

Tugas Besar

II3160 Teknologi Sistem Terdistribusi

*Milestone 6: Implementasi Final
Online Food Delivery (OFD)*



Disusun oleh :

Anggita Najmi Layali 18223122

**PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025**

A. Penjelasan

Dalam pengembangan sistem Online Food Delivery (OFD) ini, setiap aktor dalam platform memiliki serangkaian kebutuhan dan proses bisnis yang berbeda. Oleh karena itu, layanan dibagi menjadi beberapa kelompok endpoint sesuai bounded context, antara lain Customer, Driver, dan Restaurant. Pada tugas ini, bounded context yang diimplementasikan berfokus pada Customer saja.

Tautan repositori: https://github.com/gitaa001/II3160_TB_18223122.git

B. Penyesuaian dan Batasan

- **Implementasi context**

Implementasi API berfokus pada *Customer-facing Ordering Context*, sebagai realisasi dari model taktis yang sudah dirancang pada M03. Seluruh endpoint yang diimplementasikan mengoperasikan entitas Order, OrderItem, dan *value object* ScheduledTime yang sebelumnya telah dibangun dalam model domain M03. Seluruh endpoint dikelompokkan di bawah prefix /customer/order/... sebagai *public API surface* dari Ordering Context yang digunakan Customer.

- **Status order**

Pada implementasi M06 ini, siklus status pemesanan disesuaikan kembali menjadi:

PENDING → SCHEDULED → CHECKOUT → CANCELED/DELVERED

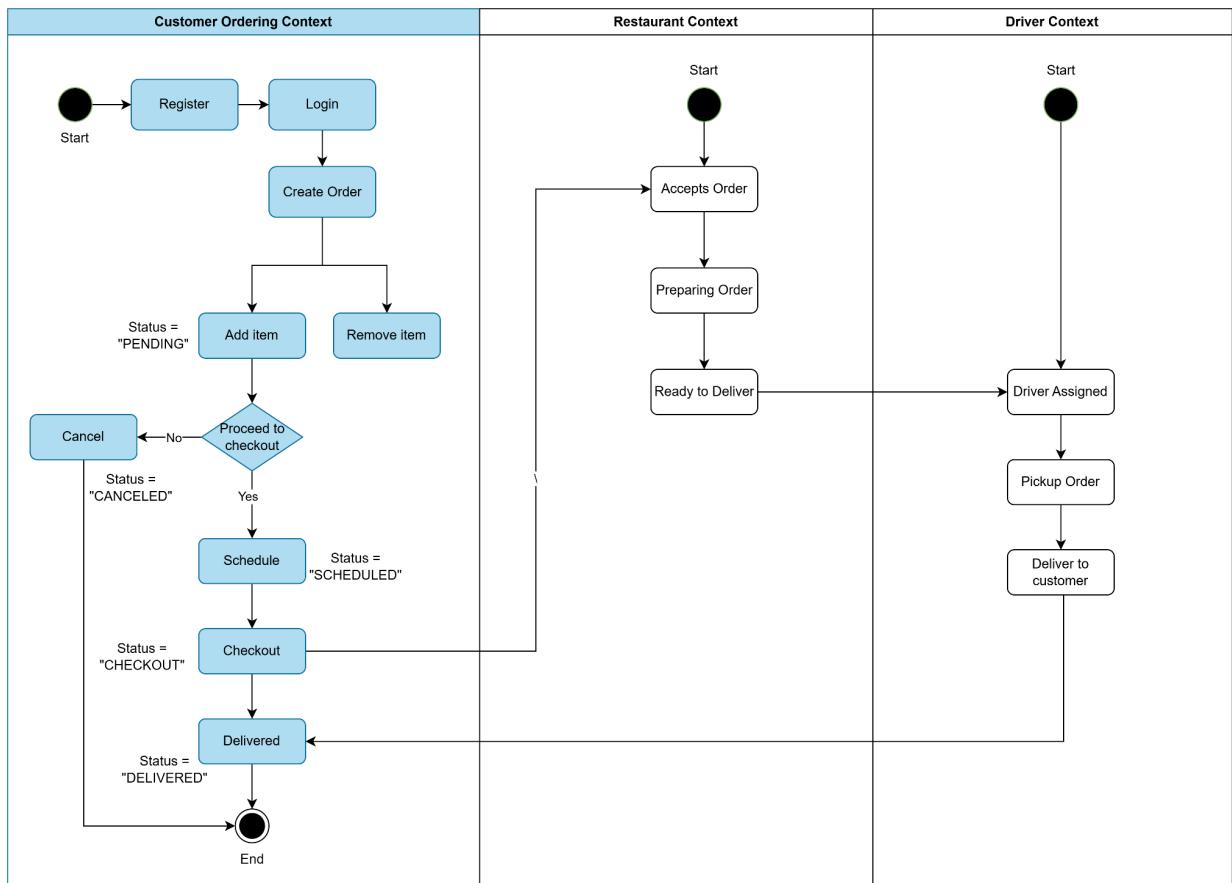
Pada milestone 3, model domain mencakup sejumlah status tambahan seperti *Approved*, *Preparing*, *Assigned*, *Delivering*, dan *Completed* untuk mencerminkan aktivitas lintas konteks antara Restaurant Context dan Driver (Delivery) Context. Namun, implementasi dibatasi pada perspektif Customer dalam Ordering Context, sehingga hanya status yang relevan dengan interaksi pelanggan yang diimplementasikan.

Pada milestone 5, implementasi hanya Pending, Canceled, dan Completed. Namun, untuk alasan konsistensi dengan nama endpoint dan proses yang lebih detail, status disesuaikan menjadi Pending, Scheduled, Checkout, dan Canceled/Delivered.

- **Payment Context**

Pada implementasi ini, Payment Context tidak direalisasikan. Proses *checkout* hanya mensimulasikan konfirmasi pelanggan tanpa pembayaran aktual. Hal ini karena implementasi hanya fokus pada Ordering (Customer) Context sehingga konsistensi dalam konteks ini dapat dicapai tanpa ketergantungan pada konteks selain itu.

C. Activity Diagram

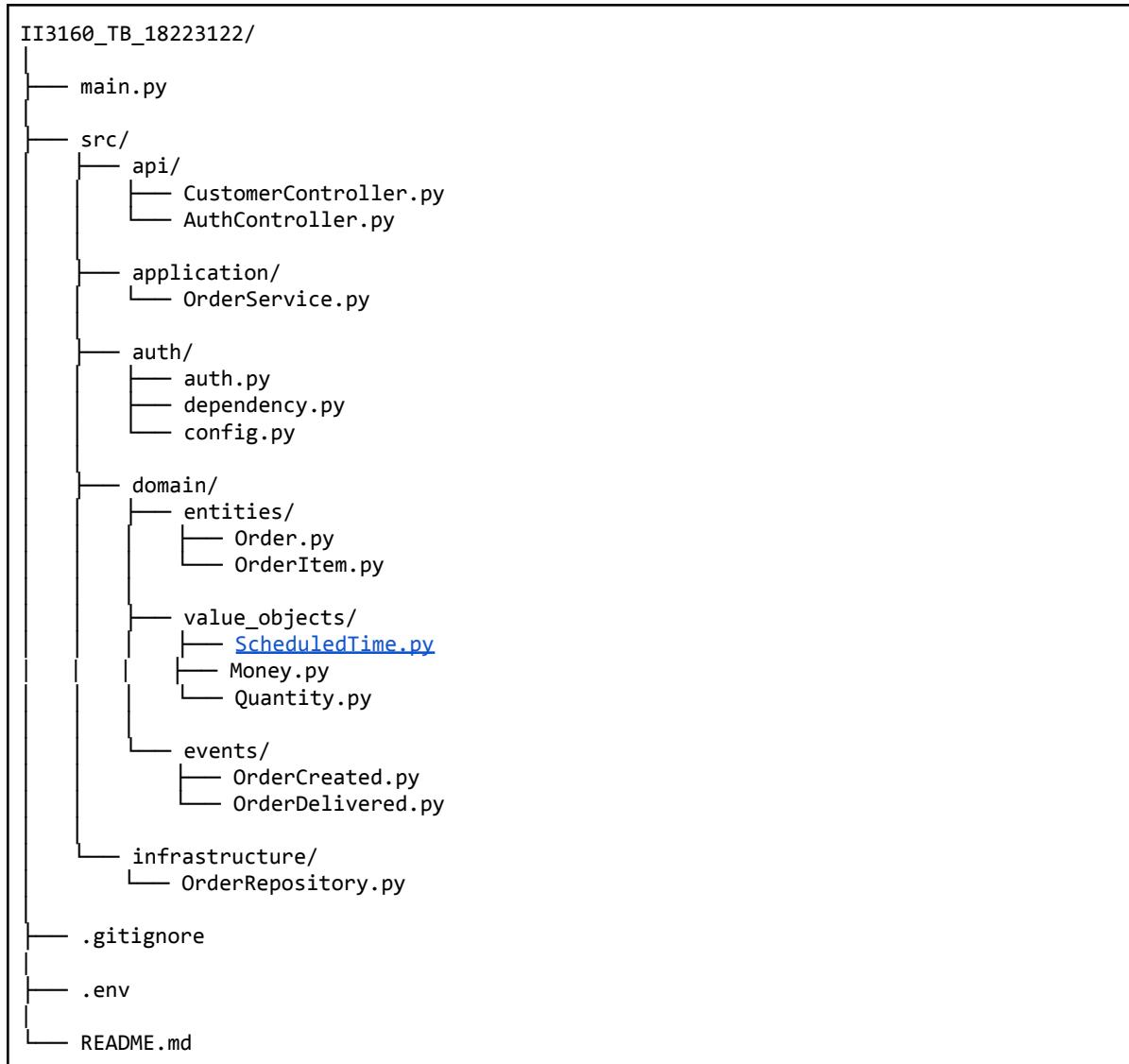


Gambar 1. Activity Diagram Customer Ordering Context

Activity diagram di atas adalah alur yang dilalui customer untuk memesan makanan melalui sistem Online Food Delivery. Bagian berwarna biru adalah context yang diimplementasikan pada tugas besar ini.

D. Implementasi

Berikut adalah struktur direktori dalam implementasi sistem OFD ini.



Aplikasi ini menerapkan arsitektur DDD dengan pembagian layer sebagai berikut:

1. Domain Layer (src/domain/)

Layer inti yang berisi business logic murni, tidak bergantung pada framework atau infrastruktur:

- Entities: Objek dengan identitas unik (Order, OrderItem)
- Value Objects: Objek tanpa identitas, immutable (Money, Quantity, ScheduledTime)

- Domain Events: Event yang terjadi dalam domain (OrderCreated, OrderDelivered)

2. Application Layer (src/application/)

Orchestration layer yang mengkoordinasikan use case dan business logic:

- Services: Mengelola alur bisnis yang melibatkan multiple entities

3. API Layer (src/api/)

Interface layer untuk komunikasi dengan client:

- Controllers: REST API endpoints yang menerima HTTP requests

4. Infrastructure Layer (src/infrastructure/)

Layer untuk integrasi dengan sistem eksternal:

- Repositories: Abstraksi akses data dan persistence

5. Auth Module (src/auth/)

Cross-cutting concern untuk autentikasi dan otorisasi menggunakan JWT. JWT (JSON Web Token) adalah standar terbuka yang digunakan untuk pertukaran informasi secara aman antar sistem dalam format JSON. JWT biasanya digunakan untuk mekanisme *authentication* dan *authorization*.

Setelah pengguna berhasil login menggunakan username dan password, server akan mengeluarkan sebuah token JWT yang berisi klaim identitas pengguna dan waktu kadaluarsa token. Token ini kemudian dikirim pada setiap request berikutnya melalui header Authorization: Bearer <token>, sehingga server dapat memvalidasi identitas pengguna tanpa menyimpan session di server. Algoritma yang digunakan dalam autentikasi ini adalah bcrypt HS256 dengan batas usia token 30 menit.

E. Daftar Endpoint Context

Berikut adalah endpoints yang telah diimplementasikan

Method	Endpoint	Deskripsi Fungsionalitas
POST	/auth/register	Membuat user baru dan menambahkan ke dalam dummy DB.

POST	/auth/login	Masuk dengan password dan user yang sudah ada. Dengan login, akan didapatkan token autentikasi yang digunakan untuk memproses method lain.
POST	/customer/order/create	Membuat order baru berdasarkan customer_id dan restaurant_id. Sistem menghasilkan order_id unik dan status awal <i>PENDING</i> .
GET	/customer/order/{order_id}	Mengambil detail order tertentu berdasarkan order_id, termasuk item, status, dan waktu terjadwal.
GET	/customer/orders/user/{user_id}	Mengambil seluruh order yang dimiliki oleh customer tertentu. Berguna untuk halaman <i>order history</i> .
GET	/customer/orders	Menampilkan seluruh order dalam sistem. Hanya digunakan untuk debugging atau admin.
POST	/customer/order/{order_id}/add-item	Menambahkan item makanan baru ke dalam order. Sistem membuat objek OrderItem yang berisi menu, harga, dan kuantitas.
DELETE	/customer/order/{order_id}/remove-item/{menu_id}	Menghapus salah satu item dari order berdasarkan menu_id.
POST	/customer/order/{order_id}/schedule	Menjadwalkan order pada waktu tertentu. Endpoint ini membuat ScheduledTime sebagai Value Object.
POST	/customer/order/{order_id}/checkout	Mengonfirmasi order sebelum dikirim ke restoran. Checkout hanya valid jika order sudah dijadwalkan. Status berubah menjadi <i>PENDING</i> .
POST	/customer/order/{order_id}/cancel	Membatalkan order. Status menjadi <i>Canceled</i> . Order yang di-cancel TIDAK dapat di-deliver.

POST	<code>/customer/order/{order_id}/deliver</code>	Menandai order sebagai selesai (<i>Completed</i>). Terdapat validasi bahwa order tidak boleh dibatalkan sebelumnya.
-------------	---	---

F. Testing dan Dokumentasi

Berikut adalah server (local host), request payload, dan response yang diterima yang digunakan selama pengujian.

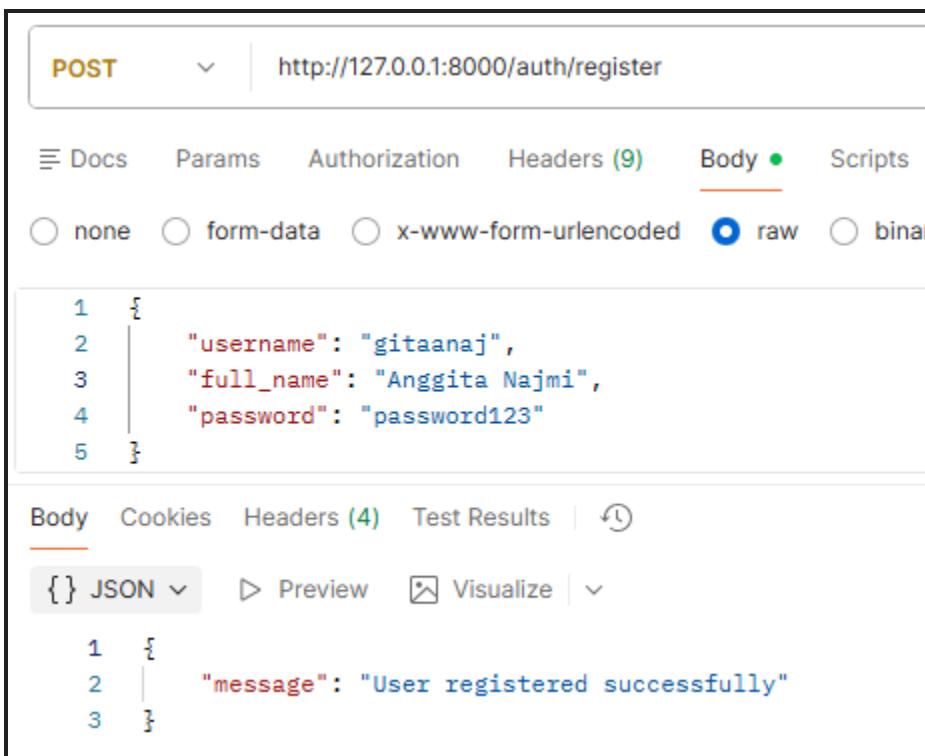
Endpoint	Request	Response
<code>http://127.0.0.1:8000/auth/register</code>	<pre>{ "username": "gitaanaj", "full_name": "Anggita 2", "password": "password123" }</pre>	<pre>{ "detail": "User registered successfully" }</pre>
<code>http://127.0.0.1:8000/auth/login</code>	Body (x-www-form-urlencoded) username=gitaanaj password=password123	<pre>{ "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJzdWIiOiJnaXRhYW5haiIiLCJ4cCI6MTc2NTUyMzIwNSwiaWF0IjoxNzY1NTIxNDA1fQ.6HuASk-D8lyozCcmLgLysY1E2PlhsHVU7XKf3WOGO3o", "token_type": "bearer" }</pre>
<code>http://127.0.0.1:8000/customer/order/create</code>	<pre>{ "customer_id": "gitaanaj", "restaurant_id": "restoA" }</pre>	<pre>{ "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [], "status": "PENDING", "scheduled_time": null }</pre>
<code>http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/add-item</code>	<pre>{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": 15000, "quantity": 2 }</pre>	<pre>{ "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": { "value": 15000 }, "quantity": { "value": 2 } }], "status": "PENDING", }</pre>

		<pre> "scheduled_time": null } } </pre>
http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/remove-item/M002	-	<pre> { "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": { "value": 15000 }, "quantity": { "value": 2 } }], "status": "PENDING", "scheduled_time": null } </pre>
http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/schedule	{ "datetime": "2025-01-15 18:30" }	<pre> { "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": { "value": 15000 }, "quantity": { "value": 2 } }], "status": "SCHEDULED", "scheduled_time": { "datetime": "2025-01-15T18:30:00" } } </pre>
http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/checkout	-	<pre> { "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": { "value": 15000 }, "quantity": { "value": 2 } }] } </pre>

		<pre> } }, "status": "CHECKOUT", "scheduled_time": { "datetime": "2025-01-15T18:30:00" } } </pre>
http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/cancel	-	<pre> { "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468", "customer_id": "gitaanaj", "restaurant_id": "restoA", "items": [{ "menu_id": "M001", "menu_name": "Nasi Goreng", "price": { "value": 15000 }, "quantity": { "value": 2 } }], "status": "CANCELED", "scheduled_time": { "datetime": "2025-01-15T18:30:00" } } </pre>
http://127.0.0.1:8000/customer/order/9b9b35ce-baaa-4c51-8ea4-3a708103b536/deliver	-	<pre> { "order_id": "9b9b35ce-baaa-4c51-8ea4-3a708103b536", "customer_id": "gitaanaj", "restaurant_id": "restoB", "items": [{ "menu_id": "M002", "menu_name": "Bakso", "price": { "value": 15000 }, "quantity": { "value": 2 } }], "status": "DELIVERED", "scheduled_time": { "datetime": "2025-01-15T18:30:00" } } </pre>

Dokumentasi dan Skema Pengujian

REGISTER



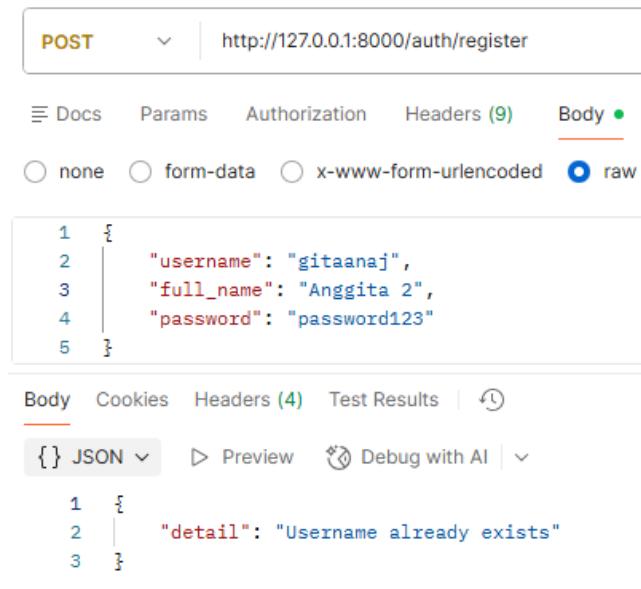
A screenshot of the Postman application interface. The request method is POST, the URL is `http://127.0.0.1:8000/auth/register`, and the body is raw JSON:

```
1 {  
2   "username": "gitaanaj",  
3   "full_name": "Anggita Najmi",  
4   "password": "password123"  
5 }
```

The response status is 201, and the response body is:

```
1 {  
2   "message": "User registered successfully"  
3 }
```

Kita akan mencoba mendaftarkan user dengan username yang sudah terdaftar.



A screenshot of the Postman application interface. The request method is POST, the URL is `http://127.0.0.1:8000/auth/register`, and the body is raw JSON:

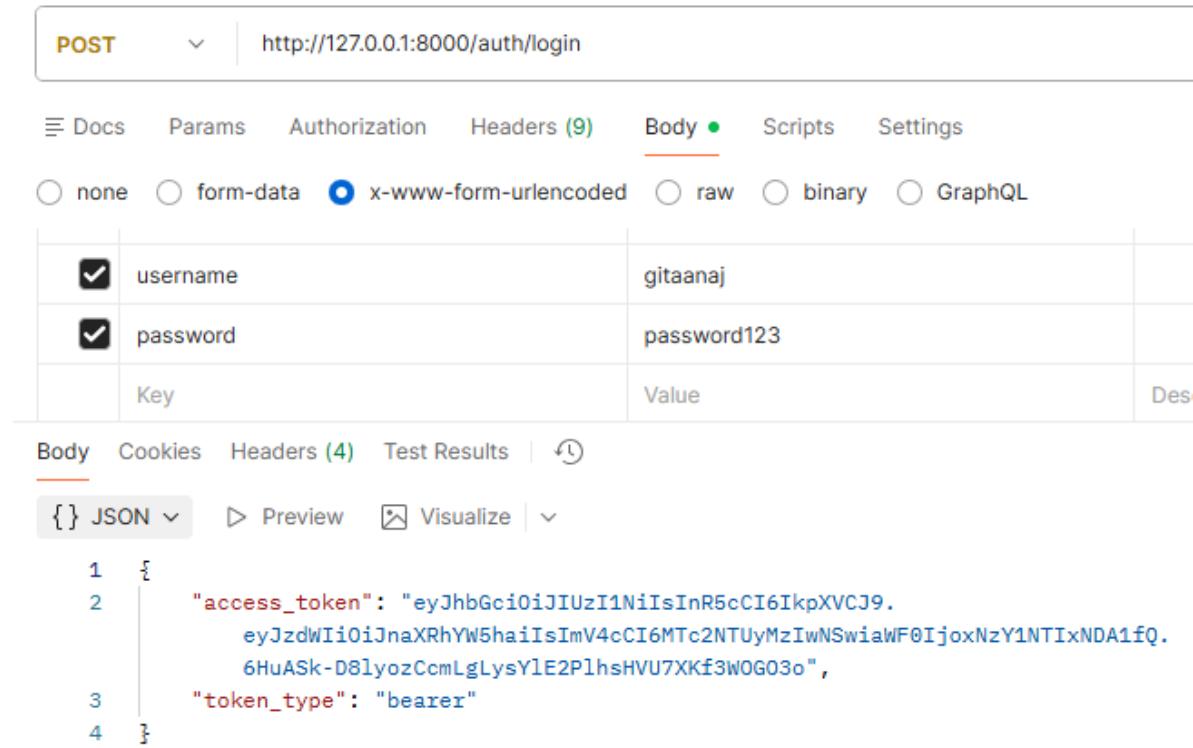
```
1 {  
2   "username": "gitaanaj",  
3   "full_name": "Anggita 2",  
4   "password": "password123"  
5 }
```

The response status is 400, and the response body is:

```
1 {  
2   "detail": "Username already exists"  
3 }
```

Tangkapan layar di atas menunjukkan logika sistem sudah benar, bahwa tidak bisa mendaftar dengan username yang sudah ada di dummy DB.

LOGIN



The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: http://127.0.0.1:8000/auth/login
- Body type: x-www-form-urlencoded
- Body content:

username	gitaanaj
password	password123
Key	Value
- Response (JSON):

```
1 {  
2     "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
3         eyJzdWIiOiJnaXRhYW5haiIsImV4cCI6MTc2NTUyMzIwNSwiaWF0IjoxNzY1NTIxNDA1fQ.  
4             6HuASK-D8lyozCcmLgLySYLE2PlhsHVU7XKf3WOGO3o",  
      "token_type": "bearer"  
}
```

Dari sini, didapat akses token yang akan kita gunakan untuk mengakses endpoint-endpoint customer:

```
"access_token":  
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJzdWIiOiJnaXRhYW5haiIsImV4cCI6MTc2NTUyMzIwNS  
wiaWF0IjoxNzY1NTIxNDA1fQ.6HuASK-D8lyozCcmLgLySYLE2PlhsHVU7XKf3WOGO3o"
```

Selanjutnya, endpoint-endpoint yang digunakan sama dengan milestone sebelumnya. Hanya saja, karena ada autentikasi kita harus terdaftar sebagai "authenticated user" untuk dapat melakukan method (akan expired dalam 30 menit). Oleh karena itu, kita mencantumkan token di atas pada kolom "authorization" pada Postman, dengan Auth Type "Bearer Token" dan masukkan token yang sesuai, contohnya sebagai berikut.

The screenshot shows the Postman interface for creating a new order. At the top, a header section shows a POST request to `http://127.0.0.1:8000/customer/order/create`. Below this, the 'Authorization' tab is selected, showing a 'Bearer Token' input field containing `cmLgLySIE2PlhsHVU7XKf3WOGO3o`. The main content area is titled 'CREATE ORDER' and contains another POST request to the same endpoint. The 'Body' tab is selected, showing raw JSON data:

```
1 {
2   "customer_id": "gitaanaj",
3   "restaurant_id": "restoA"
4 }
5
```

Below the body, the response is shown as:

```
1 {
2   "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3   "customer_id": "gitaanaj",
4   "restaurant_id": "restoA",
5   "items": [],
6   "status": "PENDING",
7   "scheduled_time": null
8 }
```

A note below the response states: "Dari hasil create order di atas, didapatkan order_id yang akan digunakan pada endpoint setelahnya, yaitu: `order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468"`". It also mentions that the order is currently in status "PENDING".

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/add-item
- Headers:** (10) - Headers tab is selected.
- Body:** raw JSON
- JSON Response:**

```
1 {
2     "menu_id": "M001",
3     "menu_name": "Nasi Goreng",
4     "price": 15000,
5     "quantity": 2
6 }
```

```
1 {
2     "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3     "customer_id": "gitaanaj",
4     "restaurant_id": "restoA",
5     "items": [
6         {
7             "menu_id": "M001",
8             "menu_name": "Nasi Goreng",
9             "price": {
10                 "value": 15000
11             },
12             "quantity": {
13                 "value": 2
14             }
15         }
16     ],
17     "status": "PENDING",
18     "scheduled_time": null
19 }
```

REMOVE ITEM

Tambahkan dahulu item lain dalam order agar dapat dihapus.

```
"customer_id": "gitaanaj",
"restauzant_id": "restoA",
"items": [
    {
        "menu_id": "M001",
        "menu_name": "Nasi Goreng",
        "price": {
            "value": 15000
        },
        "quantity": {
            "value": 2
        }
    },
    {
        "menu_id": "M002",
        "menu_name": "Bakso",
        "price": {
            "value": 15000
        },
        "quantity": {
            "value": 2
        }
    }
],
"status": "PENDING",
```

Kemudian dilakukan testing remove-item.

DELETE http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/remove-item/M002

Docs Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (4) Test Results

{ } JSON ▾ Preview Visualize

```
1 {
2     "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3     "customer_id": "gitaanaj",
4     "restaurant_id": "restoA",
5     "items": [
6         {
7             "menu_id": "M001",
8             "menu_name": "Nasi Goreng",
9             "price": {
10                 "value": 15000
11             },
12             "quantity": {
13                 "value": 2
14             }
15         }
16     ],
17     "status": "PENDING",
18     "scheduled_time": null
19 }
```

SCHEDULE

POST http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/schedule

Docs Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (4) Test Results

{ } JSON ▾ Preview Visualize

```
1 {
2     "datetime": "2025-01-15 18:30"
3 }
```

```
1 {
2     "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3     "customer_id": "gitaanaj",
4     "restaurant_id": "restoA",
5     "items": [
6         {
7             "menu_id": "M001",
8             "menu_name": "Nasi Goreng",
9             "price": {
10                 "value": 15000
11             },
12             "quantity": {
13                 "value": 2
14             }
15         }
16     ],
17     "status": "SCHEDULED",
18     "scheduled_time": {
19         "datetime": "2025-01-15T18:30:00"
20     }
}
```

CHECKOUT

POST | http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/checkout

Docs Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (4) Test Results

{ } JSON ▾ ▶ Preview Visualize

```

1 {
2     "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3     "customer_id": "gitaanaj",
4     "restaurant_id": "restoA",
5     "items": [
6         {
7             "menu_id": "M001",
8             "menu_name": "Nasi Goreng",
9             "price": {
10                 "value": 15000
11             },
12             "quantity": {
13                 "value": 2
14             }
15         }
16     ],
17     "status": "CHECKOUT",
18     "scheduled_time": {
19         "datetime": "2025-01-15T18:30:00"
20     }
21 }
```

CANCEL ORDER

POST | http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/cancel

Docs Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (4) Test Results

{ } JSON ▾ ▶ Preview Visualize

```

1 Ctrl+Alt+P to Ask AI
```

```

1 {
2     "order_id": "5239c4b8-29b3-44e8-8b8b-51ebe36d7468",
3     "customer_id": "gitaanaj",
4     "restaurant_id": "restoA",
5     "items": [
6         {
7             "menu_id": "M001",
8             "menu_name": "Nasi Goreng",
9             "price": {
10                 "value": 15000
11             },
12             "quantity": {
13                 "value": 2
14             }
15         }
16     ],
17     "status": "CANCELED",
18     "scheduled_time": {
19         "datetime": "2025-01-15T18:30:00"
20     }
21 }
```

Uji coba 1: deliver order yang sudah di-cancel

POST http://127.0.0.1:8000/customer/order/5239c4b8-29b3-44e8-8b8b-51ebe36d7468/deliver

Docs Params Authorization Headers (9) Body Scripts Settings

Body Cookies Headers (4) Test Results | ⚙️

{ } JSON ▾ ▶ Preview ⚡ Debug with AI | ▾

```
1  {
2    "detail": "Canceled orders cannot be delivered"
3 }
```

Hasil di atas menunjukkan bahwa order dengan order_id yg sudah di-cancel tidak dapat di-deliver ke customer. Berarti, skema endpoint berjalan dengan baik.

Uji coba 2: dibuat order baru untuk mendapatkan id yang dapat di-deliver.

```
1  {
2    "order_id": "9b9b35ce-baaa-4c51-8ea4-3a708103b536",
3    "customer_id": "gitaanaj",
4    "restaurant_id": "restoB",
5    "items": [],
6    "status": "PENDING",
7    "scheduled_time": null
8 }
```

Setelah dilakukan add-item, schedule, dan checkout, kita coba deliver order tersebut.

POST http://127.0.0.1:8000/customer/order/9b9b35ce-baaa-4c51-8ea4-3a708103b536/deliver

Docs Params Authorization Headers (9) Body Scripts Settings

Body Cookies Headers (4) Test Results | ⚙️

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

```
1  {
2    "order_id": "9b9b35ce-baaa-4c51-8ea4-3a708103b536",
3    "customer_id": "gitaanaj",
4    "restaurant_id": "restoB",
5    "items": [
6      {
7        "menu_id": "M002",
8        "menu_name": "Bakso",
9        "price": {
10          "value": 15000
11        },
12        "quantity": {
13          "value": 2
14        }
15      ],
16    "status": "DELIVERED",
17    "scheduled_time": {
18      "datetime": "2025-01-15T18:30:00"
19    }
20  }
```

Deliver berhasil pada user002 dan status berubah menjadi "DELIVERED".

GET ORDER

GET <http://127.0.0.1:8000/customer/order/9b9b35ce-baaa-4c51-8ea4-3a708103b536>

Docs Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (4) Test Results

{ } JSON ▾ ▷ Preview Visualize ▾

```

1  {
2    "order_id": "9b9b35ce-baaa-4c51-8ea4-3a708103b536",
3    "customer_id": "gitaanaj",
4    "restaurant_id": "restoB",
5    "items": [
6      {
7        "menu_id": "M002",
8        "menu_name": "Bakso",
9        "price": {
10          "value": 15000
11        },
12        "quantity": {
13          "value": 2
14        }
15      ],
16    "status": "DELIVERED",
17    "scheduled_time": {
18      "datetime": "2025-01-15T18:30:00"
19    }
20  }
21 }
```

Menampilkan detail order untuk suatu order_id. Di atas, digunakan order_id milik user gitaanaj dan response sukses menampilkan detail order user002 yang berstatus "DELIVERED" karena telah di-deliver di pengujian poin sebelumnya.

GET CUSTOMER ORDER

GET <http://127.0.0.1:8000/customer/orders>

Docs Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (4) Test Results

{ } JSON ▾ ▷ Preview Visualize ▾

```

19  {
20    "scheduled_time": {
21      "datetime": "2025-01-15T18:30:00"
22    }
23  },
24  {
25    "order_id": "9b9b35ce-baaa-4c51-8ea4-3a708103b536",
26    "customer_id": "gitaanaj",
27    "restaurant_id": "restoB",
28    "items": [
29      {
30        "menu_id": "M002",
31        "menu_name": "Bakso",
32        "price": {
33          "value": 15000
34        },
35        "quantity": {
36          "value": 2
37        }
38      ],
39    "status": "DELIVERED",
40    "scheduled_time": {
41      "datetime": "2025-01-15T18:30:00"
42    }
43  }
44 ]
```