

ABSTRACT

We present the design and Implementation of a laundry management system (LMS) used in a laundry establishment.

Laundry firms are usually faced with difficulties in keeping detailed records of customers clothing; this little problem as seen to most laundry firms is highly discouraging as customers are filled with disappointments, arising from issues such as customer clothes mix-ups and untimely retrieval of clothes.

The aim of this application is to determine the number of clothes collected, in relation to their owners, as this also helps the users fix a date for the collection of their clothes. Also customer's information is secured, as a specific id is allocated per registration to avoid contrasting information.

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We would like to express our gratitude towards our parents & member of “CDAC-ACTS” for their kind co-operation and encouragement which help us in completion of this project. We would like to express our special gratitude and thanks to industry people for giving us such attention and time. Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

INTRODUCTION

Laundry services system is a system that manages services for consumers, especially customers who have signed up as members. With this system a good cleaning service, it will ensure the availability of needed services to consumers. Due to the many cleaning services system as manual that affects waste or inefficiency of time, data inaccuracies, errors repeated so the writer tries to computerized valet service process in the hope of providing data or information fast, precise, accurate and efficient.

A significant part of the operation of any laundry firm involves the acquisition, management and timely retrieval of great volumes of information. This information typically involves; customer personal information and clothing records history, user information, price of delivery and retrieval period, users scheduling as regards customers details and dealings in service rendered, also our products package waiting list. All of this information must be managed in an efficient and cost wise fashion so that the organization resources may be effectively utilized.

The goal of laundry management system is to automate the management of the laundry firm making it more efficient and error free. It aims at standardizing data, consolidating data ensuring data integrity and reducing inconsistencies, through the use of highly computerized process that is stress free, reliable and quick through the use of Java computer programming language and SQL database application to both the users and the staff in charge of the registration and laundry management processes. HTML would be at the front-end and provide the graphical user interface that relates with the user, while the SQL database will be at the back-end to handle the data storage process.

The project titled “**(Dhobhi-Ghaat) Online Laundry Management System**” is designed using Spring Boot which was developed using the eclipse IDE in java language. The front end was handled using Angular-6.

The project contains four main modules.

- Customer.
- Order.
- Address.
- Service.

Product Overview and Summary

Purpose:

This project is to connect with people online and provide services related to laundry. Using this system the client can post an order for a particular laundry service, view the status of his order, view the price list of the particular services.

This document gives the brief summary of the system of Laundry Management. It will be helpful for the developer to actually get the outline of the system.

Scope:

The Dhobhi-Ghaat is a software application to assist a user to post a request online without actually going to the laundry shop.

The system provides benefit to the customer to post an order according to his convenience of day and time.

The system provides the user friendly environment for laundry management to the customer.

Feasibility Study

Feasibility is determination of whether a projects worth doing or not. Before actually recommending the new system it is important to investigate if it is feasible to develop the new system.

Before developing and implementing a system we have sure that our system is feasible in the following ways:

➤ **Technical Feasibility:**

In the type of feasibility study, the system analyst has to check whether it is possible or not to develop the requested system with availability of manpower, software, hardware, etc...The system which we run in Linux as well as windows platform and hence are suitable for the end-user. The system is technically feasible because it does not require too much manpower and runs with the basic available equipment.

➤ **Operational Feasibility:**

In this type of feasibility study the operation implementation of the system is considered. Checking is done regarding whether it is feasible for the user department to use the software or will there be any inertial resistance from the users. Thus the proposed system is said to be operationally feasible only if the end users are able to understand the system clearly and correctly and can use the system with ease and with the minimum training.

We need to train our staff so that system will be handled efficiently. As the system developed is very user-friendly and easy to operate for any person with minimum computer knowledge of computer is also able to handle our system. It is also easy to operate due to the user-friendly interface developed using Java.

➤ **Economical Feasibility:**

In this type of feasibility study, the benefits of the system to the organization are considered by taking into consideration the cost-benefit analysis. The basic software, which is required for the implementation of the system, is Java which is easily available. Also with the basic training user can use this software thus reducing the training cost to the organization. Thus, using this system is feasible for the organization and learning Java and the proposed system is economically feasible for the organization. As our system goes online we will have a lot of customers adding to our publicity. This in turn will increase our profit.

Overall Description

Product Features

The main feature of this system is the customer can post laundry orders and view the status of his order as well as the total cost of his order. The user must be a registered before he orders any service

User Classes

There are two types of user to this system one is the **Admin** which can

1. Get all customers.
2. Remove any customer by his respective ID.
3. Get list of all orders.
4. Get order details by order status.
5. Get order details by order ID.
6. Get order details by Customer ID.
7. Update order by status.
8. Get address by customer ID.
9. Update service by customer ID.

The second is the **Customer** of this system can –

1. Login.
2. Register.
3. Create order.
4. Add address.
5. Remove address.
6. Track order.
7. See the price list of particular laundry service.

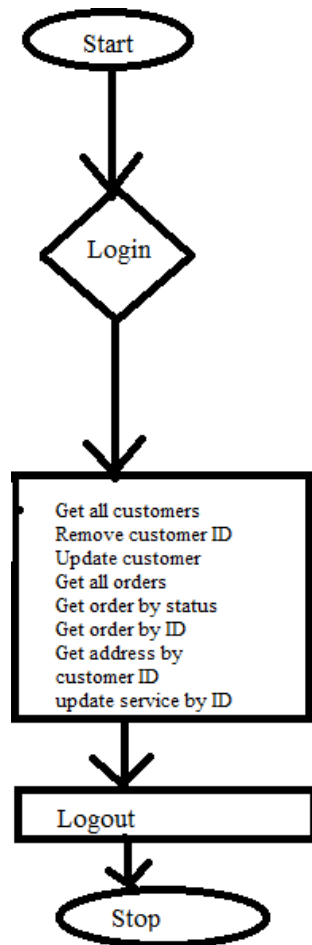
General Constraints

The “Dhobhi-Ghaat” should run on all Internet Browser and all processors which supports the Internet Browser.

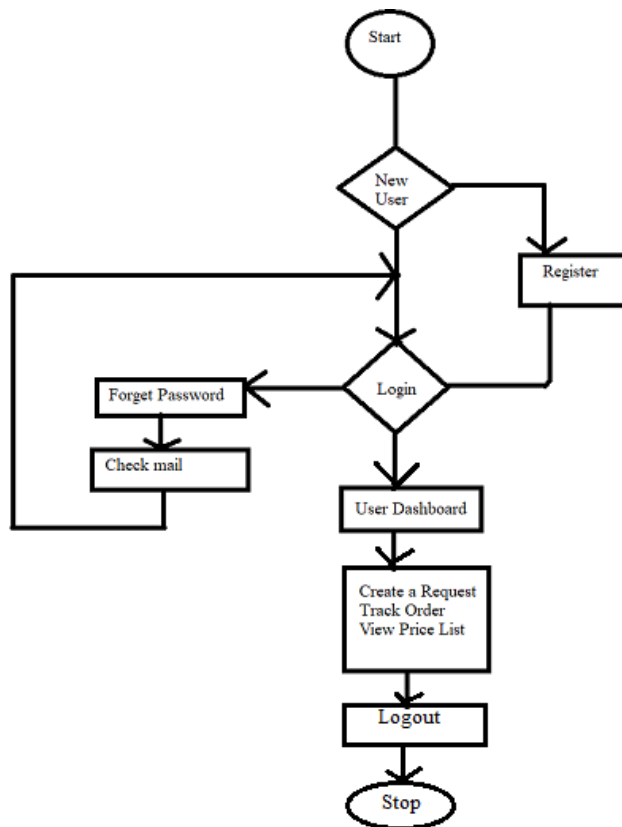
Requirements

Functional Requirements

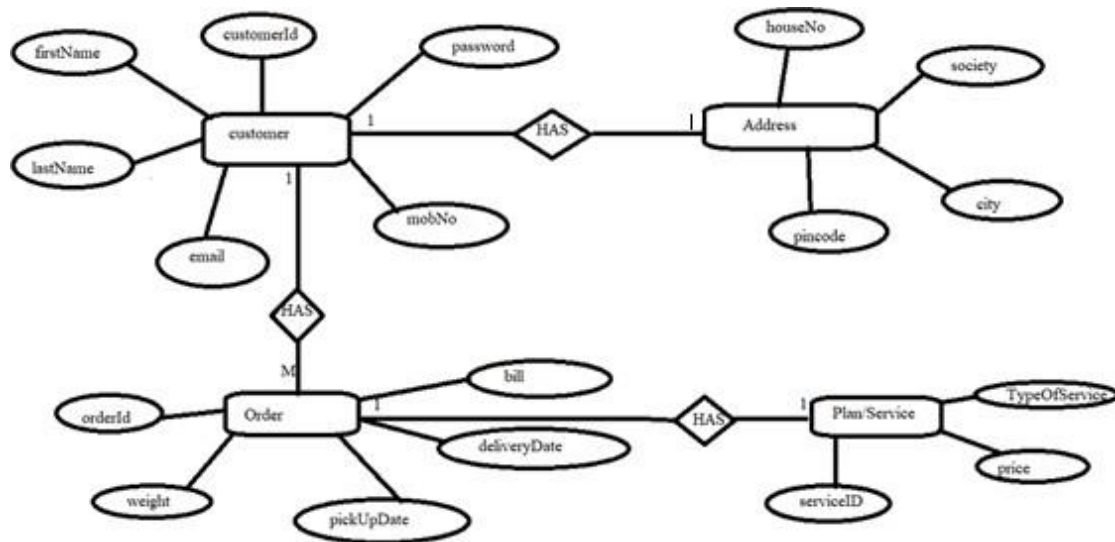
ADMIN



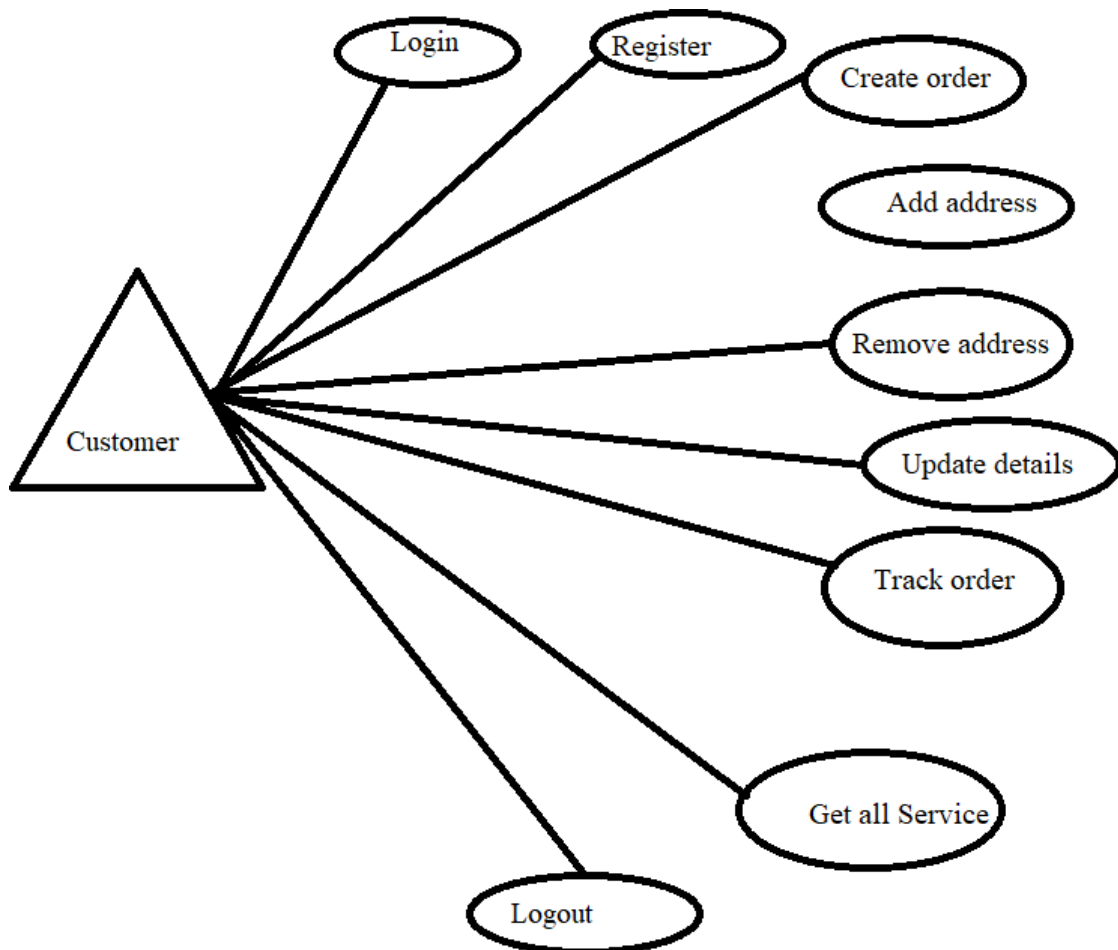
USER



Entity Relation Diagram



CUSTOMER TASK



API: Customer Registration:

There is an entry interface that is intended to facilitate the Customer to register to posts an order and view and track the order.

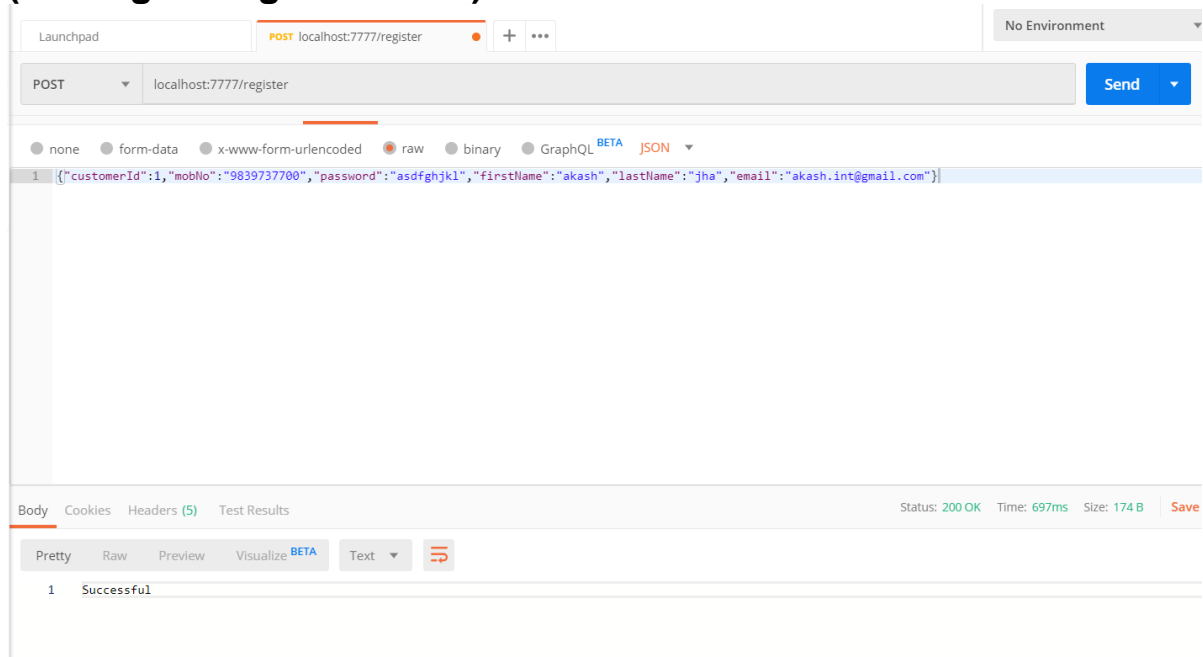
Scenario 1: Mainline Sequence:

1. User: Clicks on Register link.
2. System: User redirect to registration form containing fields for information like first name, last name, email, phone number, etc.
3. User: Enters the required information in the fields, after that presses submit.
4. System: System calls confirm action. And control is transferred to to the method containing @RequestMapping (value = /register) which inserts the data provided by the user in the corresponding table in the database.

(The Mapping to corresponding method)

```
@RequestMapping(value = "/register", method = RequestMethod.POST)  
public ResponseEntity<?> m2(@RequestBody Customer customer) {  
  
    if (service.register(customer)) {  
        return new ResponseEntity<String>("Successful", HttpStatus.OK);  
    }  
    return new ResponseEntity<String>("Email or Mobile No already exist", HttpStatus.OK);  
}
```

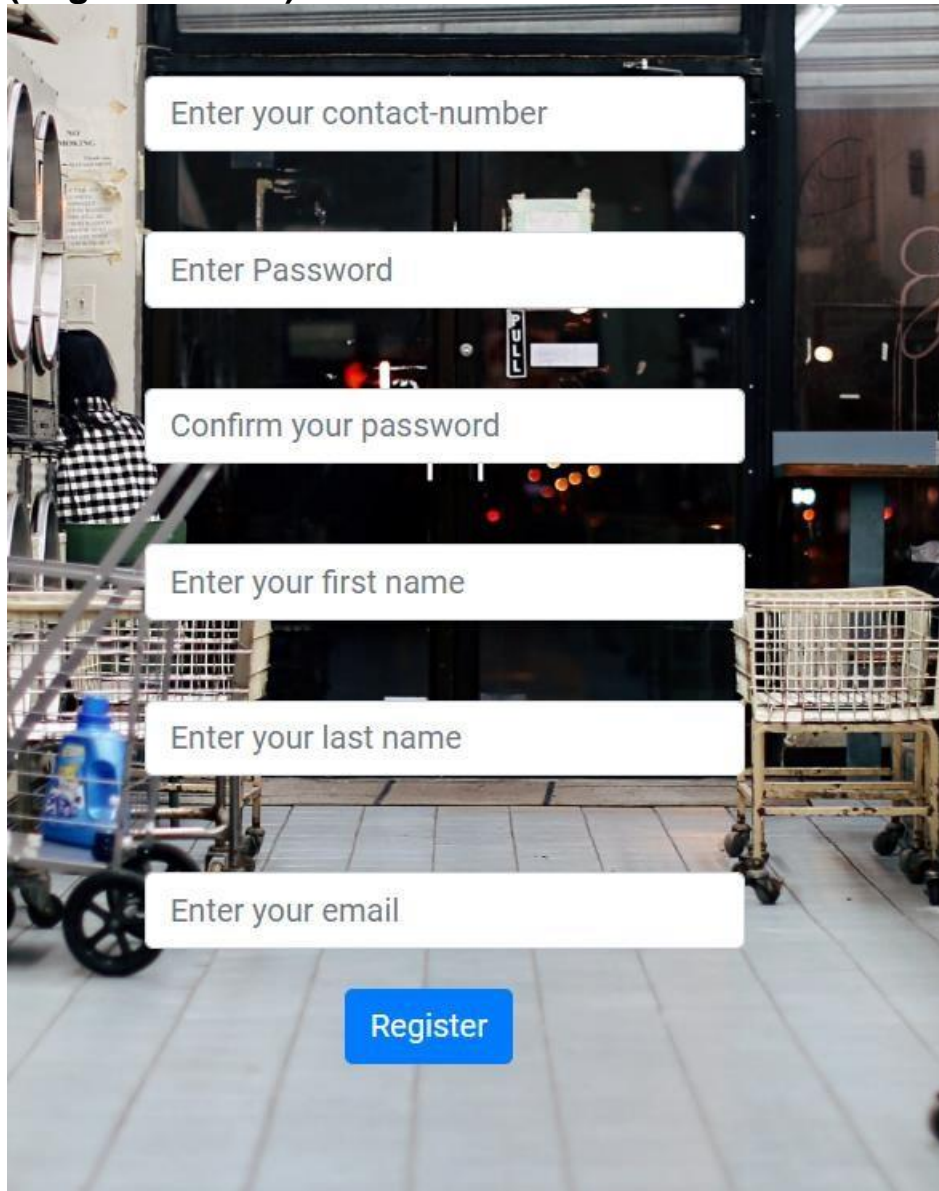
(Testing through Postman)



(Registered customer in database)

```
mysql> select * from customers;
+-----+-----+-----+-----+-----+-----+
| customer_id | email                | first_name | last_name | mob_no  | password |
+-----+-----+-----+-----+-----+-----+
| 1           | akash.int@gmail.com  | akash     | jha      | 9839737700 | asdfghjkl |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(Registration UI)

A registration form is overlaid on a background image of a store interior. The form consists of six white input fields with rounded corners, stacked vertically. Each field contains a placeholder text: 'Enter your contact-number', 'Enter Password', 'Confirm your password', 'Enter your first name', 'Enter your last name', and 'Enter your email'. Below the input fields is a blue rectangular button with the word 'Register' in white text. The background shows a person in a checkered shirt, shopping carts, and store shelves.

Enter your contact-number

Enter Password

Confirm your password

Enter your first name

Enter your last name

Enter your email

Register

API-Customer Login

1. The customer clicks on the login link.
2. The system checks for the login credentials in database, if valid it redirects it to user dashboard else pops up invalid credentials.

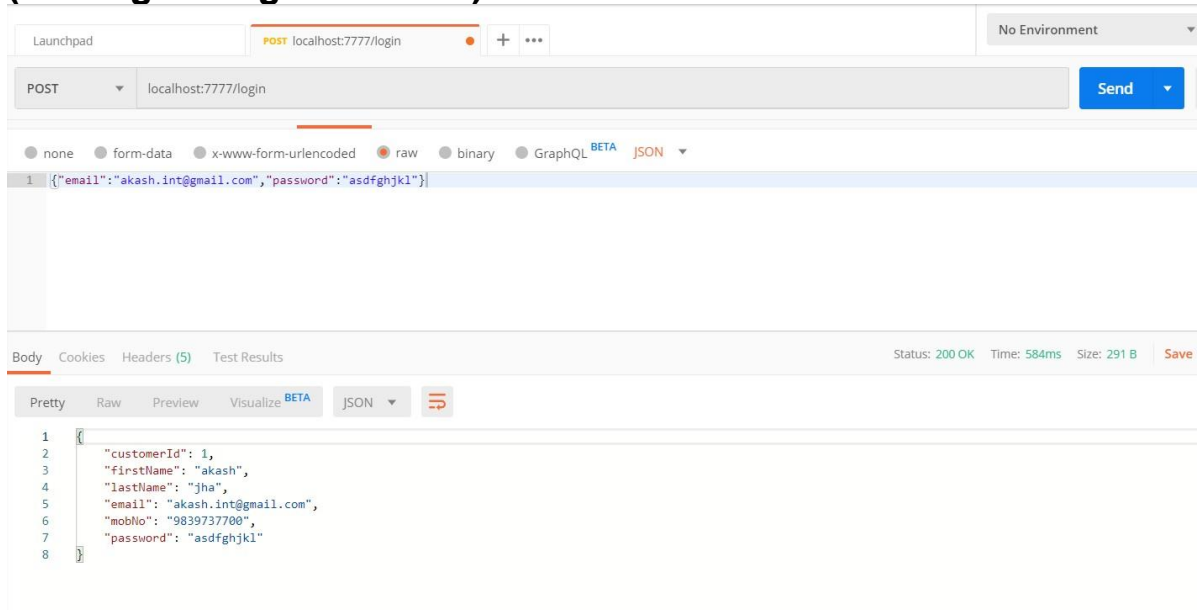
(The Mapping to corresponding method)

```
@RequestMapping(value = "/login", method = RequestMethod.POST)
public ResponseEntity<?> m1(@RequestBody Customer customer) {

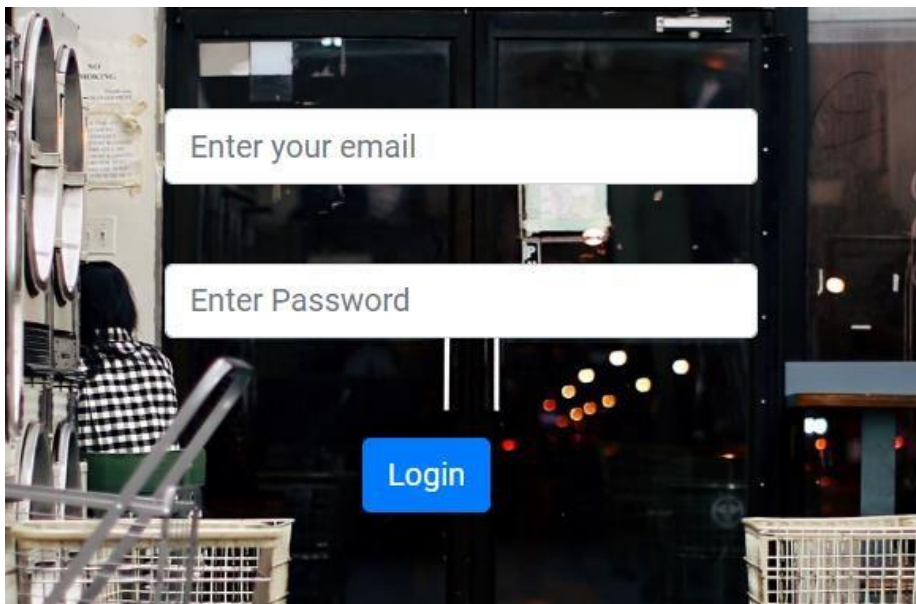
    Customer temp = service.login(customer);

    if (temp != null) {
        return new ResponseEntity<Customer>(temp, HttpStatus.OK);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials", HttpStatus.OK);
}
```

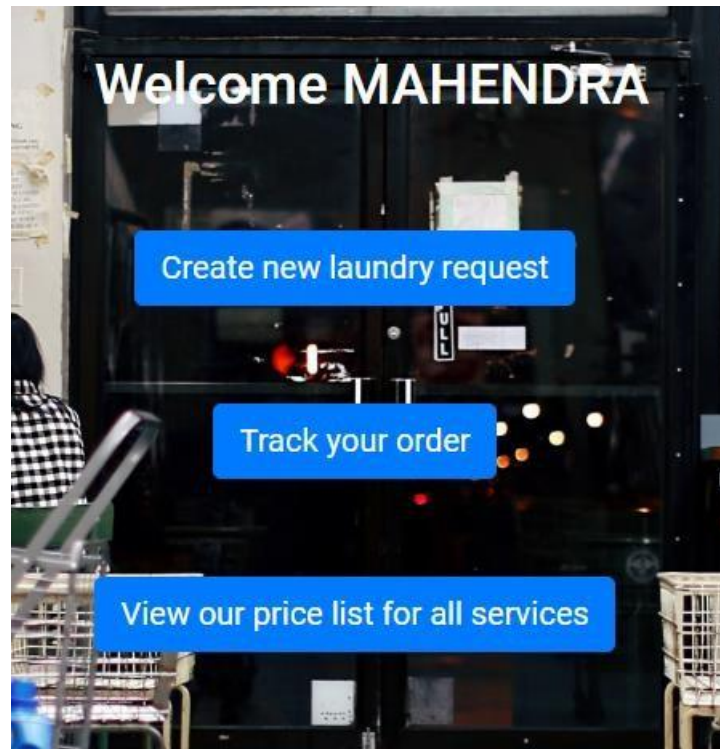
(Testing through Postman)



(Login UI)



Customer Dashboard



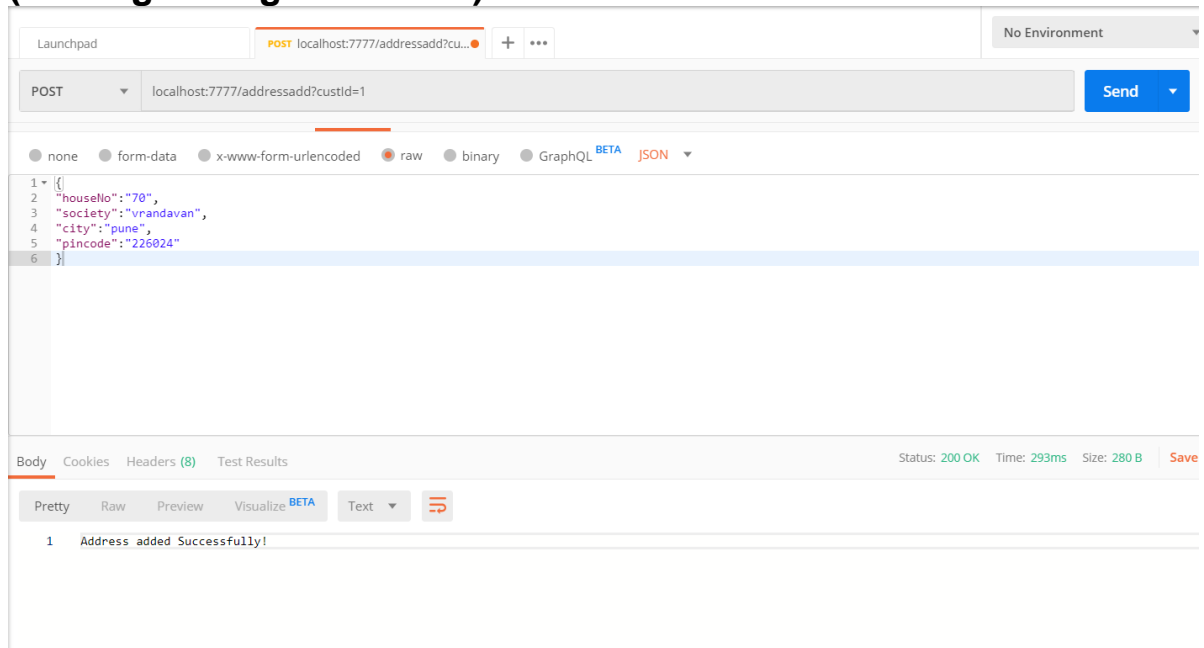
API-Add Address:

Customer can add address in his profile using the following method.

(The Mapping to corresponding method)

```
@RequestMapping(value = "/addressadd", method = RequestMethod.POST)  
public ResponseEntity<?> m1(@RequestBody Address address,@RequestParam Integer custId) {  
    if (service.addAddress(address,custId)) {  
        return new ResponseEntity<String>("Address added Successfully!", HttpStatus.OK);  
    }  
    return new ResponseEntity<String>("Address addition failed!", HttpStatus.OK);  
}
```


(Testing through Postman)



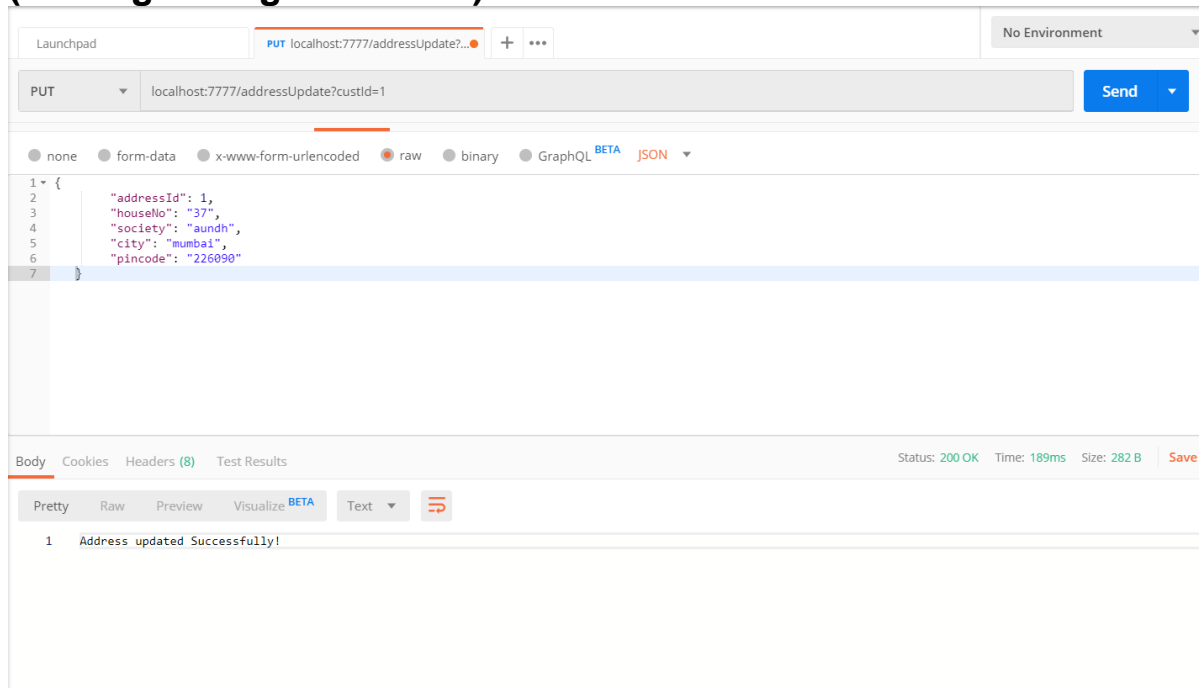
API-Address Update

The customer can update his address with the following method

(The Mapping to corresponding method)

```
@RequestMapping(value = "/addressUpdate", method = RequestMethod.PUT)
public ResponseEntity<?> m4(@RequestBody Address address) {
    if (service.updateAddress(address)) {
        return new ResponseEntity<String>("Address updated Successfully!", HttpStatus.OK);
    }
    return new ResponseEntity<String>("Address update Failed!", HttpStatus.OK);
}
```

(Testing through Postman)

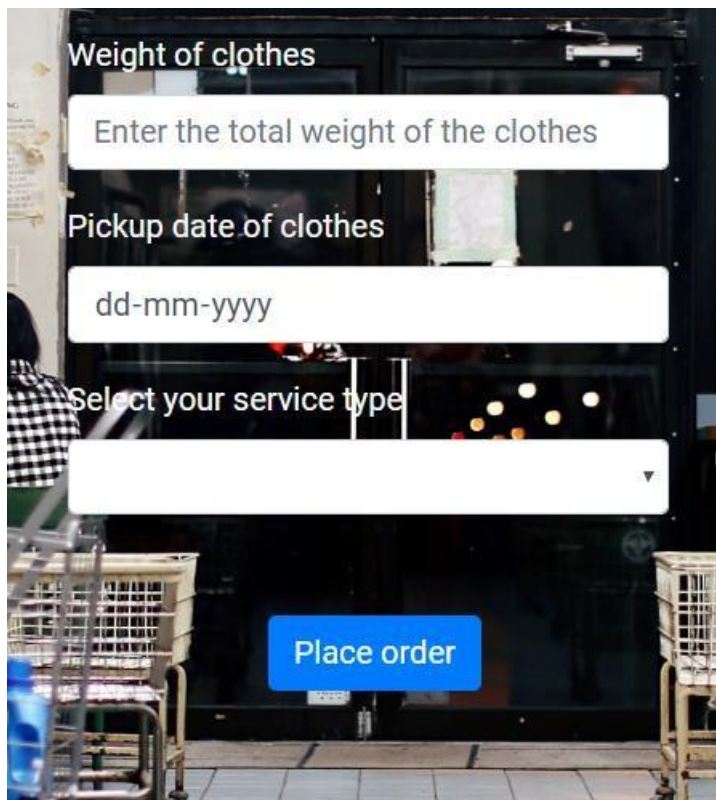


(Updated address in database)

```
mysql> select * from cust_address;
+-----+-----+-----+-----+-----+-----+
| address_id | city   | house_no | pincode | society | cust_id |
+-----+-----+-----+-----+-----+-----+
|          1 | mumbai | 37       | 226090  | aundh   |        1 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

API-Create Order

The customer can post an order for a particular laundry service.



Weight of clothes

Enter the total weight of the clothes

Pickup date of clothes

dd-mm-yyyy

Select your service type

Place order

(The Mapping to corresponding method)

```
@RequestMapping(value= "/createorder/{id}",method=RequestMethod.POST)
public ResponseEntity<> createOrder(@RequestBody Order order,@PathVariable Integer id,@RequestParam
{
    try
    {
        if(service.createOrder(order,id,serviceType,status))
        {
            return new ResponseEntity<String>("Order Successfully added",HttpStatus.OK);
        }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }

    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```




API-Get all services

This API allows users to get the list of the services offered by the application.

(The Mapping to corresponding method)

```
@RequestMapping(value= "/getallservice",method =RequestMethod.GET)
public ResponseEntity<?> getallService()
{
    List<ServiceType> services =null;
    try
    {
        services=service.getAllService();
        return new ResponseEntity<List<ServiceType>>(services,HttpStatus.OK);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

(Service Types UI)

		
IRON	WASH	DRYCLEAN
₹ 70	₹ 50	₹ 100

ADMIN TASK



API-Get All Order

This API allows the admin to get a list of all current orders.

(The Mapping to corresponding method)

```
@RequestMapping(value= "/getallorder",method =RequestMethod.GET)
public ResponseEntity<?> getOrder()
{
    List<Order> orders=null;
    try
    {
        orders=service.getAllOrder();
        return new ResponseEntity<List<Order>>(orders,HttpStatus.OK);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

API-Get Order By Customer ID

This API allows admin to get all the orders under one customer ID.

(The Mapping to corresponding method)

```
@RequestMapping(value= "/getorderbycustid",method =RequestMethod.GET)
public ResponseEntity<?> getOrderByCustomerId(@RequestParam int id)
{
    List<Order> order=null;
    try
    {
        order=service.getOrderByCustomerId(id);
        return new ResponseEntity<List<Order>>(order,HttpStatus.OK);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

API-Get Order By Order ID

This API allows admin to get order details by order id.

(The Mapping to corresponding method)

```
@RequestMapping(value= "getorderbyid/{id}",method =RequestMethod.GET)
public ResponseEntity<?> getOrderById(@PathVariable int id)
{
    Order order=null;
    try
    {
        order=service.getOrderById(id);
        return new ResponseEntity<Order>(order,HttpStatus.OK);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }

    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

API-Get Order By Order Status

This API allows admin to get the status of the order.

(The Mapping to corresponding method)

```
@RequestMapping(value= "/getorderbystatus",method =RequestMethod.GET)
public ResponseEntity<?> getOrderByStatus(@RequestParam String status)
{
    List<Order> orders=null;
    try
    {
        orders=service.getOrderByStatus(OrderStatus.valueOf(status.toUpperCase()));
        return new ResponseEntity<List<Order>>(orders,HttpStatus.OK);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

API-List of Customers

This API allows admin to get the list of customers.

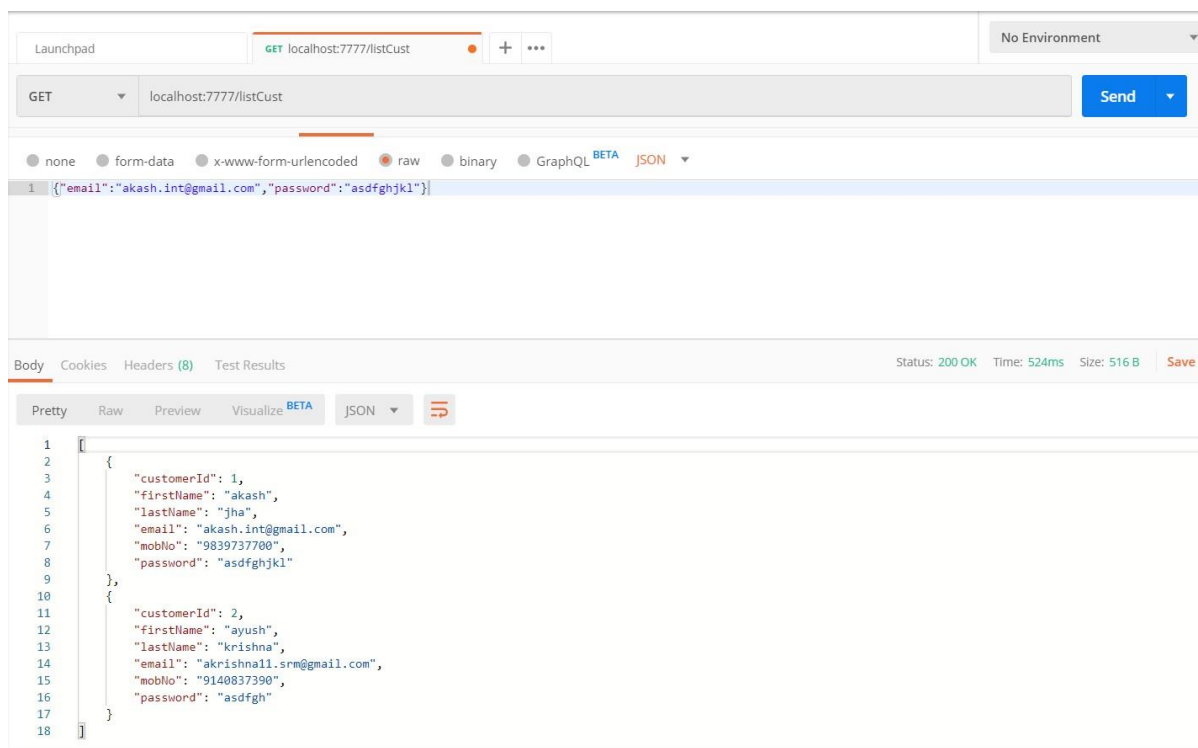
(The Mapping to corresponding method)

```
@RequestMapping(value = "/listCust", method = RequestMethod.GET)
public ResponseEntity<?> m3() {

    List<Customer> allCustomers = service.getAllCustomers();

    if (allCustomers.size()==0) {
        return new ResponseEntity<Void>(HttpStatus.NO_CONTENT);
    }
    return new ResponseEntity<List<Customer>>(allCustomers,HttpStatus.OK);
}
```

(Testing through Postman)



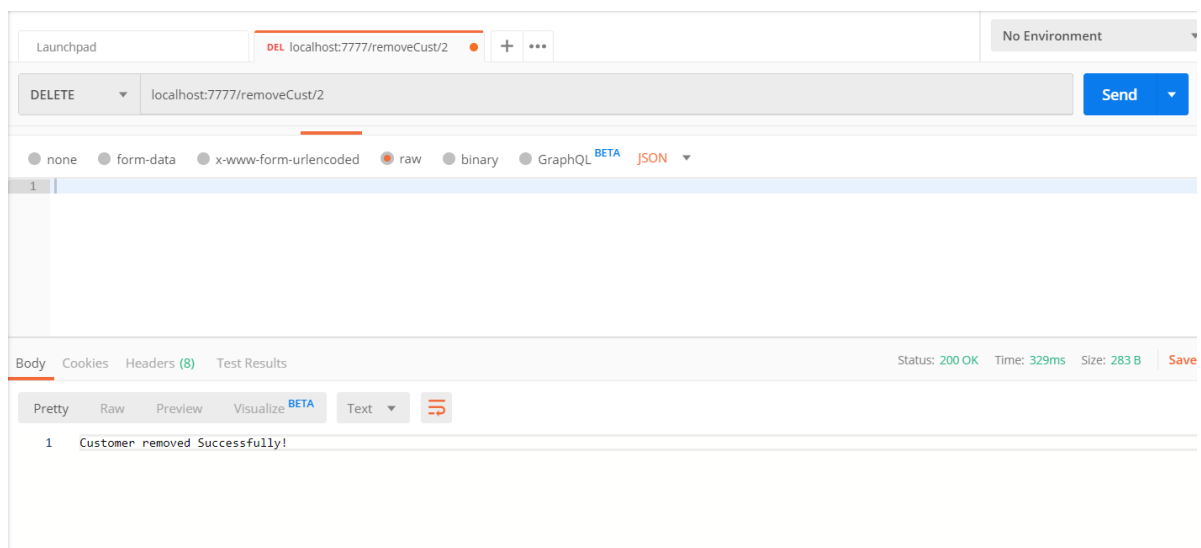
API-Remove a customer by a customer ID

This API allows admin to remove a customer by customer ID.

(The Mapping to corresponding method)

```
@RequestMapping(value = "/removeCust/{custId}", method = RequestMethod.DELETE)
public ResponseEntity<?> m3(@PathVariable Integer custId) {
    if (service.removeCustomer(custId)) {
        return new ResponseEntity<String>("Customer removed Successfully!", HttpStatus.OK);
    }
    return new ResponseEntity<String>("Customer removal failed!", HttpStatus.OK);
}
```

(Testing through Postman)



API-Remove a customer by a customer ID

The admin can remove a customer by his respective ID using this API

(The Mapping to corresponding method)

```
@RequestMapping(value= "/updateOrderByStatus",method =RequestMethod.PUT)
public ResponseEntity<?> updateOrderStatus(@RequestBody Order order)
{
    try
    {
        if(service.updateOrderByStatus(order))
        {
            return new ResponseEntity<String>("Successfully updated",HttpStatus.OK);
        }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

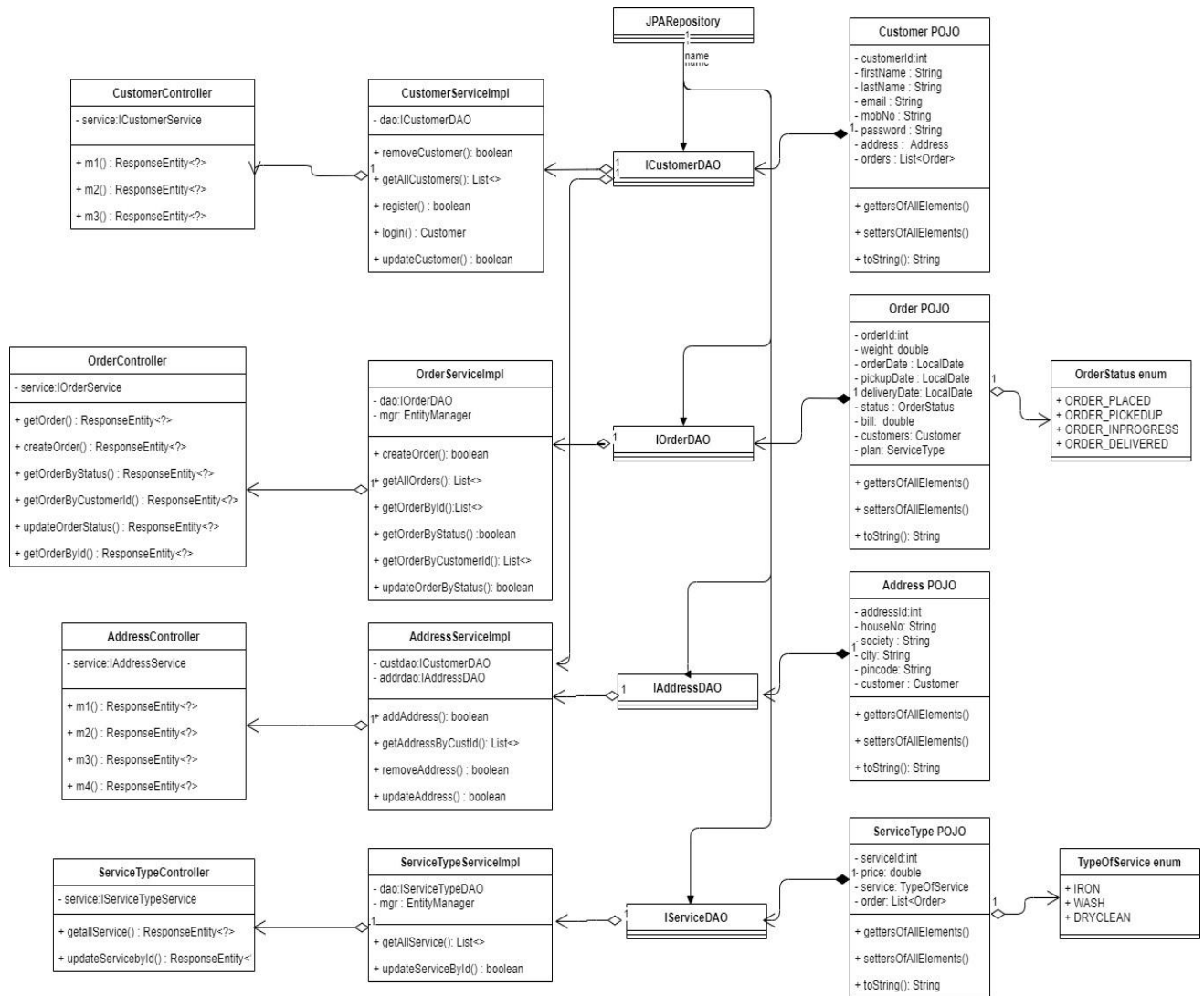
API-Update service by customer ID

The admin can update service of a customer by ID

(The Mapping to corresponding method)

```
@RequestMapping(value= "/updateservice/{id}",method =RequestMethod.PUT)
public ResponseEntity<?> updateServicebyId(@PathVariable int id,@RequestParam double price)
{
    try
    {
        if(service.updateServiceById(id, price))
        {
            return new ResponseEntity<String>("Updated Successfully",HttpStatus.OK);
        }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    return new ResponseEntity<String>("Authentication Failed: Invalid Credentials",HttpStatus.OK);
}
```

Class Diagram



Test Report

The report of the testing is given here under.

Project Name :- Dhobhi-Ghaat

Sr. No	Test Case Title	Description	Expected Outcome	Error Message	Result
1	Login Page-Admin	If User Id=Admin ID, Password= Admin Password	If Validated allow for Admin Home Page If not redirect to same page	Username and password required /Invalid Credentials	Passed
2	Login Page –User	If User Id=User ID, Password= User Password	If Validated allow for User Dashboard If not redirect to same page	Username and password required /Invalid Credentials	Passed
3	User Dashboard Displayed	Homepage display for every successful log in.	Home Page Displayed	No Error	Passed
4	Create New Order	Only Customer can access this to enter new order details	New order for laundry entered	Pickup date/weight Is required	Passed
5	Track order	Customer can see status of the current order	Order Details	No Error	Passed
7	Service List	Customer can see list of Services.	List of services	No Error	Passed
8	Get All orders	Admin see all the list of Current orders.	List of orders	Invalid credentials	Passed
9	Get order by status	Admin can get the list of order by status	List of orders	Invalid Credentials	Passed
10	Get order by ID	Admin can get the list of order by ID	List of orders	Invalid Credentials	Passed
10	Update Order by status	Admin can update the order Status of an order	Order Details	Invalid Credentials	Passed
11	Add address	Customer can address	Customer details	Address addition failed.	Passed
12	Get Address by ID	Admin can see the address of a particular customer.	Customer Address	CORS error	Passed
13	Update Address	Customer can update the address	Customer Address	Address update failed	Passed
14	Update Service By ID	Admin can update the service by customer ID	Updated Successfully Message	Invalid Credentials	Passed
15	Log out	User / Admin can logout by using this link	Successfully logout message	No Error	Passed

Future Scope

- Feedback data can be analyzed to figure which areas we need to improve and which areas are good currently.
- We can add various payment methods apart from cash on delivery only.
- We can include maps in order to provide navigation tracking of orders.

References

- <http://www.roseindia.net/jsp/jsp.htm>
- <http://java.sun.com/javaee/5/docs/tutorial/doc/bnafd.html>
- <http://www.exforsys.com/tutorials/jsp.html>
- <http://www.jmarshall.com>