**Team-3**

Shuvankar Biswas

Rijula Batabyal

Imantika Biswas

Akash Ghanwate



GREEN WASH

On-Demand Car Wash

POC & LLD

**Table of Contents**

[1.0 Document Purpose](#_Toc94636300)

[2.0 Intended Audience](#_Toc94636301)

[3.0 Project Background, Objective(s)](#_Toc94636302)

[4.0 Design Pattern](#_Toc94636305)

[5.0 Solution Steps](#_Toc94636307)

[6.0 Classes/function name](#_Toc94636308)

[7.0 Data model/Tables Diagram](#_Toc94636309)

[8.0 Use Case Diagram](#_Toc94636310)

[9.0 Data Flow Diagram](#_Toc94636314)

***10.0 Microservices Architecture***

[11.0 API Canvas](#_Toc94636311)

**1.0 Document Purpose**

The documents contain a detailed description of the solution architecture of the on-demand Car Wash System.

**2.0 Intended Audience**

|  |  |
| --- | --- |
| Role | Nature of Engagement in the On Demand Car Wash System Architecture |
| Product Owner/SME | Key stakeholder to ensure that the architecture is aligned with business goals. |
| Business Analysts | Business analysts are one of the stakeholders who are informed with the key architectural decisions. |
| Enterprise Architects | To enforce Customer management Platform Architecture is aligned to business goals and architecture, architectural guidelines. |
| Developers | Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with Customer management Microservice |
| End-User | An End- user can check the bookings done, cost for the booking and other package information and book/cancel a Wash. |

**3.0 Project Background & Objectives**

**3.1 Project Background**

On Demand Car Wash System leads to perform Management of Car wash details where one can register themselves and perform various operations related to washing Cars.

**3.2 Project Objectives**

On Demand Car Wash System will perform various operations like instantly booking the car wash service or scheduling it for later.

The user of this system should first register with his/her email ID and their password for any interaction with the system. Once registered and after logging in the user should select the kind of activity, he would like to perform.

**3.3Technologies Used**

* Frontend:- HTML, CSS, JavaScript, Angular
* Backend:- Spring Microservices, Spring Boot , Spring rest
* Database:- MongoDB

**4.0 Design Pattern**

|  |  |  |
| --- | --- | --- |
| **Serial no.** | **Name** | **Description** |
| 1 | Angular | Creating a user interface |
|  |  | (Front-end), and |
|  |  | consuming API services. |
| 2 | Database | For storing, maintaining |
|  |  | and accessing customer, |
|  |  | admin, car washer and |
|  |  | booking details. |
| 3 | API | Using HTTP requests, |
|  |  | we will use the |
|  |  | respective action to |
|  |  | trigger various |
|  |  | operations |
| **4** | RabbitMQ | RabbitMQ is used as an |
|  |  | external messaging |
|  |  | service |
| **5** | Spring(boot) Framework | Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier to create standalone applications |
|  |  |  |

**5.0 Solution Steps**

**5.1 Customer**

**Registering Customer**

1. User will be able to register himself by entering the details like Name, address, Phone, email, and Password.
2. After filling the user credentials the form is validated.
3. If the validation is successful, by clicking the submit button browser directs the request to customer registration API.
4. The call reaches the API gateway.
5. API gateway does the routing and saves the data in the database.
6. Once a user is successfully registered an alert is displayed and the user is redirected to the login page.

**Viewing Wash package and Payment Details**

1. After login, users can book their first car wash within few taps.
2. They can even add a new car and include some extra wash add-ons if they wish to.
3. users can schedule car wash for upcoming dates by selecting their preferred date, time, location, and package.
4. A user can confirm the booking and pay washers through their debit/credit cards.
5. User gets a payment receipt from washer after the successful car wash.
6. After the successful car wash, users can share their thoughts and opinions for washers through reviews and ratings.
7. Users can view and edit their profile information and can view order details like Current orders and Past orders.

**5.2 Car Washer**

1. Washer can login using – Email
2. Wash Request is sent to the washer along with the user details. The washer can either accept or decline the request.
3. On accepting the request, a washer can navigate through customer's address by Google Maps Integration.
4. Washers can view and update their profile information like profile picture and contact information.
5. Washers will be notified in the below scenario: -
   1. Scheduled wash notification before 2 hours.
   2. New wash order.
   3. When users cancel wash request.
   4. On successful Payment by users.
6. Washer and can view order details like Current orders and Past orders.

**5.3 Admin**

1. Admin will be able to edit and add new car washer, add news car details, and payment details.
2. Admin has access to all users and car washer’s details.
3. Admin can Add/Edit service plan details and Active/Inactive it.
4. Admin can Add/Edit Add-On list, Promo codes and can inactive also.
5. Admin can view pending, accepted, under process, completed and cancelled orders.
6. GetAllCars() will let the admin view all the car details.
7. GetCustomerById() will allow Admin to view customer details by ID.

**6.0 Classes/Functions**

|  |  |  |
| --- | --- | --- |
| Serial no. | Class | Description |
| 1 | Model Class | Model for holding the booking schema details for user. |
| 2 | Repository | The Interface in Data Access Layer for the user. |
| 3 | Controller | Controller handles the incoming HTTP requests and send the response back to the caller. |
| 4 | Services | It’s the Business Access Layer holding the Business Logic and meditates the communication between the controller and repository (Data Access) Layer. |
| 5 | Exception Handlers | Exception Handlers handles all the exceptions that which are revealed during runtime. |

**7.0 Database Diagram**

**User Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **ID** | **Varchar(10)** |
| 2 | **Full Name** | **Varchar(50)** |
| 3 | **Email** | **Varchar(50)** |
| 4 | **Token** | **Varchar(100)** |
| 5 | **Enabled** | **Boolean** |
| 6 | **Password** | **Varchar(20)** |

**Wash Packs**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **ID** | **Varchar(20)** |
| 2 (FK) | **Package Name** | **Varchar(50)** |
| 3 | **Cost** | **Numeric** |
| 4 | **Description** | **Varchar(100)** |

**Ratings**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **ID** | **Numeric** |
| 2 | **Name** | **Varchar(50)** |
| 3 | **Comments** | **Varchar(20)** |
| 4 | **Rating** | **Numeric** |

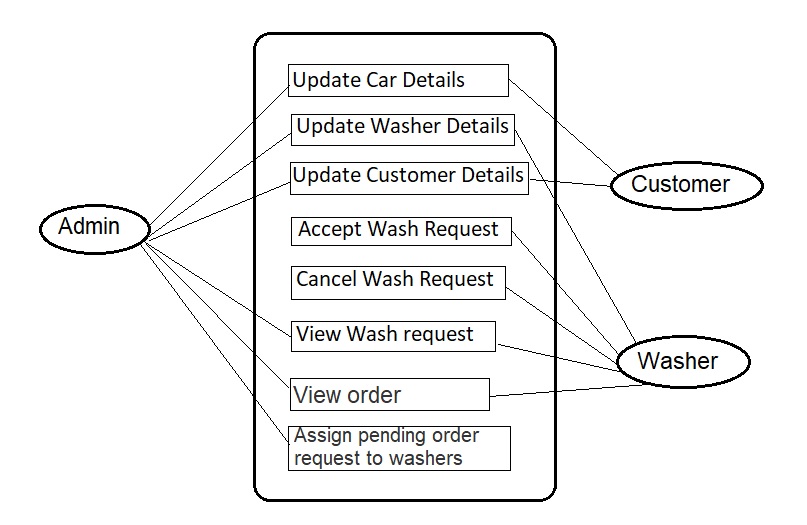
**Car**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **ID** | **Numeric** |
| 2 | **Name** | **Varchar(20)** |
| 3 | **Model** | **Varchar(20)** |

**Order details**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **OrderID** | **Varchar(20)** |
| 2 (FK) | **UserEmailID** | **Varchar(20)** |
| 3 | **WasherName** | **Varchar(20)** |
| 4 | **Washpack** | **Varchar(20)** |
| 5 | **PhoneNo** | **Numeric** |
| 6 | **Area Pincode** | **Numeric** |
| 7 | **Status** | **Varchar(10)** |

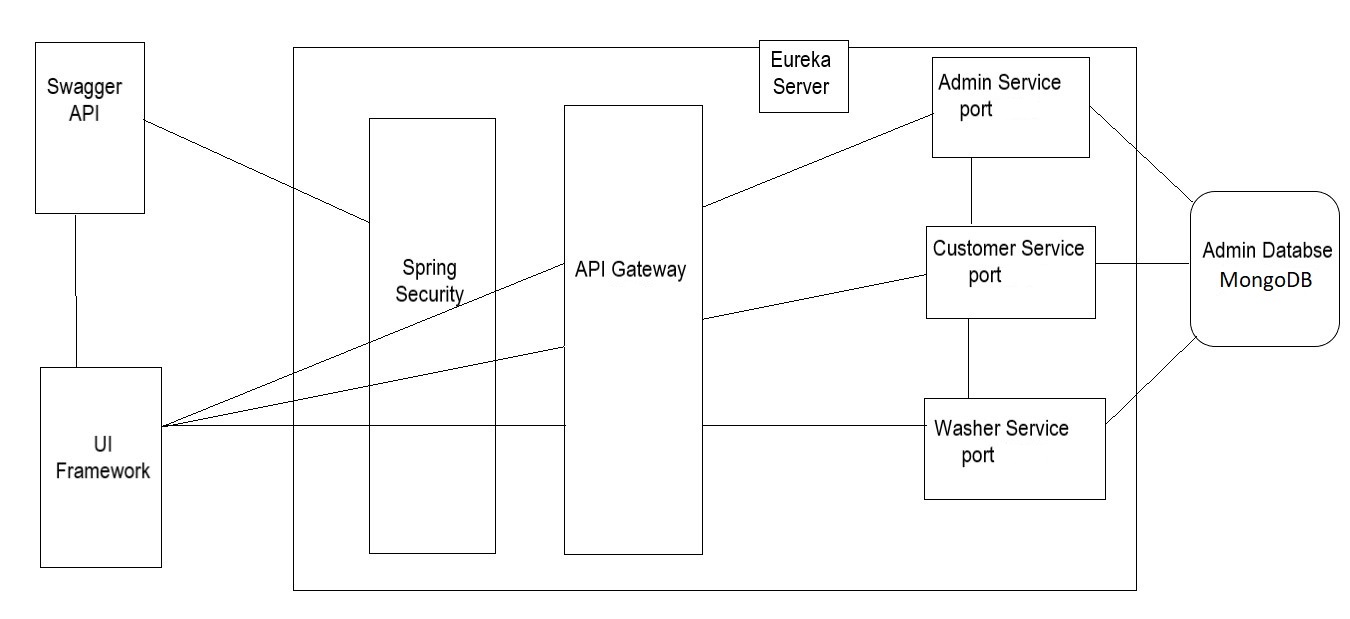
**8.0 Use Case Diagram**

****

Diagram

Description automatically generated with medium confidence**9.0 Microservices Architecture**

**10.0 Data Flow Diagram**

****

**11.0 API Canvas**

**11.1 User**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| User-management | /Customer | POST | To register a customer | No | True |
| User-management | /Customer/Id | GET | To get a customer by Id | Admin | True |
| User- management | /Customer | GET | To get the list of customers | Admin | True |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Booking-management | /booking | POST | To reserve a wash | No | True |
| Booking-management | /Booking/Id | DELETE | To delete a wash | No | True |
| Booking-management | /Booking | PUT | To update date of wash | No | True |
| Booking-management | /Booking/Id | GET | To get the wash  Details by Id | No | True |

**11.2 Booking**