Name: Pooja Shashikant Chavan

Project Name: Insurance Management Platform.

Introduction

Insurance Management Platform is developed to manage Insurance policy-related activity by a web application, where an admin is a user, can check the list of the insurance client, can add a client, add a policy, and claims etc. The user will be a customer who can view the client details, policy and can apply for a policy listed.

This is a Spring Boot project using the Spring Initializr and it includes necessary dependencies like Spring Boot DevTools, Spring web, Spring DataJPA(to use hibernate), MySQL(to connect SQL). Here, used postman for send data from front end and using spring boot performs operations on the table.
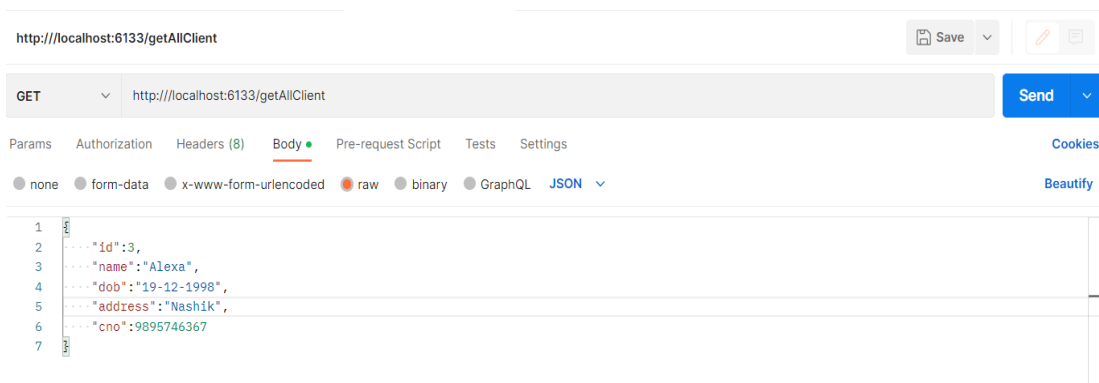
How to run the application

Step 1: Import the project in eclipse framework

Step 2: Add Database details and user details in application.properties file
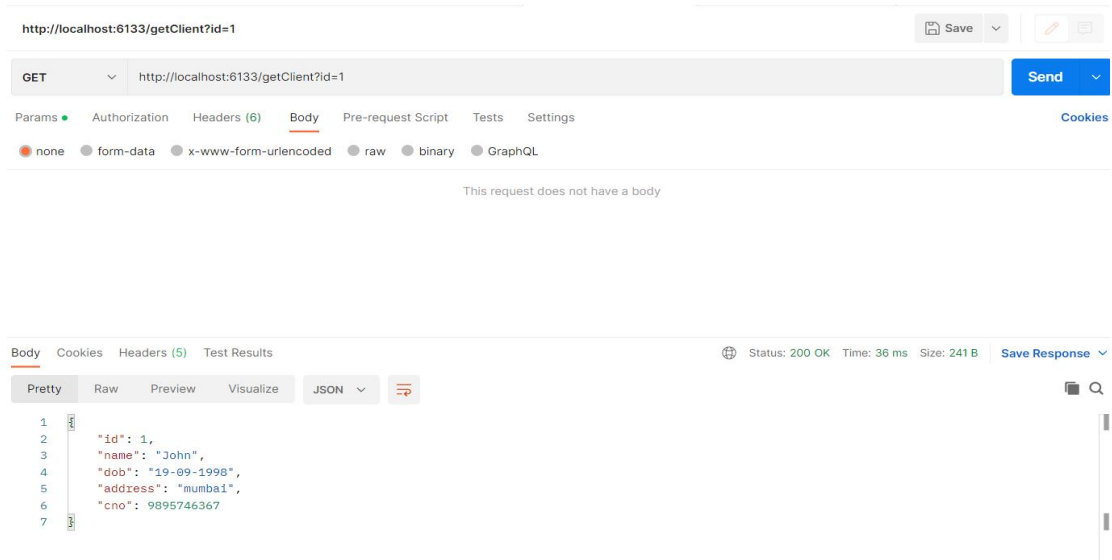
Step 3: Open Postman and create new request

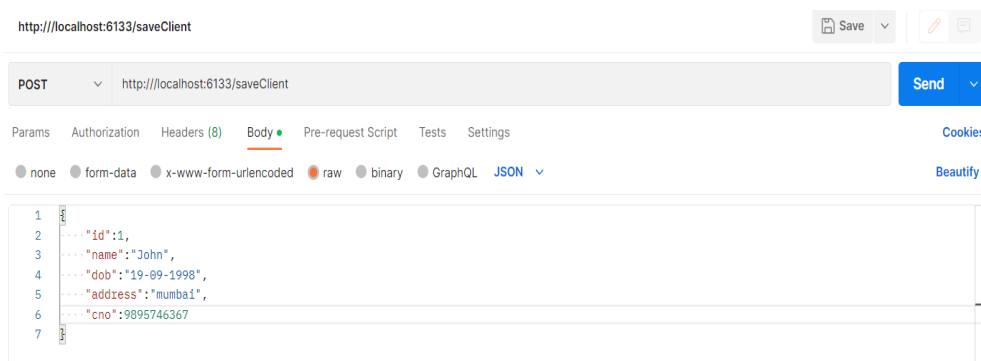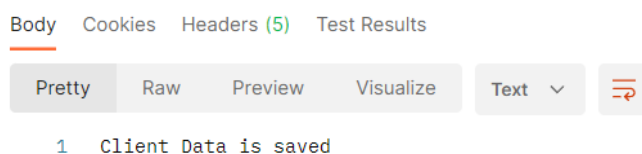Select **GET** fetch all details of client and write url- http://localhost:6133/getAllClient

It will return all data of client.

Step 4: Select **GET** fetch details of client by id and

write url- http://localhost:6133/getClient?id=1

It will return client detail of id 1.

```
http://localhost:6133/getClient?id=1                    Save  ∨      🖉  ▭

GET   ∨    http://localhost:6133/getClient?id=1                    Send  ∨

Params ●   Authorization   Headers (6)   Body   Pre-request Script   Tests   Settings          Cookies

● none  ○ form-data  ○ x-www-form-urlencoded  ○ raw  ○ binary  ○ GraphQL

                        This request does not have a body


Body   Cookies   Headers (5)   Test Results          ⊕ Status: 200 OK  Time: 36 ms  Size: 241 B   Save Response ∨

Pretty   Raw   Preview   Visualize   JSON ∨   ⇆                                    ▣ Q

1  {
2      "id": 1,
3      "name": "John",
4      "dob": "19-09-1998",
5      "address": "mumbai",
6      "cno": 9895746367
7  }
```

Step 5: Select **POST** to save details of client and

write url- http://localhost:6133/saveClient

```
http:///localhost:6133/saveClient                    Save  ∨      🖉  ▭

POST  ∨    http:///localhost:6133/saveClient                    Send  ∨

Params   Authorization   Headers (8)   Body ●   Pre-request Script   Tests   Settings          Cookies

○ none  ○ form-data  ○ x-www-form-urlencoded  ● raw  ○ binary  ○ GraphQL   JSON ∨        Beautify

1  {
2    "id":1,
3    "name":"John",
4    "dob":"19-09-1998",
5    "address":"mumbai",
6    "cno":9895746367
7  }
```

It will save client details.

```
Body   Cookies   Headers (5)   Test Results

Pretty   Raw   Preview   Visualize   Text ∨   ⇆

1   Client Data is saved
```

Clients are saved into SQL Database.

id	address	cno	dob	name
1	mumbai	9895746367	19-09-1998	John
2	Pune	9895746367	19-09-1998	Doe
3	Nashik	9895746367	19-12-1998	Alexa
NULL	NULL	NULL	NULL	NULL

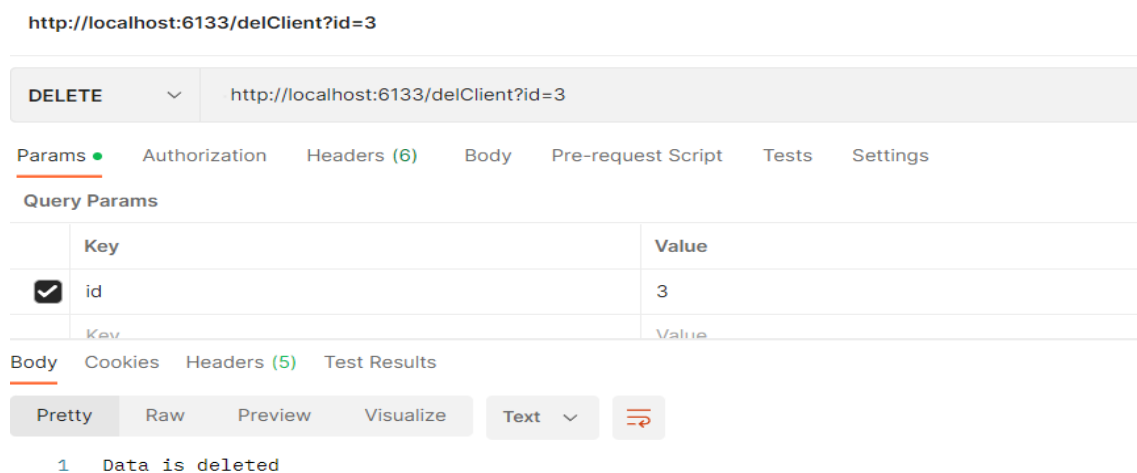Step 6: Select **PUT** to update details of client and

write url- http://localhost:6133/updateClient

http:///localhost:6133/updateClient

| PUT | ⌄ | http:///localhost:6133/updateClient |

Params   Authorization   Headers (8)   Body •   Pre-request Script   Tests   Settings

⚪ none   ⚪ form-data   ⚪ x-www-form-urlencoded   🔘 raw   ⚪ binary   ⚪ GraphQL   **JSON** ⌄

```
1  {
2      "id":3,
3      "name":"Alexa",
4      "dob":"19-12-1998",
5      "address":"Mumbai",
6      "cno":9895746367
7  }
```
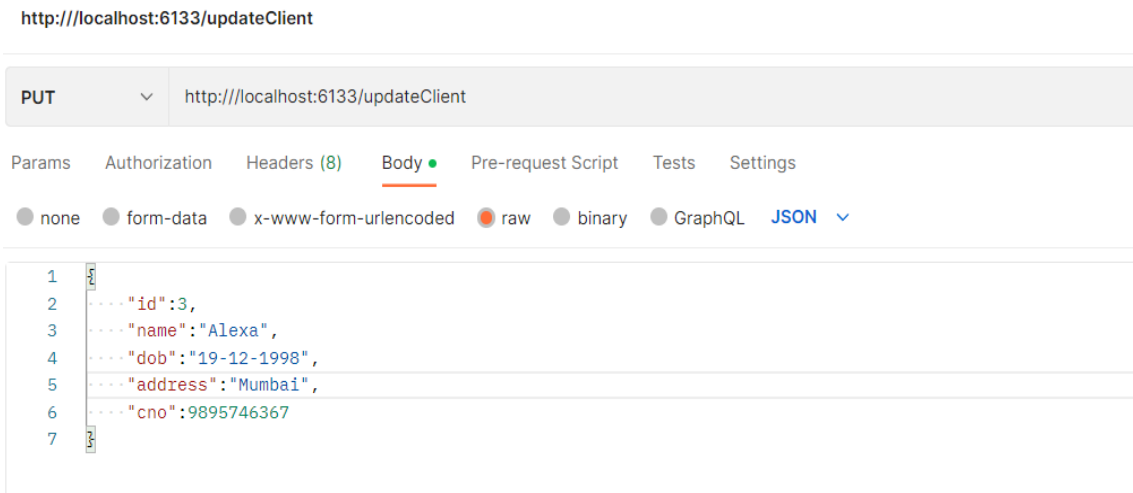
It will be update the client details and display the messa

Body   Cookies   Headers (5)   Test Results

Pretty   Raw   Preview   Visualize

Data is updated

Step 7: Select **DELETE** to delete details of client and

write url- http://localhost:6133/delClient?id=3

http://localhost:6133/delClient?id=3

| DELETE | ⌄ | http://localhost:6133/delClient?id=3 |

Params •   Authorization   Headers (6)   Body   Pre-request Script   Tests   Settings

Query Params

	Key	Value
☑	id	3
	Key	Value

Body   Cookies   Headers (5)   Test Results

Pretty   Raw   Preview   Visualize   Text ⌄   ⇥

```
1   Data is deleted
```

‹

Result Grid | ▦ | ↻ Filter Rows: [   ] | Edit: 🖉

	id	address	cno	dob	name
▶	1	mumbai	9895746367	19-09-1998	John
	2	Pune	9895746367	19-09-1998	Doe
*	NULL	NULL	NULL	NULL	NULL

Step 8: Similar steps to perform crud operation on InsuarancePolicy and claims.

Insurance Policy Database

pid	camount	end_date	preminum	ptype	start_date	c_id
1	25000	12-05-2023	1000	Health	12-05-2015	1
2	20000	17-05-2023	1200	Health	17-05-2015	1
3	28000	13-08-204	1400	Educ...	13-08-2016	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Claims Database

cno	cdate	cdesc	cstatus
1	12-05-2015	Health care	active
2	12-09-2016	Education	active
3	18-09-2016	Medical	active
4	24-09-2016	Medical	active
5	28-10-2016	Job	active
NULL	NULL	NULL	NULL

Claim associated with insurance Policy

claims_cno	ip_pid
1	1
2	1
3	2
4	2
NULL	NULL

Structure (ERD Diagram):
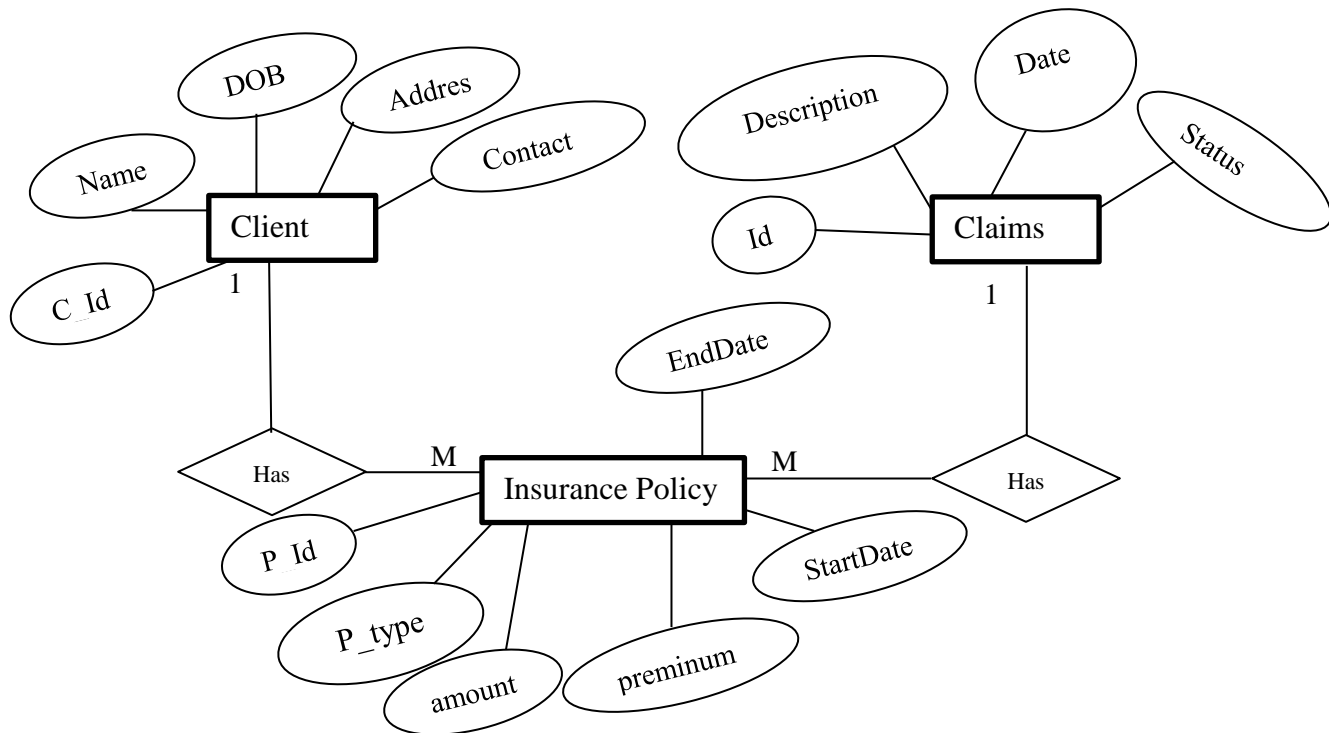
Features:

Today, as we are living in uncertainty, what may happen tomorrow no one knows, whether it is related to our health or the objects around us that we use in our daily life. So, why not take Insurance. Insurance is all about providing financial safety to the individual. It helps us to enjoy financial security. If we have a proper application to access all the Insurance related activity then it will be cheery on top.

Tools and Technologies

Front-End: Postman API

Server-side: Spring Boot.

Back-end: MYSQL, Hibernate.