German International University
Faculty of Informatics and Computer Science
Dr. Iman Awaad
Eng. Ahmed Sherif
Amir Haythem
Mohamed Essam
Yassein Eldamasy

Software Engineering, Winter 2025
Project Milestone 3

Submission: Tuesday December 3rd 2025 at 23:59

The objective of milestone 3 is to build backend for the GIU Food-truck System. The first step is to create database tables in the schema FoodTruck, such as Users, Trucks, MenuItems, Orders, OrderItems, and Carts. **A complete SQL script file (script.sql) is provided with this document to create all necessary tables in PostgreSQL.** Execute this script in pgAdmin to set up your database schema.

Hint: You can view the SQL table definitions online at: https://ideone.com/0GCJeJ

Note: You are welcome to use this system directly or use it as a reference to build your own custom food truck management system. The provided structure and endpoints are designed to help you get started quickly. In addition, you will be required to implement all server-side endpoints to receive a full grade.

Users Table in schema FoodTruck

Column	Type	Constraint	Note
userId	serial	primary key	-
name	text	not null	-
email	text	not null & unique	-
password	text	not null	hashed password
role	text	default “customer”	“customer” or “truckOwner”
birthDate	date	default “current_timestamp”	-
createdAt	timestamp	default “current_timestamp”	-

Trucks Table in schema FoodTruck

Column	Type	Constraint	Note
truckId	serial	primary key	-
truckName	text	not null & unique	-
truckLogo	text	null allowed	url to logo
ownerId	integer	foreign key	reference Users(userId)
truckStatus	text	default “available”	“available” or “banned”
orderStatus	text	default “available”	“available” or “unavailable”
createdAt	timestamp	default “current_timestamp”	-

MenuItems Table in schema FoodTruck

Column	Type	Constraint	Note
itemId	serial	primary key	-
truckId	integer	foreign key	references Trucks(truckId)
name	text	not null	-
description	text	null allowed	-
price	numeric(10,2)	not null	e.g. 25.50
category	text	not null	-
status	text	default “available”	“available” or “unavailable”
createdAt	timestamp	default “current_timestamp”	-

Orders Table in schema FoodTruck

Column	Type	Constraint	Format
orderId	serial	primary key	-
userId	integer	foreign key	references Users(userId)
truckId	integer	foreign key	references Trucks(truckId)
orderStatus	text	not null	-
totalPrice	numeric(10,2)	not null	e.g. 75.00
scheduledPickupTime	timestamp	-	e.g. 2025-12-03 14:30:00
estimatedEarliestPickup	timestamp	-	e.g. 2025-12-03 14:00:00
createdAt	timestamp	default “current_timestamp”	-

OrderItems Table in schema FoodTruck

Column	Type	Constraint	Note
orderItemId	serial	primary key	-
orderId	integer	foreign key	references Orders(orderId)
itemId	integer	foreign key	references MenuItems(itemId)
quantity	integer	not null	e.g. 2
price	numeric(10,2)	not null	e.g. 12.50

Carts Table in schema FoodTruck

Column	Type	Constraint	Note
cartId	serial	primary key	-
userId	integer	foreign key	references Users(userId)
itemId	integer	foreign key	references MenuItems(itemId)
quantity	integer	not null	-
price	numeric(10,2)	not null	-

Sessions Table in schema FoodTruck

Column	Type	Constraint	Note
id	serial	primary key	-
userId	integer	foreign key	references Users(userId)
token	text	not null	-
expiresAt	timestamp	default “current_timestamp”	-

It is highly recommended to use Knex Query Builder Syntax.

```
Example
await knex("FoodTruck.Users").insert({name: "Mo",email: "mo@example.com"});
```

Hint: You can also view and copy the SQL table definitions from the following link: https://ideone.com/0GCJeJ

The async function getUser is implemented in milestoneBackend/utils/session.js. The function get user information who is currently logged into website and returns a user object.

user object

key	Type
id	integer
userId	integer
token	string
expiresAt	date
name	string
birthDate	date
email	string
password	string
role	string
truckId	integer

The second part of project is to implement server-side user stories. Make sure to add try catch block to catch errors whenever you are performing a query to the database. Make sure that each user story corresponds to its correct user role.

The following steps are required to test your server-side endpoints :

- **Download Postman or install thunderclient from visual studio code extensions**
- **Type npm run server inside your terminal of folder milestoneBackend**
- **Type localhost:3000 inside your browser**
- **You need to create an account and sign in**
- **You must check the TestingPrivateApi.docx before testing private api**
- **You can start to test your api using thunderclient or postman**

API Request and Response Examples

This section provides concrete JSON examples for API requests and responses to help you understand the expected format. All responses should include appropriate HTTP status codes.

Success Response Format

All successful operations should return JSON with appropriate status codes:

- **200: Success with data**
- **201: Resource created successfully**

Error Response Format

All errors should return JSON in this format:

```
{
  "error": "Error message describing what went wrong"
}
```

Common error status codes:

- **400: Bad Request (validation errors, business logic errors)**
- **401: Unauthorized (not logged in)**
- **403: Forbidden (logged in but not authorized for this resource)**
- **404: Not Found (resource doesn't exist)**
- **500: Internal Server Error (unexpected errors)**

Important Notes:

- **All date fields should be returned in ISO 8601 format**
- **Price fields should be numbers, not strings**
- **Always validate user permissions before performing operations**
- **Use try-catch blocks to handle database errors gracefully**
- **Return appropriate HTTP status codes with all responses**

Create a Menu Item User Story

- **Request Type: POST**
- **End Point: /api/v1/menuItem/new**
- **Request body: menuItem object**
- **Output: object**

menuItem object

key	Type
name	string
price	number
description	string
category	string

As a truck owner, I should be able to add items in to my menu. You need to insert a new menu item record in your MenuItems database table. You must use the default value of status column and createdAt column. The server should respond with a message that "menu item was created successfully".
Hint: You must use the function getUser to retrieve the current truck owner's truckId

Example Request Body:

```
{
  "name": "Beef Burger",
  "price": 45.99,
  "description": "Delicious beef burger with cheese",
  "category": "Main Course"
}
```

Example Success Response (200):

```
{
  "message": "menu item was created successfully"
}
```

View My Truck's Menu Items User Story

- **Request Type: GET**
- **End Point: /api/v1/menuItem/view**
- **Output: array of menuItem objects**

menuItem object

key	Type
itemId	integer
truckId	integer
name	string
description	string
price	number
category	string
status	string
createdAt	string

As a truck owner, I should be able to view all my menu items where status has a value of "available". I am not allowed to view other truck owner's menu items. The server should respond with an array of menuItem objects. The results should be sorted according to column itemId in table MenuItems.
Hint: You must use the function getUser to retrieve the current truck owner's truckId

Example Success Response (200):

```
[
  {
    "itemId": 1,
    "truckId": 5,
    "name": "Beef Burger",
    "description": "Delicious beef burger with cheese",
    "price": 45.99,
    "category": "Main Course",
    "status": "available",
    "createdAt": "2025-11-18T10:30:00.000Z"
  },
  {
    "itemId": 2,
    "truckId": 5,
    "name": "Chicken Wrap",
    "description": "Grilled chicken wrap",
    "price": 35.50,
    "category": "Main Course",
    "status": "available",
    "createdAt": "2025-11-18T11:00:00.000Z"
  }
]
```

View a Specific Menu Item User Story

- **Request Type: GET**
- **End Point: /api/v1/menuItem/view/:itemId**
- **Request params: itemId**
- **Output: menuItem object**

menuItem object

key	Type
itemId	integer
truckId	integer
name	string
description	string
price	number
category	string
status	string
createdAt	string

As a truck owner, I should be able to view all information of a specific menu item using its itemId. I am not allowed to view menu items from other truck owners. The server should respond with a menuItem object.

Example Success Response (200):

```
{
  "itemId": 1,
  "truckId": 5,
  "name": "Beef Burger",
  "description": "Delicious beef burger with cheese",
  "price": 45.99,
  "category": "Main Course",
  "status": "available",
  "createdAt": "2025-11-18T10:30:00.000Z"
}
```

Edit a Menu Item User Story

- **Request Type: PUT**
- **End Point: /api/v1/menuItem/edit/:itemId**
- **Request body: menuItem object**
- **Request params: itemId**
- **Output: object**

menuItem object

key	Type
name	string
price	number
category	string
description	string

As a truck owner, I should be able to edit my menu items only. I am not allowed to edit other truck owner's menu items. You need to update a menu item record in your table MenuItems. You are allowed to edit name, price, category, and description only. The server should respond with a message "menu item updated successfully".
Hint: You must use the function getUser to retrieve the current truck owner's truckId

Example Request Body:

```
{
  "name": "Premium Beef Burger",
  "price": 49.99,
  "category": "Main Course",
  "description": "Premium beef burger with special sauce"
}
```

Example Success Response (200):

```
{
  "message": "menu item updated successfully"
}
```

Delete a Menu Item User Story

- **Request Type: DELETE**
- **End Point: /api/v1/menuItem/delete/:itemId**
- **Request params: itemId**
- **Output: object**

As a truck owner, I should be able to delete my menu items only. I am not allowed to delete other truck owner's menu items. You need to update a menu item record in your table MenuItems by setting its status to "unavailable". The server should respond with a message "menu item deleted successfully".
Hint: You must use the function getUser to retrieve the current truck owner's truckId

Example Success Response (200):

```
{
  "message": "menu item deleted successfully"
}
```

View All Available Trucks User Story

- **Request Type: GET**
- **End Point: /api/v1/trucks/view**
- **Output: array of truck objects**

truck object

key	Type
truckId	integer
truckName	string
truckLogo	string
ownerId	integer
truckStatus	string
orderStatus	string
createdAt	string

As a customer, I should be able to view all trucks that have truckStatus as "available" and orderStatus as "available". The server should respond with an array of truck objects. The results should be sorted according to column truckId in table Trucks.

Example Success Response (200):

```
[
  {
    "truckId": 5,
    "truckName": "Tasty Tacos Truck",
    "truckLogo": "https://example.com/logo1.png",
    "ownerId": 10,
    "truckStatus": "available",
    "orderStatus": "available",
    "createdAt": "2025-10-15T08:00:00.000Z"
  },
  {
    "truckId": 8,
    "truckName": "Burger Paradise",
    "truckLogo": "https://example.com/logo2.png",
    "ownerId": 12,
    "truckStatus": "available",
    "orderStatus": "available",
    "createdAt": "2025-11-01T09:30:00.000Z"
  }
]
```

View Menu Items for a Specific Truck User Story

- **Request Type: GET**
- **End Point: /api/v1/menuItem/truck/:truckId**
- **Request params : truckId**
- **Output: array of menuItem objects**

menuItem object

key	Type
itemId	integer
truckId	integer
name	string
description	string
price	number
category	string
status	text
createdAt	date

As a customer, I should be able to view all menu items for a specific truck where status is "available". The server should respond with an array of menuItem objects. The results should be sorted according to column itemId in table MenuItems.

Example Success Response (200):

```
[
  {
    "itemId": 1,
    "truckId": 5,
    "name": "Beef Burger",
    "description": "Delicious beef burger with cheese",
    "price": 45.99,
    "category": "Main Course",
    "status": "available",
    "createdAt": "2025-11-18T10:30:00.000Z"
  },
  {
    "itemId": 3,
    "truckId": 5,
    "name": "French Fries",
    "description": "Crispy golden fries",
    "price": 15.00,
    "category": "Sides",
    "status": "available",
    "createdAt": "2025-11-18T10:45:00.000Z"
  }
]
```

Add Menu Item to Cart User Story

- **Request Type: POST**
- **End Point: /api/v1/cart/new**
- **Request body: cart object**
- **Output: object**

cart object

key	Type
itemId	integer
quantity	integer
price	number

As a customer, I should be able to add any available menu item to my shopping cart. You need to insert a new cart record in your database table Carts. You must use the getUser function to retrieve the current customer id (userId). Before adding the menu item to cart, verify that all items in the cart belong to the same truck (truckId). If items are from different trucks, respond with error message "Cannot order from multiple trucks". Otherwise, the server should respond with a message "item added to cart successfully".

Example Request Body:

```
{
  "itemId": 1,
  "quantity": 2,
  "price" : 45.99
}
```

Example Success Response (200):

```
{
  "message": "item added to cart successfully"
}
```

Example non Successful Response (400):

```
{
  "message": "Cannot order from multiple trucks",

}
```

View Cart User Story

- **Request Type: GET**
- **End Point: /api/v1/cart/view**
- **Output: array of cartItem objects**

cartItem object

key	Type
cartId	integer
userId	integer
itemId	integer
itemName	string
price	number
quantity	integer

As a customer, I should be able to view all items in my shopping cart based on my userId. You must use the function getUser to retrieve the current customer id (userId). The server should respond with an array of cartItem objects. This array should be sorted according to column cartId in table Carts.

Example Success Response (200):


```
[
  {
    "cartId": 1,
    "userId": 1,
    "itemId": 1,
    "itemName": "Beef Burger",
    "price": 45.99,
    "quantity": 2
  },
  {
    "cartId": 2,
    "userId": 1,
    "itemId": 3,
    "itemName": "French Fries",
    "price": 15.00,
    "quantity": 1
  }
]
```

Delete Item from Cart User Story

- **Request Type: DELETE**
- **End Point: /api/v1/cart/delete/:cartId**
- **Request params: cartId**
- **Output: object**

As a customer, I should be able to remove any item from my shopping cart. You need to delete a cart record using request params cartId. You must verify that the cart item belongs to the current user before deleting. The server should respond with a message "item removed from cart successfully".

Example Success Response (200):

```
{
  "message": "item removed from cart successfully"
}
```

Edit Cart Item Quantity User Story

- **Request Type: PUT**
- **End Point: /api/v1/cart/edit/:cartId**
- **Request params: cartId**
- **Request Body: cart object**
- **Output: string**

cart object

key	Type
quantity	integer

As a customer, I should be able to edit the quantity of items in my shopping cart. You need to update a cart record using request params cartId. You must verify that the cart item belongs to the current user before updating. The server should respond with a message "cart updated successfully".

Example Request Body:

```
{
  "quantity": 3
}
```

Example Success Response (200):

```
{
  "message": "cart updated successfully"
}
```

Place an Order User Story

- **Request Type: POST**
- **End Point: /api/v1/order/new**
- **Request body: orderRequest object**
- **Output: object**

orderRequest object

key	Type
scheduledPickupTime	timestamp

As a customer, I should be able to place an order from my shopping cart with a scheduled pickup time. You must use the function getUser to retrieve the current customer id (userId).

Before creating the order, verify that all items in the cart belong to the same truck (truckId). If items are from different trucks, respond with error message "Cannot order from multiple trucks". Then perform the following operations :

1. Calculate the totalPrice by summing all (price * quantity) from the user's cart items
2. Insert a new order record in table Orders with userId, truckId, orderStatus as "pending", totalPrice, scheduledPickupTime (from request), and estimatedEarliestPickup
3. Insert records in table OrderItems for each cart item with orderId, itemId, quantity, and price
4. Delete all cart records related to current customer id (userId)

The server should respond with a message
Server Response

Message	Case	code
Cannot order from multiple trucks	cart contains items from different trucks	400
order placed successfully	order created successfully	200

Example Request Body:

```
{
  "scheduledPickupTime": "2025-12-03T14:30:00.000Z"
}
```

Example Success Response (200):

```
{
  "message": "order placed successfully"
}
```

Example Error Response (400):

```
{
  "error": "Cannot order from multiple trucks"
}
```

View My Orders User Story

- **Request Type: GET**
- **End Point: /api/v1/order/myOrders**
- **Output: array of order objects**

order object

key	Type
orderId	integer
userId	integer
truckId	integer
truckName	string
orderStatus	text
totalPrice	number
scheduledPickupTime	timestamp
estimatedEarliestPickup	timestamp
createdAt	date

As a customer, I should be able to view all my past and current orders. You must use the function getUser to retrieve the current customer id (userId). The server should respond with an array of order objects including the truck name by joining with the Trucks table. The results should be sorted by orderId in descending order (most recent first).

Example Success Response (200):


```
[
  {
    "orderId": 15,
    "userId": 3,
    "truckId": 5,
    "truckName": "Tasty Tacos Truck",
    "orderStatus": "pending",
    "totalPrice": 106.98,
    "scheduledPickupTime": "2025-11-19T14:30:00.000Z",
    "estimatedEarliestPickup": "2025-11-19T14:00:00.000Z",
    "createdAt": "2025-11-19T01:45:00.000Z"
  },
  {
    "orderId": 12,
    "userId": 3,
    "truckId": 8,
    "truckName": "Burger Paradise",
    "orderStatus": "completed",
    "totalPrice": 85.50,
    "scheduledPickupTime": "2025-11-17T16:00:00.000Z",
    "estimatedEarliestPickup": "2025-11-17T15:30:00.000Z",
    "createdAt": "2025-11-17T14:20:00.000Z"
  }
]
```

View Order Details for Customer User Story

- **Request Type: GET**
- **End Point: /api/v1/order/details/:orderId**
- **Request params: orderId**
- **Output: orderDetails object**

orderDetails object

key	Type
orderId	integer
truckName	string
orderStatus	text
totalPrice	number
scheduledPickupTime	timestamp
estimatedEarliestPickup	timestamp
createdAt	date
items	array of orderItem objects

orderItem object

key	Type
itemName	string
quantity	integer
price	number

As a customer, I should be able to view the full details of a specific order including all items. You must verify that the order belongs to the current user before returning the details. The server should respond with an orderDetails object that includes order information and an array of items from the OrderItems table joined with MenuItems table.

Example Success Response (200):


```
{
  "orderId": 15,
  "truckName": "Tasty Tacos Truck",
  "orderStatus": "pending",
  "totalPrice": 106.98,
  "scheduledPickupTime": "2025-11-19T14:30:00.000Z",
  "estimatedEarliestPickup": "2025-11-19T14:00:00.000Z",
  "createdAt": "2025-11-19T01:45:00.000Z",
  "items": [
    {
      "itemName": "Beef Burger",
      "quantity": 2,
      "price": 45.99
    },
    {
      "itemName": "French Fries",
      "quantity": 1,
      "price": 15.00
    }
  ]
}
```

View Orders for My Truck User Story

- **Request Type: GET**
- **End Point: /api/v1/order/truckOrders**
- **Output: array of truckOrder objects**

truckOrder object

key	Type
orderId	integer
userId	integer
customerName	string
orderStatus	text
totalPrice	number
scheduledPickupTime	timestamp
estimatedEarliestPickup	timestamp
createdAt	date

As a truck owner, I should be able to view all orders placed for my truck. You must use the function getUser to retrieve the current truck owner's truckId. The server should respond with an array of truckOrder objects including the customer name by joining with the Users table. The results should be sorted by orderId in descending order (most recent first).

Example Success Response (200):


```
[
  {
    "orderId": 15,
    "userId": 3,
    "customerName": "Ahmed Mohamed",
    "orderStatus": "pending",
    "totalPrice": 106.98,
    "scheduledPickupTime": "2025-11-19T14:30:00.000Z",
    "estimatedEarliestPickup": "2025-11-19T14:00:00.000Z",
    "createdAt": "2025-11-19T01:45:00.000Z"
  },
  {
    "orderId": 14,
    "userId": 7,
    "customerName": "Sara Ali",
    "orderStatus": "preparing",
    "totalPrice": 75.50,
    "scheduledPickupTime": "2025-11-19T13:00:00.000Z",
    "estimatedEarliestPickup": "2025-11-19T12:30:00.000Z",
    "createdAt": "2025-11-19T01:20:00.000Z"
  }
]
```

Update Order Status User Story

- **Request Type: PUT**
- **End Point: /api/v1/order/updateStatus/:orderId**
- **Request params: orderId**
- **Request body: orderStatus object**
- **Output: object**

orderStatus object

key	Type
orderStatus	string
estimatedEarliestPickup	string

As a truck owner, I should be able to update the status of orders for my own truck only. Valid values for orderStatus are "pending", "preparing", "ready", "completed", and "cancelled". Before updating an order, the server must verify that the order belongs to the currently logged-in truck owner's truck. The estimatedEarliestPickup field may also be included in the update, but it is optional and should only be updated if provided. After successfully applying the update, the server should respond with the message: "order status updated successfully".

Example Request Body:

```
{
  "orderStatus": "preparing",
  "estimatedEarliestPickup": "2025-12-03 14:00:00"
}
```

Example Success Response (200):

```
{
  "message": "order status updated successfully"
}
```

View Order Details for Truck Owner User Story

- **Request Type: GET**
- **End Point: /api/v1/order/truckOwner/:orderId**
- **Request params: orderId**
- **Output: orderDetails object**

orderDetails object

key	Type
orderId	integer
truckName	string
orderStatus	text
totalPrice	number
scheduledPickupTime	timestamp
estimatedEarliestPickup	timestamp
createdAt	date
items	array of orderItem objects

orderItem object

key	Type
itemName	string
quantity	integer
price	number

As a truck owner, I should be able to view the full details of a specific order including all items. You must verify that the order belongs to the current truck owner before returning the details. The server should respond with an orderDetails object that includes order information and an array of items from the OrderItems table joined with MenuItems table.

Example Success Response (200):

```
{
  "orderId": 15,
  "truckName": "Tasty Tacos Truck",
  "orderStatus": "pending",
  "totalPrice": 106.98,
  "scheduledPickupTime": "2025-11-19T14:30:00.000Z",
  "estimatedEarliestPickup": "2025-11-19T14:00:00.000Z",
  "createdAt": "2025-11-19T01:45:00.000Z",
  "items": [
    {
      "itemName": "Beef Burger",
      "quantity": 2,
      "price": 45.99
    },
    {
      "itemName": "French Fries",
      "quantity": 1,
      "price": 15.00
    }
  ]
}
```

Update Truck Order Availability User Story

- **Request Type: PUT**
- **End Point: /api/v1/trucks/updateOrderStatus**
- **Request body: truckOrderStatus object**
- **Output: response**

truckOrderStatus object

key	Type
orderStatus	string

As a truck owner, I should be able to toggle my truck's order availability. Valid orderStatus values are: "available" or "unavailable". You must use the function getUser to retrieve the current truck owner's truckId. You need to update the orderStatus column in the Trucks table for your truck only. The server should respond with a message "truck order status updated successfully".

Example Request Body:

```
{
  "orderStatus": "unavailable"
}
```

Example Success Response (200):

```
{
  "message": "truck order status updated successfully"
}
```

Search Menu Items by Category User Story

- **Request Type: GET**
- **End Point: /api/v1/menuItem/truck/:truckId/category/:category**
- **Request params: truckId, category**
- **Output: array of menuItem objects**

menuItem object

key	Type
itemId	integer
truckId	integer
name	string
description	string
price	number
category	string
status	string
createdAt	string

As a customer, I should be able to search and filter menu items by category for a specific truck. Only items with status "available" should be returned. The server should respond with an array of menuItem objects matching the specified category. The results should be sorted according to column itemId in table MenuItems.

Example Success Response (200):


```
[
  {
    "itemId": 1,
    "truckId": 5,
    "name": "Beef Burger",
    "description": "Delicious beef burger with cheese",
    "price": 45.99,
    "category": "Main Course",
    "status": "available",
    "createdAt": "2025-11-18T10:30:00.000Z"
  },
  {
    "itemId": 2,
    "truckId": 5,
    "name": "Chicken Wrap",
    "description": "Grilled chicken wrap",
    "price": 35.50,
    "category": "Main Course",
    "status": "available",
    "createdAt": "2025-11-18T11:00:00.000Z"
  }
]
```

View My Truck Information User Story

- **Request Type: GET**
- **End Point: /api/v1/trucks/myTruck**
- **Output: truck object**

truck object

key	Type
truckId	integer
truckName	string
truckLogo	string
ownerId	integer
truckStatus	text
orderStatus	text
createdAt	date

As a truck owner, I should be able to view my truck's information including current status settings. You must use the function getUser to retrieve the current truck owner's truckId. The server should respond with a truck object containing all information about the truck owner's truck.

Example Success Response (200):

```
{
  "truckId": 5,
  "truckName": "Tasty Tacos Truck",
  "truckLogo": "https://example.com/logo.png",
  "ownerId": 10,
  "truckStatus": "available",
  "orderStatus": "available",
  "createdAt": "2025-10-15T08:00:00.000Z"
}
```

API Endpoints Quick Reference

This table summarizes all the endpoints you need to implement. Use it as a checklist to ensure you have completed all user stories.

Method	Endpoint	Role	Description
Menu Item Management			
POST	/api/v1/menuItem/new	Truck Owner	Create menu item
GET	/api/v1/menuItem/view	Truck Owner	View my menu items
GET	/api/v1/menuItem/view/:itemId	Truck Owner	View specific menu item
PUT	/api/v1/menuItem/edit/:itemId	Truck Owner	Edit menu item
DELETE	/api/v1/menuItem/delete/:itemId	Truck Owner	Delete menu item
Truck Management			
GET	/api/v1/trucks/view	Customer	View all available trucks
GET	/api/v1/trucks/myTruck	Truck Owner	View my truck info
PUT	/api/v1/trucks/updateOrderStatus	Truck Owner	Update truck availability
Browse Menu			
GET	/api/v1/menuItem/truck/:truckId	Customer	View truck's menu
GET	/api/v1/menuItem/truck/:truckId/category/:category	Customer	Search menu by category
Cart Management			
POST	/api/v1/cart/new	Customer	Add item to cart
GET	/api/v1/cart/view	Customer	View cart
PUT	/api/v1/cart/edit/:cartId	Customer	Update cart quantity
DELETE	/api/v1/cart/delete/:cartId	Customer	Remove from cart
Order Management			
POST	/api/v1/order/new	Customer	Place order
GET	/api/v1/order/myOrders	Customer	View my orders
GET	/api/v1/order/details/:orderId	Customer	View order details
GET	/api/v1/order/truckOwner/:orderId	Truck Owner	View order details
GET	/api/v1/order/truckOrders	Truck Owner	View truck's orders
PUT	/api/v1/order/updateStatus/:orderId	Truck Owner	Update order status

Total Endpoints to Implement: 20

- **Truck Owner Endpoints: 10**
- **Customer Endpoints: 10**

Submission guidelines:

- **Submit all your code in ONE .zip file.**
- **Rename the .zip folder in the following format ID-Tutorial#ID-Tutorial (1000546-T8#1000678-T9).**
- **Include all your teams ID and tutorial same as in the above format**
- **Ensure the script.sql file is included in the milestoneBackend/connectors/ folder**
- **You must invite the following emails as collaborators to your GitHub repository:**
  - **amir.haytham@giu-uni.de**
  - **ahmed.sherif@giu-uni.de**
  - **mohamed.essam@giu-uni.de**
  - **yassein.eldamasy@giu-uni.de**
- **The submission link is https://forms.gle/PmH8sQMCejjo88br7**
- **It is YOUR responsibility to ensure that you have submitted the correct .zip file and saved the correct content before submitting. Make sure to submit before deadline.**
- **Good luck! :)**