Design Documents

1.**System architecture**:

2.T**echnologies**

- .NET 9
- GraphQL
- gRPC
- Entity Framework Core (PostgreSQL)
- xUnit + Moq cho testing
- Hangfire
- Docker

3. Data Model

```
4 references
public class Invoice : BaseEntity<Guid>
{
    3 references | 2/2 passing
    public Guid OrderId { get; set; }
    1 reference
    public decimal Amount { get; set; }
    1 reference
    public DateTimeOffset IssuedAt { get; set; }
}
```

```
22 references
public class Order : BaseEntity<Guid>
{
    12 references | 5/5 passing
    public string Name { get; set; }           // fo
    10 references | 5/5 passing
    public decimal Amount { get; set; }
    16 references | 5/5 passing
    public EOrderStatus Status { get; set; }
    3 references
    public DateTimeOffset CreatedAt { get; set; }
    6 references
    public DateTimeOffset UpdatedAt { get; set; }


}
```

```
17 references
public enum EOrderStatus
{
    Created,
    PaymentFailed,
    CheckedOut,
    PushOrderFailed,
    Complete
}
```

4. GraphQL Endpoint (API Gateway)

```
type Query {
  orders(name: String!): [Order!]!
}

type Mutation {
  checkoutOrder(orderId: String!): CheckoutResult!
}

type Order {
  id: String!
  name: String!
  status: OrderStatus!
}

type CheckoutResult {
  success: Boolean!
  message: String!
}
```

5. gRPC Services

product.proto:

```
syntax = "proto3";

option csharp_namespace = "DemoProject.GrpcServer.ProductService";

package productservice;

service ProductionService {
    rpc PushOrder (PushOrderRequest) returns (PushOrderResponse);
}
message PushOrderRequest {
    string order_id = 1;
}
message PushOrderResponse {
    bool success = 1;
    string message = 2;
}
```

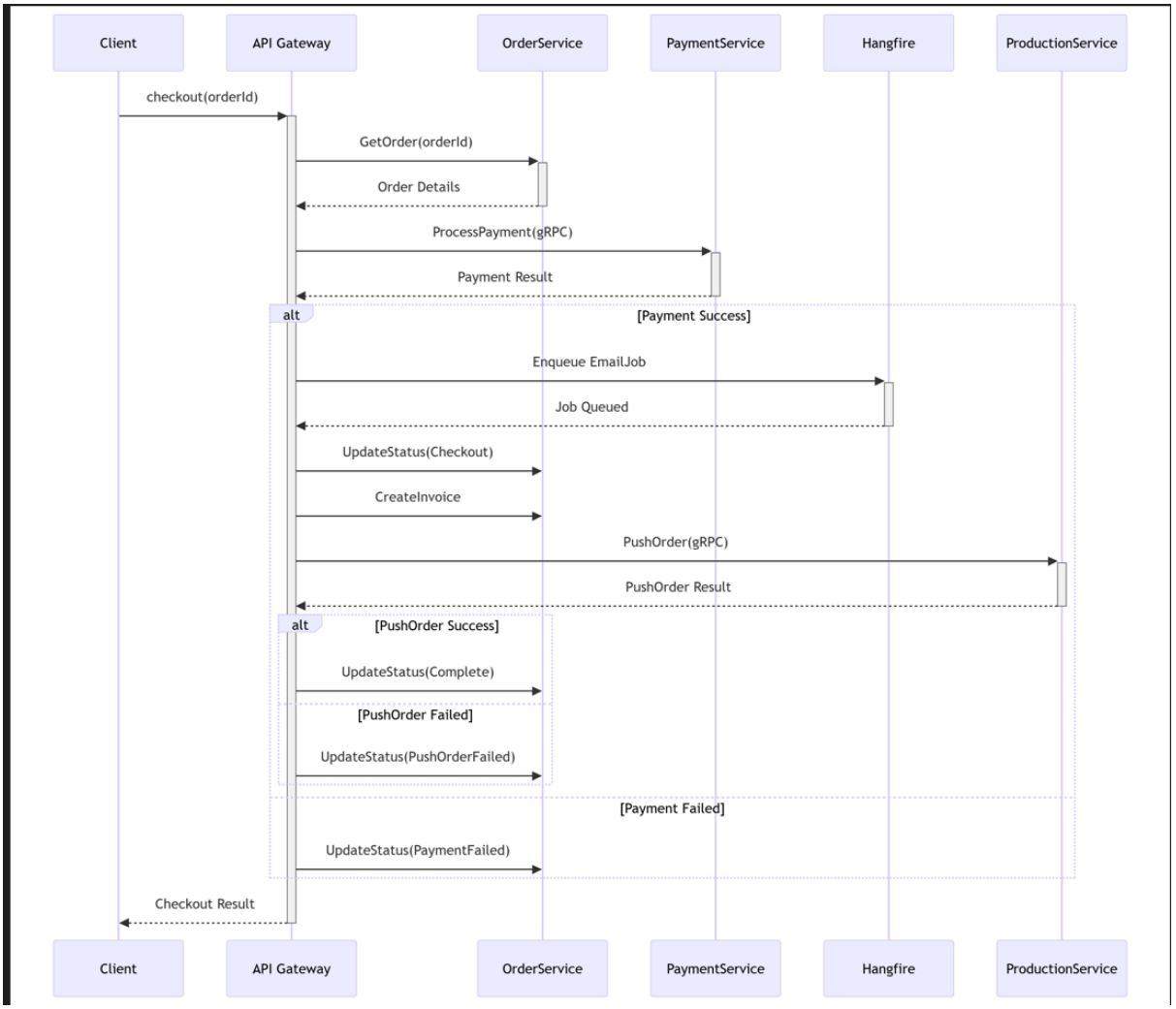
email.proto:

```
syntax = "proto3";

option csharp_namespace = "DemoProject.GrpcServer.EmailService";

package emailservice;

service EmailService {
    rpc SendOrderConfirmation (SendOrderConfirmationRequest) returns (SendOrderConfirmationResponse);
}
message SendOrderConfirmationRequest {
    string order_id = 1;
}
message SendOrderConfirmationResponse {
    bool success = 1;
    string message = 2;
}
```

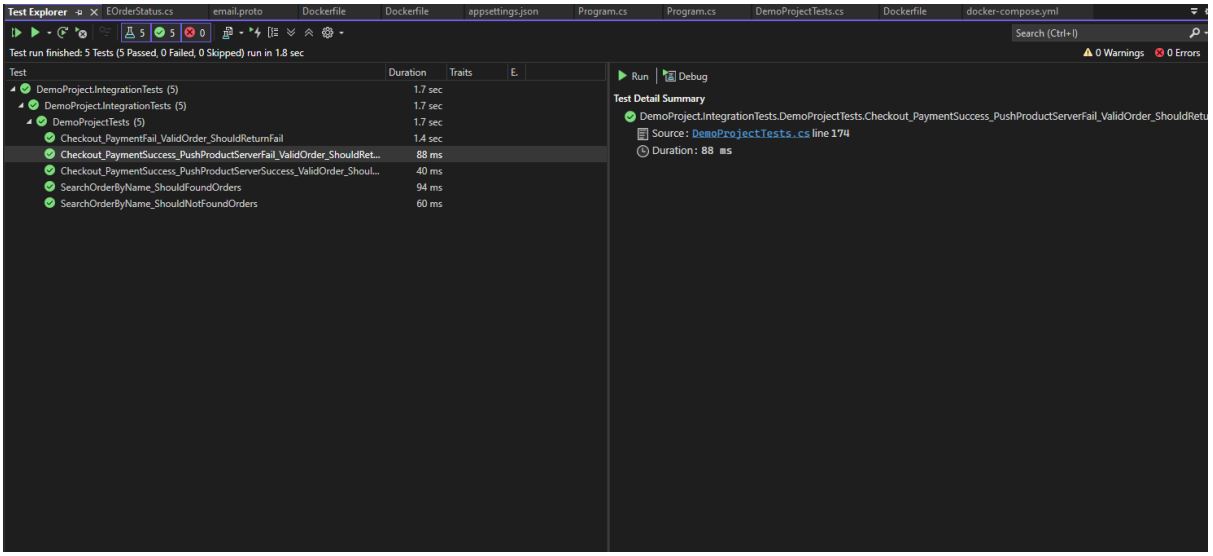
payment.proto:

```
syntax = "proto3";

option csharp_namespace = "DemoProject.GrpcServer.PaymentService";

package paymentservice;

service PaymentService {
    rpc ProcessPayment (ProcessPaymentRequest) returns (ProcessPaymentResponse);
}
message ProcessPaymentRequest {
    string order_id = 1;
}
message ProcessPaymentResponse {
    bool success = 1;
    string message = 2;
}
```

6.**CheckOut Flow**



7.**Integration test**



8 Estimation

	A	B
	Process	**Hours**
	Design system	8
	Design gRPC	2
	Design GraphQL	1
	Integration test	6
	Self-test	4
	Docker Deploy	1
	Document	1
	Total	**23**

9.How Test

-Clone Source code:

https://github.com/voynguyendev/projectdemo_pixelz_demo.git

-Run

```
docker-compose build --no-cache

docker-compose up -d

docker-compose restart (Option :make sure all services running)
```

-Open PostMan

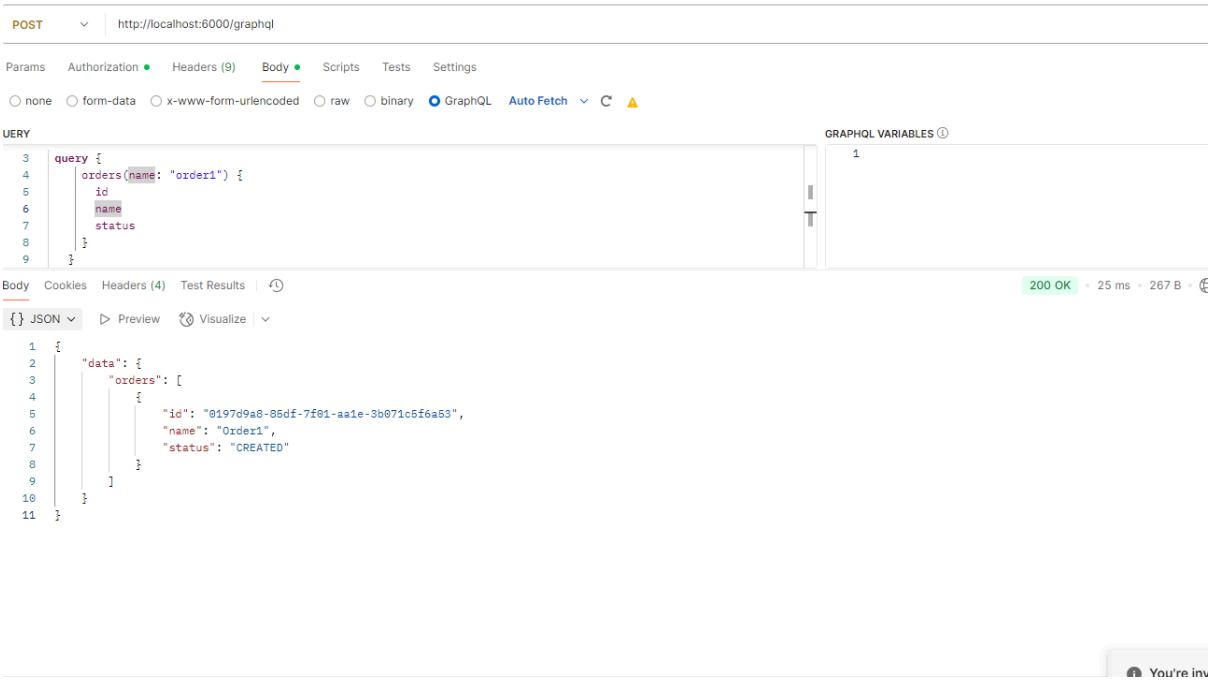
+ Search Order By Name :

End point: http://localhost:6000/graphql

Query GrapghQL :

```
query {
  orders(name: "order1") {
    id
    name
    status
  }
}
```

Result:



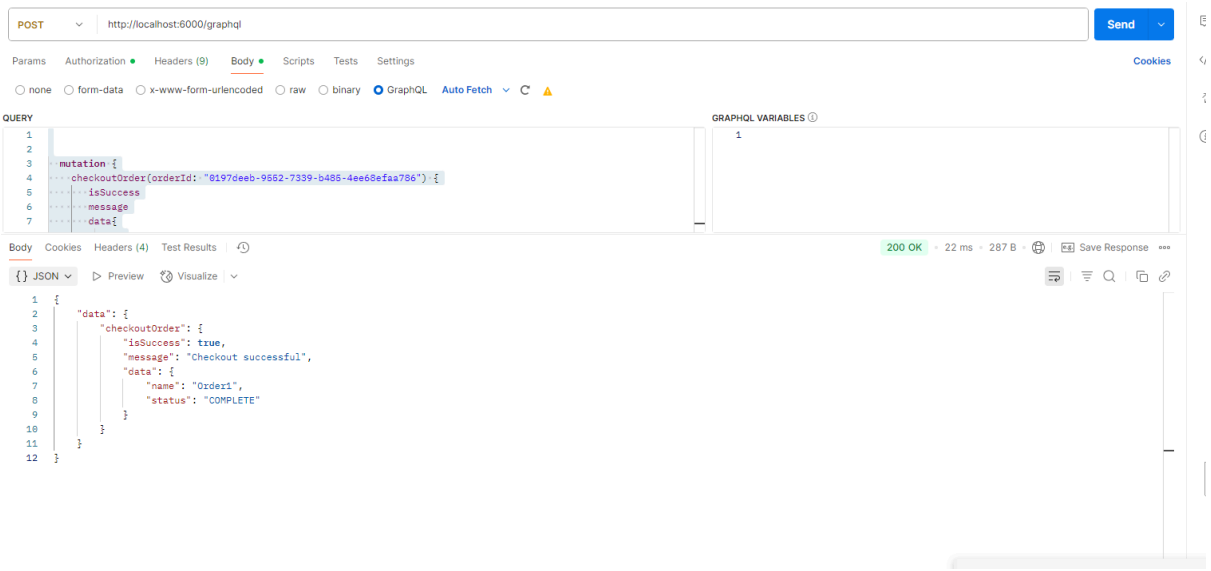
+Check Out Order:

End point: http://localhost:6000/graphql

`Mutation` GrapghQL :

```
mutation {
  checkoutOrder(orderId: "0197deeb-9552-7339-b485-4ee68efaa786") {
     isSuccess
     message
     data{
       name
       status

     }
   }
}
```

Result:


POST
http://localhost:6000/graphql
Send
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Auto Fetch
QUERY
GRAPHQL VARIABLES
mutation {
checkoutOrder(orderId: "0197deeb-9552-7339-b485-4ee68efaa786") {
isSuccess
message
data{
Body
Cookies
Headers (4)
Test Results
200 OK
22 ms
287 B
Save Response
JSON
Preview
Visualize
{
"data": {
"checkoutOrder": {
"isSuccess": true,
"message": "Checkout successful",
"data": {
"name": "Order1",
"status": "COMPLETE"
}
}
}
}