Developer’s Manual

Pharma Server-Side NodeJS Application

Zacharias Christos Argyropoulos

Back-End Development

SAE Athens 2021

Overview

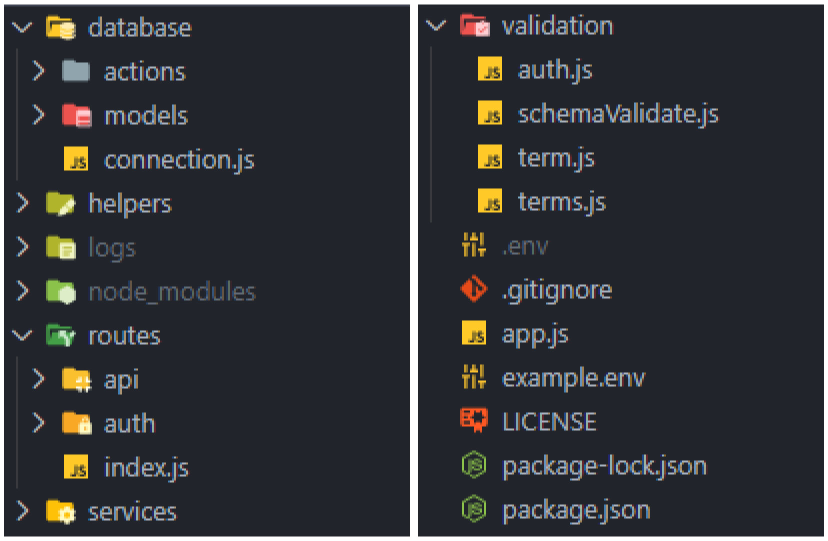
This project is a RESTful API that provides data and functionality to a frontend application named pharma-react. This node js server side application in its core uses express js for the app to listen for connections and uses the mongoose library to create schemas/models and connect to a MongoDB db/cluster. Some of the features of this application are:

- Fetching a 3rd party api with axios to get various terms and save them in the database.
- Returning terms from the database.
- Updating, creating and deleting various pharmaceutical terms.
- Registering new users, while checking if the user already exists in the database.
- Hash the password of the user using salt and bcrypt.
- Send verification/forgot password emails (with nodemailer) for the user to verify his email address and reset his password.
- Reset password and verify user with given token.
- Login the users after checking the request with the database (authentication).
- Perform validation on every route/request with Joi library.

- Throw and catch various logical and functional errors and send back the appropriate responses.
- Log errors to a file in the server.
- Log errors to a database collection named ‘logs’.
- Modularity, Separation of Concerns, DRY principles.

And many more.

Folder Structure



```
database
  actions
  models
  connection.js
helpers
logs
node_modules
routes
  api
  auth
  index.js
services
validation
  auth.js
  schemaValidate.js
  term.js
  terms.js
.env
.gitignore
app.js
example.env
LICENSE
package-lock.json
package.json
```

Database: In the models folder we can find the various schemas used to save data in the database (like user, term) and in actions we have various actions based on these schemas that require interactions with the database.

Helpers: General functions that work as help for other functionalities.

Logs: Includes an error.log file where the application writes all the error logs.

Routes: We have two big endpoints; Api and Auth. Api includes all the terms based actions like getTerms, create, update, delete terms whereas the Auth endpoint includes all actions related to authentication like login,register,confirm-reset password, verify email etc.

Services: Here we do actions like creating tokens, send the appropriate emails as well as fetch the 3rd party api to get some terms and save them into the database.

Validation: Here we have the Joi validation schemas to validate the request body/params. Also we have a set of schemaValidations that take the req, the next function and one of the schemas (when called) so it validates and returns the value for the next handle in the router. Also handles errors in validation that passes it to the next middleware (error handling/logging).

Example.env / .env: Included some hidden config vars. The structure and names of these variables are given in the example.env file in order to create the .env file. The original .env file will be provided only in the uploaded folder in canvas (and of course these values are used in Heroku).

Technologies/Libraries used

NodeJS

Express

Mongoose

Axios

Bcrypt

CORS

Joi

JWT (Json Web Token)

Morgan/Mongoose Morgan

Nodemailer

Crypto

Postman

Robo 3T

Nodemon

Dotenv

Helmet/Compression

Heroku

Links

Github Repo: https://github.com/HRSArgyropoulos/pharma-server-zach.git

Heroku App: https://pharma-server-side-zach.herokuapp.com/

HRSArgyropoulos