Sartar Backend Program

**Nodejs version 10**

METHOD TYPES **GET**, **POST**, **PUT**, **DELETE**

|  |  |
| --- | --- |
| **Method Types** | **Description** |
| GET | Show Data |
| POST | Insert Data |
| PUT | Update Data |
| DELETE | Remove Data |

Database Used is MongoDB

In order to access the Rest API use the URL or Base URL => <https://YOUR_URL_ADDRESS/api/>   
FIRST time installation

Go to your server and locate your backend API folder, once you locate it the next to do is install the dependencies.   
  
To install the dependencies type in the command (***npm install***)

Once the installation is finished type in the command pm2 start

If you have not installed the pm2. Please install the pm2 dependencies in order for your backend NODEJS to run smoothly. ( npm install pm2 -g

For more information on how to install the pm2 please refer to the url [pm2 - npm (npmjs.com)](https://www.npmjs.com/package/pm2)

**Users API**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | API URL | Parameter | description |
| **POST** | /api/users/signup | { username, password, full\_name, uers\_type } | User\_type is role. For mobile app dev always set the role to ‘customers’ |
| **POST** | /api/users/login | { username: string,  password: string  } | Authenticate users and generate token to securely login |
| **GET** | /api/users/logout | - | Token will be destroyed |
| **GET** | /api/users/:id |  | Shows specific users |
| **PUT** | /api/users/:id | - | Allows to update the user profile |
| **PUT** | /api/users/password/update | { old\_password, new\_password } | Updates the password |
| **PUT** | /api/uses/password/forgot\_password | { email: email\_address } | Email sent to the user for password reset |

Customer API (Mobile APP USERS)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | API URL | Role | PARAMETER | DESCRIPTION |
| **GET** | /api/customers/ | ADMIN |  | Shows all users for the admin |
| **GET** | /api/customers/search | ADMIN |  |  |
| **GET** | /api/customers/bought\_items | CUSTOMER |  | Shows lists of item bought by the logged in user |
| **GET** | /api/customers/carts/:id | Customer and Admin |  | Showing lists of cart items made by the customers logged in the mobile app |
| **GET** | /api/customers/favorite |  |  | Shows lists of favorite |
| **GET** | /api/customers/:id | ADMIN |  | View specific customers by the admin |
| **PUT** | /api/customers/profileUpdate | Customer | - | Customer update Profile |
| **PUT** | /api/customers/profilePicture | Customer | File upload | Upload Profile Picture |
| **PUT** | /api/customers/:id | ADMIN |  | Update customers information |
| **DELETE** | /api/customers/:id | ADMIN | Id | Admin will delete the customers info |
| **DELETE** | /api/customers/profilePic | Customer |  | Remove the profile picture |

BLOG API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | Parameters | Descriptions |
| **GET** | /api/blog | ALL |  | View all blogs |
| **GET** | /api/blog/view | Salers |  | Salers views its own blog posts |
| **GET** | /api/blog/:id | All | Blog id | Specific blog profile |
| **POST** | /api/blog | Salers,admin | { title, story, and file uploads} | Creating blog |
| **POST** | /api/blog/search | Salers,customers,admin | { title: String, story: String } | Search blog from title and story |
| **PUT** | /api/blog/:id | Salers,admin | Id | update title or story |
| **DELETE** | /api/blog/:id | Salers,admin | Id | REMOVE BLOG |

Brands API ENDPOINT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | Parameters | Descriptions |
| **GET** | /api/brands/ | All roles |  | View all brands |
| **GET** | /api/brands/top | All roles | {is\_top: true } | Shows all brand with the flag is\_top is true |
| **GET** | /api/brands/products/:id | All roles |  | View all products under the specific brand category |
| **GET** | /api/brands/sales/:id | All roles |  | View all products that are sold. This endpoints helpers to count how many products under a specific brand name has been sold |
| **GET** | /api/brands/follow/:id | Customers |  | Showing lists of followers for a specific brand |
| **GET** | /api/brands/likes/:id | Customers |  | Showing lists of likes for a specific brand |
| **GET** | /api/brands/:id | All roles |  | View speicif brand |
| **GET** | /api/messages/:id | Customers, salers |  | Shows lists of communication made by customers and specific brand salers |
| **POST** | /api/brands | Salers,super\_admin | { title } | Create brand by passing a parameter title |
| **POST** | /api/brands/search | All customers | { title } | Searching the title of the brand |
| **PUT** | /api/brands/:id | Super\_admin, admin | { title: [string] | Update the brand |
| **DELETE** | /api/brands/:id | Super\_admin, admin | Id | Remove brands |

Carts/bags API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTIONS |
| **GET** | /api/carts | Super\_admin, admin |  | Showing all lists of bags |
| **GET** | /api/carts/lists | Customers |  | Showing all lists of items in the bag for the logged in customers |
| **GET** | /api/carts/:id | Customers, admin |  | Showing specific item in the bag |
| **POST** | /api/carts | Customers | { product: \_id } | Storing item in the bag |
| **PUT** | /api/carts/:id | Super\_admin | {id: \_id } | Updating a specific cart item |
| **DELETE** | /api/carts/:id | Customers, super\_admin |  | REMOVE specific item in thebag |

CATEGORY API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PATAMETERS | DESCRIPTION |
| **GET** | /api/categories | All roels |  | View all categories |
| **GET** | /api/categories/:id | super\_admin | {\_id: passed id } | View specific categories |
| **POST** | /api/categories/create | Admin, super\_admin | { name: entered\_string } | Create categories |
| **PUT** | /api/categories/:id | Admin, super\_admin | { name: entered\_string } | Update category specific |
| **DELETE** | /api/categories/:id | Super\_admin, admin | { id: entered\_id } | Remove specific category from the database |

CHECKOUT API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PATAMETERS | DESCRIPTION |
| **GET** | /api/checkout | Super\_admin, admin |  | Show all checkout or bought transactions |
| **GET** | /api/checkout/lists | Customers |  | Showing all checkout for the logged in customer only |
| **GET** | /api/checkout/:id | Customer, superadmin | {\_id: passed\_para } | Showing specific transaction |
| **POST** | /api/checkout | Customer | {  Amount, billing,  Mobile, country, state, zipcode, cardname, cardnumber, cardExpirmont, cardExpirYear  cvv | This will create checkout order. |
| **PUT** | /api/checkout/:id | Customer |  | Update the checkout but it is not necessary to use this checkout. |
| **DELETE** | /api/checkout/:id | Super\_admin |  | It will remove this checkout |

Favorite API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PATAMETERS | DESCRIPTION |
| **GET** | /api/favorite | Super\_admin, amdin |  | Shows all favorite lists |
| **GET** | /api/favorite/lists | Customers |  | Shows all lists of favorite made by the customers |
| **GET** | /api/favorite/:id | Customers, superadmin |  | Shows specific favorite |
| **POST** | /api/favorite | Customers | { product: String } | Create or removes product from personal favorite fields |
| **PUT** | /api/favorite | Super\_admin |  | Update the favorite item lists (not necessary to update) |
| **DELETE** | /apu/favorite/:id | Super\_admin |  | Remove item from favorite lists |

Internal Users meaning admin super\_admins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PATAMETERS | DESCRIPTION |
| **GET** | /api/internals | Super admin |  | Show all internal users |
| **GET** | /api/internals/:id | Super\_amdin |  | Shows specific internal users |
| **PUT** | /api/internals/profilepic | Super\_admin, admin | Filename | Update the profile picture of the user |
| **PUT** | /api/internals/:idi | Superadmin |  | Update the string data of the internal users |
| **DELETE** | /api/internals/:id | Super\_admin, admin |  | Remove specific user |

Messages API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PATAMETERS | DESCRIPTION |
| **GET** | /api/message | Super admin |  | Show all messages |
| **GET** | /api/message/lists | Customers |  | Shows the message by the users |
| **GET** | /api/message/:id | Super\_admin, admin | Filename | Specific message |
| **POST** | /api/message/ | Customers, salers | { message: type: string } | Add message |
| **DELETE** | /api/message/:id | Super\_admin, admin |  | Remove specific message |

Notifications API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTION |
| **GET** | */api/notification* | Super admin |  | Show all Notification |
| **GET** | */api/notification/lists* | Customers |  | Shows the message by the users |
| **GET** | */api/notification/customers/unread/open* | Super\_admin, admin |  | Update the unread to read |
| **GET** | */api/notification/customer/unread* | Customers, salers |  | Shows lists of unread messages |
| **GET** | */api/notification/customer/read* |  |  | Shows lists of read notification |
| **POST** | */api/notification/* | Super\_admin, admin | { title: string, message: string } | Create notification |
| **PUT** | */api/notitication/:id* | Super\_admin | { title: string, message: string } | Update |
| **DELETE** | */api/notification/:id* | Super\_admin |  | Remove the notfication |

Product api

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTION |
| **GET** | /api/product | ALL ROLES |  | View all products |
| **GET** | /api/product/view | Salers |  | Salers own product entries |
| **GET** | /api/product/:id | All roles | {\_id: id } | View specific product |
| **POST** | /api/product | Salers | { name, store, price, category, brands } and 7 images with one video. Upload files. Basically there will be 8 defaults | Enter products. |
| **POST** | /api/product/search | All roles | { name: search query } | Search and display the result based on the product’s name or store or price |
| **PUT** | /api/product/:id | Salers |  | Update products |
| **DELETE** | /api/product/:id | Salers |  | Remove products |

Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTION |
| **GET** | /api/reviews | All roles |  | Views all reviews |
| **GET** | /api/reviews/lists | Customers |  | Shows all reviews of logged in customers |
| **GET** | /api/reviews/:id | Admin, customers |  | Specific reviews |
| **POST** | /api/reviews | Customers | { product, review, rate and uploaded image } | Create reviews |
| **POST** | /api/reviews/search | super\_admin, admin, salers | { review: search parameters } | Search result of all reviews |
| **PUT** | /api/reviews/:id | Customers | { product, review, rate } | Update reviews |
| **DELETE** | /api/reviews/:id | Customers |  | Remove reviews |

Salers api

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTION |
| **GET** | /api/salers/ | Super\_admin |  | View all salers |
| **GET** | /api/salers/profile | Salers |  | View own profile for the logged in salers |
| **GET** | /api/salers/:id | Super\_admin |  | View specific salers |
| **PUT** | /api/salers/:id | Salers |  | Update own profile of strings |
| **PUT** | /api/salers/updateProfile  /:id | Salers |  | Upload profile images |
| **DELETE** | /api/salers/:id | Super\_admin |  | Remove saler |

Wish lists api

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD | URL | ROLE | PARAMETERS | DESCRIPTION |
| **GET** | /api/whishlists/ | Super\_admin |  | Vew all wishlists |
| **GET** | /api/wishlists/lists | Customers |  | View all wishlists made by the customer |
| **GET** | /api/wishlists/:id | Customers, admin |  | Specific wishlists |
| **POST** | /api/wishlists | Customers | { product: string\_name } | Add to wishlists by the customers |
| **PUT** | /api/wishlists/:id | Customers | {product: pass the id } | Update the wishlists |
| **DELETE** | /api/wishlists/:id | Customers |  | Remove from wishlists |

Other Information

1. Stripe has been integrated on the backend, every checkouts will pass through the stripe API, your Stripe dashboard will display every transaction made by your platform customers