Basic Concepts

1. Coding

A. Code Smell

Data Clumps

Data Clumps is usually occured when one piece of the data is often found together

Example:

Booking.java

```
public class Booking {

   // attr
   private Long orderId;
   private Long roomId;
   private LocalDate startDate;
   private LocalDate endDate;

   // def construct
   public Booking() {}

   // getter & setter
   public Long getOrderId() { return orderId; }
   public void setOrderId(Long orderId) { this.orderId = orderId; }
   public Long getRoomId() { return roomId; }
   public void setRoomId(Long roomId) { this.roomId = roomId; }
   public LocalDate getStartDate() { return startDate; }
   public void setStartDate(LocalDate startDate) { this.startDate; }
   public LocalDate getEndDate() { return endDate; }
   public void setEndDate(LocalDate endDate) { this.endDate; }

}
```

startDate and endDate often found together, so we can solve it by extracting an object that contains startDate and endDate. this solution called as *Extraction Class.*

Solution:

DateRange.java

```
public class DateRange {

   // attr
   private LocalDate startDate;
   private LocalDate endDate;

   // def construct
   public DateRange() {}

   // getter & setter
   public LocalDate getStartDate() { return startDate; }
   public void setStartDate(LocalDate startDate) { this.startDate; }
   public LocalDate getEndDate() { return endDate; }
   public void setEndDate(LocalDate endDate) { this.endDate; }

}
```

Booking.java

```
public class Booking {

    // attr
    private Long orderId;
    private Long roomId;
    private DateRange dateRange = new DateRange();

    // def construct
    public Booking() {}

    // getter & setter
    public Long getOrderId() { return orderId; }
    public void setOrderId(Long orderId) { this.orderId = orderId; }
    public Long getRoomId() { return roomId; }
    public void setRoomId(Long roomId) { this.roomId = roomId; }
    public LocalDate getStartDate() { return dateRange.getStartDate(); }
    public void setStartDate(LocalDate startDate) { this.dateRange.setStartDate(startDate); }
    public LocalDate getEndDate() { return dateRange.getEndDate(); }
    public void setEndDate(LocalDate endDate) { this.dateRange.setEndDate(endDate); }

}
```

Prevention:

Data clumps usually occured when we used agile methodology to develop a project. To prevent such things we have to see a bigger picture when creating a model or an object.

B. Dependency Injection

What is dependency injection ?

Dependency Injection means Injecting the current class with the other class that the current class depends on.

Why is it important ?

1. To instantiate a class we have to create a new object that used by many classes and stored in the heap, which we don't know when the garbage is collected. and Dependency Injection Framework can solve that problem.

2. To make App testing easier, which we don't have to instantiate a new class by making a real object , which later on connected to the real database.

3. Dependency injection makes the instantiated class as a singleton, which means the same instance will be reused for many classes that injected by that instantiated class.

Basic Concepts

Before Dependency Injection:

```
public class EmailService {

   private final ContactListService contactListService;

   // constructor
   public EmailService() {
      this.contactListService = new ContactListService();
   }

   // methods...

}

public class ContactListService {

   // constructor
   public ContactListService() {}

   // methods...

}
```

After Dependency Injection *(SpringBoot)*:

```
public class EmailService {

   private final ContactListService contactListService;

   // constructor
   @Inject
   public EmailService(ContactListService contactListService) {
      this.contactListService = contactListService;
   }

   // methods...

}

@Service
public class ContactListService {

   // constructor
   public ContactListService() {}

   // methods...

}
```

A. Rest API

POST Request

Don't: send 200 OK response status if the request is successful.

Do: send 201 Created response status instead.

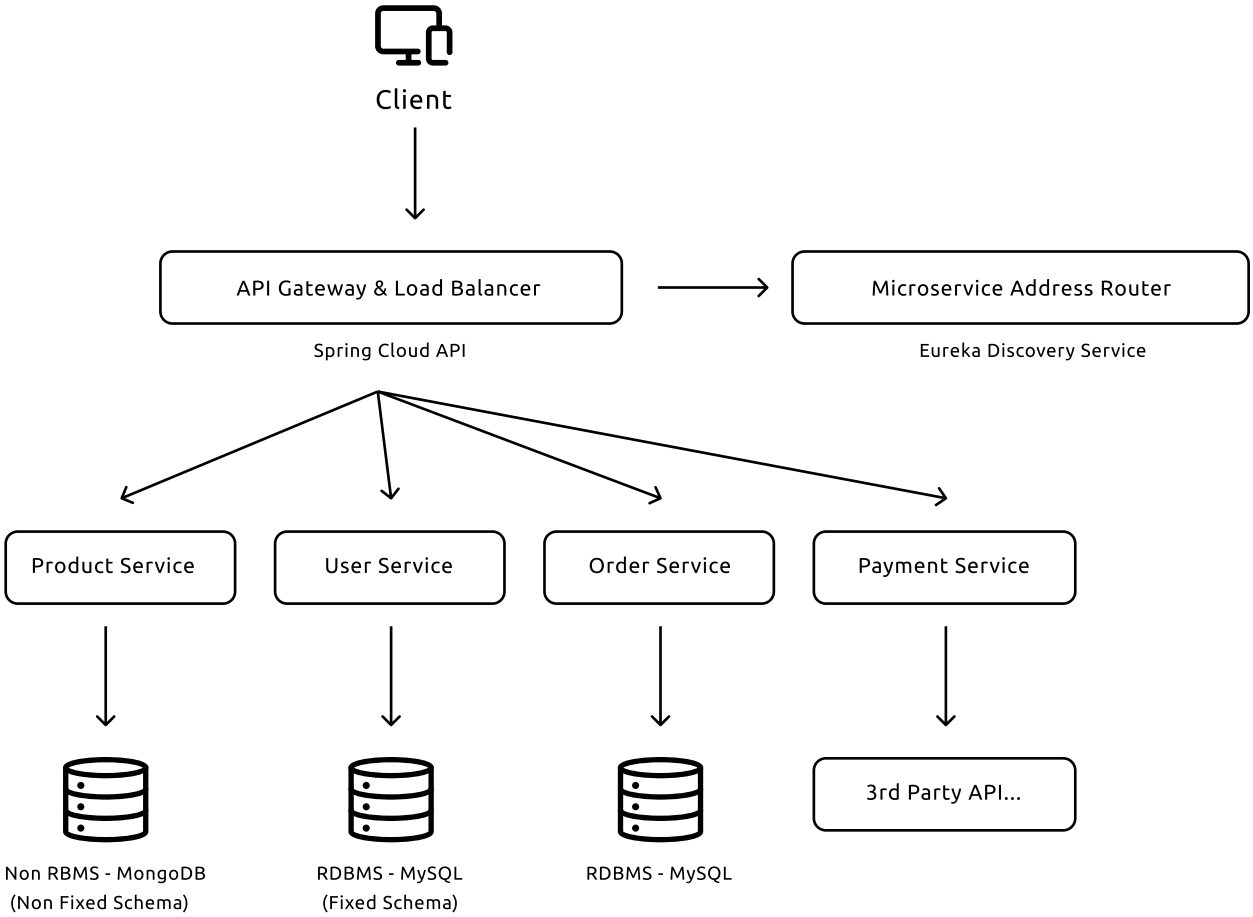
GET Request

Don't: send 400 BadRequest response status if the request is unsuccessful.

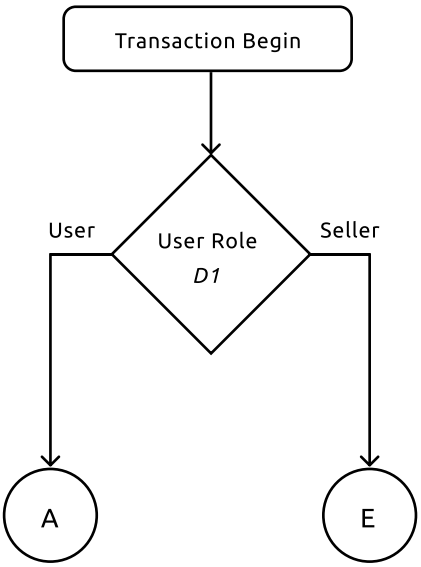
Do: send 503 Service Unavailable response status instead.

Basic Coding

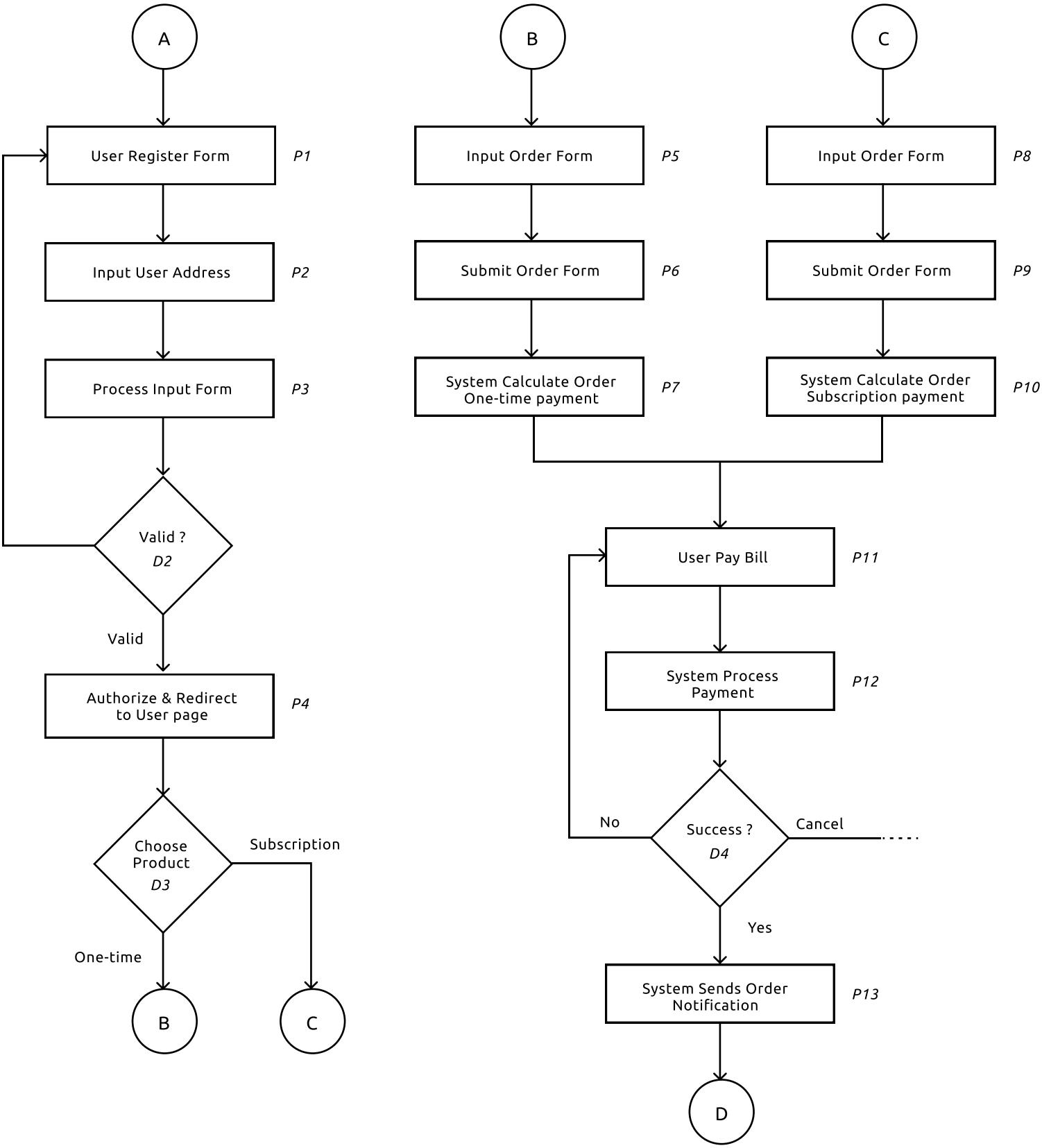
1. System Design (*Case Java SpringBoot Microservice*)



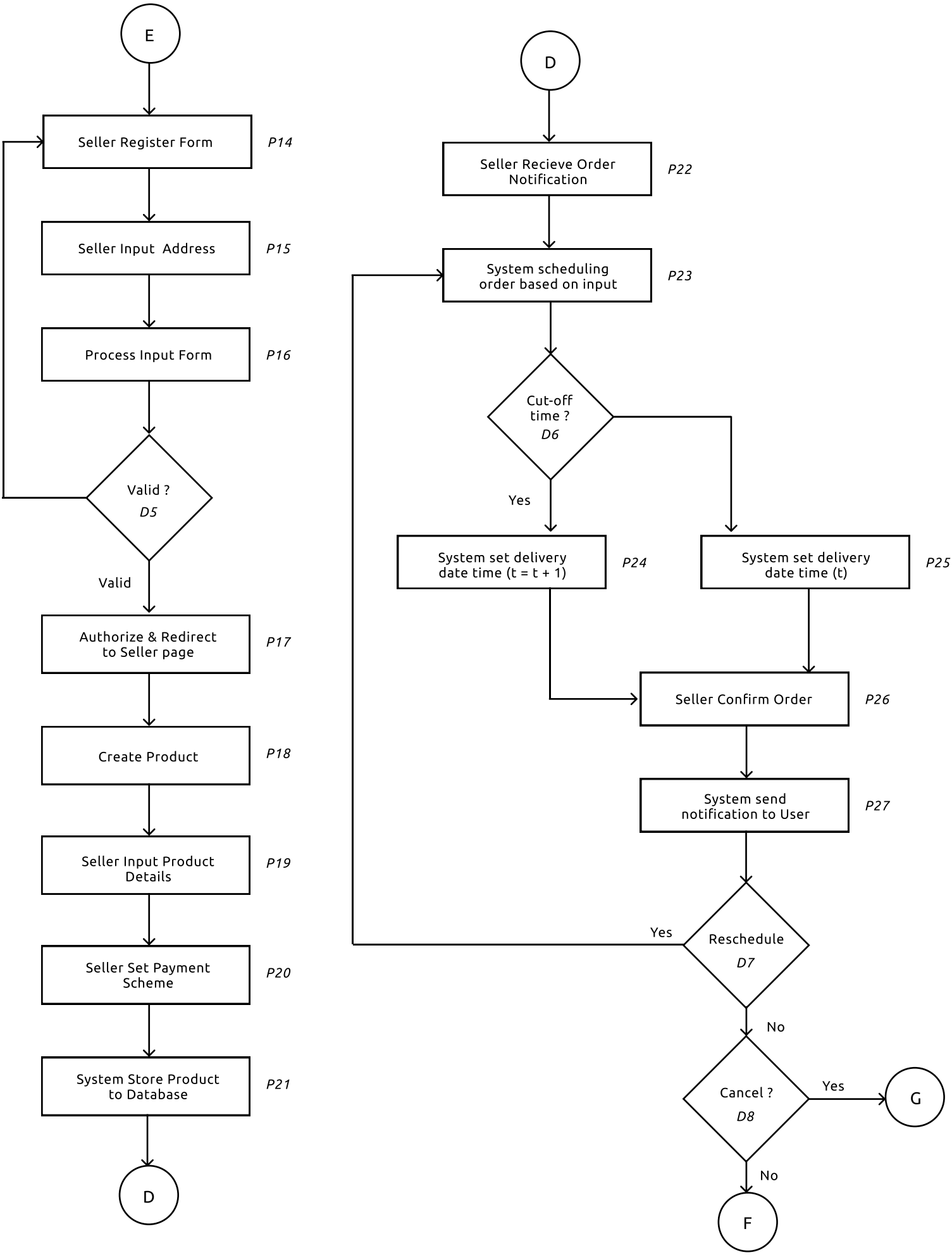
2. Transaction Flow



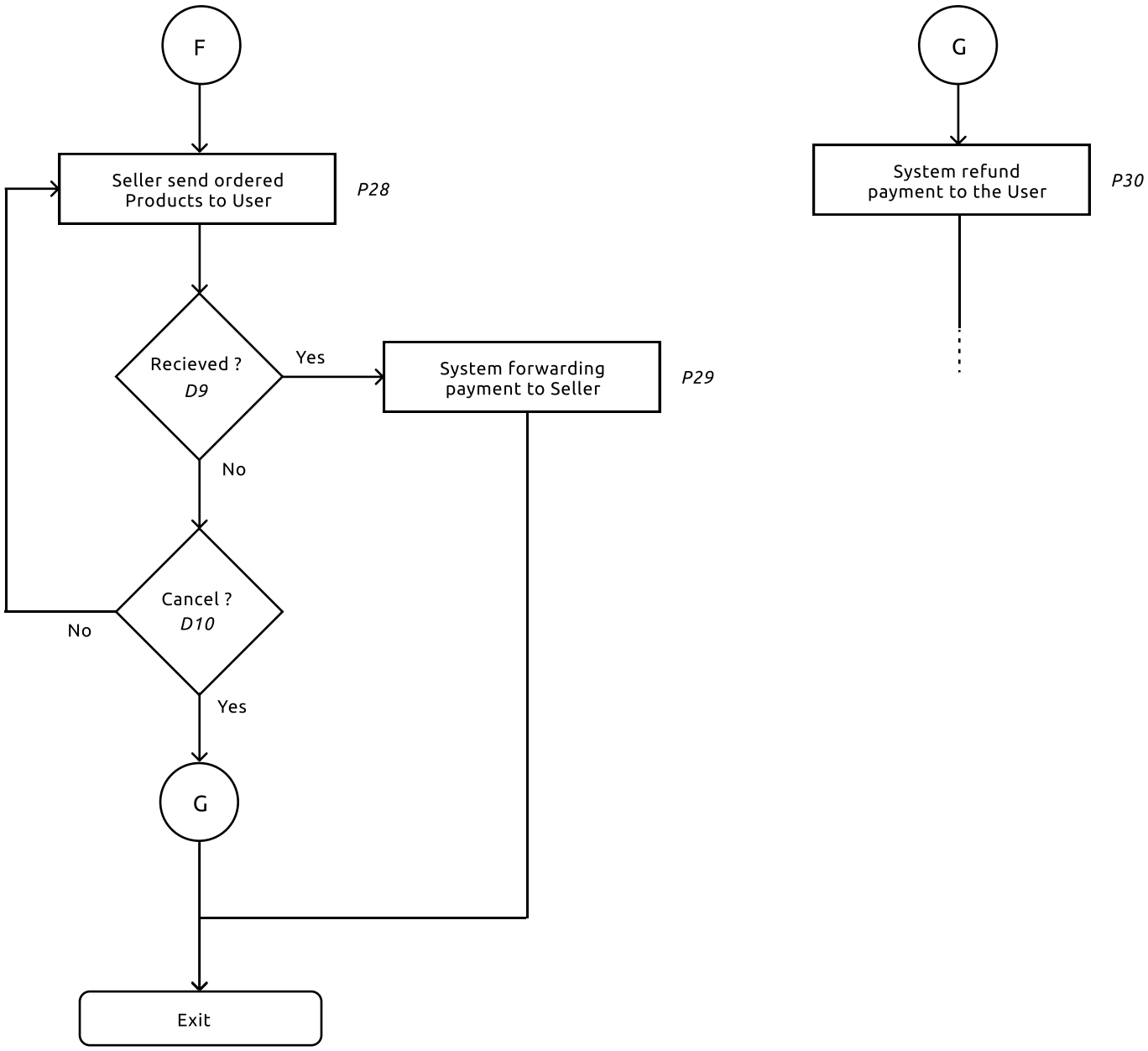
Basic Coding



Basic Coding



Basic Coding



Algorithm

Solutions

Problem Solutions can be found in this Github Repo :
https://github.com/daffaarravi/kulina-backend-challenge

Thank You..