



# Ecommerce & Banking Rest API

2017331088 (Roman Raihan)

2017331010 (M. Ebrahim Sazin)

**Github:**

<https://github.com/CaptainMissile/ecommerce-rest-api>

## Overview

<http://localhost:8000>

( This root url is for API Documentation & API Testing )

### Ecommerce & Banking API alpha

[ Base URL: localhost:8000/api ]  
<http://localhost:8000/?format=openapi>

Build Your Test Ecommerce and Banking Frontend Using our API  
[Contact the developer](#)

Schemes  
HTTP

Django LoginAuthorize

Filter by tag

auth

POST

/auth/login/

auth\_login\_create

PATCH

/auth/password-reset-complete/

auth\_password-reset-complete\_partial\_update

GET

/auth/password-reset-confirm/{uidb64}/{token}/

auth\_password-reset-confirm\_read

POST

/auth/password-reset-req/

auth\_password-reset-req\_create

PATCH

/auth/profile-update/{username}

auth\_profile-update\_partial\_update

GET

/auth/profile/{username}

auth\_profile\_read

POST

/auth/register/

auth\_register\_create

GET

/auth/verify-email/

auth\_verify-email\_list

bank

POST

/bank/add-money-request/

bank\_add-money-request\_create

GET

/bank/approve-cash-in-request/{transaction\_id}

bank\_approve-cash-in-request\_read

GET

/bank/approve-send-money-request/{transaction\_id}

bank\_approve-send-money-request\_read

POST

/bank/create-account/

bank\_create-account\_create

DELETE

/bank/delete-account/{account\_no}

bank\_delete-account\_delete

GET

/bank/filter-transaction/

bank\_filter-transaction\_list

GET

/bank/list-account/

bank\_list-account\_list

GET

/bank/list-transaction/

bank\_list-transaction\_list

POST

/bank/send-money-request/

bank\_send-money-request\_create

GET

/bank/single-account/{account\_no}

bank\_single-account\_read

GET

/bank/single-transaction/{transaction\_id}

bank\_single-transaction\_read

PATCH

/bank/update-account/{account\_no}

bank\_update-account\_update

cart

POST

/cart/add-update-to-cart/

cart\_add-update-to-cart\_create

DELETE

/cart/delete-cart/

cart\_delete-cart\_delete

DELETE

/cart/delete-cart/{id}

cart\_delete-cart\_delete

GET

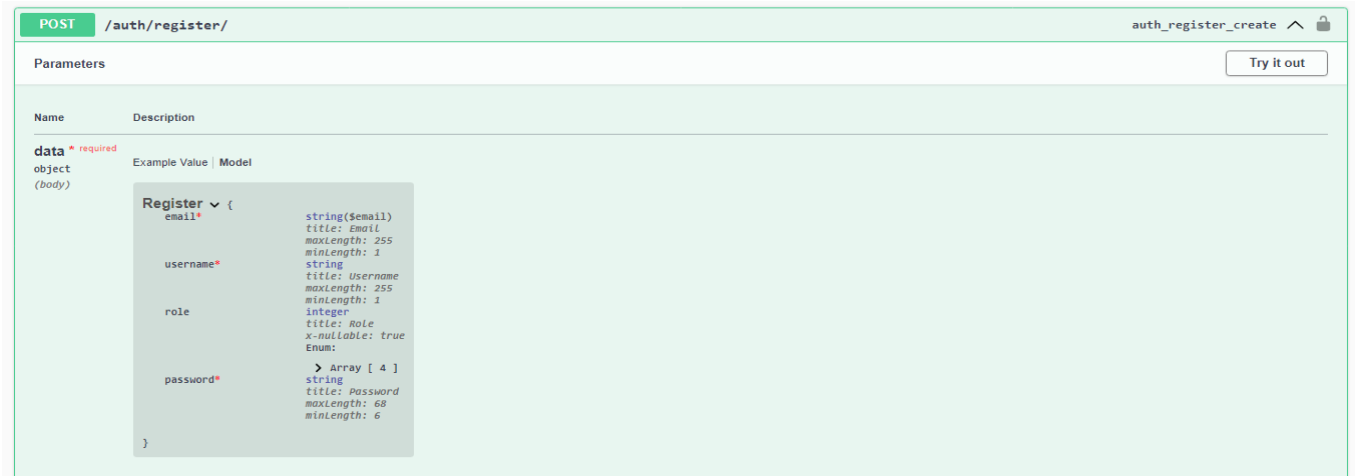
/cart/list-cart/

cart\_list-cart\_list

## API Description In Terms of Requirements:

### 1. User first registers and then logs in to the e-commerce website.

1. To Register an Account: <http://localhost:8000/api/auth/register/>



POST /auth/register/ auth\_register\_create

Parameters Try it out

Name	Description
data* required object (body)	<p>Example Value   Model</p> <pre> Register {   email* string(\$email)     title: Email     maxLength: 255     minLength: 1   username* string     title: Username     maxLength: 255     minLength: 1   role integer     title: Role     x-nullable: true     Enum:       &gt; Array [ 4 ]   password* string     title: Password     maxLength: 68     minLength: 6 }</pre>

**Explanation:** email, username and password fields are self explanatory. There are 4 types of user based on their role.

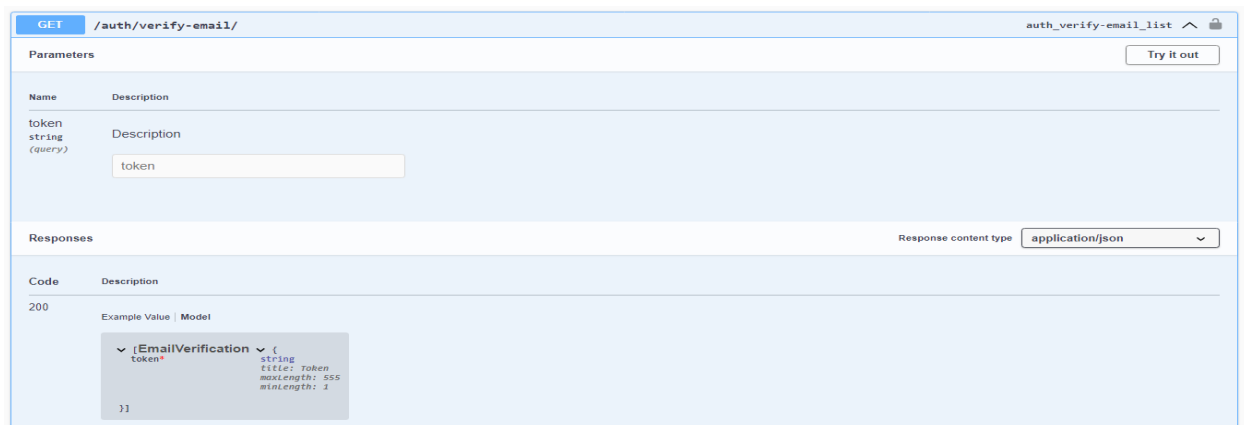
role : 1 implies Bank Manager type user.

role : 2 implies Ecommerce Admin type user.

role : 3 implies Seller type user.

role : 4 implies Customer type user.

2. To Verify Email The system will send back a token:



GET /auth/verify-email/ auth\_verify\_email\_list

Parameters Try it out

Name	Description
token string (query)	<p>Description</p> <input type="text" value="token"/>

Responses Response content type: application/json

Code	Description
200	<p>Example Value   Model</p> <pre> {EmailVerification {   token* string     title: Token     maxLength: 555     minLength: 1 }}</pre>

3. To Login: <http://localhost:8000/api/auth/login/>

**Example:**

POST /auth/login/ auth\_login\_create

Parameters Try it out

Name	Description
<b>data</b> * required	Example Value   Model
object (body)	<pre> Login {   email*   password* }</pre> <div> <div>string(\$email)</div> <div>title: Email</div> <div>maxLength: 255</div> <div>minLength: 1</div> </div> <div> <div>string</div> <div>title: Password</div> <div>maxLength: 128</div> <div>minLength: 1</div> </div>

**Response:** If both email and password are valid, response will be sent back with access token and refresh token. For Every 'login required' request the user has to send an 'access token' in header as a proof of authenticated request.

Other than these, 'password reset request', 'password reset complete', 'set new password' are implemented.

## 2. Set up his/her bank account and add a secret

1. To Create an Bank Account and Add credential: <http://localhost:8000/api/bank/create-account/>

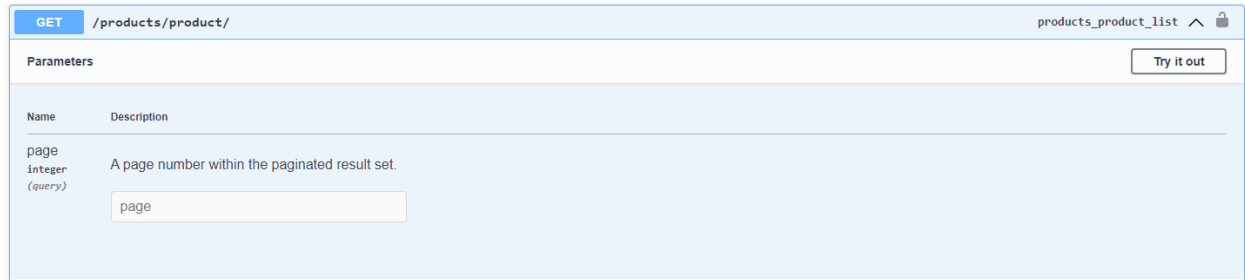
POST /bank/create-account/ bank\_create-account\_create

Parameters Try it out

Name	Description
<b>data</b> * required	Example Value   Model
object (body)	<pre> BankAccountCreateSerializer_ {   credential* }</pre> <div> <div>string</div> <div>title: Credential</div> <div>maxLength: 100</div> <div>minLength: 1</div> </div>

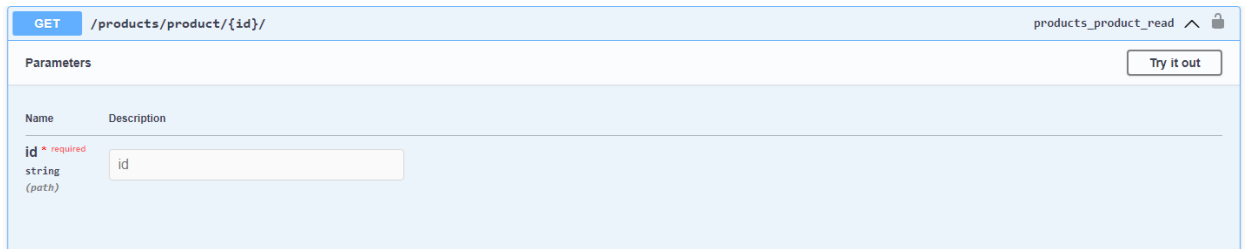
## 3 & 4. View and Add to Cart products

1. To View Product List: <http://localhost:8000/api/products/product/>



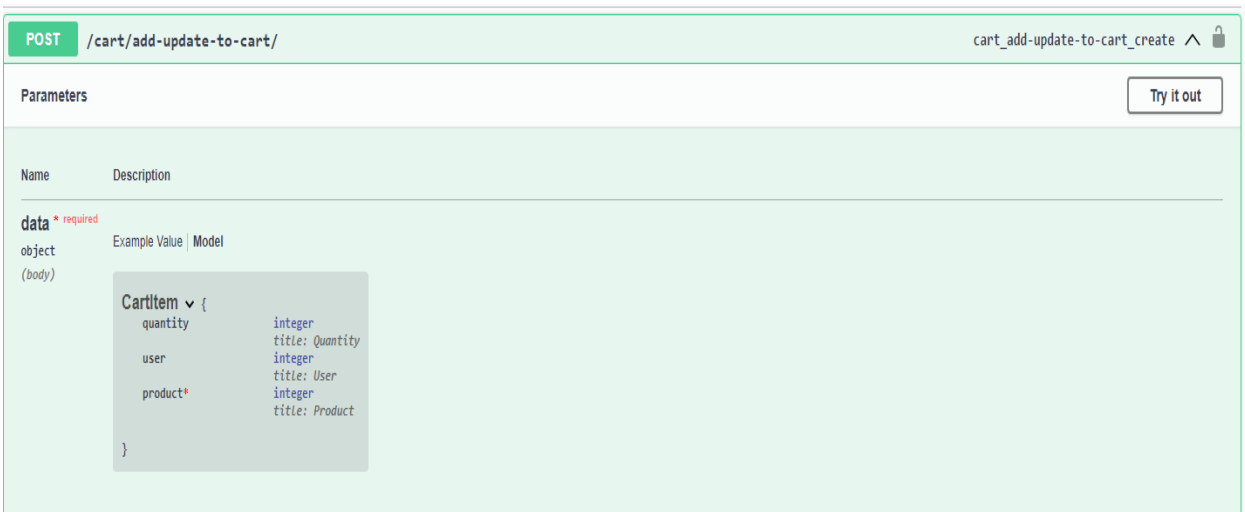
Swagger UI for GET /products/product/. The interface shows a table with columns 'Name' and 'Description'. The 'Name' column contains 'page' with type 'integer' and '(query)'. The 'Description' column contains 'A page number within the paginated result set.' There is a text input field with the value 'page'.

2. To View Single Product : <http://localhost:8000/api/products/product/{id}>



Swagger UI for GET /products/product/{id}/. The interface shows a table with columns 'Name' and 'Description'. The 'Name' column contains 'id' with type 'string' and '(path)'. The 'Description' column contains 'id'.

3. Add to Cart : <http://localhost:8000/api/cart/add-update-to-cart/>



Swagger UI for POST /cart/add-update-to-cart/. The interface shows a table with columns 'Name' and 'Description'. The 'Name' column contains 'data' with type 'object' and '(body)'. The 'Description' column contains 'Example Value | Model'. The 'Model' column shows a JSON object: 

```
CartItem {
  quantity: integer (title: Quantity)
  user: integer (title: User)
  product*: integer (title: Product)
}
```

## 5. Amount to buy them is calculated and a transaction request is sent

1. **Place The Order** : <http://localhost:8000/api/order/place-the-order/> (It will collection product quantity from cart and request for an order)

POST /orders/place-the-order/ orders\_place-the-order\_create

Parameters Try it out

Name	Description
<b>data</b> * required	
object	Example Value   Model
(body)	<pre>PlaceOrder {   full_name* string     title: Full name     maxLength: 50     minLength: 1   address* string     title: Address     maxLength: 250     minLength: 1   city* string     title: City     maxLength: 100     minLength: 1   phone* string     title: Phone     maxLength: 100     minLength: 1   post_code* string     title: Post code     maxLength: 20     minLength: 1 }</pre>

2. **Payment Gateway** : <http://localhost:8000/api/payments/make-payment/>

POST /payments/make-payment/ payments\_make-payment\_create

Parameters Try it out

Name	Description
<b>data</b> * required	
object	Example Value   Model
(body)	<pre>MakePayment {   order_id* integer     title: Order id     string(\$decimal)   amount* string     title: Amount   credential* string     title: Credential     minLength: 1 }</pre>

3. **Send Money Request**: <http://localhost:8000/api/bank/send-money-request/>

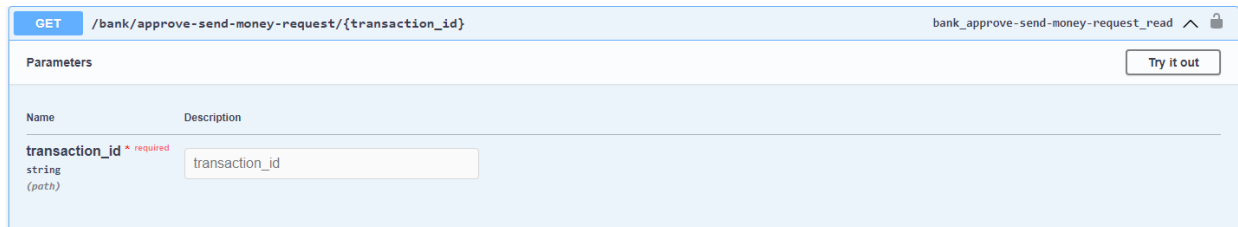
POST /bank/send-money-request/ bank\_send-money-request\_create

Parameters Try it out

Name	Description
<b>data</b> * required	
object	Example Value   Model
(body)	<pre>SendMoneyRequestSerializer_ {   account_from integer     title: Account from     x-multiple: true   credential string     title: Credential     maxLength: 100     x-multiple: true   account_to* integer     title: Account to     string(\$decimal)   amount* string     title: Amount }</pre>

## 6&7. bank settles the transaction and returns a tranxId and transfer amount to the supplier's account.

1. Approve Send Money Request: <http://localhost:8000/api/bank/approve-send-money-request/{tranxID}>



GET /bank/approve-send-money-request/{transaction\_id} bank\_approve-send-money-request\_read

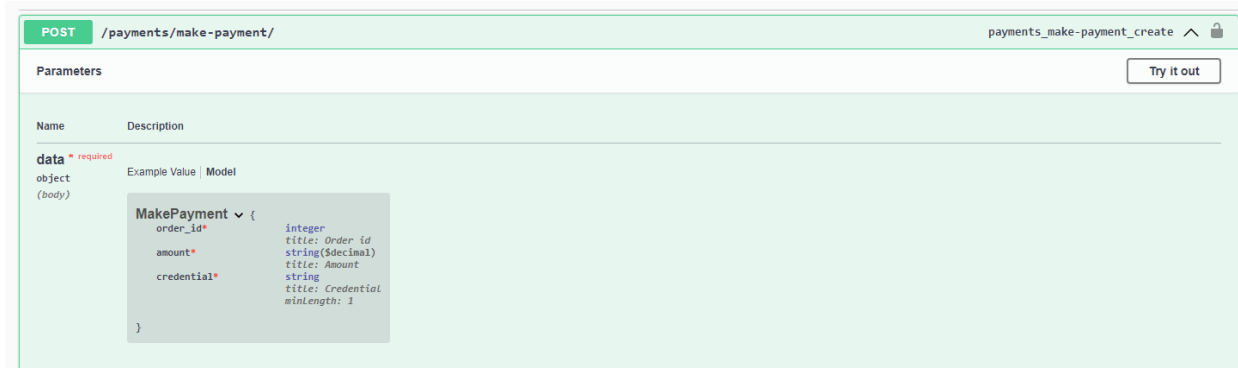
Parameters

Name	Description
transaction_id * required string (path)	transaction_id

Try it out

## 8. Bank settling the transaction and returns a transaction number

1. Make Payment : <http://localhost:8000/api/payments/make-payment/>



POST /payments/make-payment/ payments\_make-payment\_create

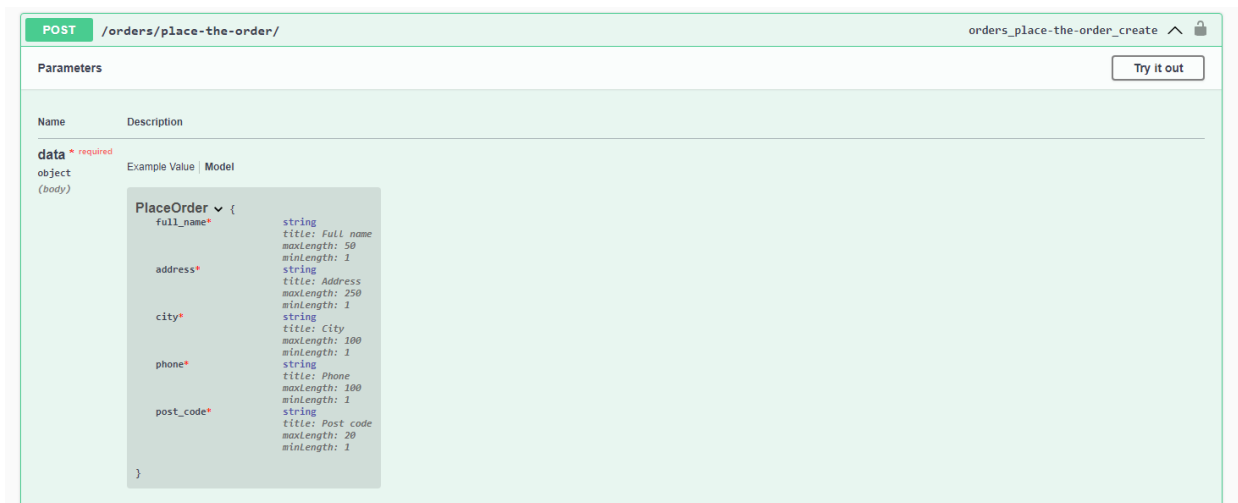
Parameters

Name	Description
data * required object (body)	<p>Example Value   Model</p> <pre>MakePayment {   order_id* integer     title: Order id     string(\$decimal)   amount* string     title: Amount     string     title: Credential     minlength: 1   credential* string     title: Credential     minlength: 1 }</pre>

Try it out

## 9. Submits a request of the ordered products to the supplier with the tranxID

1. Place The Order : <http://localhost:8000/api/order/place-the-order/>



POST /orders/place-the-order/ orders\_place-the-order\_create

Parameters

Name	Description
data * required object (body)	<p>Example Value   Model</p> <pre>PlaceOrder {   full_name* string     title: Full name     maxlength: 50     minlength: 1   address* string     title: Address     maxlength: 250     minlength: 1   city* string     title: City     maxlength: 100     minlength: 1   phone* string     title: Phone     maxlength: 100     minlength: 1   post_code* string     title: Post code     maxlength: 20     minlength: 1 }</pre>

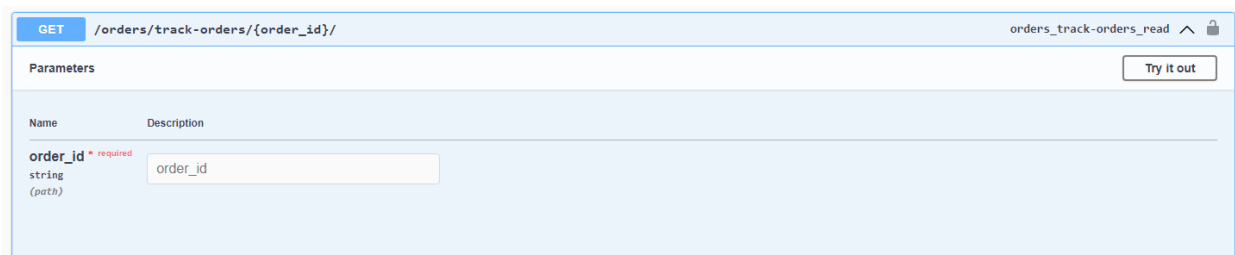
Try it out

## 10. Successful message

1 .IT IS ALREADY COVERED IN THE “*Make Payment*” SECTION

## 11. Updates its record so that the user can see that the chosen products have been supplied

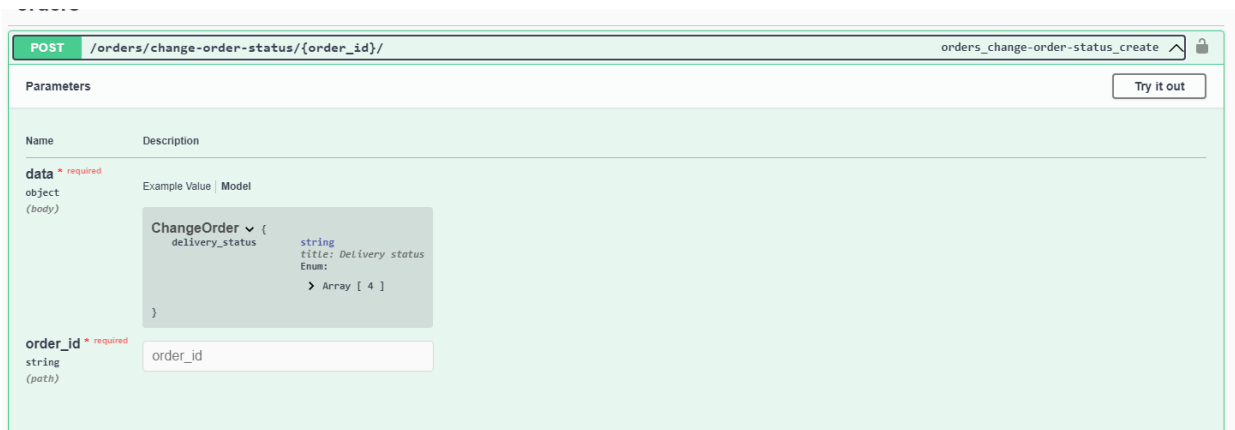
1 . Track Order Status By Customer:



Swagger UI for the endpoint `GET /orders/track-orders/{order_id}/`. The interface shows the method `GET` and the path `/orders/track-orders/{order_id}/`. A "Try it out" button is present. The parameters section lists a required path parameter `order_id` of type `string`.

Name	Description
<code>order_id</code> * required string (path)	order_id

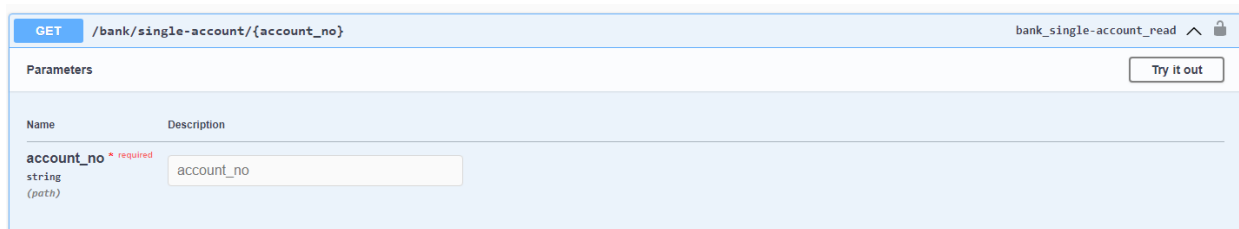
2 . Update Order Status By Ecommerce Admin:



Swagger UI for the endpoint `POST /orders/change-order-status/{order_id}/`. The interface shows the method `POST` and the path `/orders/change-order-status/{order_id}/`. A "Try it out" button is present. The parameters section lists a required body parameter `data` of type `object` and a required path parameter `order_id` of type `string`. An example JSON body is provided for the `data` parameter.

Name	Description
<code>data</code> * required object (body)	Example Value   Model  <pre>ChangeOrder {   delivery_status: string }</pre> <p>string title: Delivery status Enum: Array [ 4 ]</p>
<code>order_id</code> * required string (path)	order_id

## 12. Mechanism for these entities to show their bank balance.



Swagger UI for the endpoint `GET /bank/single-account/{account_no}`. The interface shows the method `GET` and the path `/bank/single-account/{account_no}`. A "Try it out" button is present. The parameters section lists a required path parameter `account_no` of type `string`.

Name	Description
<code>account_no</code> * required string (path)	account_no