**WingifyRestAPI Design Doc**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | 22-May-2017 | Ashwani Kumar | Development |

## Github Link :

<https://github.com/ashwinsingh2007/wingifyrestapi>

## Heroku Link :

https://wingifyrestapi.herokuapp.com/

## 

## 

## 

Index

[**1. About**](#_bit7idv2ancx)

**Design**

**Authentication**

**Installation**

[2.](#_bit7idv2ancx) Signup

[Request](#_kd6tjl5w5wjo)

[Response](#_8b7ijrpuamb8)

3. POST (Insert product data)

[Request](#_38pypefkb82)

[Response](#_47awg09ehquu)

[4. GET](#_9mc75gzfq3m3) (get all product with/without filteration)

[Request](#_xtszbus9yomc)

[Response](#_xuhh3y4ufz7)

5. PUT (Update Product data)

[Request](#_iaps2dh7l2m2)

[Response](#_fvf6lulvf6wv)

6. DELETE (Delete Product data)

[Request](#_iaps2dh7l2m2)

[Response](#_fvf6lulvf6wv)

Conventions

[Status Codes](#_of6ke88xwee4)

## 

## About

WingifyrestAPI is RESTful API for an online store which can be used to manage different products.

## Design :

* + This Restful API is build using NodeJS and uses MySQL for database interaction.
  + It uses simple basic authentication and only authenticated user can **create, update, delete** the products details
  + Data Flow :

1. User will first signup for the API. See below document for signup process.
2. On each and every POST,PUT,DELETE, he needs to send the

**Basic Authentication Key** generated using his signup username and password.

1. For inserting product details he needs to request a URL,authentication key in header and JSON formatted data in Body. See below section for more details.
2. He can further Update and Delete the product details which are only created by him. See below section for update and delete operation.
3. In this way security is maintained.
4. Update of product details can be proceeded by PUT method and request

a URL as mentioned in below sections.

1. Similarly Deletion can be done but he can only delete those data which are created by him.

## Implementation Description :

* NodeJS :
  + Backend is implemented in NodeJS without using any Framework.
  + main.js is main file to be executed in cmd or terminal.
  + db.js file contains codes for interaction with MySQL.
  + dbOperation.js file is a middleware file in between requests and MySQL interaction.
* Modules used in NodeJS:
  + Mysql – For MySQL interaction.

## Installation :

wingifyrestapi can be installed in nodejs environment. Config the app by following guidelines below :

1. Config the db.js file in Compressed file inside the Wingify folder ,

the credentials for your MySQL.

1. Import the wingify.sql in your Mysql as provided inside

a dump file inside compressed Wingify/mydb folder..

3. Remember Database name should be "Wingify" and

two tables name should

be "StubUser" and "ITEMS" respectively.

4. Download all dependencies by "npm install" command.

5. Start the server by running command "npm start" or “node main.js”.

## 

## 

Methods

## 2. Signup

To allow user to sign up for the API uses and generation of basic authentication key.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | https://wingifyrestapi.herokuapp.com/signup |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEAD  POST  POST | api\_key  username  password | string  string  string |

**api\_key**

api\_key must be sent with all client requests. The api\_key helps the server to validate the request source.

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | Signup Successfully ! |
| 404 | Invalid Request :( |
| 500 | Internal Server failure :( |
| 403 | Login Failure. May be invalid username or password |
| 500 | Oops! Something went wrong. Check your request :( |

## 3. POST (Insert product data)

Used to insert product data only by registered users or authorized users.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | https://wingifyrestapi.herokuapp.com/online/products |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEAD  BODY | auth\_key(basic authentication key build from username and password. Use PostMan for this)  {"Id":"[STRING]","Name":"[STRING]","Price":[NUMBER],"Description":"[STRING]","Quantity":[NUMBER]}  (JSON Format) | String  JSON |

**auth\_key**

The basic auth\_key which is created by username and password.

**\*Note1** : The API may give you Invalid parameter or Wrong JSON Format in case of wrong URL request or wrong JSON respectively.

**\*Note2** : Only the authorized person with the username and password will be able to create data. Also He can update/delete the product data's which is only created by him

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **Response will be an JSON object containing the list of recipes (array)**  **as well as the updated recipe database. Each item in the recipe array has the following structure.**  Successfully Inserted !! |
| 404 | Invalid Request :( |
| 403 | Wrong JSON Format !! OR Item with ID is already Present!! |
| 403 | Authorization Failure |
| 500 | Internal Server failure :( |

## 4. GET (get all product with/without filtration).

Get the product information and details. There are two versions of this method provided.

The one used by all end users without authentication.

The other is used by authenticated user to get the product data which are created by them.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | https://wingifyrestapi.herokuapp.com/online/products?[options] |
| **URL Params** | 1. "?limit=[NUMBER]" :- Will get specified number of product data. 2. "?where=[STRING]/[NUMBER]" :- Will get product data by filter criteria like "?where=Quantity>10" or "?where=Name='Rackets'" etc. 3. "?limit=[NUMBER]&where=[STRING]/[NUMBER]" : - Will have functionality for both above two points. 4. "?where='\*'" :- Will filter product data which are created or inserted by the current user.(Pass Authentication{username, password} for this). 5. "?limit=[NUMBER]&where=\*" - Will filter product data which are created or inserted by the current user also limits the amount of data to be rendered.(Pass Authentication{username, password} for this). |

**[options] :**

1. "?limit=[NUMBER]" :- Will get specified number of product data.
2. "?where=[STRING]/[NUMBER]" :- Will get product data by filter criteria like "?where=Quantity>10" or "?where=Name='Rackets'" etc.
3. "?limit=[NUMBER]&where=[STRING]/[NUMBER]" : - Will have functionality for both above two points.
4. "?where='\*'" :- Will filter product data which are created or inserted by the current user.(Pass Authentication{username, password} for this).
5. "?limit=[NUMBER]&where=\*" - Will filter product data which are created or inserted by the current user also limits the amount of data to be rendered.(Pass Authentication{username, password} for this).

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEAD | auth\_key(basic authentication key build from username and password. Use PostMan for this)  (required only when "https://wingifyrestapi.herokuapp.com/online  /products?where=\*" is requested) | String |

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **A json containing the Product details.**  **The JSON will depend upon the filter criteria or limit requested**  **by user.Also authenticated user can have filter as “ where=’\*’ ” for getting JSON for only those product that he has created.**  Example response:-  [{"Id":"1","Name":"Tennis Ball","Price":2000,"Description":"N/A","Quantity":7,"username":""},{"Id":"200","Name":"Racket","Price":4000,"Description":"N/A","Quantity":10,"username":""}] |

|  |  |
| --- | --- |
| 404 | Invalid Request :( |
| 403 | Authorization Failure |
| 500 | Internal Server failure :( |

## 5. PUT (Update Product data)

This request will update that item/products details which is only created by the current authenticated user.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **PUT** | https://wingifyrestapi.herokuapp.com/online/products |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEAD  BODY | auth\_key(basic authentication key build from username and password. Use PostMan for this)  JSON Format - {"Id":"[STRING]","Name":"[STRING]",  "Price":[NUMBER],"Description":"[STRING]",  "Quantity":[NUMBER]}  (Product details to be updated.) | String  JSON |

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | Item Updated succesfully!! |

|  |  |
| --- | --- |
| 404 | Invalid Request :( |
| 403 | Wrong JSON Format !! OR Item with ID is already Present!! |
| 403 | Authorization Failure |
| 500 | Internal Server failure :( |

\*Note1 : The API may give you Invalid parameter or Wrong JSON Format in case of wrong url request or wrong json respectively.

\*Note2 : Incase if you the admin wants to update only on column, his body JSON must contain the updated value for the key and rest previous values for the key.

\*Note3 : Items or Products can only be updated by there Id.

\*Note4 : Only the user created/inserted the product data will be able to update it.

\*LIMITATION : Id of the Products cannot be updated.

## 5. DELETE (Product data by Id)

This request will delete that item/products details which is only created by the current authenticated user.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **DELETE** | <https://wingifyrestapi.herokuapp.com/online/products>  /delete?Id=[STRING] |
| **URL Param** | ?Id=[STRING]  Example :  <https://wingifyrestapi.herokuapp.com/online/products>  /delete?Id=”120” |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEAD | auth\_key(basic authentication key build from username and password. Use PostMan for this) | String |

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | Item deleted succesfully !! |

|  |  |
| --- | --- |
| 404 | Invalid Request :( |
| 403 | Wrong JSON Format !! OR Item with ID is already Present!! |
| 403 | Authorization Failure |
| 500 | Internal Server failure :( |

**\*Note1** : Only the user created/inserted the product data will be able to delete it.

Glossary

## Conventions

* **Client** - Client application.
* **Status** - HTTP status code of response.
* All the possible responses are listed under ‘Responses’ for each method. Only one of them is issued per request server.
* All response are in JSON format. Except Error messages.
* All request parameters are mandatory unless explicitly marked as [optional]

## Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

2XX - Success of some kind

4XX - Error occurred in client’s part

5XX - Error occurred in server’s part

|  |  |
| --- | --- |
| **Status Code** | **Description** |
| 200 | OK |
| 201 | Created |
| 202 | Accepted (Request accepted, and queued for execution) |
| 400 | Bad request |
| 401 | Authentication failure |
| 403 | Forbidden |
| 404 | Resource not found |
| 405 | Method Not Allowed |
| 409 | Conflict |
| 412 | Precondition Failed |
| 413 | Request Entity Too Large |
| 500 | Internal Server Error |
| 501 | Not Implemented |
| 503 | Service Unavailable |