## Requirement

**Scenario:**

Now a days we are seeing how much struggle the farmers are facing to sell their crop in the market for proper price. Farmer must bear the cost of transportation, wait time, negotiations for proper price, etc. Even though farmer sells the product in the market, he/she must pay lot of intermittent charges as commissions to make his way for coming out of market.

**Solution:**

Develop a platform (Android/iOS app) which helps the farmers to sell their product directly to dealers from Farm.

The App should act as a bridge between the Farmer and the Dealer (Crop purchaser). All you need to do is selecting whether you are a Farmer or the Dealer, signing up with details and the payment information. There is no need for the farmer to carry the crop till the market, paying unexpected commissions and wait for proper price from the Dealers. Whenever farmer want to sell the vegetables/fruits at the Farm itself, he just wants to select the type of Crop, quantity available, input the location/address and publish the information to Dealers. Whoever(Dealer) is interested in purchasing will connect with the Farmer, will reach the location, checks the quality of the crop, negotiates the price, weighs the crop and payment will be done to the farmer from the App itself which will be more transparent rather than paying tolls/commissions.

**Features**

**For Farmer: -**

**Sign up/Login:**  
Sign up with basic details. A user can login using Email or Facebook. ***(DP Hint: - Federated Identity)***

**Profile:**  
Farmer can view and edit their profile information.

**Publishing Crop details:**

Add the crop details like type (vegetables/fruits), select the vegetable/fruit from dropdown list, quantity availability and location.

**Bank Account Details:**  
Add bank account details for payment transfer.

**Receipts:**

Once the crop is sold to the dealer for the aggregable price, Farmer should able to see the generated receipt with details.

**For Dealer: -**

**Sign up/Login:**  
Sign up with basic details. A user can login using Email or Facebook. ***(DP Hint: - Federated Identity)***

**Profile:**  
Dealer can view and edit their profile information.

**Subscribe for Crop details:**

Subscribe for the crop details which you want to purchase so that a notification will be received as soon as any Farmer posts the information.

**Bank Account Details:**  
Add bank account details for payment transfer.

**Invoice Generation:**  
After successful purchase, an invoice is generated and sent to Farmer which include includes details like name, quantity, price, total. ***(DP Hint: - You can use CQRS pattern to query multiple data coming from multiple services)***

**Payment:**  
A Dealer can pay Farmers through their debit/credit cards. Use any freely available Payment gateway for payments else you can also use dummy payment gateways of or fake it by just adding the data in the database ***(Hint: - Event Sourcing can be used to do payment transactions/multiple write operations involved).***

**Admin Panel:**

**User Management:**It includes below modules:

Farmer:   
- Edit profile    
- Active/Inactive Farmers

Dealer:   
- Add/Edit Dealer   
- Active/Inactive Dealer   
- View Farmer’s ratings and reviews   
- Export Dealer’s report to excel

**Dealer & Farmer Management:**

Add/Edit car details

Active/Inactive

**Add-On Management:**

Add/Edit Add-On List

Active/Inactive Add-On List

**Report Management:**  
Admin can filter and generate reports based on order number, type, and date.

***(DP Hint: - You can use CQRS pattern to query multiple data coming from multiple services)***

**Advance Report Management:**  
Admin can generate advanced wash reports based on business, users and locations.

***(DP Hint: - You can use CQRS pattern to query multiple data coming from multiple services)***

You will have to break all the functional requirements into services that are automatically registered into the service registry like Eureka, Consul, etc. which can also be discovered by other services using same tools, Eureka, Consul, etc. thereby achieving one more Design Pattern – ***Service Discovery and Registry***

You are also have to put all the App configuration in a highly available decentralized location like github using tools like Spring Cloud Config Server, thereby achieving one more Design Pattern – ***Externalized Configuration***