19. Here, we add src , because we don’t need typescript files.
20. Now, your package can be used by anyone in the world.
21. Step 9: Now, install the new package wherever you want to use i.e., in frontend and in backend.

**\*\*\***dist folder should contain .d.ts file (declaration file) which gives the types of the variables in typescript. This file just contains the types, and no application code. It is a typescript file. Without that, we will get an error, saying ‘could not find a declaration for your package/module’.

How to get this declaration file.

In tsconfig.json file, add another key-value pair “declaration”:”true”.

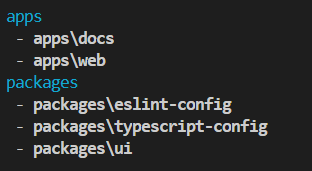
**In monorepo era:**

We have something called ‘npm workspaces’. Which take up the same job as npm packages. But, these folder can be present in the root folder of all the folders which use them.

Root>client,server,common

* TurboRepo is the most popular one. <https://turbo.build/repo/docs> >create a new monorepo > create a monorepo using the commands there.
* Step 1: npx create-turbo@latest

i)This will give the general next js files, given by turbo.



ii)By default, it is a pure frontend monorepo.

iii)It will create separate package to store tsconfig files, because the whole concept is to allow one module to import another module.

iv)Packages\ui contains all your react components. We put all the components in components folder. Packages\ui contains all the high level common useful components.

v)eslint-config is generally used for linting your project. That means, how many tabs we should have, whether we should have comma and all those things.

vi)apps\web folder will contain all the files, what you will show to the user.

* Step 2: cd <project-name> . Here, it’s ‘mono’.
* Step 3: npm install
* Step 4: npm run dev (this runs the project)

You can see what npm run dev is doing by opening the package.json files of apps, and looking at “dev” key.

In the apps>web>package.json file, you can see “ui”:”\*”. This is how our monorepos get other modules in the same repo. The main repo has packages\ui, the package.json has name:ui. If we want this ui, not from npm, but from local submodule, then we need to add ui:\* in the package.json file.

* Step 5: npm build

This will build all the applications separately, with the modules needed included in each of them.

* We can create a signup page commonly and addons in the 2 different apps, one for user and one for admin.
* Mui.com

U can see examples of turborepo in it’s github.