SYOPI
An E-Commerce Web Service

IS213 Project

Enterprise Solution Development

G3 - T1

Alexander Vincent Lewi

Emily Aurelia

Jordian Renaldi

Vincent Wesley

Vitto Surya Tedja

Yozafard Harold Siauheming

Introduction

Our project, Syopi, aims to tackle how an e-commerce site with lots of moving components would work in a microservice architecture. The three main actors in an e-commerce system are customers, sellers and couriers. Our user scenarios revolve around these three actors.

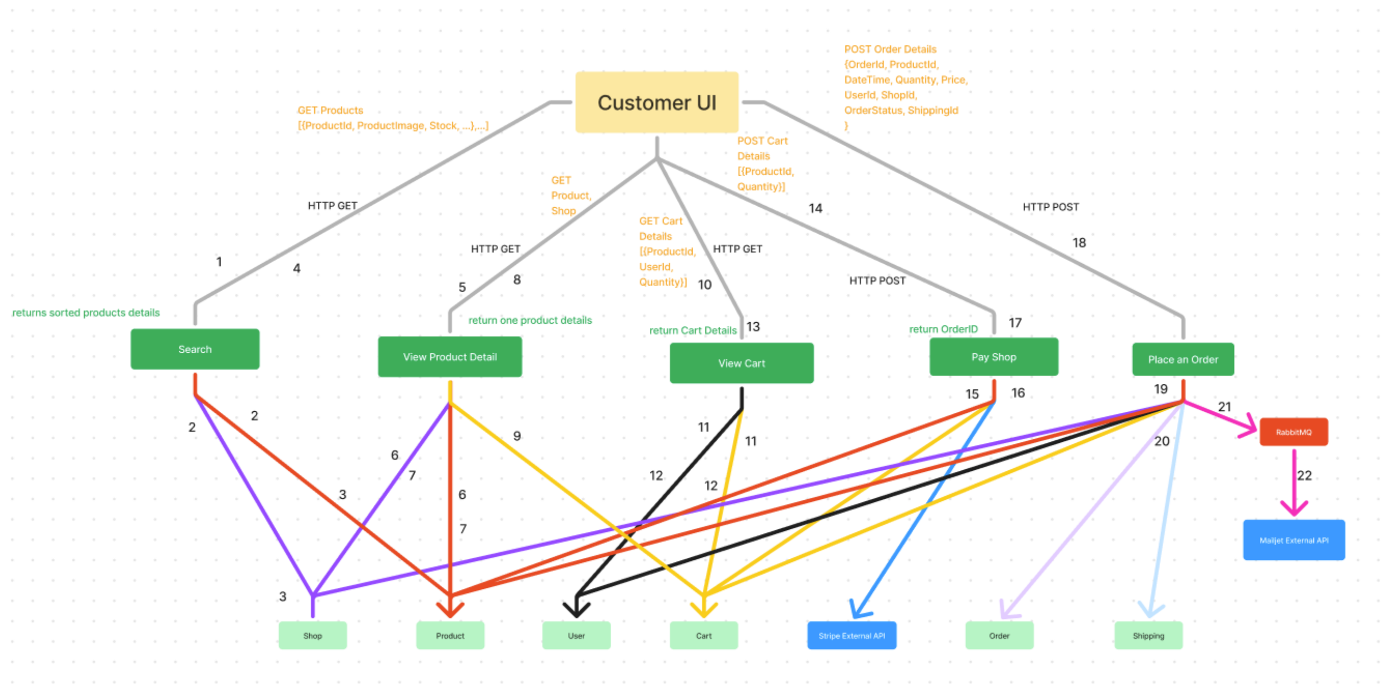
The first scenario is focused on how a customer would land on our application and browse the product that they like and then make payment and officially create an order in our application.

Our second scenario is focused on the user who can sign up as a seller and set up their own shop, add products and then later receive orders from customers.

Lastly, our third scenario follows along the storyline where the product that a user has bought arrived safely into their doorstep and they wanted to rate the product, letting the seller know how well they perform.

User Scenarios

Customer browse and purchase product


Customer UI
GET Products [{ProductId, ProductImage, Stock, ...},...]
HTTP GET
1
4
returns sorted products details
Search
GET Product, Shop
HTTP GET
5
8
return one product details
View Product Detail
GET Cart Details [{ProductId, UserId, Quantity}]
HTTP GET
10
return Cart Details
13
View Cart
POST Cart Details [{ProductId, Quantity}]
14
HTTP POST
return OrderID
17
Pay Shop
POST Order Details {OrderId, ProductId, DateTime, Quantity, Price, UserId, ShopId, OrderStatus, ShippingId }
HTTP POST
18
Place an Order
2
2
3
3
6
7
6
7
9
11
11
12
12
15
16
19
20
21
22
RabbitMQ
Mailjet External API
Shop
Product
User
Cart
Stripe External API
Order
Shipping

Customer browse and purchase product - Microservice Interaction Diagram

Customer browse and purchase product:

1. When UI loads, it calls Search Complex Microservice to get recommended products
2. Search will call product and store to get the necessary information that they need to recommend best products to the customer
3. Both microservices will send the data back by HTTP GET
4. Search returns back 10 most recommended products
5. When user clicks on one of the product card, UI request for View Product Detail Microservice
6. View Product Detail will fetch necessary information from product and shop
7. Product and Shop returns back needed data
8. View Product Details return product details to UI
9. When user click add to cart, View Product detail POST a request to cart containing ProductID and Quantity
10. When customer clicks on cart button, View Cart is called to fetch the necessary details
11. View Cart calls user and Cart to find a specific user's cart
12. User and Cart sends back data

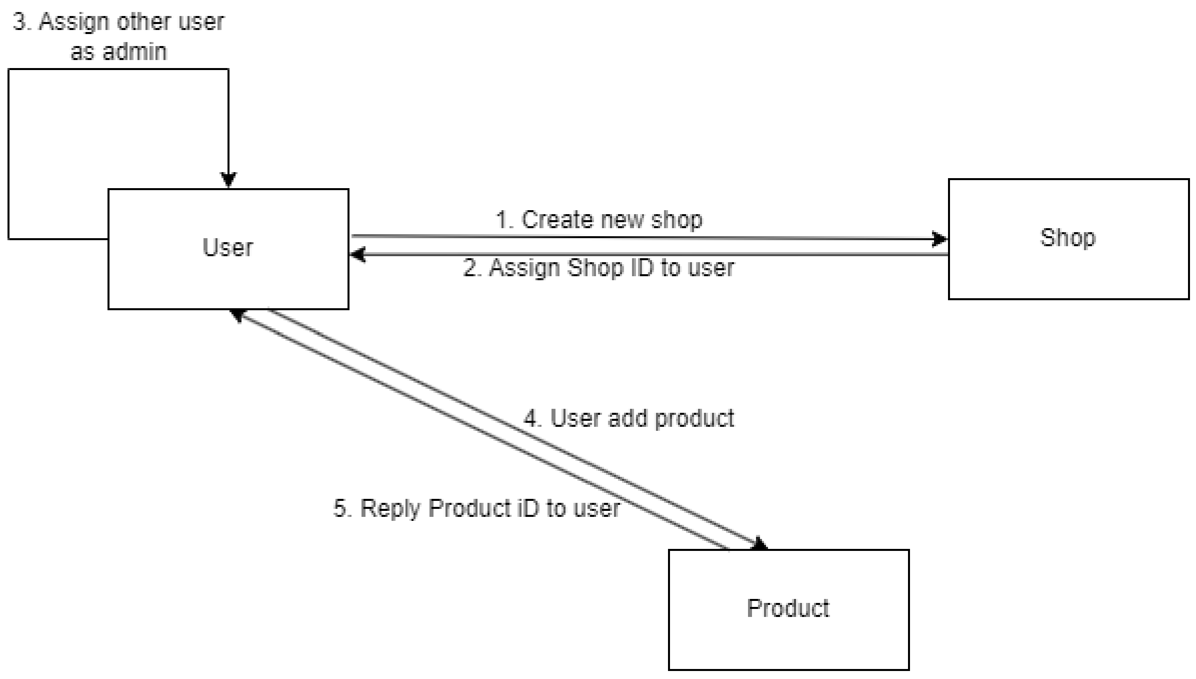
13. View Cart shows detail of Cart
14. When user press Checkout, Pay Shop is called
15. Pay Shop calls product, cart, and Stripe API to create payment Intent
16. All microservice returns necessary data
17. Pay Shop returns payment_intent as OrderID and redirects to success page
18. UI calls Place An Order
19. Place Order calls Shop, Product, User, Order and Shipping to create an order record, and at the same time removes all data in user's cart
20. Order Record is created
21. Order Record is sent to RabbitMQ to be snet to Mailjet
22. Mailjet sends email confirmation to user

External Services

Service Name	Description of the functionality	Link to external web pages that describe the detailed inputs/outputs
Supabase	*Database to store User, Order, Shop and Product information.*	https://supabase.com/
Stripe	*Make Secure Payment and create OrderID*	https://stripe.com/docs/api
Mailjet	*Send Order Info through Mail*	https://dev.mailjet.com/

Beyond the Labs

[User Scenario 2]


3. Assign other user as admin
User
1. Create new shop
2. Assign Shop ID to user
Shop
4. User add product
5. Reply Product iD to user
Product

User Opens Shop and Add Products - Microservice Interaction Diagram

User Opening a Shop and Adding Products:

1. User starts the process by opening up the user's page and clicking on a button, the button will then prompt a form for the necessary informations such as shop name, [insert some other info here]
2. After submitting the form, the UI will invoke the Shop microservice to assign a Shop ID to the user.
3. The user can now access the shop. User goes to their shop page and clicks on a button to add a new product. The button will prompt a form to fill in product details.
4. User fills in the form and after submitting, it will invoke the Product microservice to reply a Product ID to the user.

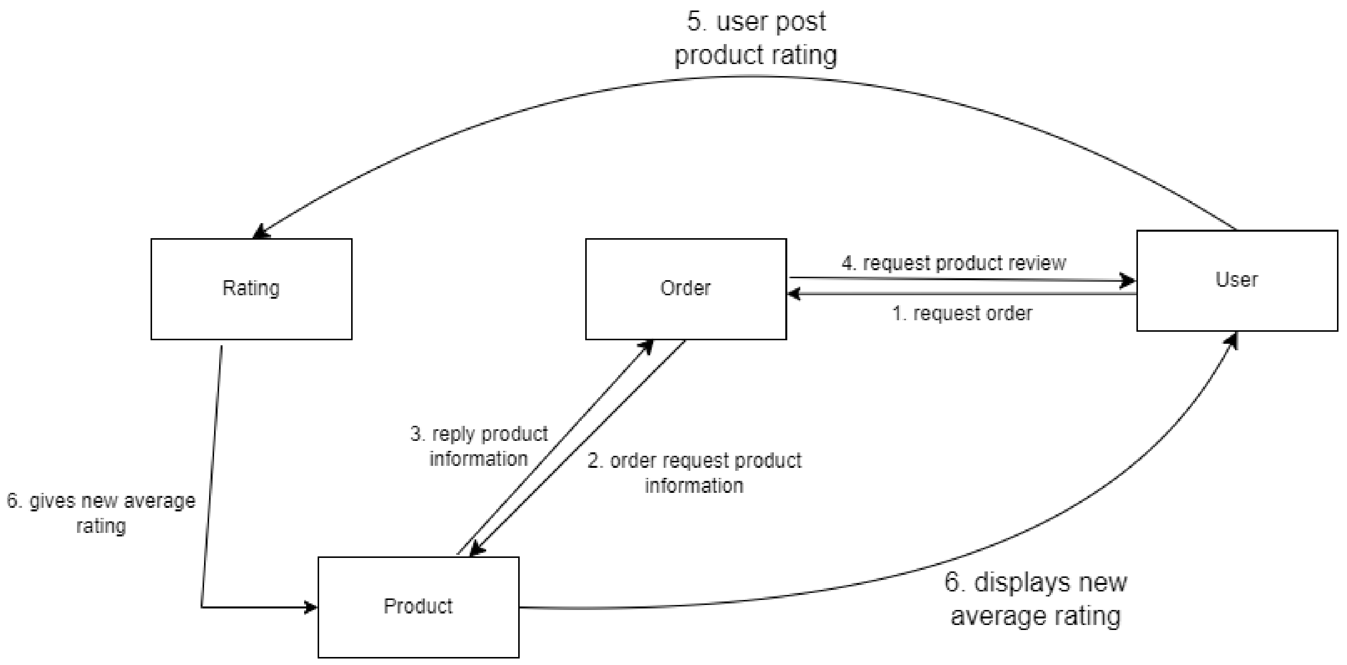
External Services

*[This is where you put in the details of any external service(s) that are **used in this User Scenario ONLY**]*

Service Name	Description of the functionality	Link to external web pages that describe the detailed inputs/outputs
Supabase	*Database to store User, Order, Shop and Product information.*	*https://supabase.com/*

Beyond the Labs

[User Scenario 3]



5. user post product rating
Rating
Order
User
4. request product review
1. request order
3. reply product information
2. order request product information
6. gives new average rating
Product
6. displays new average rating

User reviews product after it arrives - Microservice Interaction Diagram

User reviews product after it arrives:

1. User receives the product they ordered and wants to rate it, they go to the product's order page
2. it triggers the user ui to request order
3. order requests product microservice for product information
4. product microservice replies with product information
5. order requests product review from user
6. user gives product rating
7. it invokes the rating microservice to give new average rating to product
8. product displays new average rating

External Services

Service Name	Description of the functionality	Link to external web pages that describe the detailed inputs/outputs
Supabase	*Database to store User, Order, Shop and Product information.*	*https://supabase.com/*

Beyond the Labs

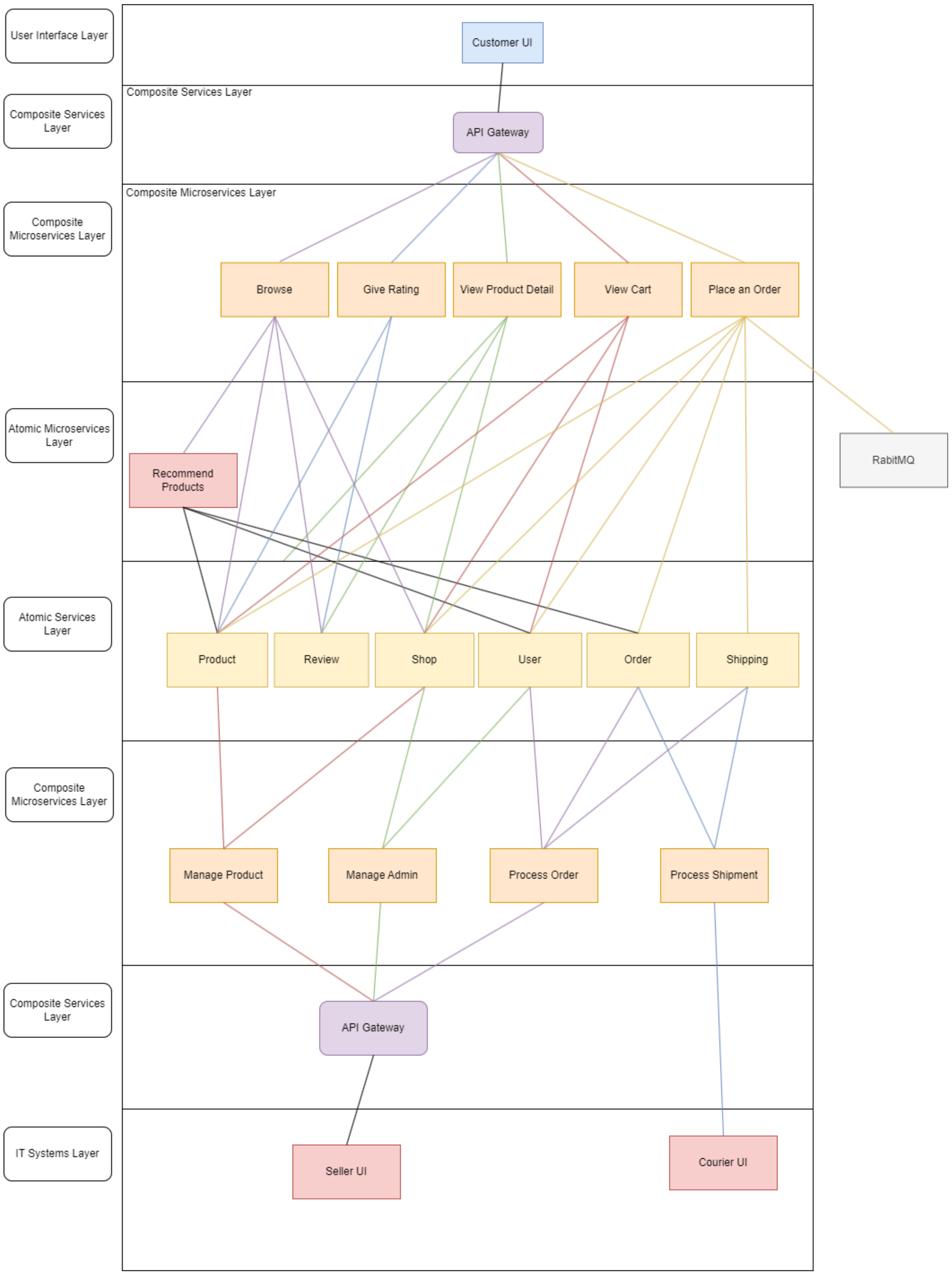
1.

Remaining Beyond the Labs not covered above

-

Appendix - Technical Overview Diagram(s) or SOA Layer Diagram(s)

(The appendix is not counted towards the page limit. It is not explicitly graded, but will be used to verify against your solution as described in the report if required.)



Appendix - API Docs

Order; HTTP-based API, AMQP-based API:

GET **/order/get_all_order** – Get all orders	
Parameters	No parameters
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
[
 {
  "DateTime": "2023-04-03T10:02:54.311233+00:00",
  "OrderId": "pi_3MskDYBJIMpkY9J20VeP9YY1",
  "OrderStatus": "Paid",
  "Price": 8.36,
  "ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2",
  "Quantity": 1,
  "ShippingId": "52b66f93-adaa-40e1-9968-3144d0e483a1",
  "ShopId": "9413c28a-3b0b-4955-9ac9-5171a3f8631d",
  "UserId": "1"
 },
 {
  "DateTime": "2023-04-03T10:02:54.327096+00:00",
  "OrderId": "pi_3MskDYBJIMpkY9J20VeP9YY1",
  "OrderStatus": "Paid",
  "Price": 7.77,
  "ProductId": "af326675-96e0-47f5-9f07-63e97db8098a",
  "Quantity": 3,
  "ShippingId": "2ae4c2a4-e0c5-42ff-8923-54a808d7950d",
  "ShopId": "9413c28a-3b0b-4955-9ac9-5171a3f8631d",
  "UserId": "1"
 },
 {
  "DateTime": "2023-04-03T10:14:53.230732+00:00",
  "OrderId": "pi_3MskPDBJIMpkY9J21aABW8lS",
  "OrderStatus": "Accepted",
  "Price": 58.52,
  "ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2",
  "Quantity": 7,
  "ShippingId": "7b484b6b-bb33-41ac-9130-1ff6430fa537",
  "ShopId": "9413c28a-3b0b-4955-9ac9-5171a3f8631d",
  "UserId": "1"
 }
]
```

Code – 404	Description – no orders Example Value

```
{
 "code": 404,
```

	`"message": "There are no orders." }`
GET **/order/find_by_orderid/<string:OrderId>** – Find order by order id	
Parameters	
Name	<string:OrderId>
OrderId string *(path)*	Order id of order to return Example Value pi_3MskDYBJIMpkY9J20VeP9YY1
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value `[{ "DateTime": "2023-04-03T10:02:54.311233+00:00", "OrderId": "pi_3MskDYBJIMpkY9J20VeP9YY1", "OrderStatus": "Paid", "Price": 8.36, "ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2", "Quantity": 1, "ShippingId": "52b66f93-adaa-40e1-9968-3144d0e483a1", "ShopId": "9413c28a-3b0b-4955-9ac9-5171a3f8631d", "UserId": "1" }, { "DateTime": "2023-04-03T10:02:54.327096+00:00", "OrderId": "pi_3MskDYBJIMpkY9J20VeP9YY1", "OrderStatus": "Paid", "Price": 7.77, "ProductId": "af326675-96e0-47f5-9f07-63e97db8098a", "Quantity": 3, "ShippingId": "2ae4c2a4-e0c5-42ff-8923-54a808d7950d", "ShopId": "9413c28a-3b0b-4955-9ac9-5171a3f8631d", "UserId": "1" }]`
Code – 404	Description – Order not found Example Value `{ "code": 404, "message": "Order not found." }`

POST **/order/create_order** – Create order	
Parameters	No parameters
Name	Description

body
object
(body)
	Order to create
	
	Example Value
	
	{
	"OrderId": "1k2s8fj-gs6aga3kshk",
	"ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2",
	"ShopId": "52b66f93-adaa-40e1-9968-3144d0e483a1",
	"Quantity": "1",
	"Price": "2.81",
	"UserId": "1"
	
	}

Responses	Content type – application/json

Code – 201	Description – successful operation
	
	Example Value
	
	{
	"DateTime": "2023-04-03T17:31:33.199441+00:00",
	"OrderId": "1k2s8fj-gs6aga3kshk",
	"OrderStatus": "Unpaid",
	"Price": 2.81,
	"ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2",
	"Quantity": 1,
	"ShippingId": "3cef98f4-69ff-44bf-9d5c-61c1a2dbd9aa",
	"ShopId": "52b66f93-adaa-40e1-9968-3144d0e483a1",
	"UserId": "1"
	}

Code – 500	Description – An error occurred creating the order
	
	Example Value
	
	{
	"code": 500,
	"message": "An error occurred while creating the order.
	}

POST **/order/accept** – Update order status

Parameters	No parameters
Name	Description
body object *(body)*	Order to create Example Value `{ "OrderId": "1k2s8fj-gs6aga3kshk", "OrderStatus": "Paid" }`
Code – 200	Description – successful operation Example Value `[{ "DateTime": "2023-04-03T17:35:25.382702+00:00", "OrderId": "1k2s8fj-gs6aga3kshk", "OrderStatus": "Paid", "Price": 2.81, "ProductId": "08d09581-e2b0-451c-83eb-0081d98b20c2", "Quantity": 1, "ShippingId": "de08fe31-e1d0-4fe4-a8c6-34f2a24c17f0", "ShopId": "52b66f93-adaa-40e1-9968-3144d0e483a1", "UserId": "1" }, { "DateTime": "2023-04-03T17:53:20.088433+00:00", "OrderId": "1k2s8fj-gs6aga3kshk", "OrderStatus": "Paid", "Price": 2.81, "ProductId": "08d09581-e2b0-451c-83eb-0031d98b20c2", "Quantity": 1, "ShippingId": "ac70a6db-d52b-4a5c-9fc0-0c08aac864b5", "ShopId": "52b66f93-adaa-40e1-9968-3144d0e483a1", "UserId": "1" }]`
Code – 404	Description – no orders Example Value `{ "code": 404, "message": "There are no orders." }`

Shop; HTTP-based API:

GET **/shop** – Get all shops	
Parameters	No parameters
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
{
  "code": 200,
  "data": [
    {
      "IsActive": "Active",
      "ShopAddress": "234 Maple St.",
      "ShopId": "b5590f6b-d95e-4675-8ea6-ec1ff54af4f4",
      "ShopName": "Target",
      "ShopPhoneNumber": "555-3456"
    },
    {
      "IsActive": "Active",
      "ShopAddress": "789 Oak St.",
      "ShopId": "885cd9dd-ed92-4f12-8c56-315773df07c8",
      "ShopName": "Walgreens",
      "ShopPhoneNumber": "555-9012"
    },
    {
      "IsActive": "Active",
      "ShopAddress": "\"PIK\"",
      "ShopId": "a22863a2-6efb-40b2-9279-37bcf03a37c2",
      "ShopName": "\"Sanur Mangga Dua\"",
      "ShopPhoneNumber": "\"999\""
    },
    {
      "IsActive": "Active",
      "ShopAddress": "\"PIK\"",
      "ShopId": "93a77f29-cabd-41f7-b2f1-81d3b70d601f",
      "ShopName": "\"Sanur Mangga Dua2\"",
      "ShopPhoneNumber": "\"999\""
    }
  ],
  "message": "Returning all shops"
}
```

Code – 404	Description – No products found Example Value

```
{
                'message': 'No products found.',
                "data": None,
                "code": 404
            }
```


GET **/shop/search/<string: search_term>** – Searches for shops that match the search term.

Parameters	
Name	Description
search_term (string) (limit:10)	The search term to be used for searching shops Example Value “Maple”
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
{
  "code": 200,
  "data": [
    {
      "IsActive": "Active",
      "ShopAddress": "234 Maple St.",
      "ShopId": "b5590f6b-d95e-4675-8ea6-ec1ff54af4f4",
      "ShopName": "Target",
      "ShopPhoneNumber": "555-3456"
    },
```

Code – 404	Description – Order not found Example Value

```
{
```

```
            “data”: None,
            "code": 404,
            "message": "No shop matching search term
found"
            })
```

POST **/shop/<string:name>** – Retrieves a specific shop by name.

Parameters	No parameters
Name	Description
name *(string)*	The name of the shop to be retrieved Example Value “Shop A”
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
{
  {
  "code": 200,
  "message": "Store is found",
  "data": [{
    "ShopId": 1,
    "ShopName": "Shop A",
    "ShopAddress": "123 Main St",
    "ShopPhoneNumber": "123-456-7890",
    "IsActive": "Active"
  }]
}
```

Code – 404	Description – No matching store found Example Value

```
{
                "code": 404,
                "message": "Store not found.",
                "data": None
            }
```

POST **/shop/add_shop** – Adds a new shop to the database.

Parameters	No Parameters
body object *(body)*	Object to be added to database Example Value

```
{
                “shopName”: “Koufu”,
                "address": “83 Prinsep Street”,
                "phone": “88899730”
            }
```

Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
({
             “code”: 200,
             "data": {
             "ShopAddress": form_address,
             "ShopPhoneNumber": form_phone_number,
             "IsActive": "Active"
                      },
                      'message': 'Insertion succesfull.'
 }, 200)
```

Code – 500	Description – Error inserting shop into database Example Value

```
({
                        'code': 500,
                        'data': None,
                        'message': 'Error inserting shop
into database.'
                    }), 500)
```

POST, GET **/shop/update_shop/<string: name>** – Updates shop in database	
Parameters	No Parameters
name (string)	The name of the shop to update
body object *(body)*	Object to be updated in the database Example Value

```
{
                "shopName": "Koufu",
                "address": "83 Prinsep Street",
                "phone": "88899730"
            }
```

Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
({
             "code": 200,
             "data": {
             "ShopAddress": form_address,
             "ShopPhoneNumber": form_phone_number,
             "IsActive": "Active"
                      },
                      'message': 'Store successfully
updated'
 }, 200)
```

Code – 500	Description – Error updating shop in the database Example Value

```
({
                        'code': 500,
                        'data': None,
                        'message': 'Error inserting shop
into database.'
                    }), 500)
```

Code - 404	Description - Shop not found Example Value

```
{
                    "code": 404,
                    "data": None,
                    "message": "Shop not found."
                }), 404
```

Shipping: HTTP-based API:

GET **/shipping/getall_shipping** – Get all shippings	
Parameters	No parameters
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	"CourierId": "NinjaVan",
	"CustomerAddress": "b",
	"DriverId": "1",
	"ShippingId": "50b30877-c8fa-4204-a11b-b9008a561aee",
	"ShippingStatus": "With Courier",
	"ShopAddress": "a"
	},
	{
	"CourierId": "c",
	"CustomerAddress": "b",
	"DriverId": "d",
	"ShippingId": "70f00af7-ab04-4313-9a61-63d467a7c5ab",
	"ShippingStatus": "Delivered",
	"ShopAddress": "a"
	}

Code – 404	Description – no orders
	
	Example Value
	
	{
	"code": 404,
	"message": "There are no orders."
	}

GET **/shipping/find_by_shippingid/<string:ShippingId>** – Find by shipping id	
Parameters	
Name	Description
ShippingId integer *(path)*	ShippingId Example Value 50b30877-c8fa-4204-a11b-b9008a561aee
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

	{
	"CourierId": "NinjaVan",
	"CustomerAddress": "b",
	"DriverId": "1",
	"ShippingId": "50b30877-c8fa-4204-a11b-b9008a561aee",
	"ShippingStatus": "With Courier",
	"ShopAddress": "a"
	}

Code – 404	Description – Order not found
	
	Example Value
	
	{
	"code": 404,
	"message": "Shipping Id not found."
	}

GET **/shipping/find_by_driverid/<string:DriverId>** – Find by driver id	
Parameters	
Name	Description
DriverId integer *(path)*	DriverId Example Value d
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	[
	{
	"CourierId": "c",
	"CustomerAddress": "b",
	"DriverId": "d",
	"ShippingId": "70f00af7-ab04-4313-9a61-63d467a7c5ab",
	"ShippingStatus": "Delivered",
	"ShopAddress": "a"
	},
	{
	"CourierId": "c",
	"CustomerAddress": "b",
	"DriverId": "d",
	"ShippingId": "c3daabb6-b05a-4831-a466-757a42cd5ce6",
	"ShippingStatus": "Successful",
	"ShopAddress": "a"
	}
	]

Code – 404	Description – Driver Id not found
	
	Example Value
	
	{
	   "code": 404,
	   "message": "Driver Id not found."
	}

GET **/shipping/getall_shipping_by_status/<string:status>** – Find by shipping status	
Parameters	
Name	Description
status integer *(path)*	Shipping Status Example Value Delivered
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	      "CourierId": "c",
	      "CustomerAddress": "b",
	      "DriverId": "d",
	      "ShippingId": "70f00af7-ab04-4313-9a61-63d467a7c5ab",
	      "ShippingStatus": "Delivered",
	      "ShopAddress": "a"
	   }

Code – 404	Description – Status not found
	
	Example Value
	
	{
	  "code": 404,
	  "data": {
	    "order_id": "<order_id>"
	  },
	  "message": "Status not found."
	}

POST **/shipping/accept** – Accept shipping	
Parameters	No parameters
Name	Description
body object *(body)*	Shipping status to change Example Value { "ShippingId": "c3daabb6-b05a-4831-a466-757a42cd5ce6", "ShippingStatus": "Delivered" }
Responses	Content type – application/json
Code – 201	Description – successful operation Example Value { "CourierId": "c", "CustomerAddress": "b", "DriverId": "d", "ShippingId": "c3daabb6-b05a-4831-a466-757a42cd5ce6", "ShippingStatus": "Delivered", "ShopAddress": "a" }
Code – 500	Description – An error occurred accepting shipping Example Value { "code": 500, "message": "An error occurred while accepting shipping.“ }

Search: HTTP-based API

GET **/recommender** – Get recommended products	
Parameters	No parameters
Responses	Content type – application/json
Code – 200	Description – successful operation

	Example Value
	
	`{`
	`  "code": 200,`
	`  "data": {`
	`    "orders": [{`
	`      "created": "Fri, 12 Jun 2020 02:14:55 GMT",`
	`      "customer_id": "Apple TAN",`
	`      "modified": "Fri, 12 Jun 2020 02:14:55 GMT",`
	`      "order_id": 1,`
	`      "order_item": [{`
	`        "book_id": "9781434474234",`
	`        "item_id": 1,`
	`        "order_id": 1,`
	`        "quantity": 1`
	`      },`
	`      {`
	`        "book_id": "9781449474212",`
	`        "item_id": 2,`
	`        "order_id": 1,`
	`        "quantity": 1`
	`      }],`
	`      "status": "NEW"`
	`    }]`
	`  }`
	`}`

Code – 404	Description – no orders
	
	Example Value
	
	`{`
	`  "code": 404,`
	`  "message": "There are no orders."`
	`}`

Product: HTTP-based API

GET **/product** – Get all products	
Parameters	No parameters
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	`{`
	`    "AmountSold": 0,`

```
      "AvgRating": 0,
      "Category": null,
      "Description": "Material: Synthetic LeatherLining: Polyester3 Colors: Pink, Grey, BlackItem Type: Women BackpacksStyle: FashionShape: SquareBag Size: Approx. 24 x 11 x 30cm/9.4 x 4.3 x 11.8inch (L x W x H)Strap Size: AdjustableClosure Type: ZipperCapacity: Below 20 LitreHandle/Strap Type: Soft HandleBackpacks size: MiddleQuantity: 1 PieceGender: Women, GirlsPattern: PlaidsFeatures: Adjustable StrapsOccasion: Fashion, Casual, School, Travel, Hiking, etc.Features:A lovely bag full of personality, so fresh and unique.Large capacity for books, umbrella, water cup, cosmetic and pocket money, phone, etc.Adjustable shoulders straps to meet different requirements for both girls and women.Each color is soft and comfortable.Ideal for fashion, casual, travel, hiking, etc.Package Content: 1 x Women Backpacks",
      "ImageUrls": [
        {
          "ImageUrl":
"https://cf.shopee.sg/file/fc8c432bca16a8b000c04790f6a24269"
        },
        {
          "ImageUrl":
"https://m.media-amazon.com/images/W/IMAGERENDERING_521856-T1/images/I/619PYbOnuHL._AC_UY500_.jpg"
        }
      ],
      "Price": 17.99,
      "ProductId": "04de3254-4a8f-4948-b1dd-e3580bf63510",
      "ProductName": "Women Backpack Synthetic Leather Backpacks Softback Bags Preppy Style Bag",
      "ShopId": "3132abdc-b1d2-45b7-b097-8a6120ada330",
      "Stock": 300
    },
    {
      "AmountSold": 0,
      "AvgRating": 0,
      "Category": null,
      "Description": "Material: LycraColor: Black, WhiteStyle: Sexy BriefsPattern: FloralDecor: LaceDesign: Low Rise, No Panty Line, Bikini Occasion: CasualGarment Care: HandWash(Max 40°C)Good quality fabric, and making style make you feel good and comfortable when wearing.Lightweight and comfortable construction.It can offer a soft and smooth feel.Cotton gusset,Full rear coveragePackage Content: 1 x Women Briefs",
      "ImageUrls": [
        {
          "ImageUrl":
"https://m.media-amazon.com/images/I/811U9nwWBNL._UL1500_.jpg"
        },
        {
          "ImageUrl":
"https://n1.sdlcdn.com/imgs/h/1/2/New-Womens-Floral-Lace-Sexy-SDL165522391-1-0ff09.JPEG"
        },
        {
          "ImageUrl":
"https://n1.sdlcdn.com/imgs/h/1/s/New-Womens-Floral-Lace-Sexy-SDL607750809-1-8a896.JPEG"
```

	}], "Price": 7.69, "ProductId": "b30ea76a-fc27-43e0-944d-067774e837f4", "ProductName": "Women Floral Lace Sexy Comfortable Low Rise No Panty Line Bikini Underwear Brief", "ShopId": "3132abdc-b1d2-45b7-b097-8a6120ada330", "Stock": 800 }
Code – 404	Description – no orders Example Value { "code": 404, "message": "There are no orders." }
GET **/product/search/<string:keyword>** – Search for product	
Parameters	
Name	<string:keyword>
keyword string *(path)*	keyword Example Value Golden
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value {"value":"3f167525-511b-4601-86d7-5990c914ff08","label":"Golden Number (M1)"},{"value":"08d948dd-7186-482f-b93f-ae1a546ae237","label":"Golden Number (M1)"},{"value":"1957516a-8b3b-473c-a03d-6010f5e9b90e","label":"Golden Mobile Numbers"},{"value":"f0a6f0cf-b67f-4583-b7d0-f38ab0f1a496","label":"Golden Mobile Numbers"},{"value":"80f9e035-0a04-4e59-bfc5-4fe72c1b9bb4","label":"Fashion Women Golden Plated Asymmetric Triangle Round "}
Code – 404	Description – Product not found Example Value { "code": 404, "message": "Order not found."

	}

GET **/product/<string:ProductId>** – Search for product

Parameters	
Name	<string:ProductId>
ProductId string *(path)*	ProductId Example Value 1957516a-8b3b-473c-a03d-6010f5e9b90e
Responses	Content type – application/json
Code – 200	Description – successful operation Example Value

```
{
    "AmountSold": 0,
    "AvgRating": 0,
    "Category": null,
    "Description": "Unregistered prepaid cards. Can port to post paid anytime1) 8144 0444  $3882) 8208 7788  $3883) 9246 2222  $3884) 8132 8000 $3005) 8132 8188  $3886) 8134 1688  $3887) 8158 5888  $6888) 8208 5588 $5889) 8208 6868  $48810) 8155 0001  $288Price negotiablePlease contact me @ 9815 2128",
    "ImageUrls": [
      {
        "ImageUrl": "https://media.karousell.com/media/photos/products/2019/04/30/golden_mobile_numbers_for_sale_postpaidprepaid_starhub_m1_singtel_golden_numbers_for_sale_1556593310_a5ecfe030_progressive"
      },
      {
        "ImageUrl": "https://media.karousell.com/media/photos/products/2023/3/7/golden_numbers_for_mobile_phon_1678200747_d1c1ba27_progressive_thumbnail.jpg"
      }
    ],
    "Price": 8,
    "ProductId": "1957516a-8b3b-473c-a03d-6010f5e9b90e",
    "ProductName": "Golden Mobile Numbers",
    "ShopId": "5a836c82-8a4a-40b7-ab52-ffac391b43bd",
    "Stock": 1
  }
```

Code – 404	Description – Product not found Example Value

```
{
  "code": 404,
```

	"message": "Order not found." }
POST **/product** – Create product	
Parameters	No parameters
Name	Description
body object *(body)*	Product to create Example Value { "ProductName": "RedBull", "ShopId": "5a836c82-8a4a-42b7-ab52-ffac391s43bd", "Category": "Drinks", "Description": "Energy Drink", "Stock": "2", "Price": "1" }
Responses	Content type – application/json
Code – 201	Description – successful operation Example Value [{ "AmountSold": 0, "AvgRating": 0, "Category": "Drinks", "Description": "Energy Drink", "Price": 1, "ProductId": "70918b26-5e2c-4fc3-9bdf-204cc12b084f", "ProductName": "RedBull", "ShopId": "5a836c82-8a4a-42b7-ab52-ffac391s43bd", "Stock": 2 }]
Code – 500	Description – An error occurred creating product Example Value { "code": 500, "message": "An error occurred while creating product." }
GET **/product/<string:ProductId>/<float:avgRating>** – Update rating	
Parameters	

Name	<string:keyword>, <float:avgRating>
ProductId string *(path)*	Product ID Example Value 1957516a-8b3b-473c-a03d-6010f5e9b90e
avgRating float *(path)*	Average rating Example Value 1.22
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	"AmountSold": 0,
	"AvgRating": 1.22,
	"Category": null,
	"Description": "Unregistered prepaid cards. Can port to post paid anytime1) 8144 0444 $3882) 8208 7788 $3883) 9246 2222 $3884) 8132 8000 $3005) 8132 8188 $3886) 8134 1688 $3887) 8158 5888 $6888) 8208 5588 $5889) 8208 6868 $48810) 8155 0001 $288Price negotiablePlease contact me @ 9815 2128",
	"Price": 8,
	"ProductId": "1957516a-8b3b-473c-a03d-6010f5e9b90e",
	"ProductName": "Golden Mobile Numbers",
	"ShopId": "5a836c82-8a4a-40b7-ab52-ffac391b43bd",
	"Stock": 1
	}

Code – 404	Description – Product not found
	
	Example Value
	
	{
	"code": 404,
	"message": "Error in Rating Update."
	}

Review: HTTP-based API

GET **/review** – Get all reviews	
Parameters	No parameters
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	`[`
	`  {`
	`    "Description": "product ini bagoes",`
	`    "OrderId": "pi_3MsQHqBJIMpkY9J21DXYT7Bn",`
	`    "ProductId": "10607759-99a3-45fa-8141-58465e1d5f97",`
	`    "Rating": 2,`
	`    "UserId": "1",`
	`    "created_at": "2023-04-03T08:11:08.428423+00:00"`
	`  },`
	`  {`
	`    "Description": "mencoba review insyaallah",`
	`    "OrderId": "pi_3MszalBJIMpkY9J20DHZveQb",`
	`    "ProductId": "10607759-99a3-45fa-8141-58465e1d5f97",`
	`    "Rating": 4,`
	`    "UserId": "1",`
	`    "created_at": "2023-04-04T04:03:33.204286+00:00"`
	`  },`
	`  {`
	`    "Description": "wow so sexy",`
	`    "OrderId": "pi_3Mt18jBJIMpkY9J21sZA2cjl",`
	`    "ProductId": "9f54c568-3d67-44b0-8ae2-ee7b22669c6c",`
	`    "Rating": 3,`
	`    "UserId": "1",`
	`    "created_at": "2023-04-04T04:07:33.016141+00:00"`
	`  },`
	`  {`
	`    "Description": "",`
	`    "OrderId": "pi_3Mt1jKBJIMpkY9J20Cl3S49i",`
	`    "ProductId": "d6013262-49e4-4db9-9923-d7363abbd6d6",`
	`    "Rating": 3,`
	`    "UserId": "1",`
	`    "created_at": "2023-04-04T04:46:43.060735+00:00"`
	`  }`
	`]`

GET **/review/<product_id>** – Find order by order id	
Parameters	
Name	Description
product_id string *(path)*	Order id of order to return Example Value

	1
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	  "Id": 1,
	  "Name": "John Doe",
	  "Email": "johndoe@gmail.com",
	  "Rating": 5,
	  "Comment": "This product is amazing!",
	  "ProductId": "123"
	 }

GET**/review/getreviewrating/<string:ProductId>** – Get rating for reviews of a product	
Parameters	No parameters
Name	Description
product_id string *(path)*	Order id of order to return Example Value 1
Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	  "Rating": 4.5
	 }

Code – 500	Description – An error occurred getting review rating
	
	Example Value
	
	[
	 {
	  "Rating": 2
	 },
	 {
	  "Rating": 4
	 }
	]

POST **/review/giverating** – Update order status	
Parameters	
Name	Description

body
object
(body)
	Order to update
	
	Example Value
	
	{
	  "Name": "John Doe",
	  "Email": "johndoe@gmail.com",
	  "Rating": 5,
	  "Comment": "This product is amazing!",
	  "ProductId": "123"
	 }

Responses	Content type – application/json

Code – 200	Description – successful operation
	
	Example Value
	
	{
	  "Name": "John Doe",
	  "Email": "johndoe@gmail.com",
	  "Rating": 5,
	  "Comment": "This product is amazing!",
	  "ProductId": "123"
	 }

Code – 404	Description – Order not found
	
	Example Value
	
	{
	  "code": 404,
	  "data": {
	    "order_id": "<order_id>"
	  },
	  "message": "Order not found."
	}

Code – 500	Description – An error occurred while giving rating
	
	Example Value
	
	{
	  "code": 500,
	  "data": {
	    "order_id": "<order_id>"
	  },

	"message": "An error occurred." } The part highlighted varies according to the error encountered.

Appendix - Technical Contribution Table

E.g. coding of feature X, deployment, configure database, etc.

Name	**Contributions**
Alexander Vincent Lewi	Product Microservices, process_order, recommender, search , database
Emily Aurelia	Shipping Microservice, Front End, view_cart, browse, database
Jordian Renaldi	Shop Microservices, Docker, Error, order_log, RabbitMq, database
Vincent Wesley	Order Microservices, report, proposal, shipping, database
Vitto Surya Tedja	Review Microservices, report, proposal, get_product_cart, give_rating, Payment, stripe API, view_cart, database
Yozafard Harold Siauheming	User Microservices, report, cart, authentication, database