

## MVP Submission Format [BE]

Berikut adalah contoh format untuk pengumpulan MVP (Minimum Viable Product) oleh peserta didik:

Judul Proyek : Pemesanan Cafe  
Nama Peserta : Muhammad Reza Hidayat  
Kelas : D-Golang

### Deskripsi Produk:

Dibuatnya aplikasi ini terinspirasi dari lama nya ngantri dalam makanan seperti di cafe cafe yang terkini ada juga pengaplikasian menggunakan remote namun si pelanggan mengharuskan mengantri dan memilih makanan kemudian balik lagi hal ini yang membuat saya memunculkan ide untuk membuat aplikasi ini agar si pelanggan tidak membuang waktu dalam memesan makanan dan juga mempermudah pemesanan makanan sekaligus membuat lebih efisien ,kemudain juga mempermudah cafe atau resto dalam memasarkan diskon di daerah dan customer secara akauran dan terukur

Spesifikasi Fitur Produk: (jelaskan spesifikasi fitur pada produk yang akan dibuat)

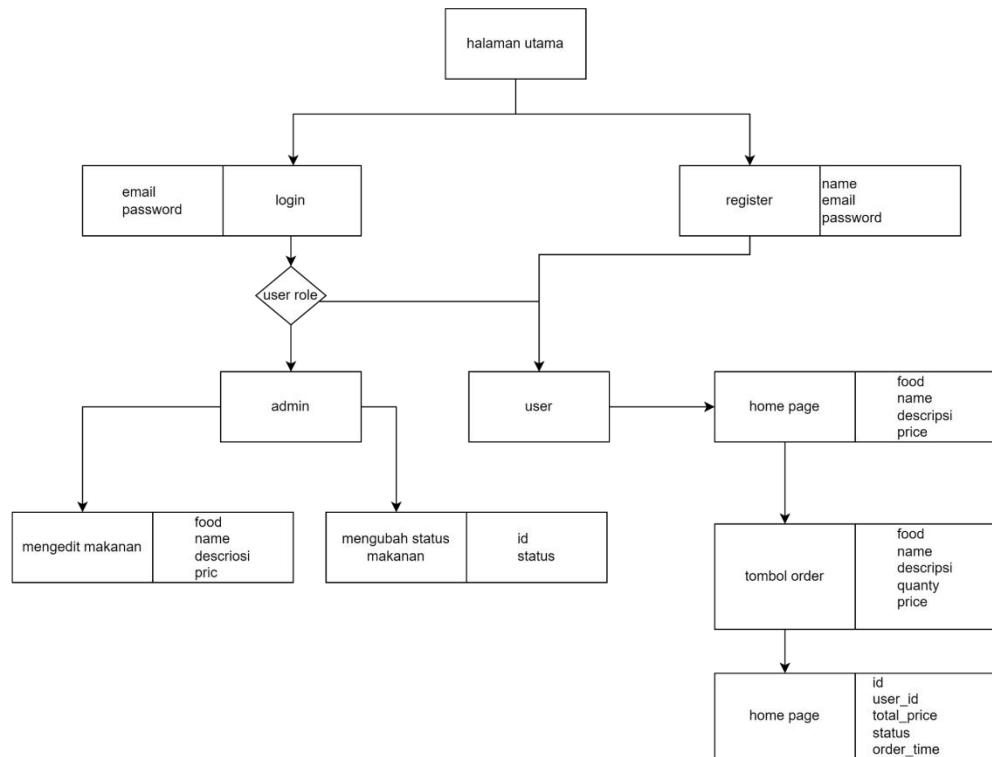
Contoh:

- User dapat registrasi dengan role sebagai customer atau kasir
- User dapat login dengan email dan password
- User dapat melihat daftar makanan
- User dapat menambahkan pesanan baru (order)
- Kasir dapat menambahkan produk makanan baru
- Kasir dapat melihat detail pesanan berdasarkan ID pesanan
- Kasir dapat mengupdate order dalam sistem
- Kasir dapat menghapus order dalam sistem
- Kasir dapat mengubah data makanan berdasarkan ID makanan
- Kasir dapat menghapus data makanan berdasarkan ID makanan
- Kasir dapat melihat status pesanan berdasarkan ID pesanan

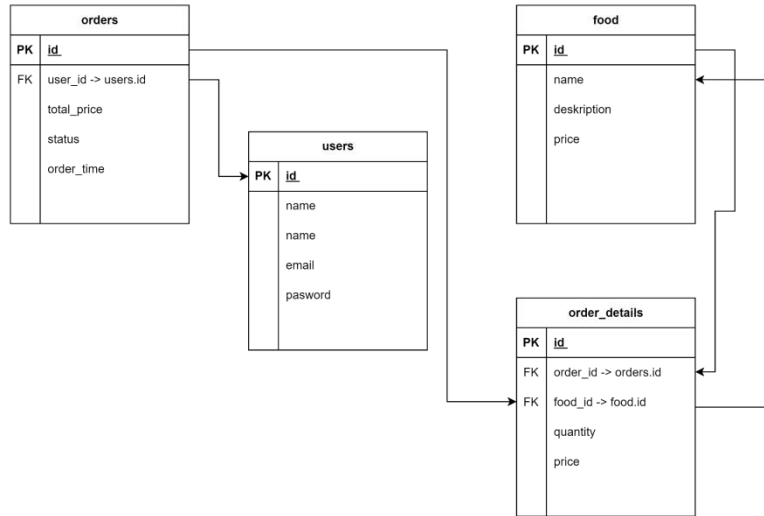
Tech Stack: jelaskan tech stack yang akan digunakan, tech stack terdiri dari:

- App Framework : Echo
- ORM Library : GORM
- DB : Mysql
- Deployment : AWS EC2
- Code Structure : MVC
- Authentication : JWT

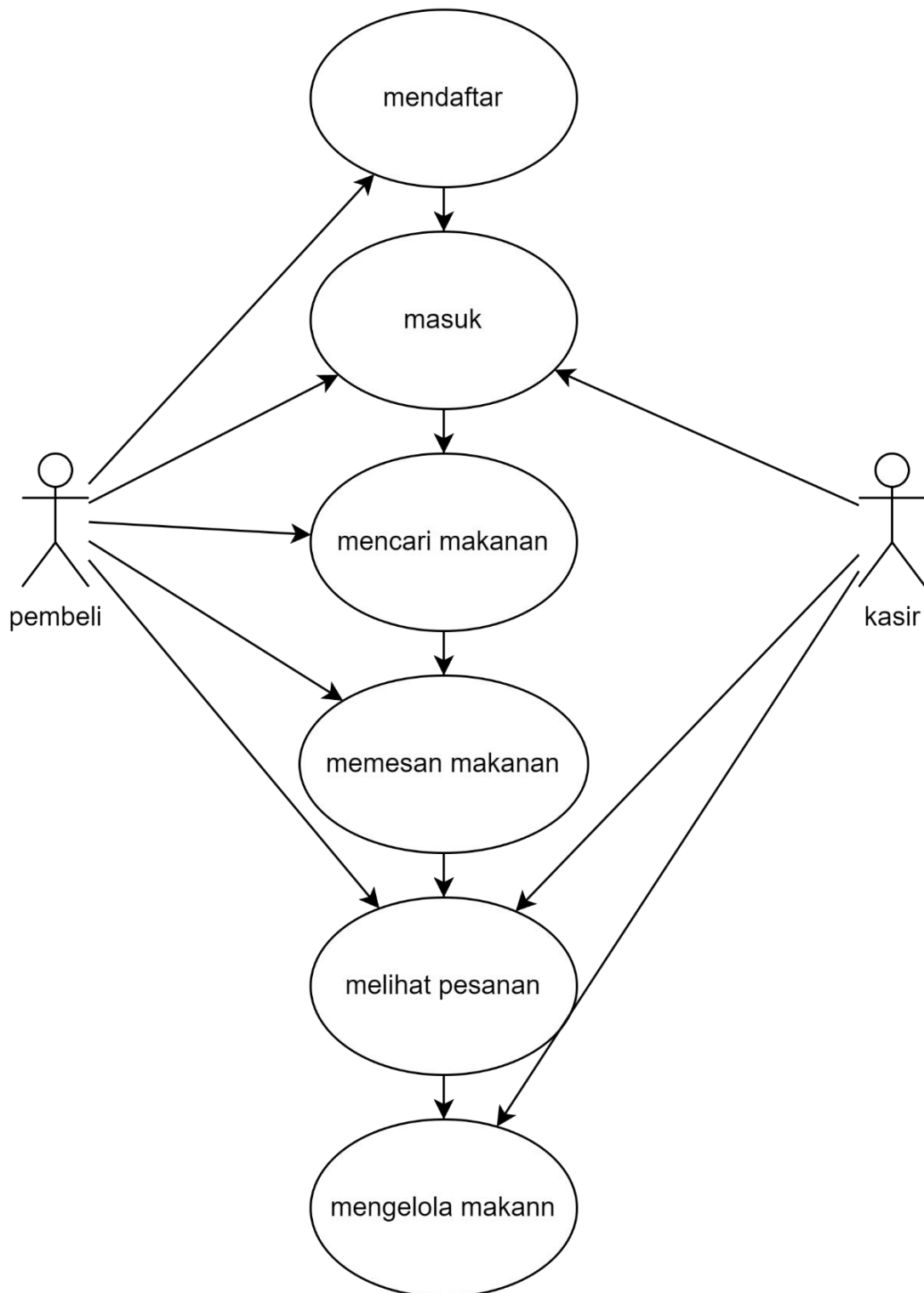
# Pemecahan Per Featur



# ERD



# DIAGRAM USECASE



# DATABASE

```
database
CREATE DATABASE
cafe;
USE cafe;
```

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE food (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description VARCHAR(255),
  price DECIMAL(10, 2) NOT NULL
);
```

```
CREATE TABLE orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  total_price DECIMAL(10, 2) NOT NULL,
  status ENUM('proses', 'selesai') NOT NULL,
  order_time TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

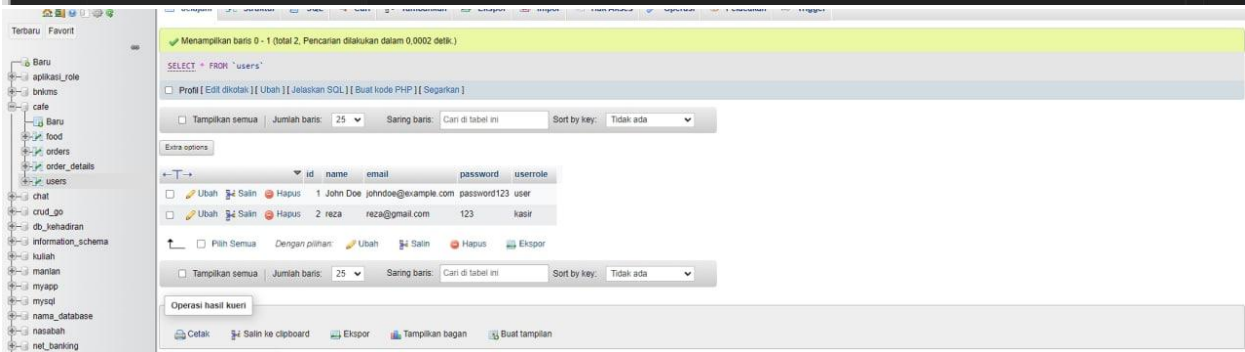
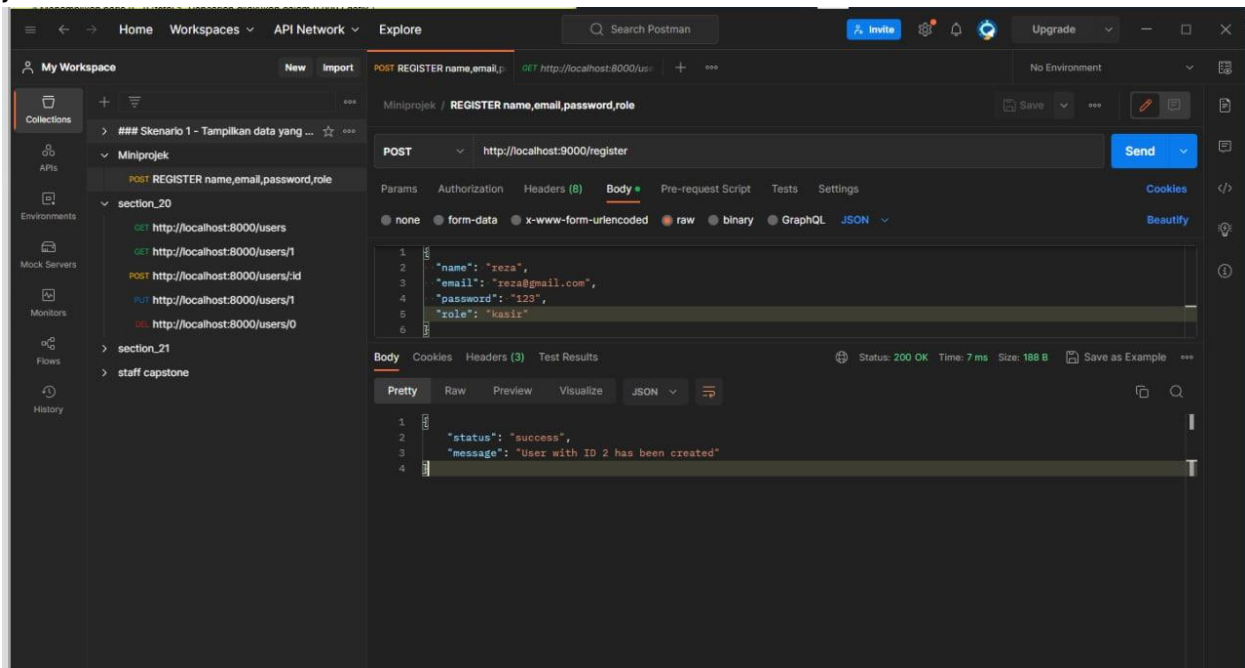
```
CREATE TABLE order_details (
  id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  food_id INT NOT NULL,
  quantity INT NOT NULL,
  price DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (order_id) REFERENCES orders(id),
  FOREIGN KEY (food_id) REFERENCES food(id)
);
```

## POSTMAN

Register

<http://localhost:9000/register>

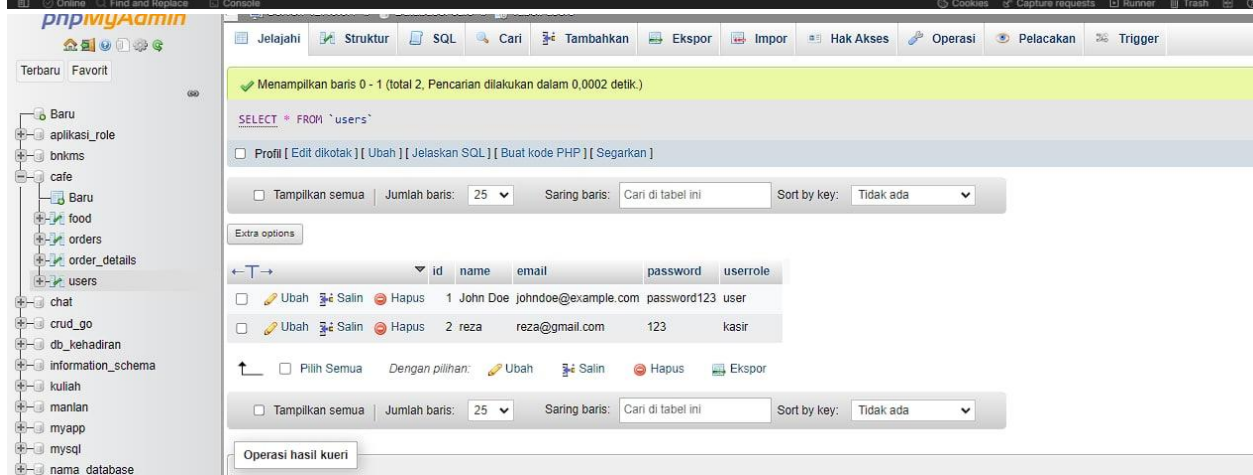
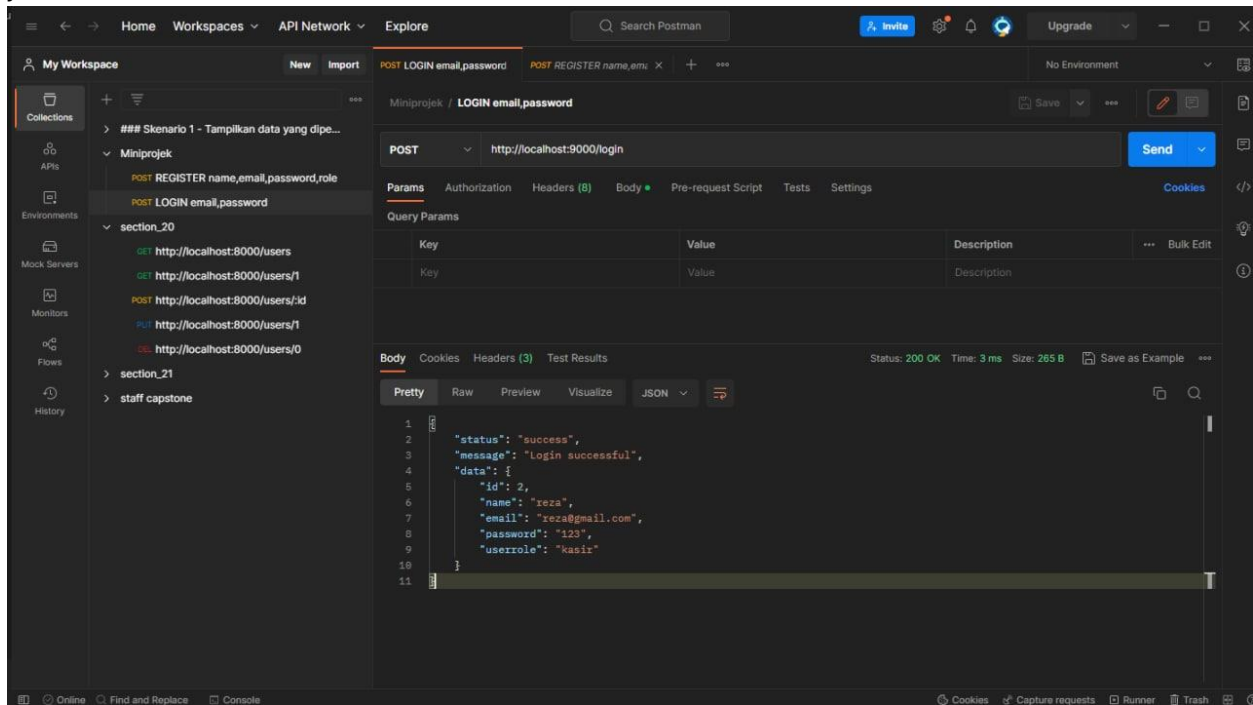
```
{
  "name": "John Doe",
  "email": "johndoe@example.com",
  "password": "password123",
  "role": "user"
}
```



Login

http://localhost:9000/login

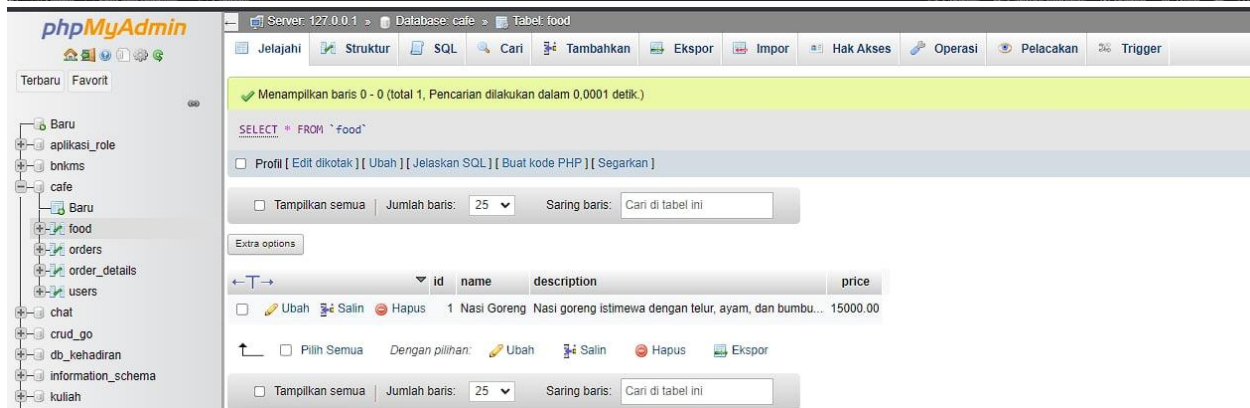
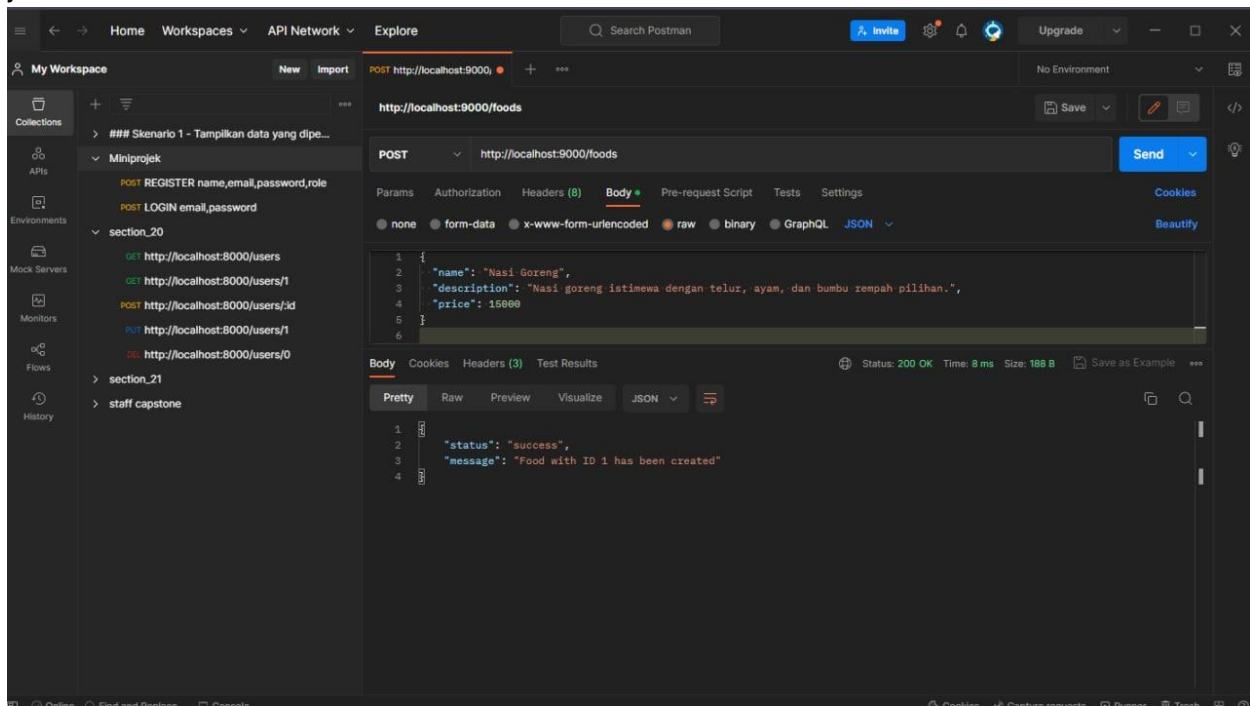
```
{  
  "email": "reza@gmail.com",  
  "password": "123"  
}
```



## Menambahkan Menu Makanan

http://localhost:9000/foods

```
{
  "name": "Nasi Goreng",
  "description": "Nasi goreng istimewa dengan telur, ayam, dan bumbu rempah pilihan.",
  "price": 15000
}
```





## List Makanan

<http://localhost:9000/foods>

The image displays two screenshots related to a web application's food list functionality.

**Top Screenshot (Postman):** Shows a GET request to `http://localhost:9000/foods` with a status of 200 OK. The response body is a JSON array containing two food items:

```
1 {
2   "status": "success",
3   "data": [
4     {
5       "id": 1,
6       "name": "Nasi Goreng",
7       "description": "Nasi goreng istimewa dengan telur, ayam, dan bumbu rempah pilihan.",
8       "price": 15000
9     },
10    {
11      "id": 2,
12      "name": "Nasi Goreng",
13      "description": "Nasi goreng istimewa dengan telur, ayam, dan bumbu rempah pilihan.",
14      "price": 15000
15    }
16  ]
17 }
```

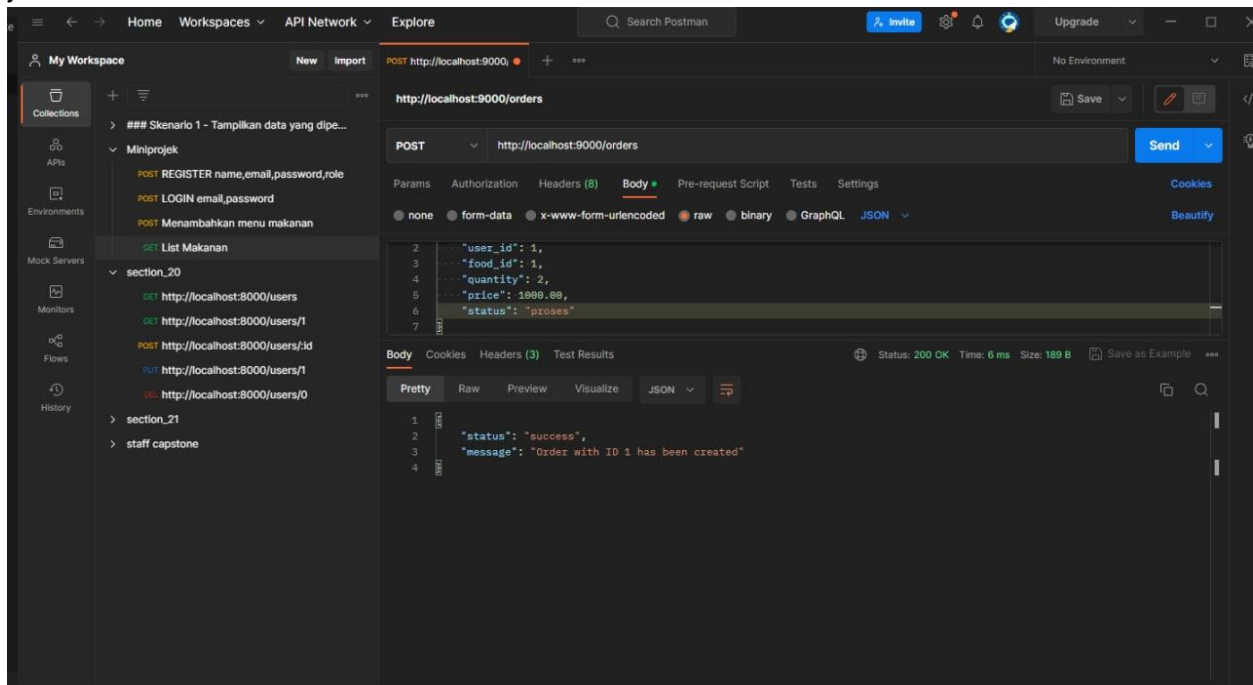
**Bottom Screenshot (phpMyAdmin):** Shows the database structure for the 'food' table. The table contains two records:

id	name	description	price
1	Nasi Goreng	Nasi goreng istimewa dengan telur, ayam, dan bumbu...	15000.00
2	Nasi Goreng	Nasi goreng istimewa dengan telur, ayam, dan bumbu...	15000.00

## Order Makanan

<http://localhost:9000/orders>

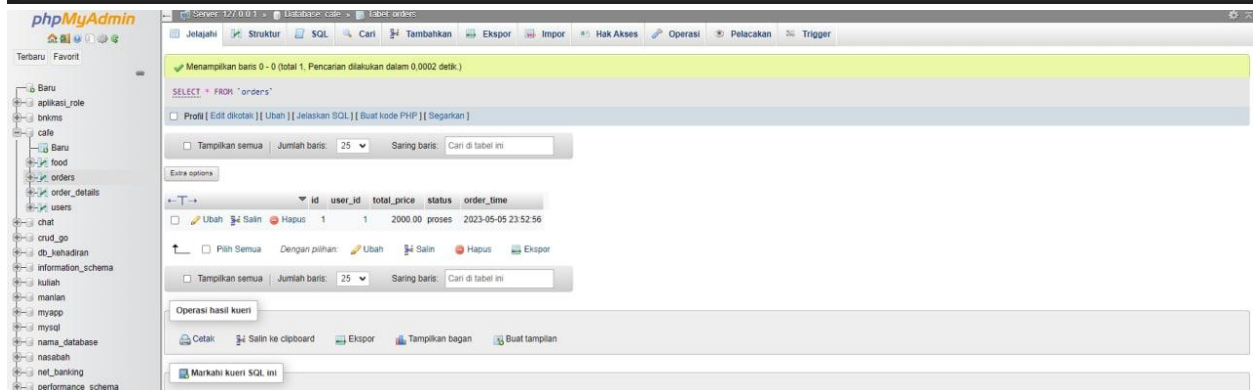
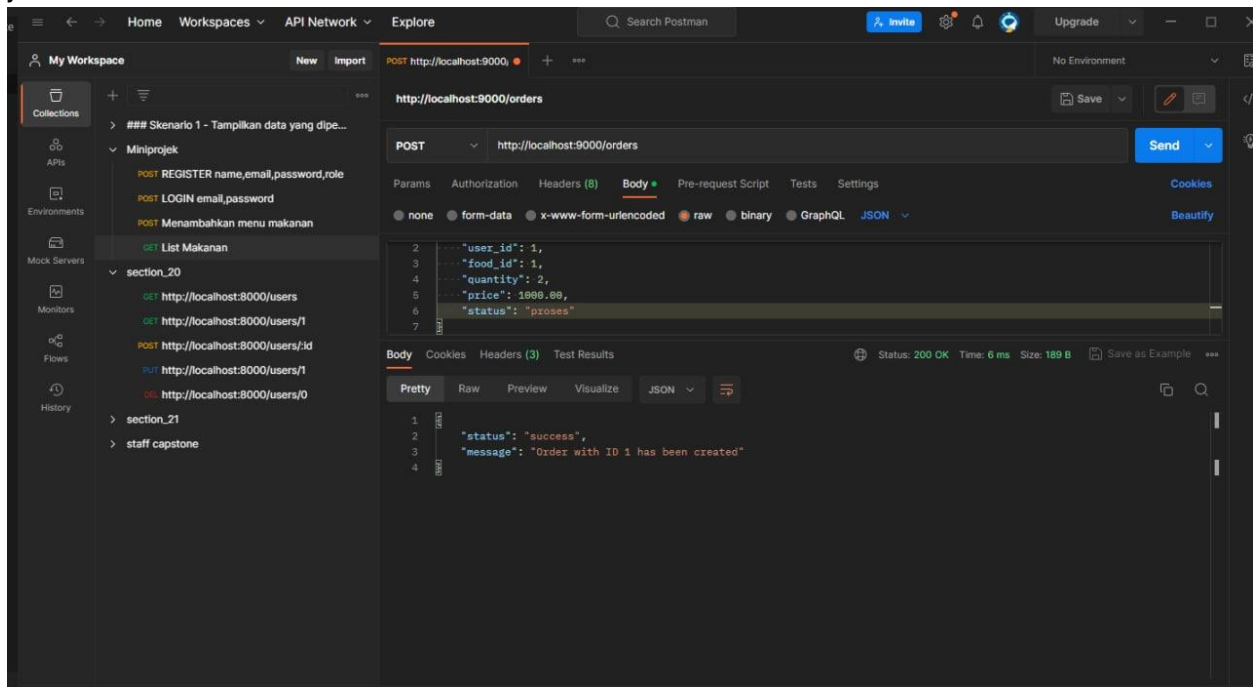
```
{  
  "user_id": 1,  
  "food_id": 1,  
  "quantity": 2,  
  "price": 1000.00,  
  "status": "proses"  
}
```

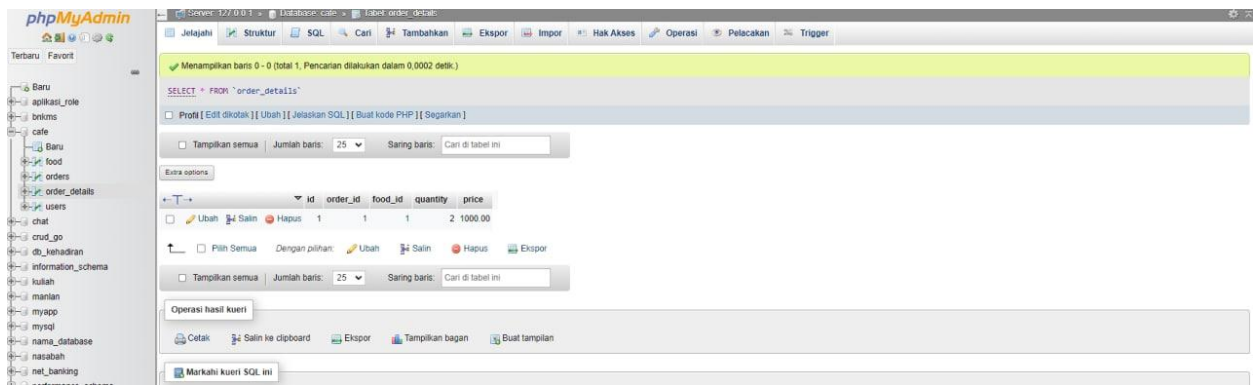


## Order Makanan

http://localhost:9000/orders

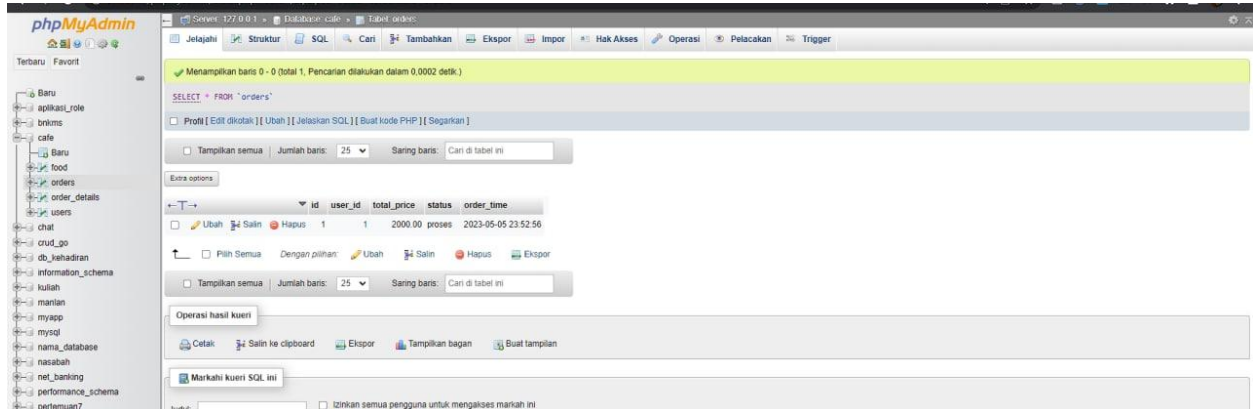
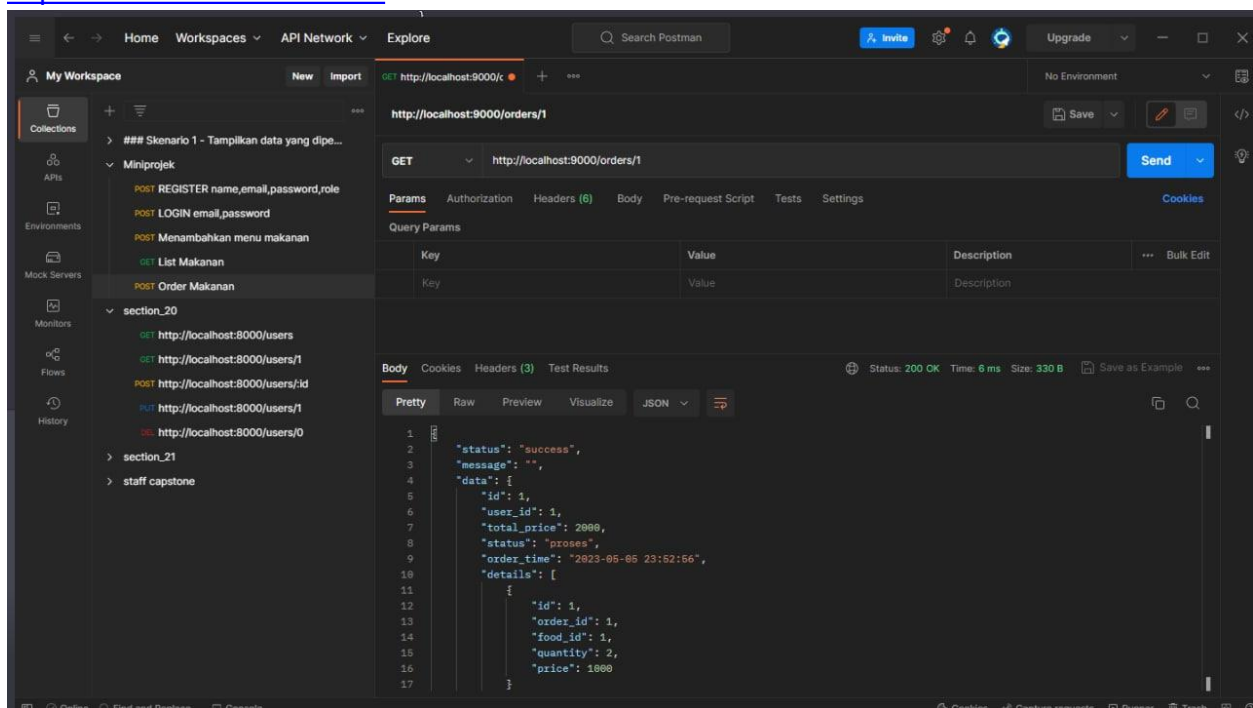
```
{
  "user_id": 1,
  "food_id": 1,
  "quantity": 2,
  "price": 1000.00,
  "status": "proses"
}
```





Cek Order Berdasarkan ID

<http://localhost:9000/orders/1>



update order

http://localhost:9000/orders/1

```
{
  "user_id": 1,
  "food_id": 1,
  "quantity": 3,
  "price": 10.50,
  "status": "selesai"
}
```

The image shows two screenshots. The top screenshot is from Postman, showing a PUT request to `http://localhost:9000/orders/1` with a JSON body. The bottom screenshot is from a web application, showing a table with the result of the update.

**Postman Request:**

Method: PUT  
URL: `http://localhost:9000/orders/1`  
Body (JSON):

```
{
  "user_id": 1,
  "food_id": 1,
  "quantity": 3,
  "price": 10.50,
  "status": "selesai"
}
```

**Postman Response:**

Status: 200 OK  
Time: 6 ms  
Size: 189 B  
Body (JSON):

```
{
  "status": "success",
  "message": "Order with ID 1 has been updated"
}
```

**Web Application:**

Menampilkan baris 0 - 0 (total 1, Pencarian dilakukan dalam 0.0003 detik.)

SELECT \* FROM `orders`

Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini

	ID	user_id	total_price	status	order_time			
<input type="checkbox"/>	Ubah	Salin	Hapus	1	1	31.50	selesai	2023-05-05 23:52:56

Pilih Semua | Dengan pilihan: Ubah | Salin | Hapus | Ekspor

Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini

## Delet Order

<http://localhost:9000/orders/1>

The image shows two screenshots related to a web application. The top screenshot is from Postman, showing a DELETE request to `http://localhost:9000/orders/1` with a status of 200 OK. The response body is a JSON object: `{ "status": "success", "message": "Order with ID 1 has been deleted" }`. The bottom screenshot is from phpMyAdmin, showing a SQL query: `SELECT * FROM `orders``. The query results show a table with columns `id`, `user_id`, `total_price`, `status`, and `order_time`. The query is executed, and the results are displayed.

**Postman Request Details:**

- Method: DELETE
- URL: `http://localhost:9000/orders/1`
- Status: 200 OK
- Time: 9 ms
- Size: 189 B
- Response Body (JSON):

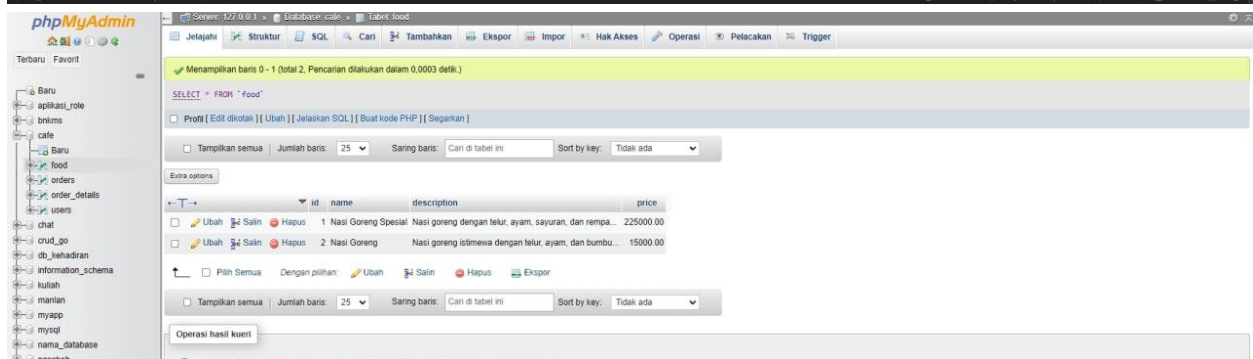
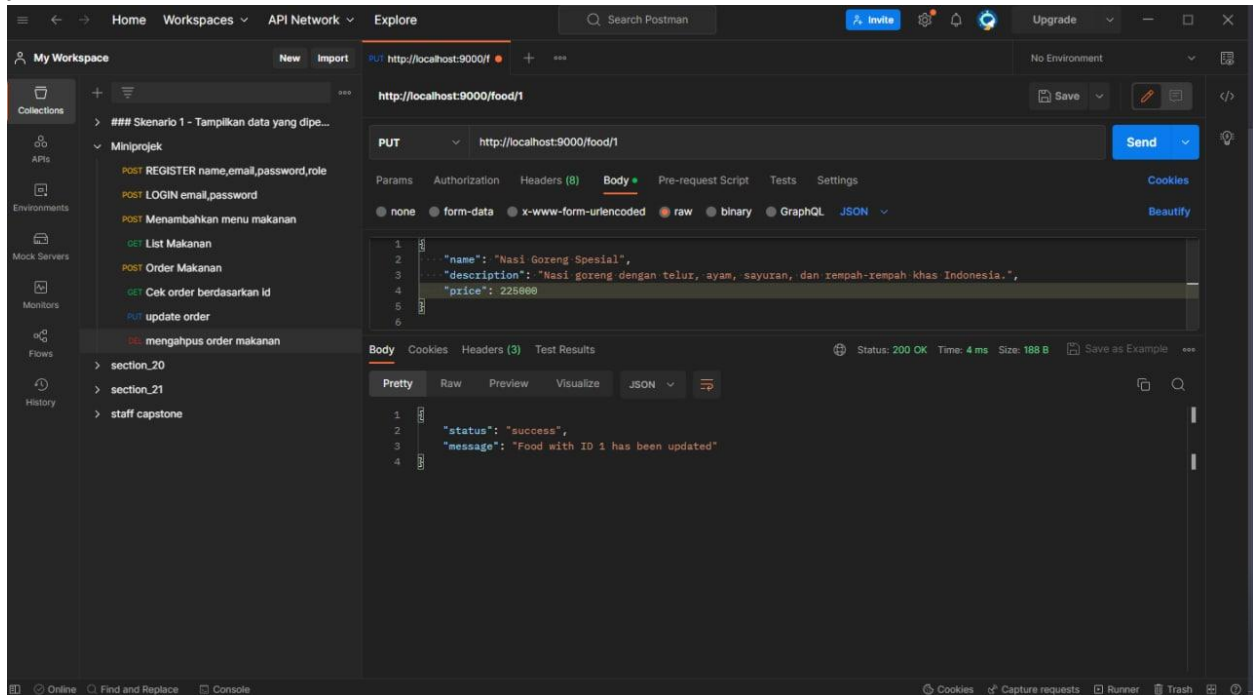
```
{  "status": "success",  "message": "Order with ID 1 has been deleted"}
```

**phpMyAdmin Query Details:**

- Query: `SELECT * FROM `orders``
- Result: MySQL memberikan hasil kosong (atau not bars). (Pencarian dilakukan dalam 0.0002 detik.)
- Table Structure: `id`, `user_id`, `total_price`, `status`, `order_time`

update menu makanan  
http://localhost:9000/food/1

```
{  
  "name": "Nasi Goreng Spesial",  
  "description": "Nasi goreng dengan telur, ayam, sayuran, dan rempah-rempah khas  
Indonesia.",  
  "price": 225000  
}
```





menghapus menu makanan

<http://localhost:9000/food/1>

