

Design Documents

1. System architecture:

```
[Client]
  | (GraphQL)
  ▼
[API Gateway] - (gRPC) - [Payment Service]
  | (gRPC)                [Email Service]
  ▼                      [Production Service]
[Order Service]
  |
  ▼
[Database]
```

```
graph TD
    Client[Client] -- "(GraphQL)" --> APIGateway[API Gateway]
    APIGateway -- "(gRPC)" --> PaymentService[Payment Service]
    APIGateway -- "(gRPC)" --> EmailService[Email Service]
    APIGateway -- "(gRPC)" --> ProductionService[Production Service]
    OrderService[Order Service] -- "(gRPC)" --> Database[Database]
```

2.Technologies

- .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; } // for
    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;  
}
```

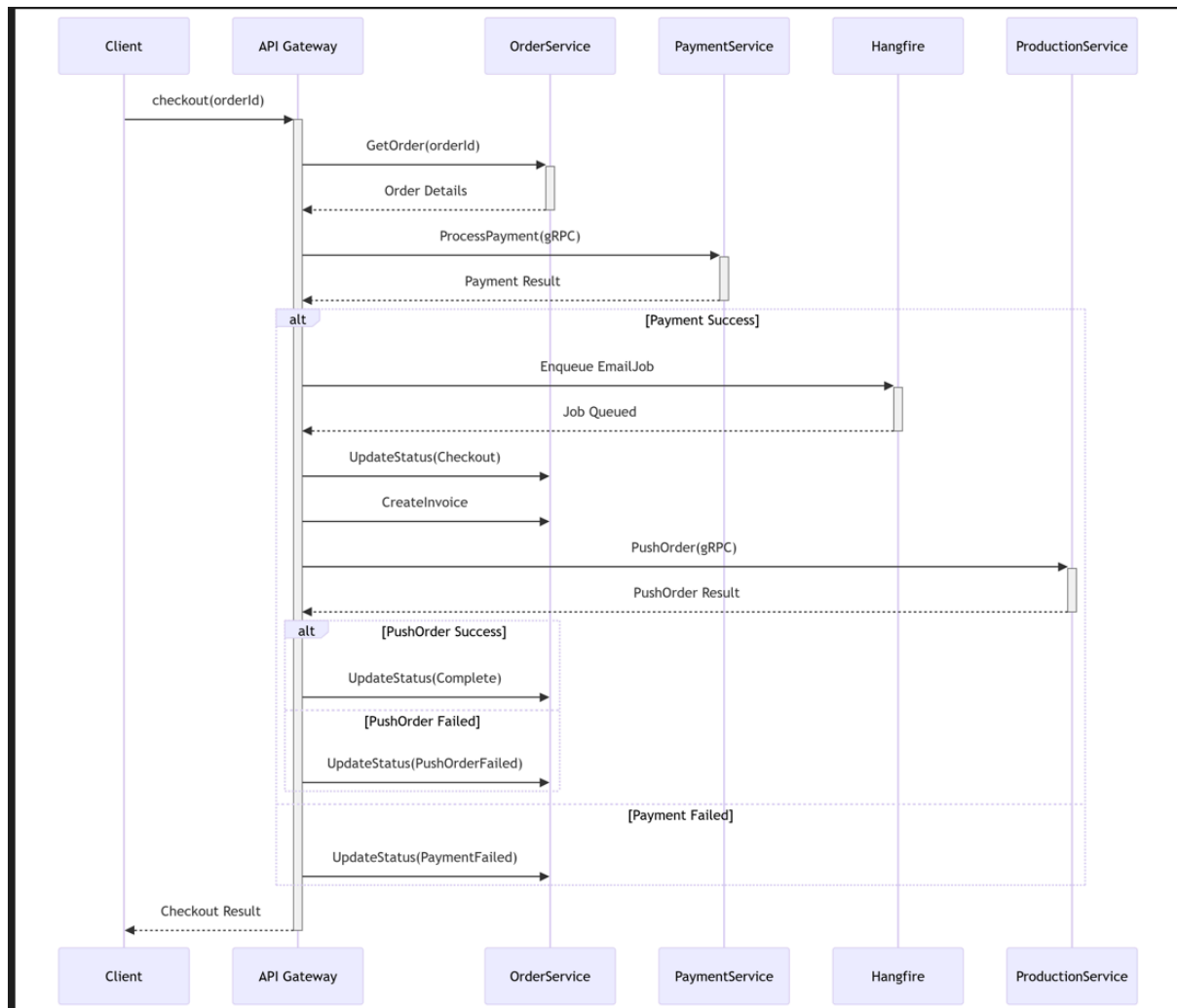
email.proto:

```
1  syntax = "proto3";
2
3  option csharp_namespace = "DemoProject.GrpcServer.EmailService";
4
5  package emailservice;
6
7  service EmailService {
8      rpc SendOrderConfirmation (SendOrderConfirmationRequest) returns (SendOrderConfirmationResponse);
9  }
10 message SendOrderConfirmationRequest {
11     string order_id = 1;
12 }
13 message SendOrderConfirmationResponse {
14     bool success = 1;
15     string message = 2;
16 }
```

payment.proto:

```
1  syntax = "proto3";
2
3  option csharp_namespace = "DemoProject.GrpcServer.PaymentService";
4
5  package paymentservice;
6
7  service PaymentService {
8      rpc ProcessPayment (ProcessPaymentRequest) returns (ProcessPaymentResponse);
9  }
10 message ProcessPaymentRequest {
11     string order_id = 1;
12 }
13 message ProcessPaymentResponse {
14     bool success = 1;
15     string message = 2;
16 }
```

6. CheckOut Flow



7. Integration test

The screenshot shows the Visual Studio Test Explorer interface. The top pane displays the test hierarchy: `DemoProject.IntegrationTests (5)`, `DemoProject.IntegrationTests (5)`, and `DemoProjectTests (5)`. The bottom pane shows the test results table:

Test	Duration	Traits	E.
✓ DemoProject.IntegrationTests (5)	1.7 sec		
✓ DemoProject.IntegrationTests (5)	1.7 sec		
✓ DemoProjectTests (5)	1.7 sec		
✓ Checkout.PaymentFail_ValidOrder_ShouldReturnFail	1.4 sec		
✓ Checkout.PaymentSuccess_PushProductServerFail_ValidOrder_ShouldRetu...	88 ms		
✓ Checkout.PaymentSuccess_PushProductServerSuccess_ValidOrder_Shoul...	40 ms		
✓ SearchOrderByName_ShouldFoundOrders	94 ms		
✓ SearchOrderByName_ShouldNotFoundOrders	60 ms		

The right pane shows the `Test Detail Summary` for the selected test: `DemoProject.IntegrationTests.DemoProjectTests.Checkout_PaymentSuccess_PushProductServerFail_ValidOrder_ShouldRetu...`. The source is `DemoProjectTests.cs` line 174, and the duration is 88 ms. The status indicates 0 Warnings and 0 Errors.

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:

The screenshot shows a GraphQL client interface with the following details:

- Method:** POST
- URL:** http://localhost:6000/graphql
- Body:** A GraphQL query to fetch an order by name:

```
query {
  orders(name: "order1") {
    id
    name
    status
  }
}
```
- Response:** 200 OK, 25 ms, 267 B. The response body is JSON:

```
{
  "data": {
    "orders": [
      {
        "id": "0197d9a8-85df-7f01-aa1e-3b071c5f6a53",
        "name": "Order1",
        "status": "CREATED"
      }
    ]
  }
}
```
- GraphQL Variables:** 1
- Footer:** You're inv

+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 C ⚠

QUERY

```
1
2
3 mutation {
4   checkoutOrder(orderId: "8197deeb-9682-7339-b485-4ee68efaa786") {
5     isSuccess
6     message
7     data {
```

GRAPHQL VARIABLES

```
1
```

Body Cookies Headers (4) Test Results 200 OK 22 ms 287 B Save Response

JSON Preview Visualize

```
1 {
2   "data": {
3     "checkoutOrder": {
4       "isSuccess": true,
5       "message": "Checkout successful",
6       "data": {
7         "name": "Order1",
8         "status": "COMPLETE"
9       }
10    }
11  }
12 }
```