HOMEWORK 4

MIDDLEWARE. FRAMEWORKS

Tasks

1. In a separate directory (e.g. *http-servers*) create three files called **plain-text-server.js**, **html-server.js** and **json-server.js** respectively. Implement basic **http server** using `http` module in each of them with the following requirements:

   - For **plain-text-server.js** file:

     a. Set **Content-Type** header to `plain text`.
     b. Send “Hello World” string as a response.

   - For **html-server.js** file:

     c. Create **index.html** file with the following content:

```
<html>
    <head></head>
    <body>
        <h1>{message}</h1>
    </body>
</html>
```

     d. Set **Content-Type** header to `html`.
     e. Read (`readFileSync`) an **index.html** file with `fs` module, replace message with a real message text.
     f. Send the response.
     g. Change `readFileSync` to be a readable stream and pipe it to **response** stream.

   - For **json-server.js** file:

     h. Set **Content-Type** header to deal with JSON
     i. Take the following sample and send it as a JSON response:

```
const product = {
   id: 1,
   name: 'Supreme T-Shirt',
   brand: 'Supreme',
   price: 99.99,
   options: [
       { color: 'blue' },
       { size: 'XL' }
   ]
};
```

2. Create an echo server in **echo-server.js**. Remember that `request` and `response` are streams so you can use all their power.
3. Install `express`.
4. Create an application structure or update an existing one to fit the following:

```
├─── bin
├─── config
├─── controllers
├─── helpers
├─── middlewares
├─── models
└─── routes
├─── app.js
├─── index.js
```

5. In index file import app from **app.js**, configure port and start an app

```
// index.js

import app from './app';
const port = process.env.PORT || 8080;
app.listen(port, () => console.log(`App listening on port ${port}!`))
```

6. Create middleware for cookie parsing.
    a. Parsed cookie should be added to request stream object as `parsedCookies` field.
7. Create middleware for query parsing.
    a. Parsed query should be added to request stream object as `parsedQuery` field.

8. Make your application to respond to the following routes:

URL	METHOD	ACTION
`/api/products`	GET	Return *ALL* products
`/api/products/:id`	GET	Return *SINGLE* product
`/api/products/:id/reviews`	GET	Return *ALL* reviews for a single product
`/api/products`	POST	Add *NEW* product and return it
`/api/users`	GET	Return *ALL* users

Evaluation Criteria

1. Application structure defined and some files are created.
2. All http servers are implemented following the requirements described in tasks 1 – 2.
3. Express is installed and the main server files are created following the requirements described in tasks 3 – 5.
4. Appropriate middleware from tasks 6 – 7 are implemented properly.
5. Application responds to all routes described in task 8.