Wingify: Online Store API

Design Document

Version 1.0.0

RESTFUL API for an online store which can be used to manage different products.

Sunny Karira
Software Developer
July 30, 2017

Contents

1 INTRODUCTION

Problem Statement: RESTFUL API for an online store which can be used to manage different products.

Details: Implement a RESTful API for an online store. It would support addition, deletion, editing and searching a product. It also would take care of authentication (i.e. only authenticated users can add / view / edit / delete items). The API will be used by mobile developer who eill use it to create mobile application. It would have sample scenarios along with the expected request / response objects.

1.1 WHAT IS REST

REST is a software design pattern typically used for web applications. A representation of a resource must be stateless. It is represented via some media type. Some examples of media types include **XML**, **JSON**, and **RDF**. Resources are manipulated by components. Components request and manipulate resources via a standard uniform interface. In the case of **HTTP**, this interface consists of standard HTTP ops e.g. **GET, PUT, POST, DELETE**.

REST is typically used over HTTP, primarily due to the simplicity of HTTP and its very natural mapping to RESTful principles. REST however is not tied to any specific protocol.

1.2 WHY REST

- Client-Server Communication
- Stateless
- Cacheble
- Uniform Interface

1.3 FEATURES OF THE API

- Based on raw NodeJS. No framework used.
- Uses JSON format data for requests and response.
- Two Step token generation feature.
- Non Persistant API tokens.
- Hard Validation.
- Deployment through DOCKER and HEROKU.
- Custom Testcases.

2 DESIGN FLOW

RESTFUL API for an online store which can be used to manage different products.

2.1 DESIGN

The OLSAPI is build on raw NodeJS and for database integration it uses MongoDB.
For authentication of routes it uses non persistant (destroy after certain time) JWT token.
NOTE: The **Local** API can be accessed through **http://localhost:10450**.
NOTE: The **Heroku** API can be accessed through **https://ols80.herokuapp.com**.
NOTE: The **Local** API postman links can be found **here.**
NOTE: The **Heroku** API postman links can be found **here.**

Basic Design Flow:

1. User will first signup for the API.
2. User will authenticate (two step authentication) to generate the api token using username and password.
3. On each GET, POST, PUT, DELETE request user need to send that api token in headers to access api routes.
4. User can insert products. For this they need to send JSON formatted data.
5. User can also update and delete the product details. This is required to send the particular params.
6. The above design steps maintain security.

2.2 IMPLEMENTATION DESCRIPTION

NodeJS

- Backend is implemented in NodeJS without any framework.
- application.js is the main file to be executed in cmd ot terminal.
- models/product.js and models/user.js contains codes for MongoDB interaction and middleware between routes.

Modules used in NodeJS:

- MongoDB - For database interaction.
- Mongoose - For schema validation.
- Jsonwebtoken - For token generation

2.3 INSTALLATION AND RUNNING

Locate to Directory where you have copied the source code.
Change permissions of script files using sudo chmod +x scriptname

Recommended Using Docker:

- Ubuntu 14

Run command **sudo ./scriptubuntu14.sh** to install on Ubuntu 14 systems.

- Ubuntu 16

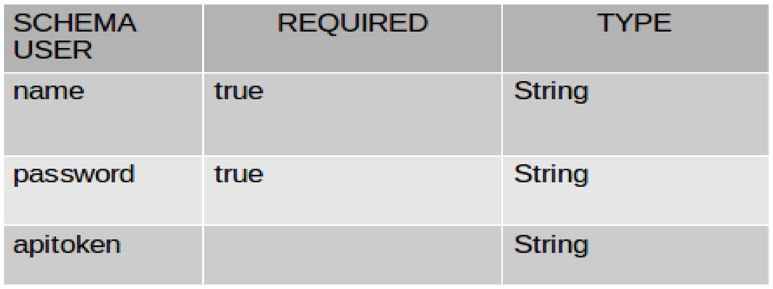
Run command **sudo ./scriptubuntu16.sh** to install on Ubuntu 16 systems.

2.4 TESTCASES

Local Testcases can be ran through command **python localtests.py** once the server is running. **Heroku Testcases** can be ran through command **python herokutests.py** once the server is running.

3 User

This is user model schema.



SCHEMA USER	REQUIRED	TYPE
name	true	String
password	true	String
apitoken		String

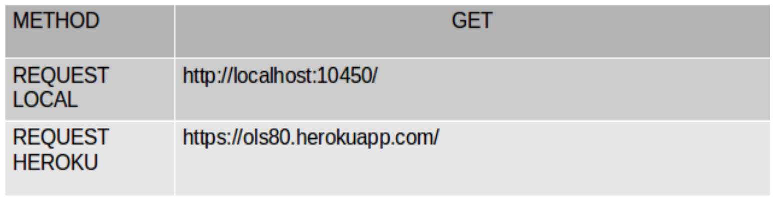
4 Product

This is product model schema.

SCHEMA PRODUCT	REQUIRED	TYPE
name	true	String
price	true	String
saleprice	false	String
category	true	String
quantity	true	Number
details		JSON object Array
description	true	String

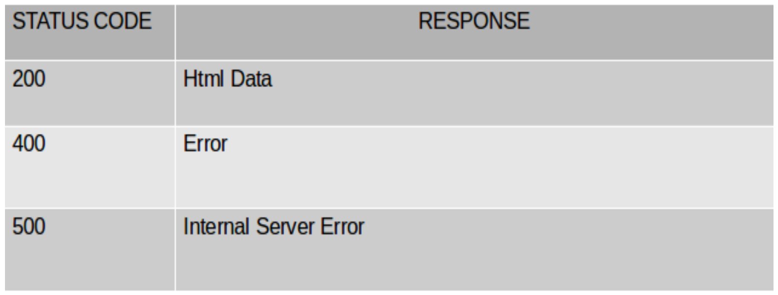
5 GET(IndexPage)

5.1 Request



METHOD	GET
REQUEST LOCAL	http://localhost:10450/
REQUEST HEROKU	https://ols80.herokuapp.com/

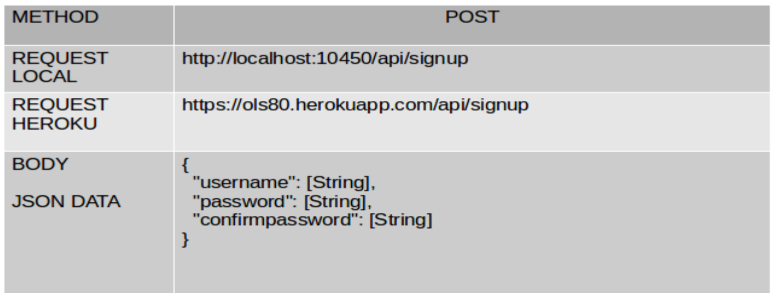
5.2 Response



STATUS CODE	RESPONSE
200	Html Data
400	Error
500	Internal Server Error

6 POST(SignUp)

6.1 Request

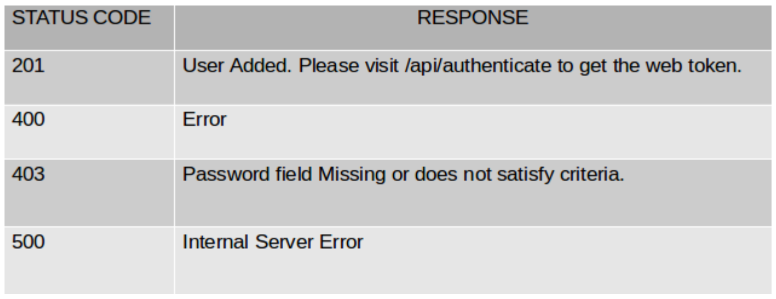


METHOD	POST
REQUEST LOCAL	http://localhost:10450/api/signup
REQUEST HEROKU	https://ols80.herokuapp.com/api/signup

BODY
 	
JSON DATA
	{
	"username": [String],
	"password": [String],
	"confirmpassword": [String]
	}

Note The body is sent as JSON data. Select Raw applicatipn/json in the request body.

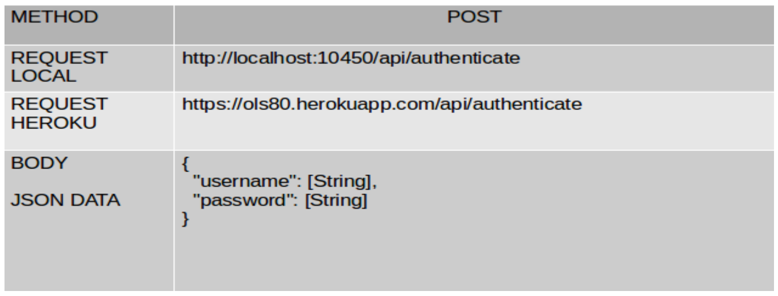
6.2 Response



STATUS CODE	RESPONSE
201	User Added. Please visit /api/authenticate to get the web token.
400	Error
403	Password field Missing or does not satisfy criteria.
500	Internal Server Error

7 POST(Authenticate)

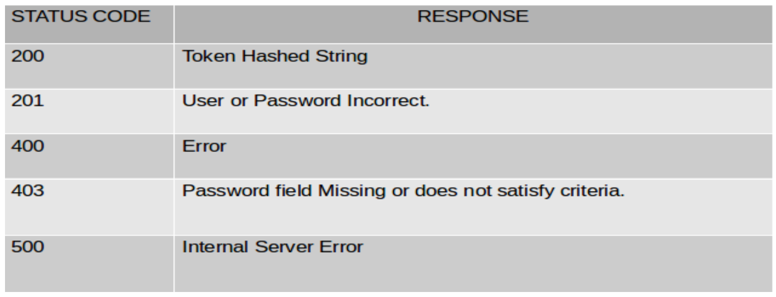
7.1 Request



METHOD	POST
REQUEST LOCAL	http://localhost:10450/api/authenticate
REQUEST HEROKU	https://ols80.herokuapp.com/api/authenticate
BODY JSON DATA	{ "username": [String], "password": [String] }

Note The body is sent as JSON data. Select Raw applicatipn/json in the request body.

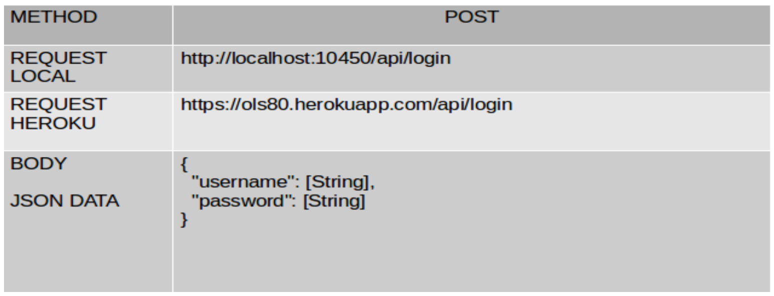
7.2 Response



STATUS CODE	RESPONSE
200	Token Hashed String
201	User or Password Incorrect.
400	Error
403	Password field Missing or does not satisfy criteria.
500	Internal Server Error

8 POST(Login)

8.1 Request

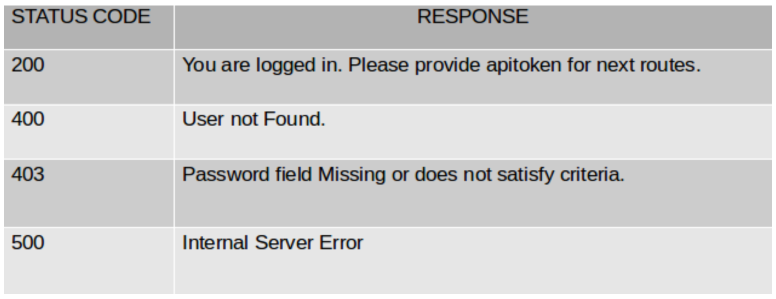


METHOD	POST
REQUEST LOCAL	http://localhost:10450/api/login
REQUEST HEROKU	https://ols80.herokuapp.com/api/login

BODY
 	
JSON DATA
	{
	  "username": [String],
	  "password": [String]
	}

Note The body is sent as JSON data. Select Raw applicatipn/json in the request body.

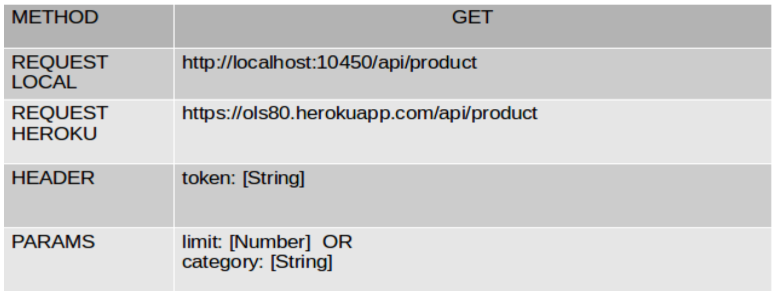
8.2 Response



STATUS CODE	RESPONSE
200	You are logged in. Please provide apitoken for next routes.
400	User not Found.
403	Password field Missing or does not satisfy criteria.
500	Internal Server Error

9 GET(Product)

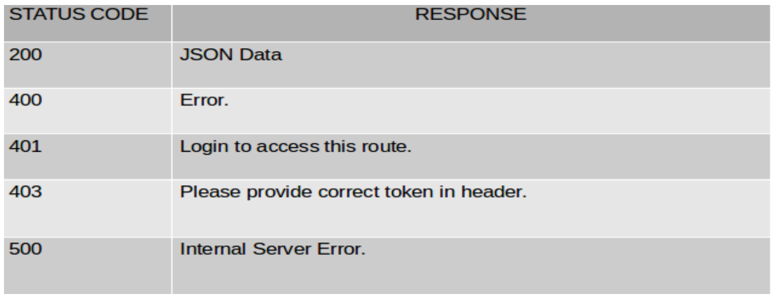
9.1 Request



METHOD	GET
REQUEST LOCAL	http://localhost:10450/api/product
REQUEST HEROKU	https://ols80.herokuapp.com/api/product
HEADER	token: [String]
PARAMS	limit: [Number] OR category: [String]

Note 1 You need to send the generated token in the headers section with the request.
Note 2 You can get product by limit or category.

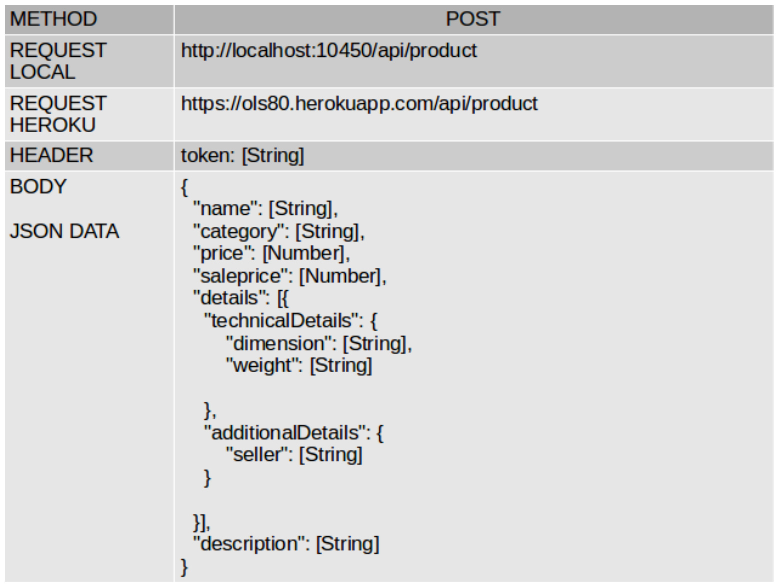
9.2 Response



STATUS CODE	RESPONSE
200	JSON Data
400	Error.
401	Login to access this route.
403	Please provide correct token in header.
500	Internal Server Error.

10 POST(Product)

10.1 Request

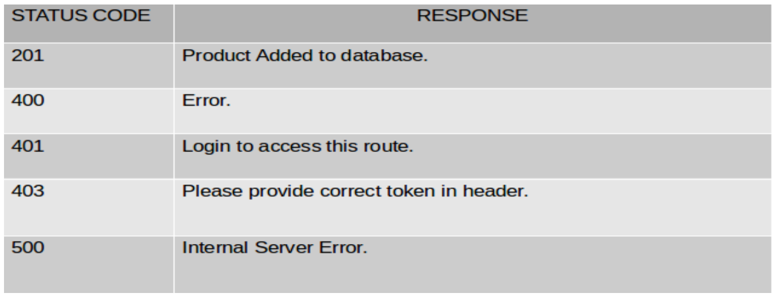


METHOD	POST
REQUEST LOCAL	http://localhost:10450/api/product
REQUEST HEROKU	https://ols80.herokuapp.com/api/product
HEADER	token: [String]

BODY
 	
JSON DATA
	{
	  "name": [String],
	  "category": [String],
	  "price": [Number],
	  "saleprice": [Number],
	  "details": [{
	    "technicalDetails": {
	        "dimension": [String],
	        "weight": [String]
	
	    },
	    "additionalDetails": {
	        "seller": [String]
	    }
	
	  }],
	  "description": [String]
	}

Note 1 You need to send the generated token in the headers section with the request.
Note 2 The body is sent as JSON data. Select Raw applicatipn/json in the request body.

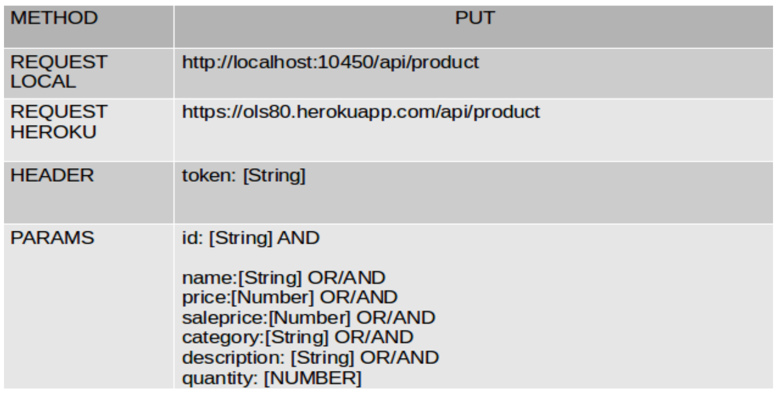
10.2 Response



STATUS CODE	RESPONSE
201	Product Added to database.
400	Error.
401	Login to access this route.
403	Please provide correct token in header.
500	Internal Server Error.

11 UPDATE(Product)

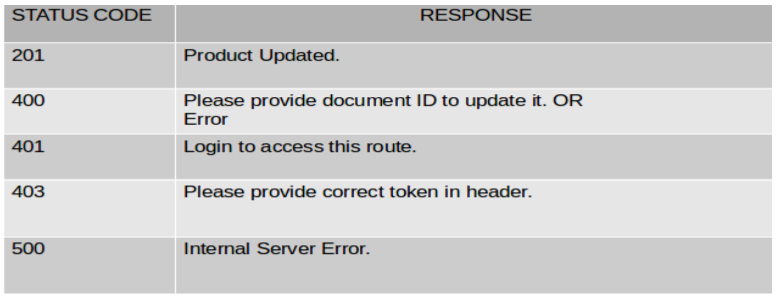
11.1 Request



METHOD	PUT
REQUEST LOCAL	http://localhost:10450/api/product
REQUEST HEROKU	https://ols80.herokuapp.com/api/product
HEADER	token: [String]
PARAMS	id: [String] AND name:[String] OR/AND price:[Number] OR/AND saleprice:[Number] OR/AND category:[String] OR/AND description: [String] OR/AND quantity: [NUMBER]

Note 1 You need to send the generated token in the headers section with the request.
Note 2 You can get product by limit or category.

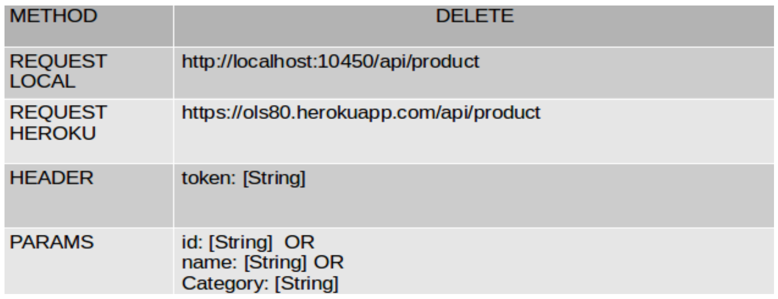
11.2 Response



STATUS CODE	RESPONSE
201	Product Updated.
400	Please provide document ID to update it. OR Error
401	Login to access this route.
403	Please provide correct token in header.
500	Internal Server Error.

12 DELETE(Product)

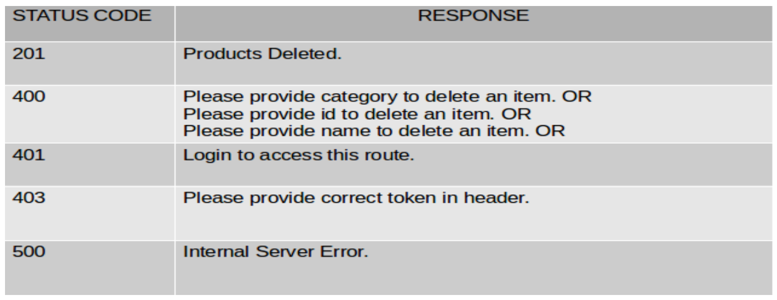
12.1 Request



METHOD	DELETE
REQUEST LOCAL	http://localhost:10450/api/product
REQUEST HEROKU	https://ols80.herokuapp.com/api/product
HEADER	token: [String]
PARAMS	id: [String] OR name: [String] OR Category: [String]

Note 1 You need to send the generated token in the headers section with the request.
Note 2 You can delete by id, name or category.

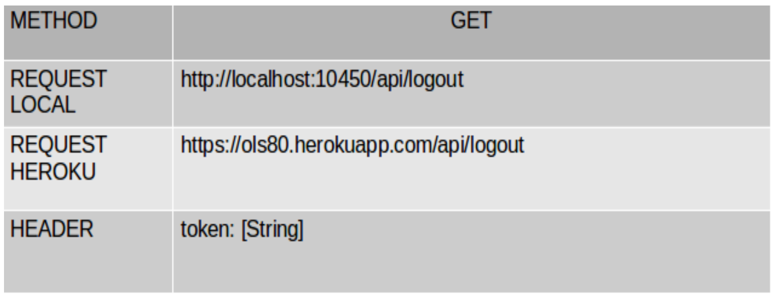
12.2 Response



STATUS CODE	RESPONSE
201	Products Deleted.
400	Please provide category to delete an item. OR Please provide id to delete an item. OR Please provide name to delete an item. OR
401	Login to access this route.
403	Please provide correct token in header.
500	Internal Server Error.

13 POST(Logout)

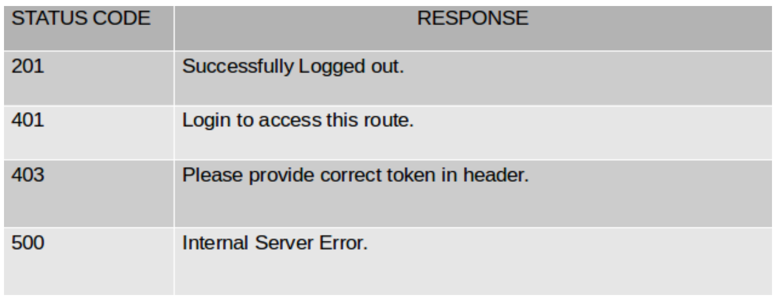
13.1 Request



METHOD	GET
REQUEST LOCAL	http://localhost:10450/api/logout
REQUEST HEROKU	https://ols80.herokuapp.com/api/logout
HEADER	token: [String]

Note 1 You need to send the generated token in the headers section with the request.
Note 2 You can delete by id, name or category.

13.2 Response



STATUS CODE	RESPONSE
201	Successfully Logged out.
401	Login to access this route.
403	Please provide correct token in header.
500	Internal Server Error.

14 Appendix

Status Codes
200 OK.
201 Created.
400 Bad Request.
401 Unauthorized.
403 Forbidden.
404 Not Found.
500 Internal Server Error.