# RESTful API Service for Product Management System using Node JS, Express JS

**Problem statement:**

Create a RESTful API application for Product Management using Node JS and Express JS.

**Description:**

The RESTful API application will be used to interact with a JSON file stored at the server, containing product data.

The RESTful API application to be built, will have following functionalities.

1. Fetch all product data from a JSON file
2. Given a product id, fetch details of that single product from file
3. Given a product id, delete that product data from the file
4. Given a product id, update the product data (except product id) in that file
5. Add a complete product data in the file

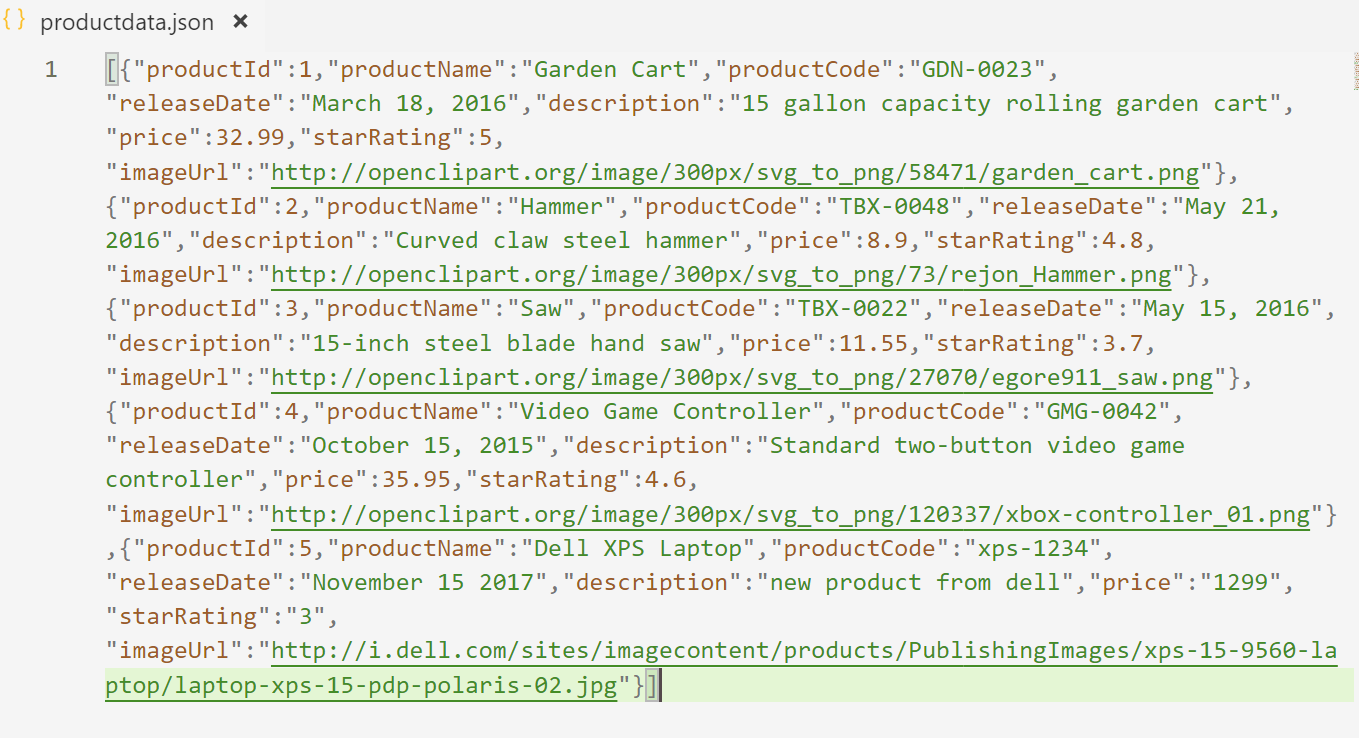
The APIs will be used by the front end application built using Angular or React or ASP.NET MVC or other applications built using different other technologies or frameworks.

**Data:**

Every product data that will be saved in the JSON file will have following characteristics. Product data will be saved in JSON format. [JSON data types are mentioned below]

1. productid: number (unique id of the product, like, 1)
2. productname: string (name of the product, like, “dell xps”)
3. productcode: string (code of the product, like, “GDN-1234”)
4. description: string (description of the product, like, “new product from dell”)
5. releasedate: string or Date (the date on which product was released to the market, like, “20-01-2018” or new Date(2018,1,25))
6. price: number (price of the product, like, 399.56)
7. rating: number (rating of the product, between 1 to 5, like, 3.5)
8. imageurl: string (URL for the image of the product. The image could be locally present in the system, like, “../images/dellxps.png” or can be a web URL, like “http://openclipart.org/image/300px/svg\_to\_png/27070/egore911\_saw.png”)

Following is a snapshot of a productdata.json file containing product data in JSON format:



**Technical Requirements:**

1. Node JS to create to the application
2. Express JS framework to create RESTful APIs
3. Any parser to parse request body for JSON data received from front end application
4. Handle cross-domain request (CORS) from the client to service
5. Any other additional packages, as you feel required for the application, should be used at your discretion (such as cookie-parser, multi-parser etc.)
6. All the methods should be tested

**Architectural Requirements:**

Three tier architecture must be used

1. Data access code to interact with JSON file, should be written in separate JS file
2. Any business logic (like sorting etc.) should be done using separate JS file, which will invoke data access layer methods
3. RESTful API service layer (created using Express JS) should be done in separate JS file and will invoke business layer methods and will receive data or pass data.

Note: If there is any file handling or data related error that should be handled at data access layer and send to RESTful API service layer.

**API Details:**

Once you complete the RESTful API application and run the same, the application must open a URL and port to receive requests and send responses. Consider the service runs on the following port: Example: http://127.0.0.1:8081/productservice.

Now,

1. **Add a product data using POST method from front end:** Front end application will send product data to the service through request body, to be saved in the JSON file.

*Request parameters:*

Request should be sent to this API as a JSON object, containing data for following characteristics (as mentioned above also)

productid, productname, productcode, description, releasedate, price, rating and imageurl

*Response parameters:*

Response should include following details as a JSON object:

{

“statusCode”: 200 (if the HTTP response is OK),

“message”: “product data added successfully”,

“status”: “success”,

“data”: [ ] (an array, containing just one product data that got added in this case)

}

Note: If there is any DB error or any other errors encounter from server, then need to handle at service layer and send to RESTful API with some customized message with statusCode and status:

For example:

statusCode: 400 (Bad Request)

status: failure

message: Input data mismatch

data: [] (blank array)

statusCode: 500 (Internal Server Error)

status: failure

message: There is some issue at server side. Please check the log.

data: [] (blank array)

statusCode: 400 (Bad Request)

status: failure

message: same product already exists (in case the product with the same id is already present in the file)

data: [] (blank array)

1. **Delete a Product using DELETE method from front end:** Front end application will send product id to the service, through URL (as URI data or may be as query string data) so that the product with the same id can be deleted from the file

Request Parameters:

productid: unique id of the product to be deleted from the file

Response Parameters:

Response should be sent back to the front end application in term of a JSON object containing the following charaterestics.

{

“statusCode”: 200, (if HTTP response code OK)

“status”: “Success”,

“message”: “product data removed successfully”,

“data”: [] (an array containing the product data that got removed from the file)

}

Note:

If there is any file error or any other errors encountered from server, then the same need to be handled at service layer and send to RESTful API with some customized message with statusCode and status.

For example:

{

“statusCode”: 400

“status”: “Failure”,

“message”: “the product doesn’t exist”, (in case the incorrect product id is sent)

“data”: [] (blank array)

}

1. **Update a Product using PUT method from front end:** Front end application will send product id along with other values for the product to the service through request body (as form data), so that the product with the same id can be updated in the file, with new set of data.

Request Parameters:

Front send application will send product data as JSON object, containing the following characterestics:

productid, productname, productcode, description, releasedate, price, rating and imageurl

productid is not updatable. It is used to search the product and then update the other details of the product.

Response Parameters:

Response should be sent back to the front end application in term of a JSON object containing the following charaterestics.

{

“statusCode”: 200, (if HTTP response code OK)

“status”: “Success”,

“message”: “product data updated successfully”,

“data”: [] (an array containing the product data after update in the file)

}

Note:

If there is any file error or any other errors encountered from server, then the same need to be handled at service layer and send to RESTful API with some customized message with statusCode and status.

For example:

{

“statusCode”: 400

“status”: “Failure”,

“message”: “the product doesn’t exist”, (in case the incorrect product id is sent, which doesn’t exist in the file)

“data”: [] (blank array)

}

1. **Fetch a Product using GET method from front end:** Front end application will send product id to the service as URI parameter or query string data through URL, so that the product with the same id can be searched in the file and returned the front send application.

Request Parameters:

Front send application will send product id as request parameter.

Response Parameters:

Response should be sent back to the front end application in term of a JSON object containing the following charaterestics.

{

“statusCode”: 200, (if HTTP response code OK)

“status”: “Success”,

“message”: “product data found”,

“data”: [] (an array containing the product data)

}

Note:

If there is any file error or any other errors encountered from server, then the same need to be handled at service layer and send to RESTful API with some customized message with statusCode and status.

For example:

{

“statusCode”: 400

“status”: “Failure”,

“message”: “the product doesn’t exist”, (in case the incorrect product id is sent, which doesn’t exist in the file)

“data”: [] (blank array)

}

1. **Fetch all Products using GET method from front end:** Front end application will just use this API to get all the product data from the file.

Request Parameters:

None

Response Parameters:

Response should be sent back to the front end application in term of a JSON object containing the following characteristics.

{

“statusCode”: 200, (if HTTP response code OK)

“status”: “Success”,

“message”: “all product data”,

“data”: [] (an array containing all the product data)

}

Note:

If there is any file error or any other errors encountered from server, then the same need to be handled at service layer and send to RESTful API with some customized message with statusCode and status.

For example:

{

“statusCode”: 400

“status”: “Failure”,

“message”: “no record found”, (in case the file is empty)

“data”: [] (blank array)

}

-----------------------------------------------------------------------------------------------------

**Project Submission Procedure:**

Submit a zip file which consists of following documents:

1. Solution folders
2. Screen capture of end to end testing using SOAP UI/PostMan tools
3. Screen capture of unit testing