**TEAM- THE CODE CRAFTERS**

**INVOICE MICRO SERVICE**

**ENDPOINTS**

**1.Create an Invoice:**

POST /invoices: Endpoint to create a new invoice.

**2.Get Invoice by ID:**

GET /invoices/{id}: Retrieve a specific invoice by its unique identifier.

**3.Update Invoice:**

PUT /invoices/{id}: Update an existing invoice identified by its ID.

PATCH /invoices/{id}: Partially update an existing invoice.

**4.Delete Invoice:**

DELETE /invoices/{id}: Delete an invoice by its ID.

**5.List Invoices:**

GET /invoices: Retrieve a list of all invoices.

**6.Filtering and Sorting:**

GET /invoices?status={status}: Get invoices based on status (e.g., paid, pending).

GET /invoices?customer={customer\_id}: Retrieve invoices for a specific customer.

GET /invoices?date={date}: Fetch invoices created on a particular date or within a date range.

GET /invoices?sort={field}: Sort invoices based on specific fields like date, amount, etc.

**7.Invoice Actions:**

POST /invoices/{id}/pay: Endpoint to mark an invoice as paid.

POST /invoices/{id}/cancel: Cancel an invoice.

POST /invoices/{id}/send: Send an invoice to the customer.

**8.Invoice Status:**

GET /invoices/{id}/status: Retrieve the status of a particular invoice.

**9.Invoice Payments:**

GET /invoices/{id}/payments: Retrieve payment history or details for a specific invoice.

POST /invoices/{id}/payments: Record a payment against an invoice.

**10.Reports and Analytics:**

GET /invoices/report: Endpoint to generate reports related to invoices (e.g., revenue, overdue invoices).

**Entities**

**1.Invoice Entity:**

An Invoice entity contains information about an invoice, such as:

ID: Unique identifier for the invoice.

Number: Invoice number or reference.

Date: Date of the invoice.

Amount: Total amount due on the invoice.

Status: Current status of the invoice (e.g., paid, pending, cancelled).

Customer ID: Identifier of the customer associated with the invoice.

Items: Details of items or services provided, including descriptions, quantities, and prices.

**2.Customer Entity:**

A c. It may include:

ID: Unique identifier for the customer.

Name: Customer's name or business name.

Email: Contact email address of the customer.

Address: Customer's address details.

Phone: Contact phone number.

**3.Payment Entity:**

A Payment entity records transactions related to an invoice's payment. It could consist of:

ID: Unique identifier for the payment.

Invoice ID: Identifier linking the payment to a specific invoice.

Amount: The amount paid.

Date: Date of the payment.

Method: Payment method (e.g., credit card, cash, bank transfer).

Status: Payment status (e.g., completed, pending, refunded).

import javax.persistence.\*;

import java.util.Date;

import java.util.List;

@Entity

public class Invoice {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String number;

private Date date;

private double amount;

private String status;

@ManyToOne

@JoinColumn(name = "customer\_id")

private Customer customer;

// Assume a one-to-many relationship between Invoice and InvoiceItems

@OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)

private List<InvoiceItem> items;

// Getters and setters, constructors, etc.

// ...

}

@Entity

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private String address;

private String phone;

// Getters and setters, constructors, etc.

// ...

}

@Entity

public class Payment {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private double amount;

private Date date;

private String method;

private String status;

@ManyToOne

@JoinColumn(name = "invoice\_id")

private Invoice invoice;

// Getters and setters, constructors, etc.

// ...

}

**Auth Microservice**

**/api/auth/register**

Description: User registration endpoint.

Functionality: Allows users to create a new account by providing necessary registration information.

**/api/auth/login**

Description: User login endpoint.

Functionality: Validates user credentials and generates an authentication token upon successful login.

**/api/auth/logout**

Description: User logout endpoint.

Functionality: Invalidates the user's authentication token, logging them out of the system.

**/api/auth/user/{id}**

Description: Get user information by ID.

Functionality: Retrieves detailed information about a specific user based on their ID.

**/api/auth/update**

Description: Update user profile information.

Functionality: Allows users to update their profile information, such as username, email, or password.

**/api/auth/reset-password**

Description: Request a password reset.

Functionality: Initiates the process for users to reset their password by sending a reset link or token.

**/api/auth/validate-token**

Description: Validate user authentication token.

Functionality: Verifies the validity of a user's authentication token.

**### Entity:**

**User ID:** Unique identifier for each user.

**Username:** The username chosen by the user.

**Email:** User's email address.

**Password:** Securely stored user password (hashed).

**Token:** Authentication token issued upon successful login.

**Profile Information:** Additional information about the user, such as name, address, etc.

**Reset Token:** Token generated for password reset requests.

**Role/Roles:** User roles defining permissions (e.g., regular user, admin).

**Search MS**

-Endpoints:

1. /api/search -basic search for products/items  
2. /api/search/product/{id} -for specific product  
3. /api/search/trending -list of popular products  
4./api/search/recentlyviewed - products/items recentlyviewed by user  
5./api/search/category -search based on specific category  
6./api/search/filter :Apply filters for attributes like price range, brand  
7./api/suggestion : Provides suggestions as users type in the search bar

**Entity:**Id  
Name/title  
Description  
price  
category  
brand  
availability  
ratings