Create a react app: npx create-react-app (name)

cd (name)

launch app: npm start

desctructuring: const Hello = (props) => {

const name = props.name const age = props.age

event handling: const handleClick = () => { console.log('clicked') }

arrays: array.map(item => item), array.find(item => item.id === id), array.filter(item => item.value !== value)

array.concat(item), array.concat(…array, item)

install json-server: npx json-server --port 3001 --watch db.json (not global)

install axios: npm install axios

Install *json-server* as a development dependency:

npm install json-server --save-dev,

"server": "json-server -p3001 --watch db.json"

npm run server

to start a backend application Node.js and Express

npm init

scripts": {

"start": "node index.js",

Run it with: node index.js

Install express: npm install express

"dependencies": {

"express": "^4.17.1"

Updeta dependencies: npm update

Installation of nodemon to watch and make automatic changes to the application without restarting it:

npm install --save-dev nodemon

"dependencies": {

"express": "^4.17.1",

},

"devDependencies": {

"nodemon": "^2.0.2"

}

We can start our application with *nodemon* like this:

node\_modules/.bin/nodemon index.js

We can now start the server in the development mode with the command:

npm run dev

The issue lies with a thing called CORS, or Cross-Origin Resource Sharing. Install *cors* with the command

npm install cors

3.B

installing mongoose: npm install mongoose

installation of dotenv: npm install dotenv, and import: require('dotenv').config()to index.js file

Eslint for backend: npm install eslint --save-dev

Initialize default config: node\_modules/.bin/eslint --init

Change: "indent": [

"error",

2

],

And: "scripts": { // ..., "lint": "eslint } ." To package.json file

Install *jest* as a development dependency:

npm install --save-dev jest

"test": "jest --verbose" in package.json file under “scripts”

And "jest": {

"testEnvironment": "node"

}

At the end of package.json file

npm install --save-dev cross-env

supertest

npm install --save-dev supertest

npm test -- tests/note\_api.test.js for a tests in the file

npm test -- -t 'a specific note is within the returned notes' for a specific test in a file

Eliminating try-catch 4.b

npm install express-async-errors

and use async-await

[bcrypt](https://github.com/kelektiv/node.bcrypt.js) package for generating the password hashes:

npm install bcrypt

[mongoose-unique-validator](https://www.npmjs.com/package/mongoose-unique-validator) npm package. Let's install it:

npm install mongoose-unique-validator

login functionality and generation of token: Part 4.d

npm install jsonwebtoken

usage of props-types: part 5.b

npm install prop-types

use of Eslint in frontend 5.b

npm install --save-dev eslint-plugin-jest

testing React apps 5.c

npm install --save-dev @testing-library/react @testing-library/jest-dom

using cypress: 5.d

npm install --save-dev cypress, ad "cypress:open": "cypress open"

run both frontend (npm start) and backend (node index.js) then run cypress with

npm run cypress:open

We can get rid of it by installing [eslint-plugin-cypress](https://github.com/cypress-io/eslint-plugin-cypress" \t "_blank) as a development dependency

npm install eslint-plugin-cypress --save-dev

add: "plugins": [

"react", "jest", "cypress"

]

REDUX

6.A

Install redux with: npm install redux

The first parameter is the *state* in the store. Reducer returns a *new state* based on the actions type.

To make testing easier, we'll first move the reducer's code to its own module to file *src/reducers/noteReducer.js*. We'll also add the library [deep-freeze](https://github.com/substack/deep-freeze), which can be used to ensure that the reducer has been correctly defined as an immutable function. Let's install the library as a development dependency

npm install --save-dev deep-freeze

¨

possibly the easiest way using the [hooks](https://react-redux.js.org/api/hooks)-api of the [react-redux](https://react-redux.js.org/) library.

First we install react-redux

npm install react-redux

6.B

Redux Devtools: npm install --save-dev redux-devtools-extension

6.C

In order to run server and axios with Redux do:

npm install json-server --save-dev

"scripts": {

"server": "json-server -p3001 --watch db.json",

// ...

}

npm install axios

enables us to create *asynchronous actions*. Installation is done with the command:

npm install redux-thunk

6.D connect

Changing dispatch with Props

Part7

Part8

Start by creating a npm-project with npm init, then install the required dependencies:

npm install --save graphql

npm install apollo-server graphql

and lastly create an index.js file in the root of the project