**On-Demand Car Wash - Basic**

We will have to accomplish the following items in our microservices implementation:

NOTE: Mentor is supposed to help trainees build this App Incrementally Phase by Phase and is free to add functionalities and complexities looking at the level of trainees and their abilities to go beyond the scope of this Case Study.

1. Each microservice exposes a set of [REST](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7baa652949-5bc0-4bc1-9ed6-daaaf5e80dfb%7d&action=edit&uid=%7bAA652949-5BC0-4BC1-9ED6-DAAAF5E80DFB%7d&ListItemId=44&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod)/ JSON endpoints for accessing business capabilities
2. Each microservice implements certain business functions using the [Spring(Boot) framework](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7b6db783b4-14e2-4813-9b7f-b14d1baee132%7d&action=edit&uid=%7b6DB783B4-14E2-4813-9B7F-B14D1BAEE132%7d&ListItemId=55&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod).
3. Each microservice stores its own persistent data using H2/MySQL/[MongoDB](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7b507ce6db-85f3-4f0a-b437-939b89cf965a%7d&action=edit&uid=%7b507CE6DB-85F3-4F0A-B437-939B89CF965A%7d&ListItemId=53&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod) database
4. Each Microservices must implement best practices such as, Exception Handling, Loggers, [Test Cases](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/Shared%20Documents/Java_Full_Stack/JUnit.pptx?d=wc2cf6dc123254f87b0d4f6fca415175f&csf=1&e=U8adCg), Static Code Analysis and build tools.
5. [Microservices](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7b6db783b4-14e2-4813-9b7f-b14d1baee132%7d&action=edit&uid=%7b6DB783B4-14E2-4813-9B7F-B14D1BAEE132%7d&ListItemId=55&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod) are built with Spring Boot, which has an embedded Tomcat server as the HTTP listener.
6. [RabbitMQ](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7b6db783b4-14e2-4813-9b7f-b14d1baee132%7d&action=edit&uid=%7b6DB783B4-14E2-4813-9B7F-B14D1BAEE132%7d&ListItemId=55&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod) is used as an external messaging service. Try finding out where it can fit in your case study.
7. The UI, Website must be implemented using [Angular](https://capgemini.sharepoint.com/:p:/r/sites/COPNA/iTransform/_layouts/15/Doc.aspx?sourcedoc=%7baff37771-4cd8-43df-bc7f-cc1a825cf2b7%7d&action=edit&uid=%7bAFF37771-4CD8-43DF-BC7F-CC1A825CF2B7%7d&ListItemId=50&ListId=%7bAF7717D0-65AD-4DE5-AA01-07A399F026B5%7d&odsp=1&env=prod)

You are also supposed to implement necessary design patterns mentioned in the table below:-

|  |  |  |
| --- | --- | --- |
| **Design Pattern** | **Design Pattern Summary** | **Demos/Examples** |
| Service Registry and Discovery | Clients of a service use either Client-side discovery or Server-side discovery to determine the location of a service instance to which to send requests. |  |
| Externalized Configuration | Move configuration information out of the application deployment package to a centralized location. |  |
| CQRS | Segregate operations that read data from operations that update data by using separate interfaces. |  |
| Event Sourcing | Use an append-only store to record the full series of events that describe actions taken on data in a domain. |  |
| Federated Identity | Delegate authentication to an external identity provider. |  |

**NOTE: - In the features below, there is *“hint”* which tells you the design pattern that you are supposed to use to implement the functionality.**

Tell us your car location and we will come to you. With Green Wash, you can get your car washed wherever you are at your convenience. Whether it's your home, salon, office or any other place, your car wash is just a few taps away. Few moments spend in signup can save your valuable time. All you need to do is just enter your car, address and payment details. A washer will come at your location and give your car a premium wash. The app also contains various packages and add-ons to wipe, clean and dress your entire car from inside out. One of the best parts of the app is, users can even schedule future car wash and give reviews and ratings to washers. Just download the app now and book your first car wash today.

## Client Requirement

Today, with the changing preferences and growing income, people are likely to roam around in sedans and SUV's. With the increasing car consumer market, car wash segment is all set to join the race. Forecasting the expanding markets of the car care services, GreenCarWash company approached Hyperlink InfoSystem to develop an interface that connects customers and service providers to avail the best wash services within the nearest proximity. The main aim of the client is to explore the untapped market opportunities by satisfying increasing consumer demands through professional car wash services. The app is developed on Android and iOS platforms and designed for the USA audience only.

**Features**

**For Customers:-**

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

**Car & Payment Details:**  
Add car details along with its image. Also, add payment details for further transaction process.

**Wash Now:**After filling appropriate details, users can book their first car wash within few taps. All they need to do is select their car and wash package. They can even add a new car and include some extra wash add-ons if they wish to. Users can also guide washers by adding some important notes and information.

**Schedule Later:**  
An app also allows users to schedule car wash for upcoming dates. Just select their preferred date, time, location and package and it's done. The wash request is sent to the washers. If no acknowledgment is received from washer regarding approve/decline of wash request, a random washer is assigned to the user by admin.

**Payment:**  
A user can confirm the booking and pay washers 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).***

**Receipt:**  
After the successful car wash, washer sends payment receipt to the user which includes after wash car photo, package, and extra add-on details. Users are also facilitated to share information on various social media platforms.

**Reviews & Ratings:**  
After the successful car wash, users can share their thoughts and opinions for washers through reviews and ratings.

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

**My Orders:** 

Here users can view order details in below form:

* **Current Orders:**View list of pending, accepted and in-process orders.
* **Past Orders:**View list of completed and cancelled orders.

**For Car Washers:-**

**Login:**  
Login using – Email or Facebook ***(DP Hint: - Federated Identity)***

**Wash Request:**  
Wash Request is sent to the washer along with the user details. The washer can either accept or decline the request. On accepting the request, a washer can navigate through customer's address by Google Maps Integration (You can also use any freely available maps).

**Invoice Generation:**  
After successful car wash service, an invoice is generated and sent to customers which include washed car image, package and add-on details. ***(DP Hint: - You can use CQRS pattern to query multiple data coming from multiple services)***

**Ratings & Reviews:**  
After the successful car wash, washers can share their experience with users through reviews and ratings

**Profile:**  
Washers can view and update their profile information like profile picture and contact information. In this section, user can view the ratings for washers too.

**My Orders:**  
Here washers can view orders in below form:

* **Current Orders:**View pending accepted and in-process order list.
* **Past Orders:**View completed and canceled orders

**Admin Panel:**

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

Customer:   
- Edit profile   
- View ratings   
- Active/Inactive customers

Washer:   
- Add/Edit Washer   
- Active/Inactive Washer   
- View Customer's ratings and reviews   
- Export Washer's report to excel

**Car Management:**

Add/Edit car details

Active/Inactive Car

**Service Plan Management:**

Add/Edit service plan details

Active/Inactive service plans

**Add-On Management:**

Add/Edit Add-On List

Active/Inactive Add-On List

**Promo Code Management:**

Add/Edit Promo Codes

**Order Management:**

Wash Now & Schedule:   
- View order details   
- Assign pending order request to washers   
- Shows the list of pending, accepted, under process, completed and canceled orders

**Report Management:**  
Admin can filter and generate reports based on order number, washer name, type, service, 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, sales, 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***