ON DEMAND CAR WASH

Low Level Design (LLD)

Date: 05/06/2022

Prepared by
(Group 1)
Korrai Bhargav
Vutukuri Naga Venkata kasi Amarnadh Kumar
Imam Gouse Peddarumala Shaik
Sujeet Kumar
Jami Mutyalarao
Sudesh K
Adrija Biswas
Priyank Joshi
Neetu Kumari
Durgesh Kumar Jha

Table of Contents

1.0 Document Purpose

This document describes the solution architecture for On-Demand Car Wash

2.0 Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the On-Demand Car wash technical architecture.

Role	Nature of Engagement in WB Classics Portal Technical Architecture
Product Owners/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 On Demand Car Wash Platform Architecture is aligned to business goals and architecture, architectural guidelines.
Solution Architects	To ensure solution design and architecture is aligned to business requirements, architectural guidelines.
Developers	Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with On Demand Car Wash

3.0 Project Background, Objective(s)

3.1 Project Background

On-Demand Car wash leads to perform

- The Customers can register themselves and book a car wash service.
- Service provider can provide car wash service at customer location. Service provider can accept/reject customer's carwash service request
- Admin can perform operations on booking list.

3.2 Project Objective

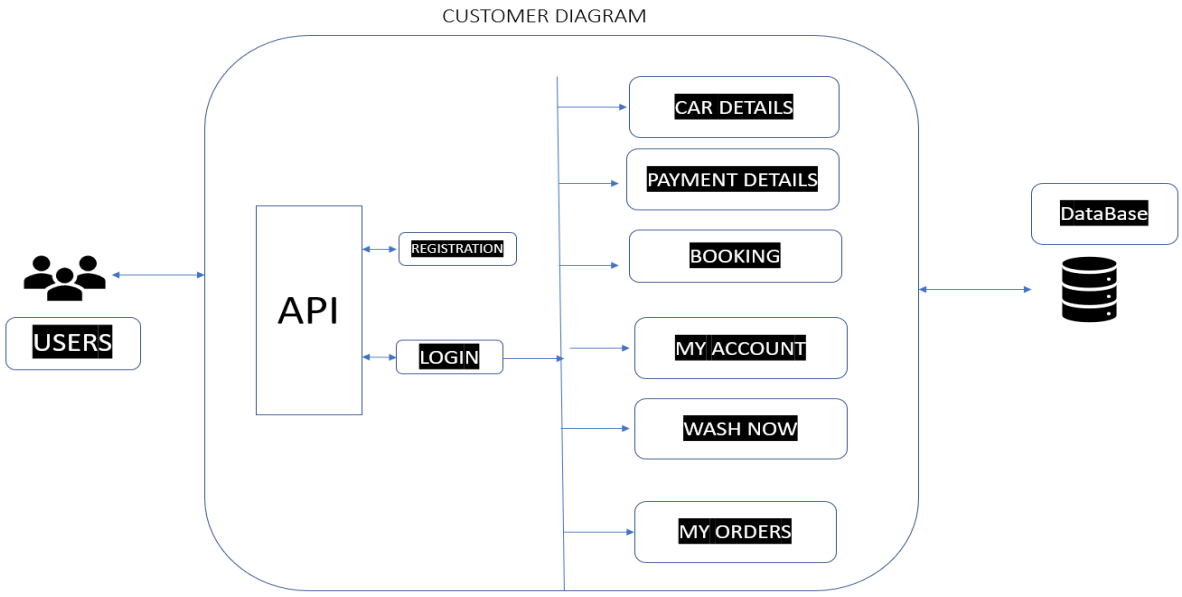
- On-Demand Car wash will perform various operations like listing, creation, updating and deletion of Customer Details.
- Customers can register themselves and then they can perform the operations such as car service.

4.0 Design Pattern

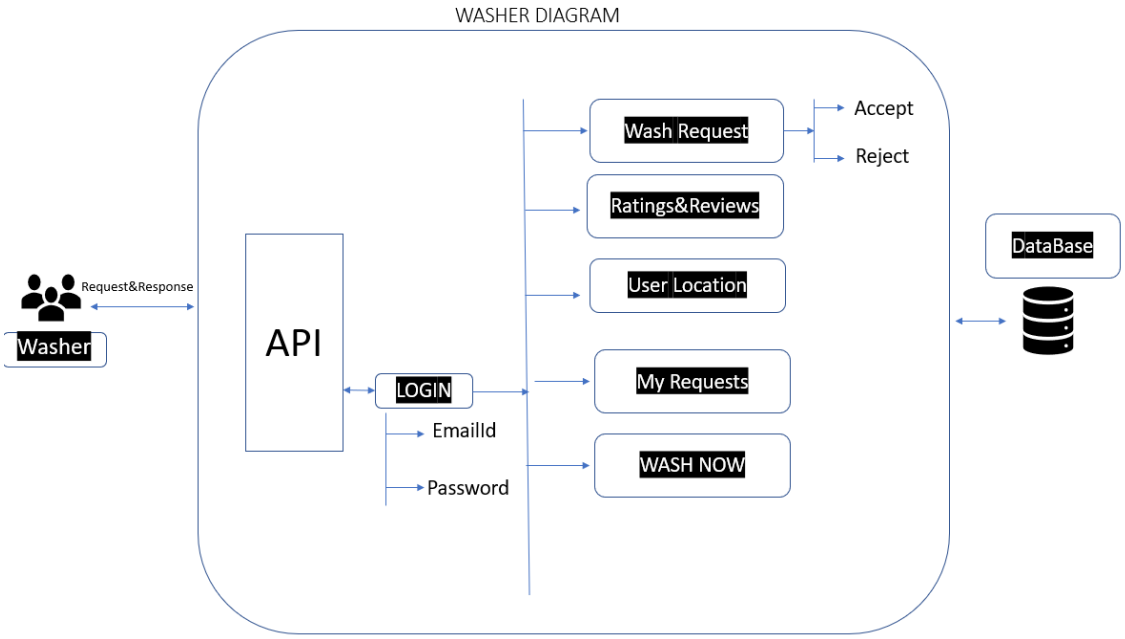
#	Name	Description
1	API	Using HTTP requests, we will use the respective action to trigger various operations

5.0 Solution Diagrams

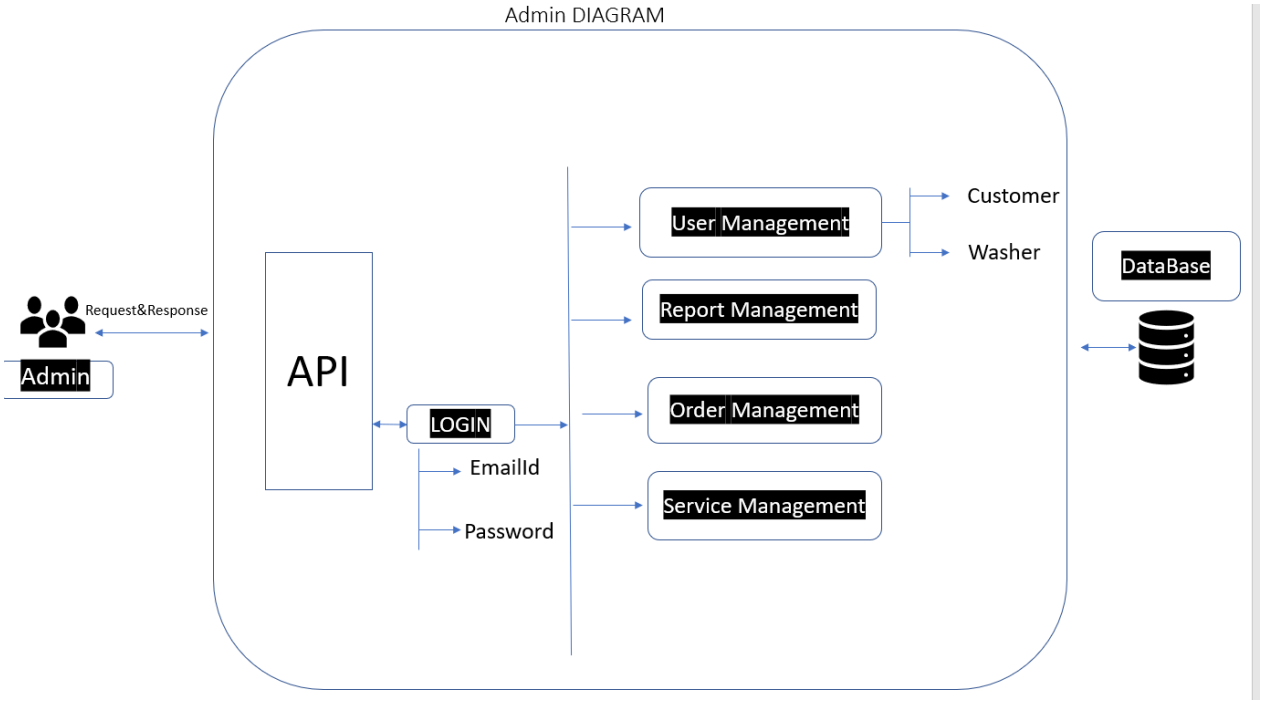
Customer Diagram



Washer Diagram



Admin Diagram



6.0 Solution Steps

6.1 Customer Steps

Customer Registration

1. Customer will enter the required details such as Name, email, phone number, and click submit button browser directs the request to customer registration API
2. call reaches the API gateway
3. API gateway does the routing and forwards the request to, Customer Controller And this register function.
4. The registration will call the validate () function to do the input validation it will have the customerValidator.validateCustomerRegistration as argument to perform the validation
   a. If validation fails, then it will return the error code and error description. with status code
   b. If validation is successful, it will store the data in database Success JSON response and HTTP status code 200 with corresponding success message.

Customer Booking List

1. customer wants to get the booking details. Click on the booking button customer wants to see the details. browser directs the request to booking List API
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to booking details Handler.
4. Booking details () method is used to fetch the data from database.
5. It sends the response body with HTTP Success response code to book detail Handler.
6. List booking details returns JSON Response
7. Success JSON response and HTTP status code 200 with corresponding success message.

Booking History

1. customer wants to view the booking details. Customers click on booking history browser directs the request to car details API
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to fetch booking details.
4. It sends the response body with HTTP Success response code to BookingHistoryHandler.
5. BookingHistoryHandler returns JSON Response
6. Success JSON response and status HTTP code 200 with corresponding success message.

Customer

1. customer enters the username and password to login into the account.
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to LoginCustomerHandler. this handle function calls the Login ()
4. Login () is responsible for fetching user credentials data from the database.
5. It sends response body with HTTP Success response code to LoginCustomerHandler.
6. LoginCustomerHandler returns JSON Response
7. Success JSON response and HTTP status code 200 with corresponding success message.

Customer Service list

1. User wants to get the service details. Browser directs the request to Wash Services List API
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to Wash Service List. This handle function will call the Service List ()
4. Service List () is responsible for fetching the data from database.
5. It sends the response body with HTTP Success response code to CustomerServiceHandler.
6. CustomerServiceHandler returns JSON Response
7. Success JSON response and HTTP status code 200 with corresponding success message.

6.2 Washer Steps

Washer login

1. Washer will enter the required details such as email id and password and click submit button browser directs the request to Washer login API.
2. API gateway does the routing and forwards the request to the washer login Handler. And this handle function will call the WasherLogin ()
3. WasherLogin () will call the dovalidate() function to do the input validation it will have the Washer Validator. It performs the validations:
    c. If validation fails, then it will return the error code and error description. with status code
    d. If validation is successful, then the handler will call the login Washer Service. Login Washer () is responsible for fetching the data from database

4.It sends a response body with HTTP Success response to login Washer Handler.

5. Login Washer Handler returns JSON Response
6. Success JSON response and HTTP status code 200 with corresponding success message.

Particular Customer Listing

1. Washer wants to view the booking details. Washer click on booking history browser directs the request to car details API
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to fetch booking details.
4. The washer decides to accept or reject the customer booking
5. It sends the response body with HTTP Success response code to BookingHistoryHandler.
6. BookingHistoryHandler returns JSON Response
7. Success JSON response and status HTTP code 200 with corresponding success message.

Admin

1. Admin wants to view the details of customer and washer.
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to admin Handler. Then this handle function calls the Admin ()
4. Admin () will call input validation.
5. If validation fails, then it will return the error code and error description with status code.
6. If validation is successful, Admin can login into the account.
7. AdminLoginHandler returns JSON Response
8. Success JSON response and status HTTP code 200 with corresponding success message.

AdminBooking History

1. Admin wants to view the booking details. Admin click on booking history browser directs the request to car details API
2. Call reaches the API gateway.
3. API gateway does the routing and forwards the request to fetch booking details.
4. It sends the response body with HTTP Success response code to AdminBookingHistoryHandler.
5. BookingHistoryHandler returns JSON Response
6. Success JSON response and status HTTP code 200 with corresponding success message.

7.0 Classes/function

#	Class	Description
1	Customer.cs	Model holds the customers schema details
2	CustomerController.cs	The controller to handle the Workflow of All Operations which calls the CustomerService class
3	CustomerService.cs	It contains the core business logic for the All Operations Which calls the database context class for Updating Data in Database
4	MyDatabaseContext.cs	It is also used to configure domain classes.
5	Washer.cs	Model holds the Washers schema details
6	WasherController.cs	The controller to handle the Workflow of All Operations which calls the Washer Service class
7	WasherService.cs	It contains the core business logic for the All Operations Which calls the database context class for Database Updation
8.	Admin.cs	Model holds the Admin schema details
9.	AdminController.cs	The controller to handle the Workflow of All Operations which calls the AdminService class
10.	CarServiceFeedback	The controller to handle feedback of car service

8.0 Data model/Table

Customer Table:

	T_CUSTOMERS	DATA_TYPES
PK	Email Id	Varchar (25)
	Name	VARCHAR (25)
	password	VARCHAR (25)

Washer Table (Service Provider)

	T_WASHER	DATA_TYPES
PK	Email Id	Varchar(25)
	Name	VARCHAR(25)
	password	VARCHAR(25)

Car Details Table

	T_CAR DETAILS	DATA_TYPES
Pk	Email Id	VARCHAR(25)
	Name	VARCHAR (25)
	Contact Number	VARCHAR (25)
	Car Model	VARCHAR (25)
	Car Type	VARCHAR(25)
	Service Type	VARCHAR(25)
	Preferred Time	VARCHAR (25)
	message	VARCHAR (25)
	address	VARCHAR (25)
	subscription	VARCHAR (25)

Admin Table:

	T_ADMIN	DATA_TYPES
PK	Id	Int
	Email Id	Varchar (25)
	Name	VARCHAR (25)
	password	VARCHAR (25)

CarService Feedback Table:

	T_carservice_Feedback	DATA_TYPES
PK	Feedback _Id	Int
	Name	Varchar(25)
	Rating_value	Decimal
	Feedback_comments	VARCHAR(25)

provider Feedback Table:

	T_provider_Feedback	DATA_TYPES
PK	Feedback _Id	Int
	Name	Varchar(25)
	Rating_value	Decimal
	Feedback_comments	VARCHAR(25)