UNIT TEST (UT)

Online Shop API

Editor	**: Arif Nur Rahman**
Create Date	**: 07 April 2021**
Revision Date	**: -**
Name	**: Online Shop API**
Version	**: 1.0**

Table of Contents

1 Preface

1.1 UT Integrated Cycles

On the UT period, tester will use resolution configuration and the process planned on the Design Conceptual stage. Tester will focus on the online shop process from login/register until confirmation payment.

1.2 UT Propose

- Do testing on Online Shop Function
- Do testing on Online Shop Process from Login/Register until Confirmation of Payment

1.3 Data Requirement

- Data IDAM
- Data SIAD

1.4 UT Implementation Assumption

- On the testing UT period, the data used is limited and it is an example data

1.5 UT Structural

Integration UT Cycle is a structure with 5 level of hierarchy. Components included as per below:

Cycle : Cycle is a process run repeatedly.

Objectives: Objective is a specific condition or prerequisite need to be fulfilled in order to complete one cycle.

Process Stage: Process stage is an activity that need to be done in one cycle.

Data:. The data type that is needed for UT is entered data.
Entered data is a data element needed in order to do the Process Stage.

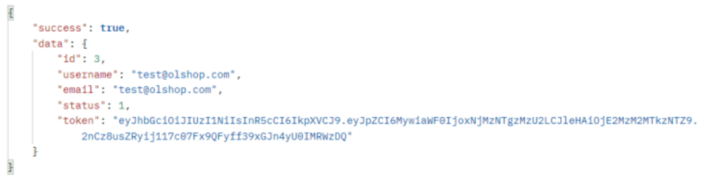
Result Expectation: Result will be received if the UT process is successfully done. The final result will appear in the level process stage from one cycle.

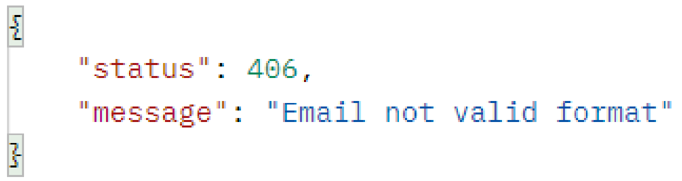
1.6 UT Cycle

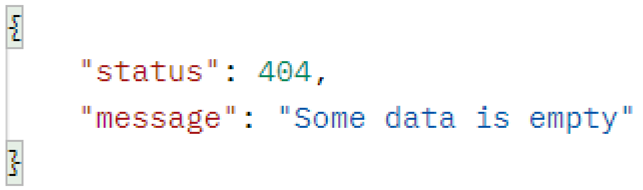
This scenario show the process that run for Online Shop API

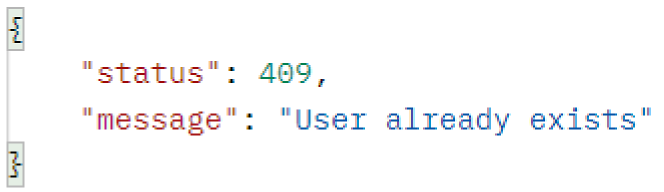
A. Authentication and Profile User

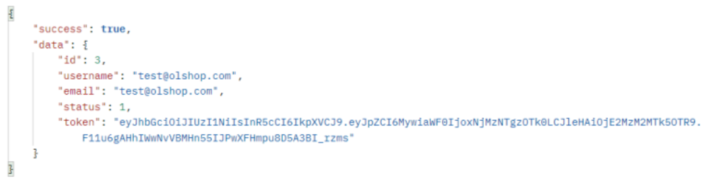
No	Process Name	Description	Data Require	Pass/Fail	Result Expectation

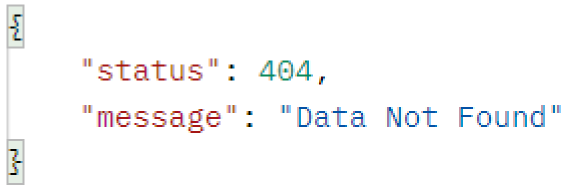
1	User Register	**API Url:** http://localhost:3000/api/auth/register
		
		Step:
		1. Access the url
		2. Fill in register data
		3. Send
			email: test@olshop.com
			
			password: test1234
				Pass	Success Register and Get Token for Login:
					
					`{`
					`"success": true,`
					`"data": {`
					`"id": 3,`
					`"username": "test@olshop.com",`
					`"email": "test@olshop.com",`
					`"status": 1,`
					`"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNjMzNTgzMzU2LCJleHAiOjE2MzM2MTkzNTZ9.2nCz8usZRyij117c07Fx9QFyff39xGJn4yU0IMRWzDQ"`
					`}`
					`}`

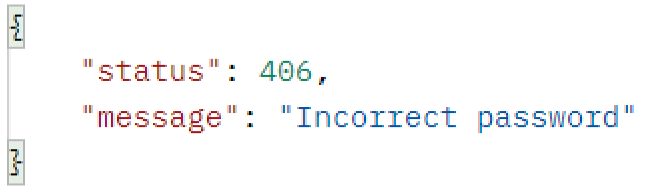
2	User Register
	
	(Negative case - Email invalid format)
		API Url: http://localhost:3000/api/auth/register
		
		Step:
		1. Access the url
		2. Fill in register data
		3. Send
			email: testolshop.com
			
			password: test1234
				Pass	Return error message email format not valid:
					
					`{`
					`"status": 406,`
					`"message": "Email not valid format"`
					`}`

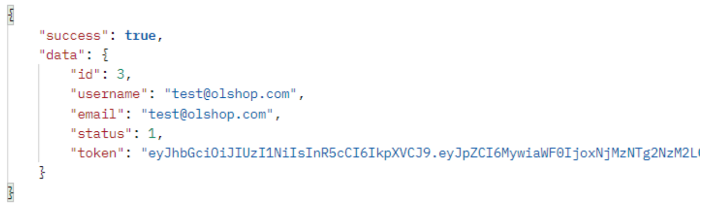
3	User Register
	
	(Negative case - some value is empty)
		API Url: http://localhost:3000/api/auth/register
		
		Step:
		1. Access the url
		2. Fill in register data
		3. Send
			password: test1234	**Pass**	Return error message some data is empty:
					
					`{`
					`"status": 404,`
					`"message": "Some data is empty"`
					`}`

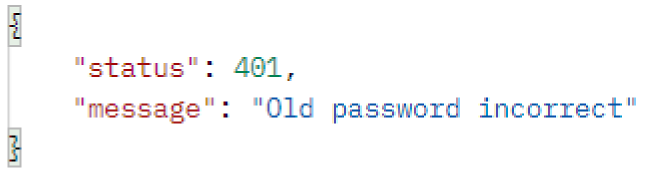
4	User Register
	
	(Negative case - User already exist)
		API Url:
		http://localhost:3000/api/auth/register
		
		Step:
		1. Access the API
		2. Fill in register data
		3. Send
			email:
			test@olshop.com
			
			password:
			test1234
				Pass	Return error message user already exists:
					
					`{`
					`"status": 409,`
					`"message": "User already exists"`
					`}`

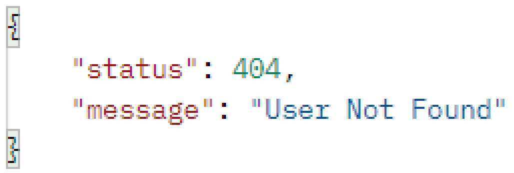
5	User Login	**API Url:**
		http://localhost:3000/api/auth/login
		
		Step:
		1. Access the API
		2. Fill in login data
		3. Send
			email:
			test@olshop.com
			
			password:
			test1234
				Pass	Success Login and Get Token Sessions:
					
					`{`
					`"success": true,`
					`"data": {`
					`"id": 3,`
					`"username": "test@olshop.com",`
					`"email": "test@olshop.com",`
					`"status": 1,`
					`"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNjMzNTgzOTk0LCJleHAiOjE2MzM2MTk5OTR9.F11u6gAHhIWwNvVBMHn55IJPwXFHmpu8D5A3BI_rzms"`
					`}`
					`}`

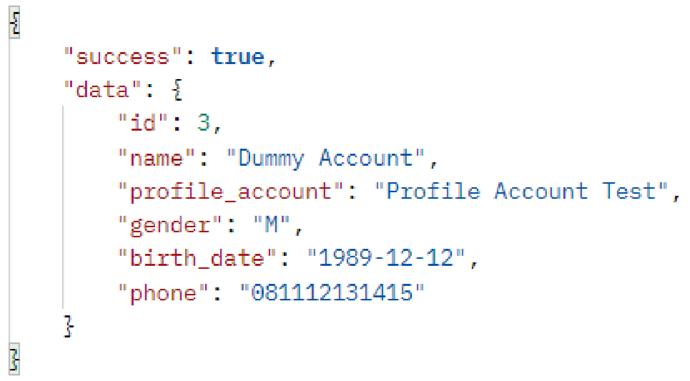
6	User Login
	
	(Negative case - User not found)
		API Url:
		http://localhost:3000/api/auth/login
		
		Step:
		1. Access the API
		2. Fill in wrong login data
		3. Send
			email:
			coba@olshop.com
			
			password:
			test1234
				Pass	Return error message data not found:
					
					`{`
					`"status": 404,`
					`"message": "Data Not Found"`
					`}`

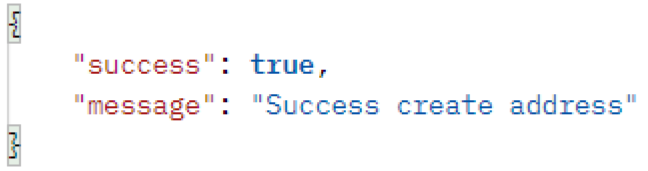
7	User Login
	
	(Negative case - incorrect password)
		API Url:
		http://localhost:3000/api/auth/login
		
		Step:
		1. Access the API
		2. Fill in login data
		3. Send
			email:
			test@olshop.com
			
			password:
			test1235
				Pass	Return error message incorrect password:
					
					`{`
					`"status": 406,`
					`"message": "Incorrect password"`
					`}`

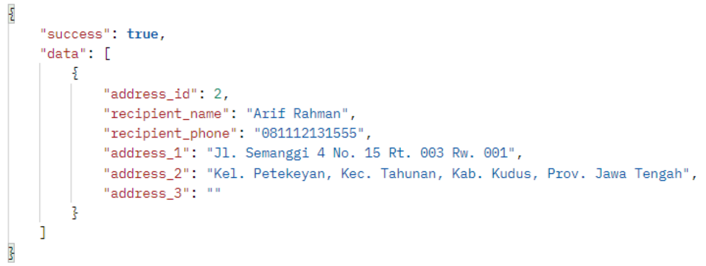
8	Update Password	**API Url:** http://localhost:3000/api/auth/update-password
		
		Step:
		1. Access the API
		2. Fill in new and old password data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			`oldPassword:`
			`test1234`
			
			`newPassword:`
			`ubahpwd1234`
				Pass	Return success message and user data:
					
					`{`
					`"success": true,`
					`"data": {`
					`"id": 3,`
					`"username": "test@olshop.com",`
					`"email": "test@olshop.com",`
					`"status": 1,`
					`"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNjMzNTg2NzM2L`
					`}`
					`}`

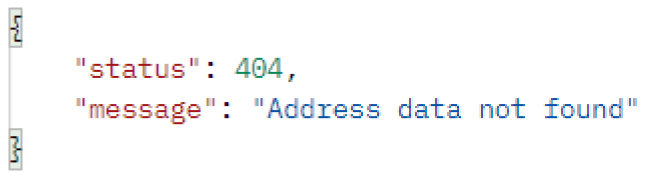
9	Update Password
	(Negative Case – Old Password Incorrect)
		API Url: http://localhost:3000/api/auth/update-password
		
		Step:
		1. Access the API
		2. Fill in new and old password data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			`oldPassword:`
			`test1234`
			
			`newPassword:`
			`ubahpwd1234`
				Pass	Return error message old password incorrect:
					
					`{`
					`"status": 401,`
					`"message": "Old password incorrect"`
					`}`

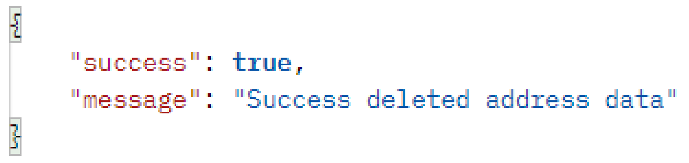
10	Update Password
	
	(Negative Case – User Not Found)
		API Url: http://localhost:3000/api/auth/update-password
		
		Step:
		1. Access the API
		2. Fill in new and old password data
		3. Fill in Authentication Header with Wrong Token from Login API (Bearer Token)
		4. Send
			oldPassword: test1234
			
			newPassword: ubahpwd1234
				Pass	Return error message user not found:
					
					`{`
					`"status": 404,`
					`"message": "User Not Found"`
					`}`

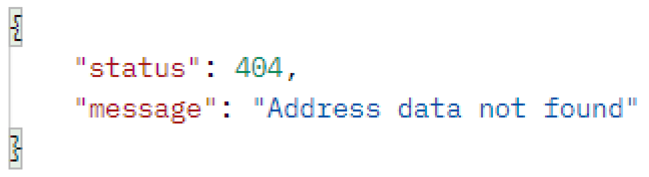
11	Update Profile	**API Url:** http://localhost:3000/api/users/update-profile
		
		Step:
		1. Access the API
		2. Fill in profile data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			name: Dummy Account
			
			profile_account: Profile Account Test
			
			gender: M
			
			birth_date: 1989-12-12
			
			phone: 081112131415
				Pass	Return success message and user profile data:
					
					`{`
					`"success": true,`
					`"data": {`
					`"id": 3,`
					`"name": "Dummy Account",`
					`"profile_account": "Profile Account Test",`
					`"gender": "M",`
					`"birth_date": "1989-12-12",`
					`"phone": "081112131415"`
					`}`
					`}`

12	Create Address	**API Url:**
		http://localhost:3000/api/users/create-address
		
		Step:
		1. Access the API
		2. Fill in address data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			recipient_name:
			`Dummy Account`
			
			recipient_phone:
			`081112131415`
			
			address_1:
			`Jl. Dummy No. 03`
			
			address_2:
			`Ciputat Timur, Tangsel`
			
			address_3:
			`-`
				Pass	Return success message create address:
					
					`{`
					`    "success": true,`
					`    "message": "Success create address"`
					`}`

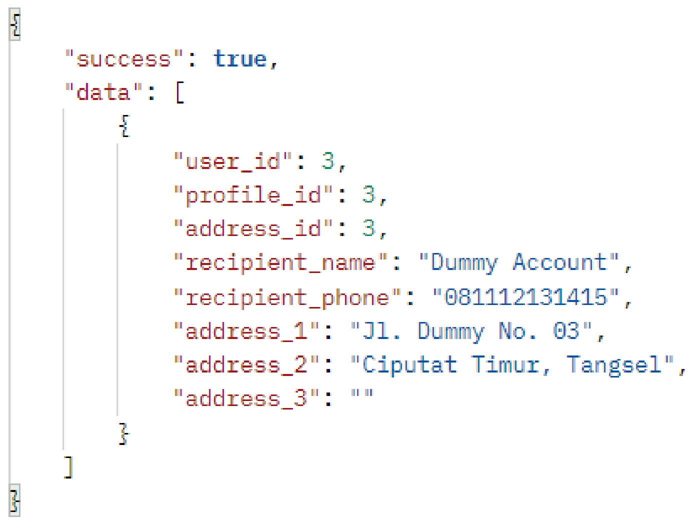
13	Update Address	**API Url:**
		http://localhost:3000/api/users/update-address
		
		Step:
		1. Access the API
		2. Fill in address data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			address_id:
			`2`
			
			recipient_name:
			`Arif Rahman`
			
			recipient_phone:
			`081112131555`
			
			address_1:
			`Jl. Semanggi 4 No. 15 Rt. 003 Rw. 001`
			
			address_2:
			`Kel. Petekeyan, Kec. Tahunan, Kab. Kudus, Prov. Jawa Tengah`
			
			address_3:
			`-`
				Pass	Return success message and address data:
					
					`{`
					`    "success": true,`
					`    "data": [`
					`        {`
					`            "address_id": 2,`
					`            "recipient_name": "Arif Rahman",`
					`            "recipient_phone": "081112131555",`
					`            "address_1": "Jl. Semanggi 4 No. 15 Rt. 003 Rw. 001",`
					`            "address_2": "Kel. Petekeyan, Kec. Tahunan, Kab. Kudus, Prov. Jawa Tengah",`
					`            "address_3": ""`
					`        }`
					`    ]`
					`}`

14	Update Address
	
	(Negative Case - Address Not Found)
		API Url:
		http://localhost:3000/api/users/update-address
		
		Step:
		1. Access the API
		2. Fill in wrong address data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			address_id: 5
			
			recipient_name: Arif Rahman
			
			recipient_phone: 081112131555
				Pass	Return error message address data not found:
					
					{
					"status": 404,
					"message": "Address data not found"
					}

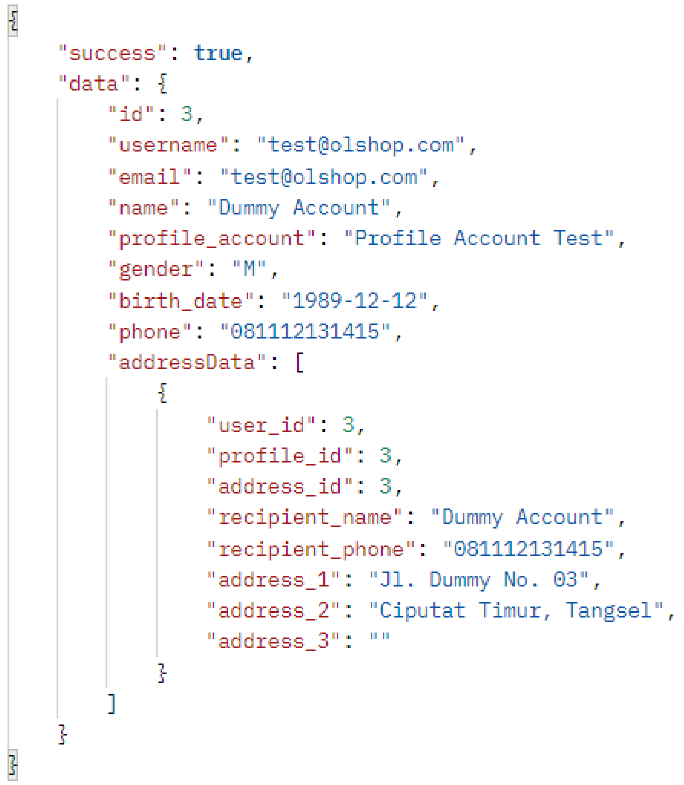
15	Delete Address	**API Url:**
		http://localhost:3000/api/users/delete-address
		
		Step:
		1. Access the API
		2. Fill in address id data
		3. Fill in Authentication Header with Token from Login API (Bearer Token)
		4. Send
			address_id: 2	**Pass**	Return success message for deleted address data:
					
					{
					"success": true,
					"message": "Success deleted address data"
					}

16	Delete Address
	
	(Negative Case - Address Not Found)
		API Url:
		http://localhost:3000/api/users/delete-address
		
		Step:
		1. Access the API
		2. Fill in wrong address id data
			address_id: 5	**Pass**	Return error message address data not found:
					
					{
					"status": 404,
					"message": "Address data not found"
					}

		3. Fill in Authentication Header with Token from Login API (Bearer Token) 4. Send			

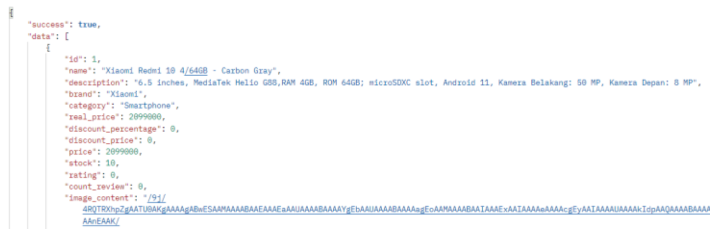
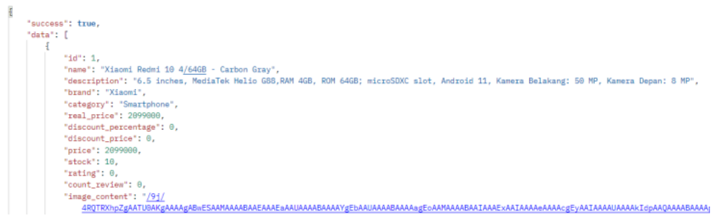
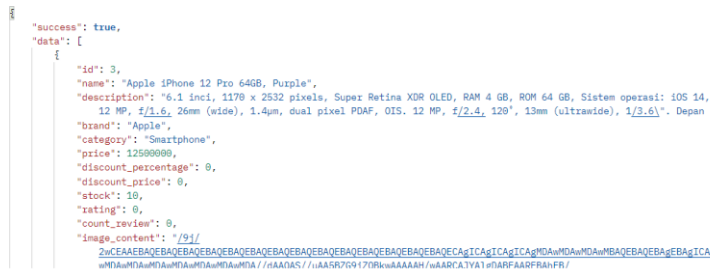
17	Get Address	**API Url:**
		http://localhost:3000/api/users/get-address
		
		Step:
		1. Access the API
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Send
			-	**Pass**	Return success message and address data:
					
					{
					"success": true,
					"data": [
					{
					"user_id": 3,
					"profile_id": 3,
					"address_id": 3,
					"recipient_name": "Dummy Account",
					"recipient_phone": "081112131415",
					"address_1": "Jl. Dummy No. 03",
					"address_2": "Ciputat Timur, Tangsel",
					"address_3": ""
					}
					]
					}

18	Get User Data	**API Url:** http://localhost:3000/api/auth/me **Step**:	-	**Pass**	Return success message and user data:

		1. Access the API 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Send			

```
{
    "success": true,
    "data": {
        "id": 3,
        "username": "test@olshop.com",
        "email": "test@olshop.com",
        "name": "Dummy Account",
        "profile_account": "Profile Account Test",
        "gender": "M",
        "birth_date": "1989-12-12",
        "phone": "081112131415",
        "addressData": [
            {
                "user_id": 3,
                "profile_id": 3,
                "address_id": 3,
                "recipient_name": "Dummy Account",
                "recipient_phone": "081112131415",
                "address_1": "Jl. Dummy No. 03",
                "address_2": "Ciputat Timur, Tangsel",
                "address_3": ""
            }
        ]
    }
}
```

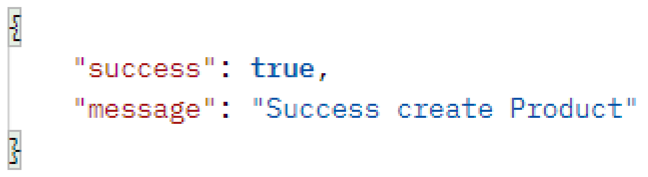
B. Product Data

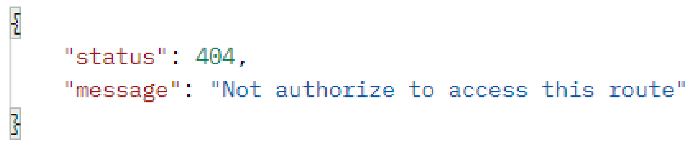
No	Process Name	Description	Data Require	Pass/Fail	Result Expectation
1	Get All Product	**API Url:** http://localhost:3000/api/products/get-all-product **Step**: 1. Access the url 2. Send	-	**Pass**	Return success message and all product data: 
2	Get Product	**API Url:** http://localhost:3000/api/products/get-product **Step**: 1. Access the url 2. Fill in product_id 3. Send	**`product_id:`** `1`	**Pass**	Return success message and product data based on id: 
3	Filter Product	**API Url:** http://localhost:3000/api/products/filter-product **Step**: 1. Access the url 2. Fill in keyword 3. Send	**`keyword:`** `Iphone 12`	**Pass**	Return success message and product data based on keyword: 

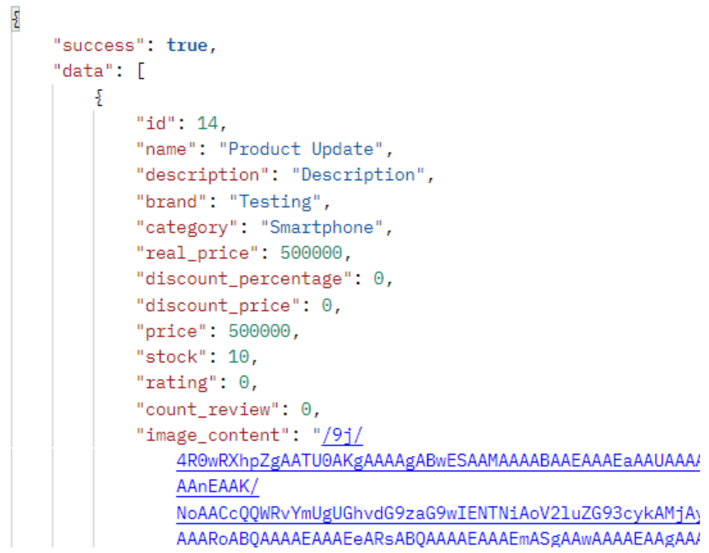
```
{
    "success": true,
    "data": [
        {
            "id": 1,
            "name": "Xiaomi Redmi 10 4/64GB - Carbon Gray",
            "description": "6.5 inches, MediaTek Helio G88,RAM 4GB, ROM 64GB; microSDXC slot, Android 11, Kamera Belakang: 50 MP, Kamera Depan: 8 MP",
            "brand": "Xiaomi",
            "category": "Smartphone",
            "real_price": 2099000,
            "discount_percentage": 0,
            "discount_price": 0,
            "price": 2099000,
            "stock": 10,
            "rating": 0,
            "count_review": 0,
            "image_content": "/9j/
                4RQTRXhpZgAATU0AKgAAAAgABwESAAMAAAABAAEAAAEaAAUAAAABAAAAYgEbAAUAAAABAAAAagEoAAMAAAABAAIAAAExAAIAAAAeAAAAcgEyAAIAAAAUAAAAkIdpAAQAAAABAAAA
                AAnEAAK/
```

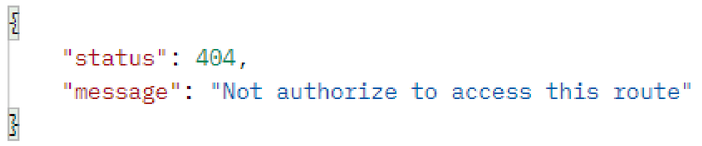
```
{
    "success": true,
    "data": [
        {
            "id": 1,
            "name": "Xiaomi Redmi 10 4/64GB - Carbon Gray",
            "description": "6.5 inches, MediaTek Helio G88,RAM 4GB, ROM 64GB; microSDXC slot, Android 11, Kamera Belakang: 50 MP, Kamera Depan: 8 MP",
            "brand": "Xiaomi",
            "category": "Smartphone",
            "real_price": 2099000,
            "discount_percentage": 0,
            "discount_price": 0,
            "price": 2099000,
            "stock": 10,
            "rating": 0,
            "count_review": 0,
            "image_content": "/9j/
                4RQTRXhpZgAATU0AKgAAAAgABwESAAMAAAABAAEAAAEaAAUAAAABAAAAYgEbAAUAAAABAAAAagEoAAMAAAABAAIAAAExAAIAAAAeAAAAcgEyAAIAAAAUAAAAkIdpAAQAAAABAAAA
```

```
{
    "success": true,
    "data": [
        {
            "id": 3,
            "name": "Apple iPhone 12 Pro 64GB, Purple",
            "description": "6.1 inci, 1170 x 2532 pixels, Super Retina XDR OLED, RAM 4 GB, ROM 64 GB, Sistem operasi: iOS 14,
                12 MP, f/1.6, 26mm (wide), 1.4µm, dual pixel PDAF, OIS. 12 MP, f/2.4, 120˚, 13mm (ultrawide), 1/3.6\". Depan
            "brand": "Apple",
            "category": "Smartphone",
            "price": 12500000,
            "discount_percentage": 0,
            "discount_price": 0,
            "stock": 10,
            "rating": 0,
            "count_review": 0,
            "image_content": "/9j/
                2wCEAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQECAgICAgICAgICAgMDAwMDAwMDAwMBAQEBAQEBAgEBAgICA
                wMDAwMDAwMDAwMDAwMDAwMDA//dAAQAS//uAA5BZG9iZQBkwAAAAAH/wAARCAJYAlgDABEAAREBAhEB/
```

4	Create Product	**API Url:** http://localhost:3000/api/products/create-product
		
		Step:
		1. Access the url
		2. Fill in product data
		3. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
		4. Send
			name:
			Product Test
			
			description:
			Product desc
			
			brand:
			testing
			
			category:
			Smartphone
			
			price:
			1100000
			
			discount_percentage:
			0
			
			Stock:
			10
			
			image_content:
			base64 image format
				Pass	Return success message create product:
					
					
					{
					"success": true,
					"message": "Success create Product"
					}

5	Create Product
	
	(Negative Case - Not authorize User)
		API Url: http://localhost:3000/api/products/create-product
		
		Step:
		1. Access the url
		2. Fill in product data
		3. Fill in Authentication Header with User Token from Login API (Bearer Token)
		4. Send
			name:
			Product Test
			
			description:
			Product desc
			
			brand:
			testing
			
			category:
			Smartphone
			
			price:
			1100000
				Pass	Return error message user not authorize:
					
					
					{
					"status": 404,
					"message": "Not authorize to access this route"
					}

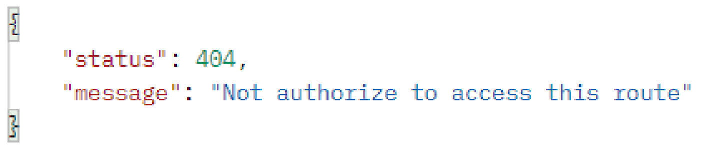
			discount_percentage: 0 **Stock:** 10 **image_content:** *base64 image format*		
6	Update Product	**API Url:** http://localhost:3000/api/products/update-product **Step**: 1. Access the url 2. Fill in product data 3. Fill in Authentication Header with Admin Token from Login API (Bearer Token) 4. Send	**product_id:** 14 **name:** Product Update **price:** 500000	**Pass**	Return success message and product data:  { "success": true, "data": [{ "id": 14, "name": "Product Update", "description": "Description", "brand": "Testing", "category": "Smartphone", "real_price": 500000, "discount_percentage": 0, "discount_price": 0, "price": 500000, "stock": 10, "rating": 0, "count_review": 0, "image_content": "/9j/ 4R0wRXhpZgAATU0AKgAAAAgABwESAAMAAAABAAEAAAEaAAUAAA AAnEAAK/ NoAACcQQWRvYmUgUGhvdG9zaG9wIENTNiAoV2luZG93cykAMjA AAARoABQAAAAEAAAEeARsABQAAAAEAAAEmASgAAwAAAAEAAgAA

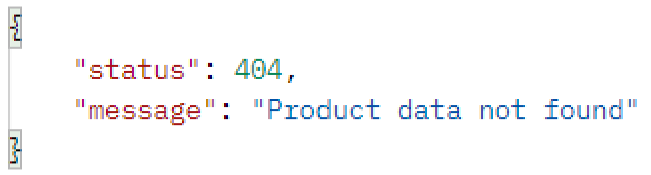
7	Update Product
	
	(Negative Case - Not authorize User)
		API Url:
		http://localhost:3000/api/products/update-product
		
		Step:
		1. Access the url
		2. Fill in product data
			product_id:
			14
			
			name:
			Product Update
			
			price:
			500000
				Pass	Return error message user not authorize:
					
					{
					"status": 404,
					"message": "Not authorize to access this route"
					}

		3. Fill in Authentication Header with User Token from Login API (Bearer Token) 4. Send			

8	Update Product
	
	(Negative Case - Product Not Found)
		API Url:
		http://localhost:3000/api/products/update-product
		
		Step:
		1. Access the url
		2. Fill in wrong product id
		3. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
		4. Send
			product_id: 18
			
			name: Product Update
			
			price: 500000
				Pass	Return error message product not found:
					
					`{`
					`"status": 404,`
					`"message": "Product data not found"`
					`}`

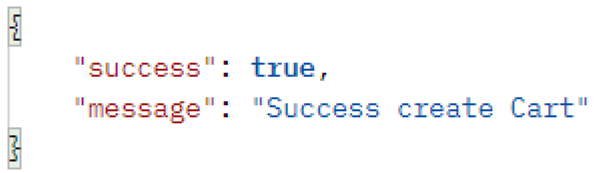
9	Delete Product	**API Url:**
		http://localhost:3000/api/products/delete-product
		
		Step:
		1. Access the url
		2. Fill in product_id data
		3. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
		4. Send
			product_id: 14	**Pass**	Return success message deleted product:
					
					`{`
					`"success": true,`
					`"message": "Success deleted Product data"`
					`}`

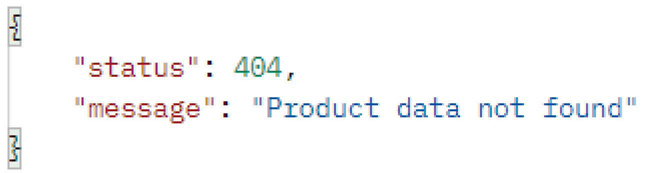
10	Delete Product
	
	(Negative Case - Not authorize User)
		API Url:
		http://localhost:3000/api/products/delete-product
		
		Step:
		1. Access the url
		2. Fill in product_id data
		3. Fill in Authentication Header with User Token from Login API (Bearer Token)
		4. Send
			product_id:
			14
				Pass	Return error message user not authorize:
					
					{
					"status": 404,
					"message": "Not authorize to access this route"
					}

11	Delete Product
	
	(Negative Case - Product Not Found)
		API Url:
		http://localhost:3000/api/products/update-product
		
		Step:
		1. Access the url
		2. Fill in wrong product id
		3. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
		4. Send
			product_id:
			18
				Pass	Return error message product not found:
					
					{
					"status": 404,
					"message": "Product data not found"
					}

C. Shopping Cart Process

No	Process Name	Description	Data Require	Pass/Fail	Result Expectation

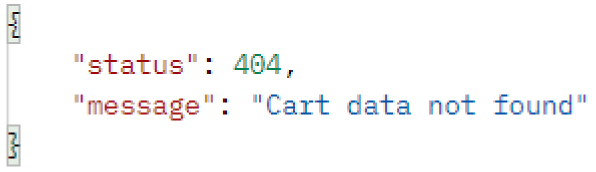
1	Add to Cart	**API Url:** http://localhost:3000/api/cart/create-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in product_id and quantity
		4. Repeat 3 Times with different product:
		id: 3 - Iphone 12 Pro
		id: 5 - Samsung Max Pro
		id: 11 - Screen Guard
		5. Send
			Cart 1:
			product_id: 3
			quantity: 1
			
			Cart 2:
			product_id: 5
			quantity: 1
			
			Cart 3:
			product_id: 11
			quantity: 1
				Pass	Return success message and all cart data based on user_id:
					
					{
					"success": true,
					"message": "Success create Cart"
					}

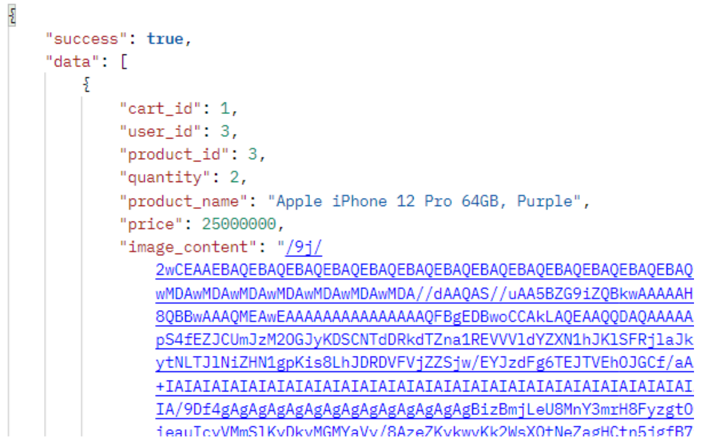
2	Add to Cart
	
	(Negative Case – Product Not Found)
		API Url: http://localhost:3000/api/cart/create-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from
			product_id: 31
			quantity: 1
				Pass	Return error message product not found:
					
					{
					"status": 404,
					"message": "Product data not found"
					}

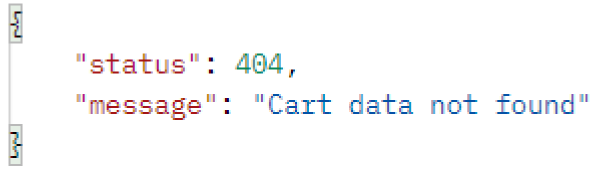
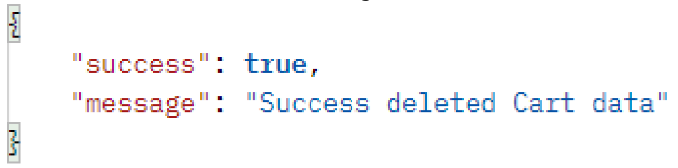
		Login API (Bearer Token) 3. Fill in wrong product_id and quantity 4. Send			

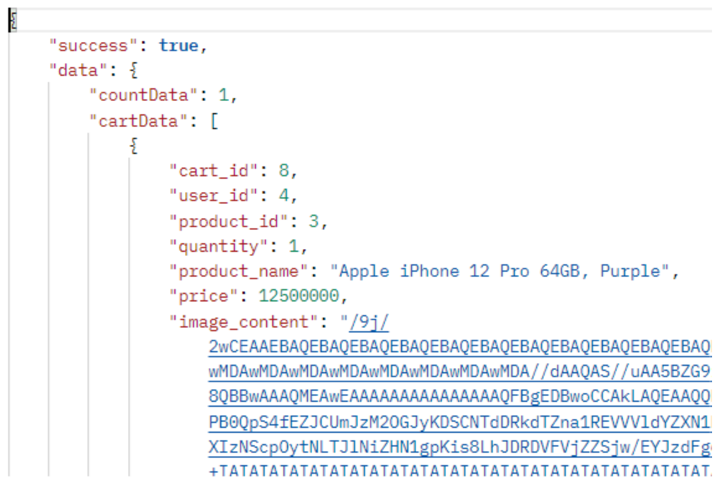
3	Add to Cart
	
	(Negative Case – Product Out of Stock)
		API Url:
		http://localhost:3000/api/cart/create-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in product_id and wrong quantity
		4. Send
			`product_id: 3`
			`quantity: 12`
				Pass	Return error message product out of stock:
					`{`
					`"status": 406,`
					`"message": "Out of Stock"`
					`}`

4	Update Cart
	
	(Negative Case – Product Out of Stock)
		API Url:
		http://localhost:3000/api/cart/update-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in product_id and wrong quantity
		4. Send
			`product_id: 3`
			`quantity: 15`
				Pass	Return error message product out of stock:
					`{`
					`"status": 406,`
					`"message": "Out of Stock"`
					`}`

5	Update Cart
	
	(Negative Case – Cart Not Found)
		API Url: http://localhost:3000/api/cart/update-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in wrong cart_id and quantity
		4. Send
			`cart_id: 31`
			
			`quantity: 1`
				Pass	Return error message cart not found
					
					`{`
					`"status": 404,`
					`"message": "Cart data not found"`
					`}`

6	Update Cart	**API Url:** http://localhost:3000/api/cart/update-cart
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in product_id and quantity
		4. Send
			`cart_id: 1`
			
			`quantity: 2`
				Pass	Return success message and cart data based on user_id:
					
					`{`
					`"success": true,`
					`"data": [`
					`{`
					`"cart_id": 1,`
					`"user_id": 3,`
					`"product_id": 3,`
					`"quantity": 2,`
					`"product_name": "Apple iPhone 12 Pro 64GB, Purple",`
					`"price": 25000000,`
					`"image_content": "/9j/`
					`2wCEAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQ`
					`wMDAwMDAwMDAwMDAwMDAwMDAwMDA//dAAQAS//uAA5BZG9iZQBkwAAAAAH`
					`8QBBwAAAQMEAwEAAAAAAAAAAAAAAAQFBgEDBwoCCAkLAQEAAQQDAQAAAAA`
					`pS4fEZJCUmJzM2OGJyKDSCNTdDRkdTZna1REVVV1dYZXN1hJK1SFRjlaJk`
					`ytNLTJ1NiZHN1gpKis8LhJDRDVFVjZZSjw/EYJzdFg6TEJTVEh0JGCf/aA`
					`+IAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAI`
					`IA/9Df4gAgAgAgAgAgAgAgAgAgAgAgAgBizBmjLeU8MnY3mrH8FyzgtO`
					`ieauTcyVMmS1KyDkyMGMYaVy/8AzeZKykwyKk2WsXOtNeZagHCtn5igfB7`

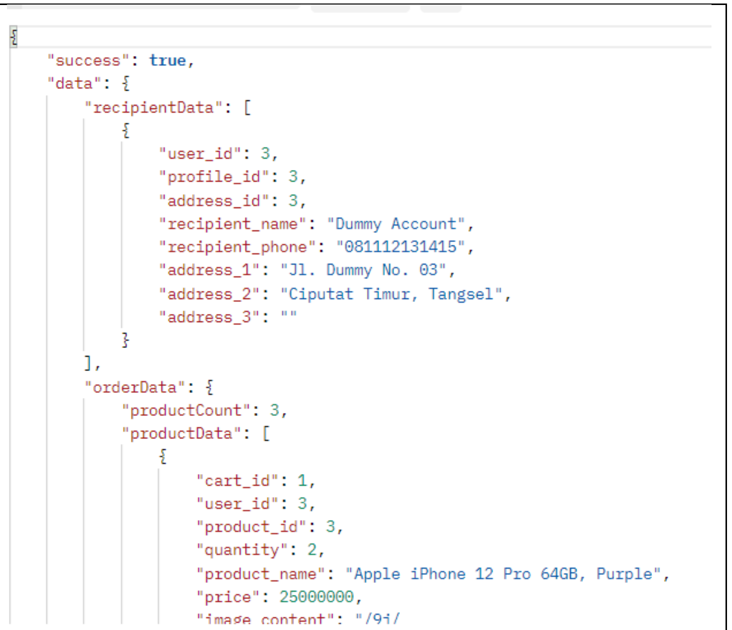
7	Delete Cart (Negative Case – Cart Not Found)	**API Url:** http://localhost:3000/api/cart/delete-cart **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in wrong cart_id 4. Send	`cart_id: 31`	**Pass**	Return error message cart not found  `{ "status": 404, "message": "Cart data not found" }`
8	Delete Cart	**API Url:** http://localhost:3000/api/cart/delete-cart **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in cart_id 4. Send	`cart_id: 3`	**Pass**	Return success message deleted cart data  `{ "success": true, "message": "Success deleted Cart data" }`
9	Get Cart	**API Url:** http://localhost:3000/api/cart/get-cart **Step**: 1. Access the url 2. Fill in Authentication Header with Token from	-	**Pass**	Return success message and cart data

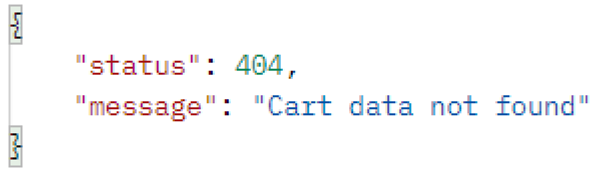
		Login API (Bearer Token) 3. Send			

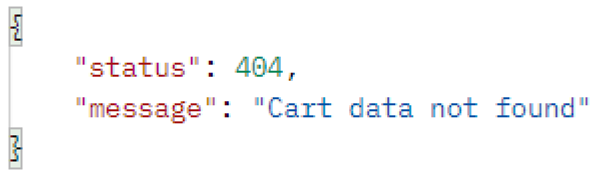
```
{
    "success": true,
    "data": {
        "countData": 1,
        "cartData": [
            {
                "cart_id": 8,
                "user_id": 4,
                "product_id": 3,
                "quantity": 1,
                "product_name": "Apple iPhone 12 Pro 64GB, Purple",
                "price": 12500000,
                "image_content": "/9j/
                    2wCEAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQ
                    wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDA//dAAQAS//uAA5BZG9
                    8QBBwAAAQMEAwEAAAAAAAAAAAAAAAQFBgEDBwoCCAkLAQEAAQQ
                    PB0QpS4fEZJCUmJzM2OGJyKDSCNTdDRkdTZna1REVVV1dYZXN1
                    XIzNScp0ytNLTJ1NiZHN1gpKis8LhJDRDVFVjZZSjw/EYJzdFg
                    +TATATATATATATATATATATATATATATATATATATATATATAT
```

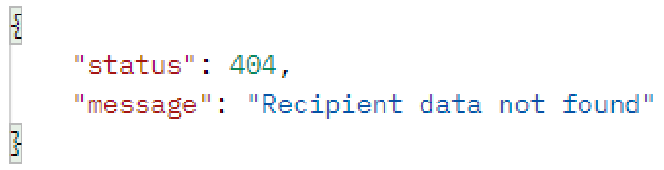
D. Checkout and Order Process

No	Process Name	Description	Data Require	Pass/Fail	Result Expectation
1	Checkout Detail	**API Url:** http://localhost:3000/api/orders/checkout-detail **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token)	-	**Pass**	Return success message and checkout detail based on user_id and cart data:

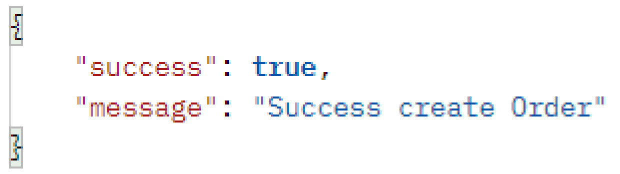
		3. Send			 `{`
					`"success": true,`
					`"data": {`
					`"recipientData": [`
					`{`
					`"user_id": 3,`
					`"profile_id": 3,`
					`"address_id": 3,`
					`"recipient_name": "Dummy Account",`
					`"recipient_phone": "081112131415",`
					`"address_1": "Jl. Dummy No. 03",`
					`"address_2": "Ciputat Timur, Tangsel",`
					`"address_3": ""`
					`}`
					`],`
					`"orderData": {`
					`"productCount": 3,`
					`"productData": [`
					`{`
					`"cart_id": 1,`
					`"user_id": 3,`
					`"product_id": 3,`
					`"quantity": 2,`
					`"product_name": "Apple iPhone 12 Pro 64GB, Purple",`
					`"price": 25000000,`
					`"image_content": "/9j/`

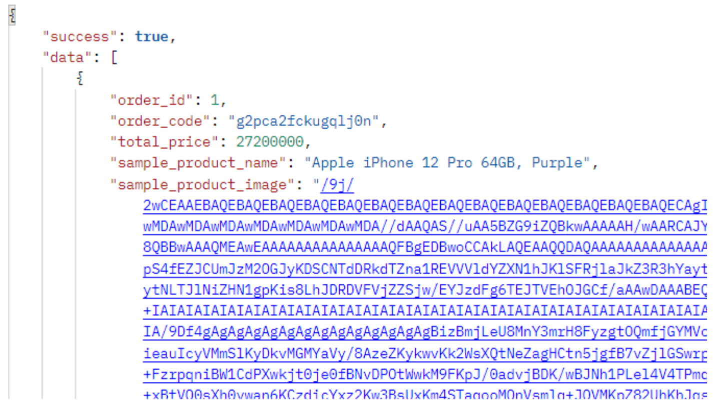
2	Checkout Detail
	
	(Negative Case – Cart data not found)
		API Url:
		http://localhost:3000/api/orders/checkout-detail
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token) which that user doesn't have a shopping cart
		3. Send
			–	**Pass**	Return error message cart data not found:
					 `{`
					`"status": 404,`
					`"message": "Cart data not found"`
					`}`

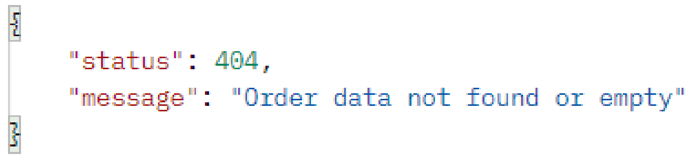
3	Create Order
	
	(Negative Case – Cart data not found)
		API Url:
		http://localhost:3000/api/orders/create-order
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token) which that user doesn't have a shopping cart
		3. Fill in order data
		4. Send
			`address_id: 2`
			
			`delivery_service: JNE`
			
			`note: -`
				Pass	Return error message cart not found
					
					`{`
					`"status": 404,`
					`"message": "Cart data not found"`
					`}`

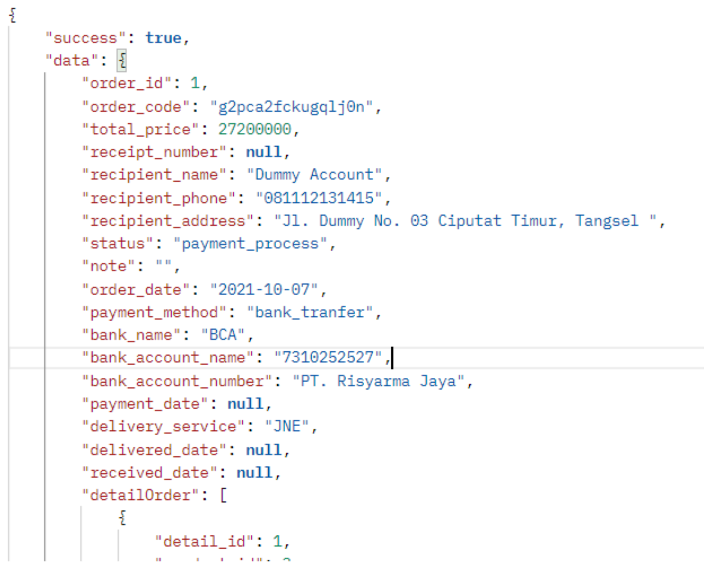
4	Create Order
	
	(Negative Case – recipient data not found)
		API Url:
		http://localhost:3000/api/orders/create-order
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in order data with wrong address_id
		4. Send
			`address_id: 5`
			
			`delivery_service: JNE`
			
			`note: -`
				Pass	Return error message recipient not found
					
					`{`
					`"status": 404,`
					`"message": "Recipient data not found"`
					`}`

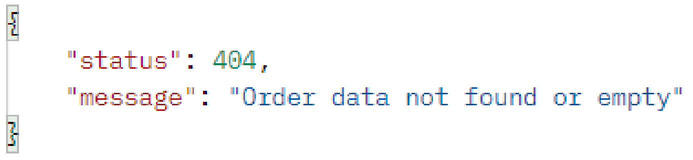
5	Create Order	**API Url:**
		http://localhost:3000/api/orders/create-order
		
		Step:
		1. Access the url
			`address_id: 3`
			
			`delivery_service: JNE`
			
			`note: -`
				Pass	Return success message create order and waiting payment

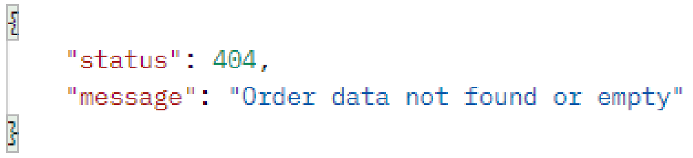
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in order data
		4. Send
					
					{
					"success": true,
					"message": "Success create Order"
					}

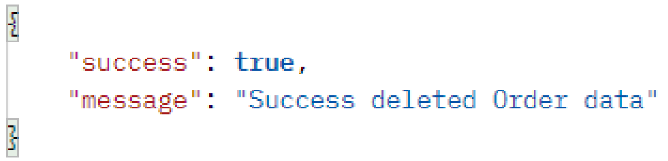
6	Get Order	**API Url:** http://localhost:3000/api/orders/get-order **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Send	-	**Pass**	Return success message and order data  { "success": true, "data": [{ "order_id": 1, "order_code": "g2pca2fckugqlj0n", "total_price": 27200000, "sample_product_name": "Apple iPhone 12 Pro 64GB, Purple", "sample_product_image": "/9j/ 2wCEAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQECAgI wMDAwMDAwMDAwMDAwMDAwMDAwMDA//dAAQAS//uAA5BZG9iZQBkwAAAAAH/wAARCAJY 8QBBwAAAQMEAwEAAAAAAAAAAAAAAAAQFBgEDBwoCCAkLAQEAAQQDAQAAAAAAAAAAAAA pS4fEZJCUmJzM2OGJyKDSCNTdDRkdTZna1REVVV1dYZXN1hJKlSFRjlaJkZ3R3hYayt ytNLTJlNiZHN1gpKis8LhJDRDVFVjZZSjw/EYJzdFg6TEJTVEhOJGCf/aAAwDAAABEQ +IAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIA IA/9Df4gAgAgAgAgAgAgAgAgAgAgAgAgAgBizBmjLeU8MnY3mrH8FyzgtOQmfjGYMVc ieauIcyVMmSlKyDkvMGMYaVy/8AzeZKykwvKk2WsXQtNeZagHCtn5jgfB7vZjlGSwrp +FzrpqniBW1CdPXwkjt0je0fBNvDPOtWwkM9FKpJ/0advjBDK/wBJNh1PLel4V4TPmc +xBtVO0sXh0vwan6KCzdicYxz2Kw3BsUxKm4STagooMOnVsmlg+JOVMKnZ82UhKhJgs

7	Get Order
	
	(Negative Case - Order Not Found)
		API Url:
		http://localhost:3000/api/orders/get-order
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token) which that user doesn't have a order data
		3. Send
			-	**Pass**	Return error message order not found
					
					{
					"status": 404,
					"message": "Order data not found or empty"
					}

8	Get Detail Order	**API Url:**
		http://localhost:3000/api/orders/get-orders
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in order_id
		4. Send
			`order_id: 1`	**Pass**	Return success message and order detail data
					
					{
					"success": true,
					"data": {
					"order_id": 1,
					"order_code": "g2pca2fckugqlj0n",
					"total_price": 27200000,
					"receipt_number": null,
					"recipient_name": "Dummy Account",
					"recipient_phone": "081112131415",
					"recipient_address": "Jl. Dummy No. 03 Ciputat Timur, Tangsel ",
					"status": "payment_process",
					"note": "",
					"order_date": "2021-10-07",
					"payment_method": "bank_tranfer",
					"bank_name": "BCA",
					"bank_account_name": "7310252527",
					"bank_account_number": "PT. Risyarma Jaya",
					"payment_date": null,
					"delivery_service": "JNE",
					"delivered_date": null,
					"received_date": null,
					"detailOrder": [
					{
					"detail_id": 1,

9	Get Detail Order
	(Negative Case - Order Not Found)
		API Url: http://localhost:3000/api/orders/get-orders
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in wrong order_id
		4. Send
			`order_id: 2`	**Pass**	Return error message order not found
					
					{
					"status": 404,
					"message": "Order data not found or empty"
					}

10	Cancel Order
	(Negative Case - Order Not Found)
		API Url: http://localhost:3000/api/orders/cancel-order
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in wrong order_id
		4. Send
			`order_id: 2`	**Pass**	Return error message order not found
					
					{
					"status": 404,
					"message": "Order data not found or empty"
					}

11	Cancel Order	**API Url:** http://localhost:3000/api/orders/cancel-order
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from
			`order_id: 1`	**Pass**	Return success message cancel or delete order
					
					{
					"success": true,
					"message": "Success deleted Order data"
					}

		Login API (Bearer Token) 3. Fill in order_id 4. Send			

E. Payment and Tracking Order

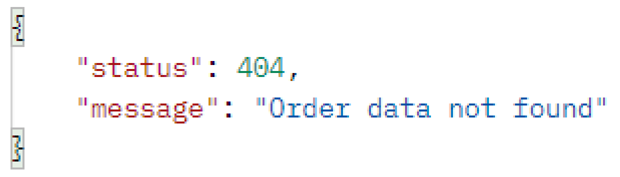
No	Process Name	Description	Data Require	Pass/Fail	Result Expectation

1	Confirm Payment
	(Negative Case - Order Not Found)
		API Url: http://localhost:3000/api/orders/confirm-payment
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in wrong order_id and valid payment_evidence
		4. Send
			order_id: 2,
			payment_evidence:
			base64_image
				Pass	Return error message order not found
					`{`
					`"status": 404,`
					`"message": "Order data not found or empty"`
					`}`

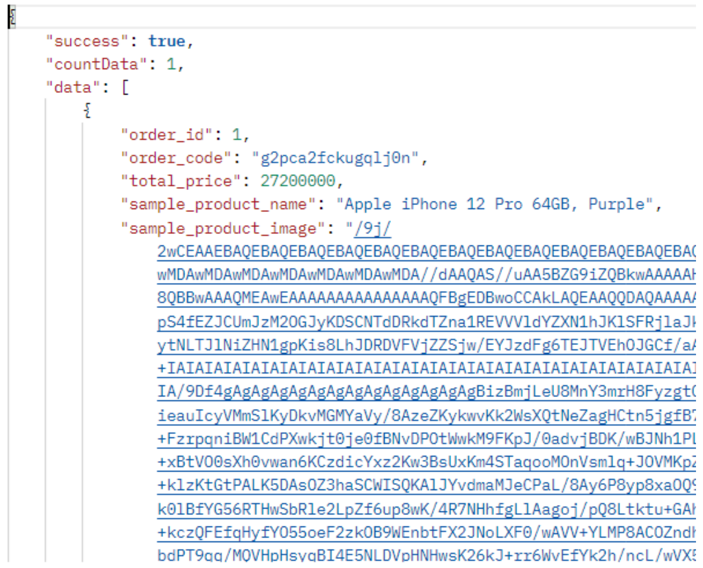
2	Confirm Payment (Negative Case - Payment	**API Url:** http://localhost:3000/api/orders/confirm-payment **Step**: 1. Access the url	**order_id**: 1, **payment_evidence**: -	**Pass**	Return error message payment evidence not found

	Evidence Not Found)	2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in order_id 4. Send			{ "status": 404, "message": "Payment evidence data not found" }
3	Confirm Payment	**API Url:** http://localhost:3000/api/orders/confirm-payment **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in valid order_id and payment_evidence 4. Send	**order_id**: 1, **payment_evidence**: *base64_image*	**Pass**	Return success message order not found { "success": true, "data": "Success upload evidence payment" }
4	Update Order (Negative Case - Not authorize User)	**API Url:** http://localhost:3000/api/orders/update-order **Note:** This API is used to update the payment_date field and order status to be **paid_off** if the evidence is valid or update status to **packed** if the package is being packed, or update **recipient_number**,	**order_id**: 1, **payment_date**: 2021-10-7 **status**: paid_off	**Pass**	Return error message user not authorize: { "status": 404, "message": "Not authorize to access this route" }

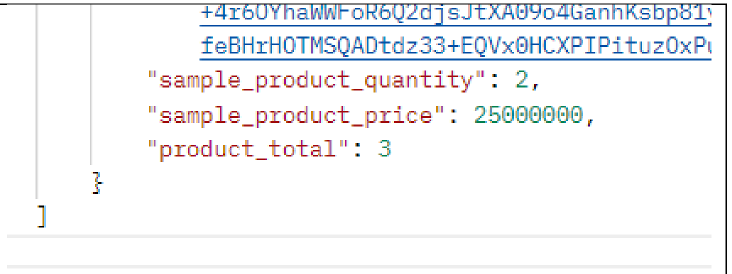
		delivered date and status to **delivered** if it has been sent, and update status to **received** if the package has been received. **Step**: 1. Access the url 2. Fill in Authentication Header with User Token from Login API (Bearer Token) and updated status field is not **received** 3. Send			

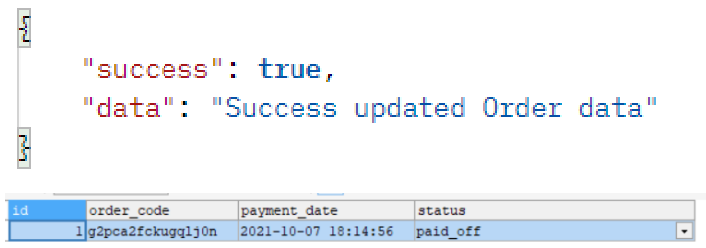
5	Update Order
	
	(Negative Case - Order Not Found)
		API Url:
		http://localhost:3000/api/orders/update-order
		
		Note:
		This API is used to update the payment_date field and order status to be **paid_off** if the evidence is valid or update status to **packed** if the package is being packed, or update **recipient_number**, delivered date and status to **delivered** if it has been sent, and update status to **received** if the package has been received.
			order_id: 14,
			payment_date: 2021-10-7
			status: paid_off
				Pass	Return error message order data not found:
					
					{
					"status": 404,
					"message": "Order data not found"
					}

		Step: 1. Access the url 2. Fill in Authentication Header with Admin Token from Login API (Bearer Token) and updated status field is not **received** 3. Send			

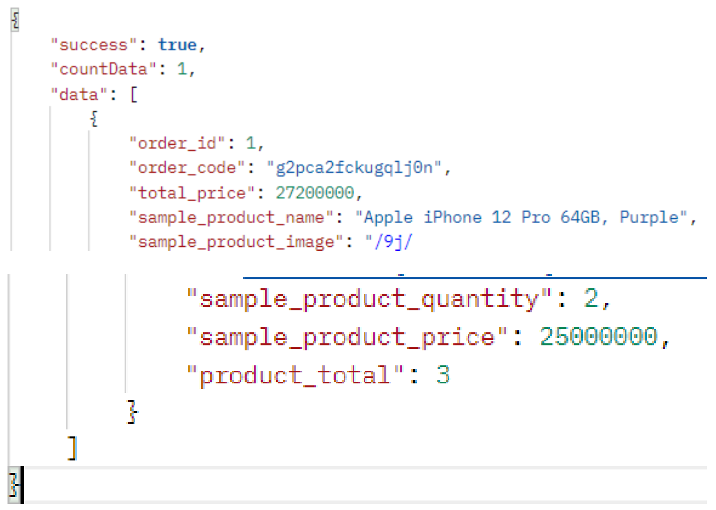
6	Tracking Status - Waiting Payment	**API Url:** http://localhost:3000/api/orders/status-orders
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in order_id, status
		4. Send
			`order_id:` `1`
			`status:`
			`payment_process`
				Pass	Return success message and order data based on status **payment_process**:
					

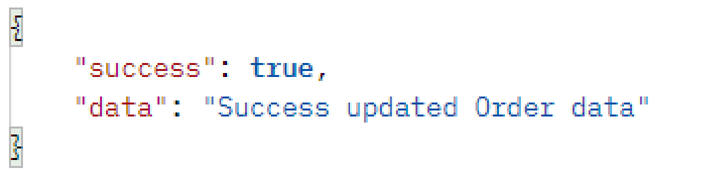
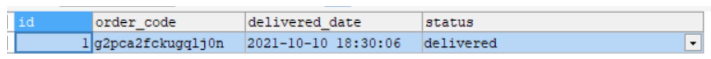
```
{
    "success": true,
    "countData": 1,
    "data": [
        {
            "order_id": 1,
            "order_code": "g2pca2fckugqlj0n",
            "total_price": 27200000,
            "sample_product_name": "Apple iPhone 12 Pro 64GB, Purple",
            "sample_product_image": "/9j/
                2wCEAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA
                wMDAwMDAwMDAwMDAwMDAwMDAwMDA//dAAQAS//uAA5BZG9iZQBkwAAAAA
                8QBBwAAAQMEAwEAAAAAAAAAAAAAAAAQFBgEDBwoCCAkLAQEAAQQDAQAAAA
                pS4fEZJCUmJzM2OGJyKDSCNTdDRkdTZna1REVVV1dYZXN1hJK1SFRjlaJ
                ytNLTJlNiZHN1gpKis8LhJDRDVFVjZZSjw/EYJzdFg6TEJTVEhOJGCf/a
                +IAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIA
                IA/9Df4gAgAgAgAgAgAgAgAgAgAgAgAgAgBizBmjLeU8MnY3mrH8Fyzgt
                ieauIcyVMmSlKyDkvMGMYaVy/8AzeZKykwvKk2WsXQtNeZagHCtn5jgfB
                +FzrpqniBW1CdPXwkjt0je0fBNvDPOtWwkM9FKpJ/0advjBDK/wBJNh1P
                +xBtVO0sXh0vwan6KCzdicYxz2Kw3BsUxKm4STaqooMOnVsmlq+JOVMKp
                +klzKtGtPALK5DAsOZ3haSCWISQKAlJYvdmaMJeCPaL/8Ay6P8yp8xaOQ
                k0lBfYG56RTHwSbRle2LpZf6up8wK/4R7NHhfgLlAagoj/pQ8Ltktu+GA
                +kczQFEfqHyfYO55oeF2zkOB9WEnbtFX2JNoLXF0/wAVV+YLMP8ACOZnd
                bdPT9qq/MOVHpHsyqBI4E5NLDVpHNHwsK26kJ+rr6WvEfYk2h/ncL/wVX
```

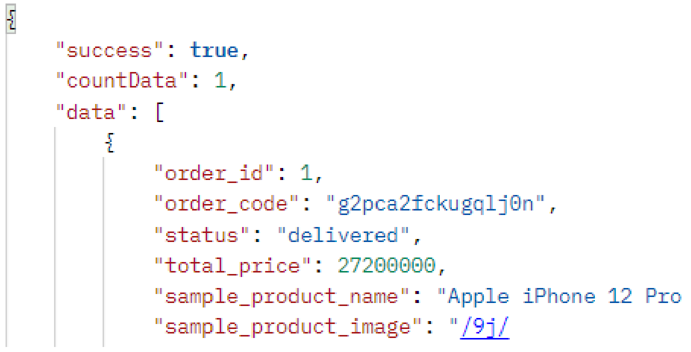
					
					`+4r6OYhaWWFoR6Q2djsJtXA09o4GanhKsbp81`
					`feBHrHOTMSQADtdz33+EQVx0HCXPIPituzOxP`
					`"sample_product_quantity": 2,`
					`"sample_product_price": 25000000,`
					`"product_total": 3`
					`}`
					`]`

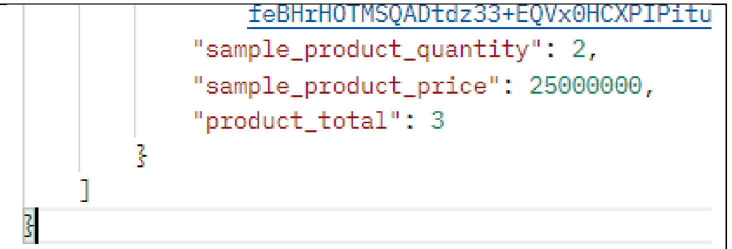
7	Update Order –
	
	Payment Date and Status **paid_off**
		API Url:
		http://localhost:3000/api/orders/update-order
		
		Note:
		This API is used to update the payment_date field and order status to be **paid_off** if the evidence is valid or update status to **packed** if the package is being packed, or update **recipient_number**, delivered date and status to **delivered** if it has been sent, and update status to **received** if the package has been received.
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
			order_id: 1,
			payment_date: 2021-10-07 18:44:56
			status: paid_off
				Pass	Return success message updated order data became **paid_off**:
					
					
					`{`
					`"success": true,`
					`"data": "Success updated Order data"`
					`}`
					
					id | order_code | payment_date | status
					1 | g2pca2fckugqlj0n | 2021-10-07 18:14:56 | paid_off

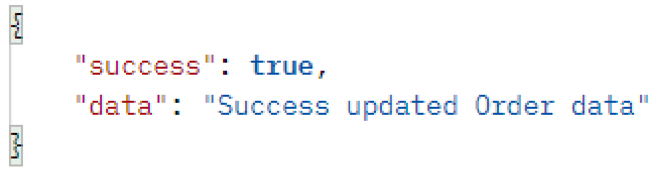
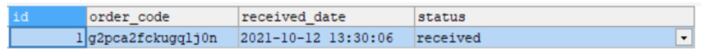
		3. Fill in order_id, payment_date, status 4. Send			

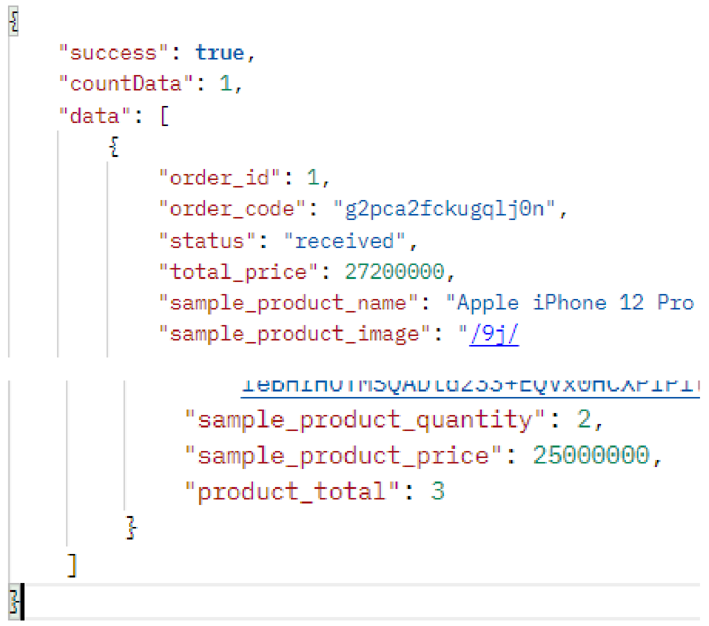
8	Tracking Status - Paid Off and waiting to Delivered	**API Url:**
		http://localhost:3000/api/orders/status-orders
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Token from Login API (Bearer Token)
		3. Fill in order_id, status
		4. Send
			order_id: 1
			status: paid_off / packed
				Pass	Return success message and order data based on status **paid_off** or **packed**:
					
					`{`
					`"success": true,`
					`"countData": 1,`
					`"data": [`
					`{`
					`"order_id": 1,`
					`"order_code": "g2pca2fckugqlj0n",`
					`"total_price": 27200000,`
					`"sample_product_name": "Apple iPhone 12 Pro 64GB, Purple",`
					`"sample_product_image": "/9j/`
					`"sample_product_quantity": 2,`
					`"sample_product_price": 25000000,`
					`"product_total": 3`
					`}`
					`]`
					`}`

9	Update Order –
	
	Delivered Date and Status **delivered**
		API Url:
		http://localhost:3000/api/orders/update-order
		
		Note:
		This API is used to update the payment_date field and order status to be **paid_off** if the evidence is valid or update status to **packed** if the package is being packed, or update
			order_id: 1,
			receipt_number: 130080013756519
			delivered_date: 2021-10-10 18:30:06
			status: delivered
				Pass	Return success message updated order data became **delivered**:
					
					`{`
					`"success": true,`
					`"data": "Success updated Order data"`
					`}`
					
					`id | order_code | delivered_date | status`
					`1 | g2pca2fckugqlj0n | 2021-10-10 18:30:06 | delivered`

		recipient_number, delivered date and status to **delivered** if it has been sent, and update status to **received** if the package has been received. **Step**: 1. Access the url 2. Fill in Authentication Header with Admin Token from Login API (Bearer Token) 3. Fill in order_id, delivered_date, status 4. Send			
10	Tracking Status - Delivered waiting to Received	**API Url:** http://localhost:3000/api/orders/status-orders **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in order_id, status 4. Send	**order_id:** 1 **status:** delivered	**Pass**	Return success message and order data based on status **delivered:**  { "success": true, "countData": 1, "data": [{ "order_id": 1, "order_code": "g2pca2fckugqlj0n", "status": "delivered", "total_price": 27200000, "sample_product_name": "Apple iPhone 12 Pro "sample_product_image": "/9j/

					
					`feBHrHOTMSQADtdz33+EQVx0HCXPIPitu`
					`"sample_product_quantity": 2,`
					`"sample_product_price": 25000000,`
					`"product_total": 3`
					`}`
					`]`
					`}`

11	Update Order –
	
	Received Date and Status **received**
		API Url:
		http://localhost:3000/api/orders/update-order
		
		Note:
		This API is used to update the payment_date field and order status to be **paid_off** if the evidence is valid or update status to **packed** if the package is being packed, or update **recipient_number**, delivered date and status to **delivered** if it has been sent, and update status to **received** if the package has been received.
		
		Step:
		1. Access the url
		2. Fill in Authentication Header with Admin Token from Login API (Bearer Token)
			order_id: 1,
			received_date: 2021-10-12 13:30:06
			status: received
				Pass	Return success message updated order data became **received**:
					
					`{`
					`"success": true,`
					`"data": "Success updated Order data"`
					`}`
					
					id | order_code | received_date | status
					1 | g2pca2fckugqlj0n | 2021-10-12 13:30:06 | received

		3. Fill in order_id, received_date, status 4. Send			
12	Tracking Status - Received	**API Url:** http://localhost:3000/api/orders/status-orders **Step**: 1. Access the url 2. Fill in Authentication Header with Token from Login API (Bearer Token) 3. Fill in order_id, status 4. Send	**order_id:** 1 **status:** received	**Pass**	Return success message and order data based on status **received:** 

```
{
    "success": true,
    "countData": 1,
    "data": [
        {
            "order_id": 1,
            "order_code": "g2pca2fckugqlj0n",
            "status": "received",
            "total_price": 27200000,
            "sample_product_name": "Apple iPhone 12 Pro
            "sample_product_image": "/9j/
                [illegible]
            "sample_product_quantity": 2,
            "sample_product_price": 25000000,
            "product_total": 3
        }
    ]
}
```

Open and Closed Issues on This Document

Open Issues

ID	Issue	Resolution	Responsibility	Due Date	Edit Date

Closed Issues

ID	Issue	Resolution	Responsibility	Due Date	

Addition

No	Process Name	Description	Data Require	Pass/ Fail	Result Expectation
1.					
2.					
3.					
4.					
5.					

Appendix