**TECHNICAL DESIGN PAGE INVOICE MICROSERVICE**

**OVERVIEW:**

Purpose of an invoice, It helps to keep track of sales made. Financial records- It serves as evidence for tax filings done each year. It is also useful during audits. The invoice micro service is responsible for billing, payment related functionalities, including

* create a new invoice
* Retrieve a specific invoice by its unique identifier
* Update an existing invoice identified by its ID
* Delete an invoice by its ID
* Retrieve a list of all invoices
* Get invoices based on status (e.g., paid, pending)
* Retrieve invoices for a specific customer
* Sort invoices based on specific fields like date, amount, etc.
* To mark an invoice as paid
* Cancel an invoice
* Send an invoice to the customer
* Retrieve the status of a particular invoice
* Retrieve payment history or details for a specific invoice
* Record a payment against an invoice
* To generate reports related to invoices (e.g., revenue, overdue invoices).

**ARCHITECTURE:**

Programming Language: Java (17)

Framework: Spring Boot (Java)

Database: MySQL/AWS

Authentication: OAuth 2.0 for secure authentication.

**HIGH-LEVEL ARCHITECTURE:**

RESTful API endpoints for communication.

It interacts with other microservices such as the order Microservice,

Order Microservice, and Authentication Microservice.

**FUNCTIONALITY:**

**1.Create an Invoice:**

POST Endpoint: osa/api/invoices: Endpoint to create a new invoice.

**2.Get Invoice by ID:**

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

**3.Update Invoice:**

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

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

**4.Delete Invoice:**

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

**5.List Invoices:**

GET Endpoint: osa/api/invoices: Retrieve a list of all invoices.

**6.Filtering and Sorting:**

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

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

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

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

**7.Invoice Actions:**

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

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

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

**8.Invoice Status:**

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

**9.Invoice Payments:**

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

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

**10.Reports and Analytics:**

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

**AUTHORIZATION:**

Implement role-based access control (RBAC) to manage user permissions.

Define roles such as User or Admin.

**SECURITY:**

Implement HTTPS for secure communication.

Store passwords securely using industry-standard hashing algorithms like bcrypt.

Use JWT tokens for authentication and authorization.

**ERROR HANDLING:**

Define clear error messages and status codes for API responses.

Log errors for debugging purposes.

**TESTING:**

Implement unit testing with code coverage.

integration tests (if Required).

end-to-end tests to ensure the reliability and stability of the microservice.

**DOCUMENTATION:**

Create comprehensive API documentation using tools like Swagger or OpenAPI.

**DEPLOYMENT:**

Utilize continuous integration and continuous deployment (CI/CD) pipelines for automated testing and deployment.

**FUTURE CONSIDERATION:**

Consider features like user notifications, password recovery.