

## ADMIN APIs

### 1. Revenue

- Summary count of revenue-related data, such as Total revenue, total orders, gross sales, service providers' payouts, and average order value
- `http://localhost:8080/revenue/summary`

- Default Data(overall data return of the current year)

The screenshot shows the Postman interface with a successful API call. The request URL is `http://localhost:8080/revenue/summary`. The response status is `200 OK`, size is `102 Bytes`, and time is `390 ms`. The response body is:

```
1 {
2   "totalRevenue": 0.0,
3   "totalOrders": 0,
4   "grossSales": 0.0,
5   "serviceProviderPayouts": 0.0,
6   "avgOrderValue": 0.0
7 }
```

- Data based on filter(Available filter values are today, this week, this month, custom(provide start and end date), overall(this year's data))

The screenshot shows a REST client interface with the following details:

- Request URL:** GET http://localhost:8081/revenue/summary
- Response Status:** 200 OK
- Response Size:** 102 Bytes
- Response Time:** 88 ms
- Response Body (JSON):**

```
1  {
2      "totalRevenue": 0.0,
3      "totalOrders": 0,
4      "grossSales": 0.0,
5      "serviceProviderPayouts": 0.0,
6      "avgOrderValue": 0.0
7  }
```

- Data based on start and end date(We can specify either start date or end date only, or we can also select both)
- Revenue details for each order(overview)
- **http://localhost:8080/revenue/total-revenue**
- Default Data(overall data return of the current year)

POST localhost:8081/login

Skills.txt

TC New Request

TC New Request\*

Send

GET http://localhost:8081/revenue/total-revenue

Query Headers 2 Auth 1 Body Tests Pre Run

None Basic Bearer OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzUxMiJ9eyJpZCI6MSwi3ViIjoiYWRtaW5AZXhhbXBsZSSjb20iLCJpYXQiOjE3NjAyNDE5MTUzMjMv4cCl6MTc2MDg0NjcxN0.UMK_34_db169x24E8AuxD2E1okZE3syd0bTiEtmiw20lC8n4tiNxGwHf_HNR0kvU_MUiLnbtZMKeqn_kg
```

Token Prefix Bearer

Status: 200 OK Size: 136 Bytes Time: 49 ms

Response Headers 11 Cookies Results Docs

```
1 [
2   {
3     "orderId": 1,
4     "date": "2025-10-12T13:24:51.686401",
5     "customerName": "John Doe",
6     "totalPaid": 62.6,
7     "providerPayout": 80.0,
8     "adminRevenue": 20.0
9   }
10 ]
```

Copy

Response Chart

Signed out Go Live

- Data based on filter(Available filter values are today, this week, this month, custom(provide start and end date), overall(this year's data))
- Data based on start and end date(We can specify either start date or end date only, or we can also select both)
- Revenue detail for a particular order(complete detail)
- `http://localhost:8080/revenue/total-revenue/{orderId}`

POST localhost:8081/login

GET http://localhost:8081/revenue/total-revenue/1

Status: 200 OK Size: 440 Bytes Time: 135 ms

```

1  {
2    "customerName": "John Doe",
3    "providerName": "FreshSpin Laundry",
4    "subServiceName": "Leak Repair",
5    "billDTO": {
6      "invoiceNumber": 101,
7      "charge": 50.0,
8      "gst": 12.6,
9      "total": 62.6
10    },
11    "paymentDTO": {
12      "paymentId": 1,
13      "transactionId": "TXN123456789",
14      "dateTime": "2025-10-12T13:24:40.826501"
15    },
16    "payoutDTO": {
17      "payoutId": 1,
18      "serviceEarning": 100.0,
19      "charge": 20.0,
20      "finalAmount": 80.0,
21      "transactionId": "TXN_PAYOUT_9876",
22      "payoutStatus": "PENDING"
23    },
24    "adminRevenueDTO": {
25      "profit": 20.0
26    }
27  }

```

- Revenue trends (Return graph for revenue trends for admin available types are gross sales, total payouts, service provider payouts based on filter monthly, quarterly, yearly)
- <http://localhost:8080/revenue/trends>
  - Default data(monthly and gross sales)

Postman interface showing a GET request to `http://localhost:8081/revenue/trends`. The **Auth** tab is selected, showing an **Bearer** token input field containing a long JWT token.

The response status is **200 OK**, size is **1.04 KB**, and time is **124 ms**. The response body is a JSON object:

```

1  {
2    "title": "Monthly Gross Sales",
3    "data": [
4      {
5        "label": "01 OCT",
6        "revenue": 0.0
7      },
8      {
9        "label": "02 OCT",
10       "revenue": 0.0
11     },
12     {
13       "label": "03 OCT",
14       "revenue": 0.0
15     },
16     {
17       "label": "04 OCT",
18       "revenue": 0.0
19     },
20     {
21       "label": "05 OCT",
22       "revenue": 0.0
23     },
24     {
25       "label": "06 OCT",
26       "revenue": 0.0
27     }
...

```

- Based on the filter(Available values are monthly, quarterly, and yearly)

Postman interface showing a GET request to `http://localhost:8081/revenue/trends`. The **Body** tab is selected, showing a **Form** data type with two fields: `filter` (set to `Quarterly`) and `type` (set to `Admin Revenue`).

The response status is **200 OK**, size is **180 Bytes**, and time is **64 ms**. The response body is a JSON object:

```

1  {
2    "title": "Quarterly Admin Revenue",
3    "data": [
4      {
5        "label": "Q1 2025",
6        "revenue": 0.0
7      },
8      {
9        "label": "Q2 2025",
10       "revenue": 0.0
11     },
12     {
13       "label": "Q3 2025",
14       "revenue": 0.0
15     },
16     {
17       "label": "Q4 2025",
18       "revenue": 0.0
19     }
20   ]
21 }

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. The main area has tabs for 'localhost:8081/login', 'Skills.txt', and several 'New Request' tabs. A search bar at the top right says 'Notes'. The current request is a 'GET' to 'http://localhost:8081/revenue/trends'. The 'Body' tab is selected, showing 'Form' data with fields: 'filter' (set to 'Yearly'), 'type' (set to 'Provider Payout'), and 'field name' (set to 'value'). Below this, it says 'Learn more about how to set a custom [content-type](#) for a field.' The 'Response' tab is active, displaying the following JSON:

```
1  {
2    "title": "Yearly Provider Payout",
3    "data": [
4      {
5        "label": "JAN",
6        "revenue": 0.0
7      },
8      {
9        "label": "FEB",
10       "revenue": 0.0
11     },
12     {
13       "label": "MAR",
14       "revenue": 0.0
15     },
16     {
17       "label": "APR",
18       "revenue": 0.0
19     },
20     {
21       "label": "MAY",
22       "revenue": 0.0
23     },
24     {
25       "label": "JUN",
26       "revenue": 0.0
27     }
28   ]
```

- Revenue analytics list for each service provider
- <http://localhost:8080/revenue/provider-analytics-list>
  - Default data(monthly)

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. The main area has tabs for 'localhost:8081/login' and 'Skills.txt'. A search bar at the top right contains the text 'Notes'. Below the tabs, a 'Send' button is visible. The 'Body' tab is selected, showing a 'Form' section with a 'filter' field containing 'Quarterly' and an unchecked 'field name' field. To the right, the response status is '200 OK', size is '192 Bytes', and time is '44 ms'. The 'Response' tab is selected, displaying the JSON response:

```
1 [  
2 {  
3   "providerId": 1,  
4   "totalRevenue": 80.0,  
5   "platformCharges": 20.0,  
6   "dateTime": "2025-10-12T13:24:47.07176",  
7   "payoutCount": 1,  
8   "startDate": "2025-08-01T00:00:00",  
9   "endDate": "2025-10-12T13:26:34.094830088"  
10 }  
11 ]
```

- Based on the filter(Available values are monthly, quarterly, and yearly)
- Revenue analytics graph data of a particular service provider
- `http://localhost:8080/revenue/provider-analytics-list/{providerId}`

Status: 200 OK Size: 1.07 KB Time: 126 ms

```

1   {
2     "title": "Monthly Revenue for service provider : John Doe",
3     "data": [
4       {
5         "label": "01 OCT",
6         "revenue": 0.0
7       },
8       {
9         "label": "02 OCT",
10        "revenue": 0.0
11      },
12      {
13        "label": "03 OCT",
14        "revenue": 0.0
15      },
16      {
17        "label": "04 OCT",
18        "revenue": 0.0
19      },
20      {
21        "label": "05 OCT",
22        "revenue": 0.0
23      },
24      {
25        "label": "06 OCT",
26        "revenue": 0.0
27    }
  
```

## 2. Service providers joining requests.

- Pending requests
- <http://localhost:8080/request/provider-requests>

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons for file operations, selection, view, and run. The main area has tabs for 'localhost:8081/login', 'localhost:8081/register', and 'Skills.txt'. A 'New Request' tab is also present. The current request is a 'GET' to 'http://localhost:8081/request/provider-requests'. The 'Auth' tab is selected, showing 'Bearer' as the type with a token input field containing a long JWT token. Other tabs include 'Basic', 'OAuth 2', 'NTLM', and 'AWS'. Below the auth section, there are fields for 'Token Prefix' and 'Bearer'. The response tab shows a status of '200 OK', size of '756 Bytes', and time of '198 ms'. The response body is a JSON object with 23 lines of data, including user details like userId, firstName, lastName, phone, email, joinedAt, businessName, businessLicenseNumber, gstNumber, profilePhoto, aadharCardPhoto, panCardPhoto, businessUtilityBillPhoto, bankName, ifscCode, bankAccountNumber, accountHolderName, addresses, name, areaName, and pincode.

```
[{"userId": 152, "firstName": "John", "lastName": "Doe", "phone": "9876543210", "email": "john@example.com", "joinedAt": "2025-10-12T11:52:37.822527Z", "businessName": "FreshSpin Laundry", "businessLicenseNumber": "LIC123456", "gstNumber": "GSTIN27AACF1234A1ZB", "profilePhoto": "https://cdn.example.com/images/freshspin_profile.jpg", "aadharCardPhoto": "https://cdn.example.com/images/freshspin_aadhar.jpg", "panCardPhoto": "https://cdn.example.com/images/freshspin_pan.jpg", "businessUtilityBillPhoto": "https://cdn.example.com/images/freshspin_bill.jpg", "bankName": "HDFC Bank", "ifscCode": "HDFC0000456", "bankAccountNumber": "123456789012", "accountHolderName": "Ramesh Patel", "addresses": [{"name": "Home", "areaName": "MG Road", "pincode": "560001"}]}
```

## - Accept request

```
- http://localhost:8080/request/accept-provider/{userId}
```

The screenshot shows the Postman application interface. The setup is identical to the previous one, with the 'Auth' tab selected for 'Bearer' and a token input field. The request method is now 'PUT' to 'http://localhost:8081/request/accept-provider/152'. The response tab shows a status of '200 OK', size of '47 Bytes', and time of '5.77 s'. The response body contains the message 'Service provider profile approved successfully.'

```
Service provider profile approved successfully.
```

The screenshot shows the Postman application interface. A PUT request is made to `http://localhost:8081/request/accept-provider/152`. The response status is 200 OK, size is 53 Bytes, and time is 26 ms. The response body contains the message: "Request for Service provider profile already accepted". The "Auth" tab is selected, showing "Bearer" as the method with a token input field containing a long JWT token.

The screenshot shows the Postman application interface. A PUT request is made to `http://localhost:8081/request/accept-provider/152`. The response status is 200 OK, size is 53 Bytes, and time is 23 ms. The response body contains the message: "Request for Service provider profile already rejected". The "Auth" tab is selected, showing "Bearer" as the method with a token input field containing a long JWT token.

- Reject request
- `http://localhost:8080/request/reject-provider/{userId}`

The screenshot shows the Postman application interface. A PUT request is being made to `http://localhost:8081/request/reject-provider/152`. The request is authenticated using a Bearer token. The response status is 200 OK, size is 34 bytes, and time taken is 5.44 s. The response body contains the message: "Service provider profile rejected."

Request URL: `PUT http://localhost:8081/request/reject-provider/152`

Auth: Bearer

Response:

```
Status: 200 OK Size: 34 Bytes Time: 5.44 s
Response Headers: 11 Cookies: Results: Docs: 1 Service provider profile rejected.
```

This screenshot is identical to the one above, showing a successful PUT request to `http://localhost:8081/request/reject-provider/152` with a Bearer token authentication. The response status is 200 OK, size is 54 bytes, and time taken is 21 ms. The response body contains the message: "Request for service provider profile already rejected."

Request URL: `PUT http://localhost:8081/request/reject-provider/152`

Auth: Bearer

Response:

```
Status: 200 OK Size: 54 Bytes Time: 21 ms
Response Headers: 11 Cookies: Results: Docs: 1 Request for service provider profile already rejected.
```

The screenshot shows the Postman application interface. A PUT request is being made to `http://localhost:8081/request/reject-provider/152`. The 'Auth' tab is selected, showing 'Bearer' authentication with a token input field containing a long JWT token. The response status is **200 OK**, size is **54 Bytes**, and time is **151 ms**. The response body is: `Request for service provider profile not accepted yet.`

This screenshot shows the same Postman session after a successful rejection. The response status is now **200 OK**, size is **53 Bytes**, and time is **21 ms**. The response body is: `Request for Service provider profile already accepted`.

### 3. Users Details

#### i. Customer

- Graph data for an overview of customers (Four types of graph data are returned: a. customer growth based on months, b. customer growth in the current month, c. customer growth in the last 5 years, d. how many customers are from each city )
- `http://localhost:8080/users/customers/graphs`

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/users/customers/graphs`. The response body is a JSON object representing user growth trends by month:

```

1  {
2    "usersThisYearMonthlyTrend": [
3      {
4        "month": "JAN",
5        "count": 0
6      },
7      {
8        "month": "FEB",
9        "count": 0
10     },
11     {
12       "month": "MAR",
13       "count": 0
14     },
15     {
16       "month": "APR",
17       "count": 0
18     },
19     {
20       "month": "MAY",
21       "count": 0
22     },
23     {
24       "month": "JUN",
25       "count": 0
26     },
27   ]
}

```

Status: 200 OK Size: 734 Bytes Time: 246 ms

```

40     "monday": 0,
41     "count": 0
42   },
43   ],
44   "newUsersThisMonthDailyTrend": [
45     {
46       "day": 1,
47       "count": 0
48     },
49     {
50       "day": 2,
51       "count": 0
52     },
53     {
54       "day": 3,
55       "count": 0
56     },
57     {
58       "day": 4,
59       "count": 0
60     },
61     {
62       "day": 5,
63       "count": 0
64     },
65     {
66       "day": 6,
67     }
    ]
  }
}

```

Status: 200 OK Size: 768 Bytes Time: 140 ms

```

94   },
95   "userGrowthTrend": [
96     {
97       "year": 2021,
98       "count": 0
99     },
100    {
101      "year": 2022,
102      "count": 0
103    },
104    {
105      "year": 2023,
106      "count": 0
107    },
108    {
109      "year": 2024,
110      "count": 0
111    },
112    {
113      "year": 2025,
114      "count": 1
115    }
116  ],
117  "regionWiseDistribution": [
118    {
119      "city": "Ahmedabad",
120      "userCount": 1
121    }
  ]
}

```

- Table data for customer details
- <http://localhost:8080/users/customers/table>

- Default data(start date is 2025-01-01, end date is the current date, and sort by user ID)

POST localhost:8081/login Skills.txt New Request \* New Request \* New Request \*

GET http://localhost:8081/users/customers/table Send

Query Headers 2 Auth 1 Body Tests Pre Run

None Basic Bearer OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzUxMiJ9eyJcI6MSwi3VijoiYWRtaW5AZXhhbXBsZSSjb20iLCJpYXQiOjE3NjAyNDE5MTUslmV4cCl6MTc2MDg0NjcxN0.UMK_34_db169x24E8AuxD2E1okZE3syd0bTiEtmiw20lC8n4tiNpxGwHf_HNR0kvU_MUiLnbtZMKeqn_kg
```

Token Prefix Bearer

Status: 200 OK Size: 227 Bytes Time: 107 ms

Response Headers Cookies Results Docs

```
[{"userId": 52, "firstName": "John", "lastName": "Doe", "phone": "9876543210", "email": "john@example.com", "joinAt": "2025-10-12T10:50:41.56629", "addresses": [{"name": "Home", "areaName": "MG Road", "pincode": "560001", "cityName": "Ahmedabad"}]}
```

Response Chart ↗ Signed out Go Live

0 Java: Ready

- Data based on keyword(Search by email or phone)

POST localhost:8081/login Skills.txt New Request \* New Request \* New Request \*

GET http://localhost:8081/users/customers/table Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

Form Fields Files Import

email john@example.com

field name value

Learn more about how to set a custom [content-type](#) for a field.

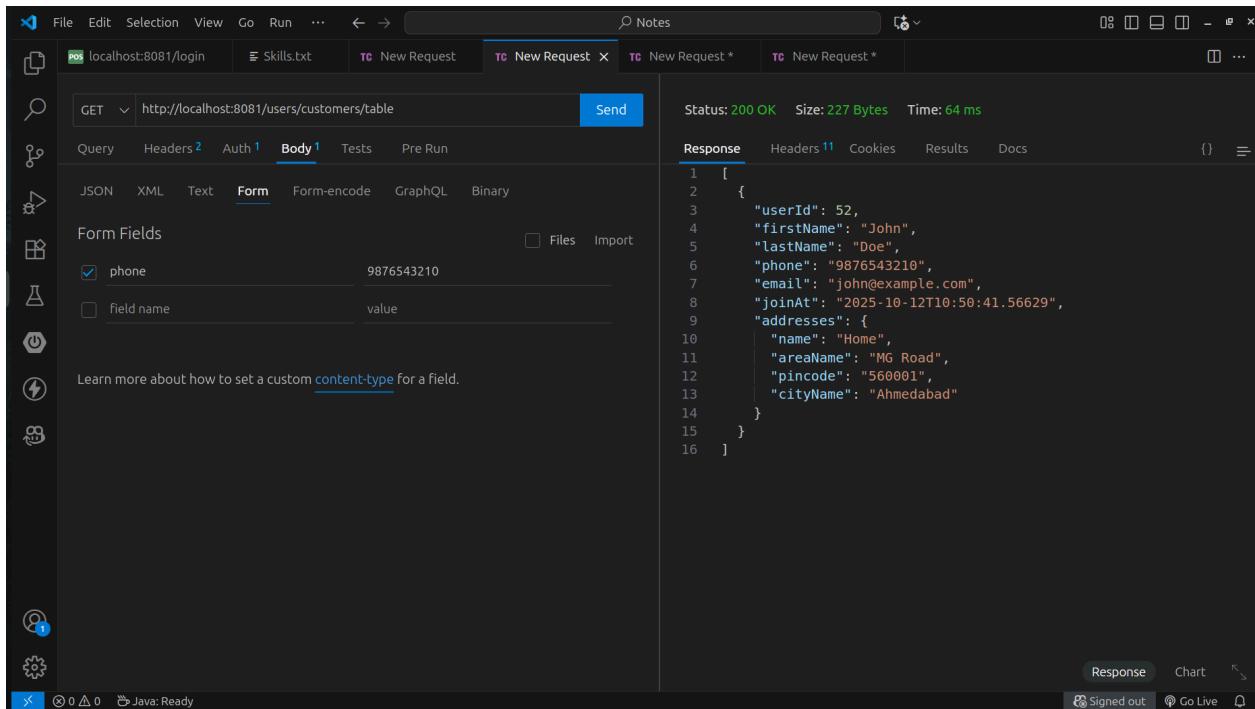
Status: 200 OK Size: 227 Bytes Time: 53 ms

Response Headers Cookies Results Docs

```
[{"userId": 52, "firstName": "John", "lastName": "Doe", "phone": "9876543210", "email": "john@example.com", "joinAt": "2025-10-12T10:50:41.56629", "addresses": [{"name": "Home", "areaName": "MG Road", "pincode": "560001", "cityName": "Ahmedabad"}]}
```

Response Chart ↗ Signed out Go Live

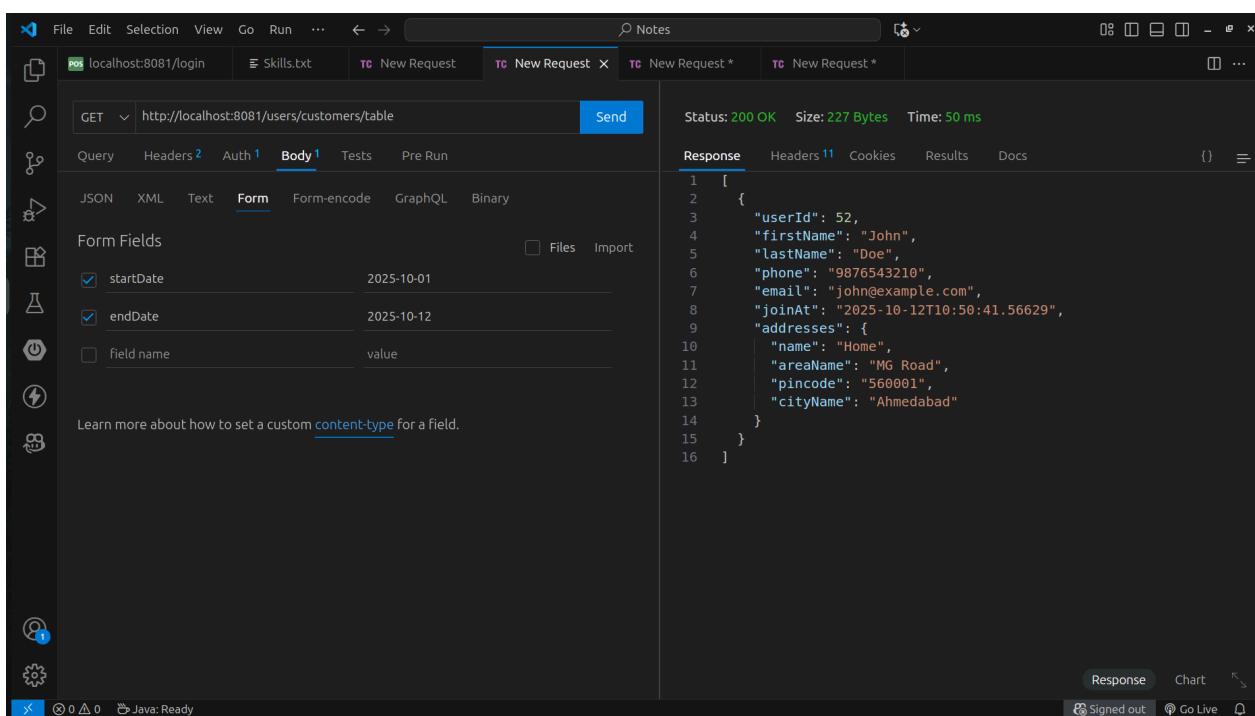
0 Java: Ready



The screenshot shows the Postman interface with a request to `http://localhost:8081/users/customers/table`. The **Body** tab is selected, showing a **Form** with one field named `phone` containing the value `9876543210`. The response status is **200 OK**, size is **227 Bytes**, and time is **64 ms**. The response body is a JSON object:

```
1 [  
2 {  
3   "userId": 52,  
4   "firstName": "John",  
5   "lastName": "Doe",  
6   "phone": "9876543210",  
7   "email": "john@example.com",  
8   "joinAt": "2025-10-12T10:50:41.56629",  
9   "addresses": [  
10     {"name": "Home",  
11       "areaName": "MG Road",  
12       "pincode": "560001",  
13       "cityName": "Ahmedabad"  
14     }  
15   ]  
16 ]
```

- Data based on start and end date(We can specify either start date or end date only, or we can also add both)



The screenshot shows the Postman interface with a request to `http://localhost:8081/users/customers/table`. The **Body** tab is selected, showing a **Form** with two fields: `startDate` and `endDate`, both set to the value `2025-10-01`. The response status is **200 OK**, size is **227 Bytes**, and time is **50 ms**. The response body is identical to the previous screenshot.

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons for file operations, search, and other tools. The main area has tabs for 'localhost:8081/login' and 'Skills.txt'. A 'New Request' tab is open, showing a GET request to 'http://localhost:8081/users/customers/table'. The 'Body' tab is selected, showing a 'Form' with a checked 'startDate' field set to '2025-10-01'. The 'Response' tab shows a status of '200 OK', size of '227 Bytes', and time of '47 ms'. The JSON response is displayed as:

```
1 [  
2 {  
3   "userId": 52,  
4   "firstName": "John",  
5   "lastName": "Doe",  
6   "phone": "9876543210",  
7   "email": "john@example.com",  
8   "joinAt": "2025-10-12T10:50:41.56629",  
9   "addresses": {  
10     "name": "Home",  
11     "areaName": "MG Road",  
12     "pincode": "560001",  
13     "cityName": "Ahmedabad"  
14   }  
15 }  
16 ]
```

This screenshot is nearly identical to the one above, showing the same Postman interface and request details. The difference is in the response body. The 'Response' tab now shows a status of '200 OK', size of '2 Bytes', and time of '40 ms'. The JSON response is displayed as:

```
1 []
```

- We can also apply sort by filter along with a date filter. This allows for sorting based on either the join date or the user ID.

File Edit Selection View Go Run ... ← → Notes

localhost:8081/login Skills.txt New Request \* New Request \*

GET http://localhost:8081/users/customers/table Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

Form Fields

startDate 2025-10-01

endDate 2025-10-12

sortBy Joining Date

field name value

Learn more about how to set a custom [content-type](#) for a field.

Status: 200 OK Size: 227 Bytes Time: 405 ms

Response Headers 11 Cookies Results Docs

```
1 [
2   {
3     "userId": 52,
4     "firstName": "John",
5     "lastName": "Doe",
6     "phone": "9876543210",
7     "email": "john@example.com",
8     "joinAt": "2025-10-12T10:50:41.56629",
9     "addresses": [
10       {
11         "name": "Home",
12         "areaName": "MG Road",
13         "pincode": "560001",
14         "cityName": "Ahmedabad"
15       }
16     ]
17   ]
18 ]
```

File Edit Selection View Go Run ... ← → Notes

localhost:8081/login Skills.txt New Request \* New Request \*

GET http://localhost:8081/users/customers/table Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

Form Fields

startDate 2025-10-01

endDate 2025-10-12

sortBy Id

field name value

Learn more about how to set a custom [content-type](#) for a field.

Status: 200 OK Size: 227 Bytes Time: 63 ms

Response Headers 11 Cookies Results Docs

```
1 [
2   {
3     "userId": 52,
4     "firstName": "John",
5     "lastName": "Doe",
6     "phone": "9876543210",
7     "email": "john@example.com",
8     "joinAt": "2025-10-12T10:50:41.56629",
9     "addresses": [
10       {
11         "name": "Home",
12         "areaName": "MG Road",
13         "pincode": "560001",
14         "cityName": "Ahmedabad"
15       }
16     ]
17   ]
18 ]
```

## - Block/Unblock(toggle) particular customer

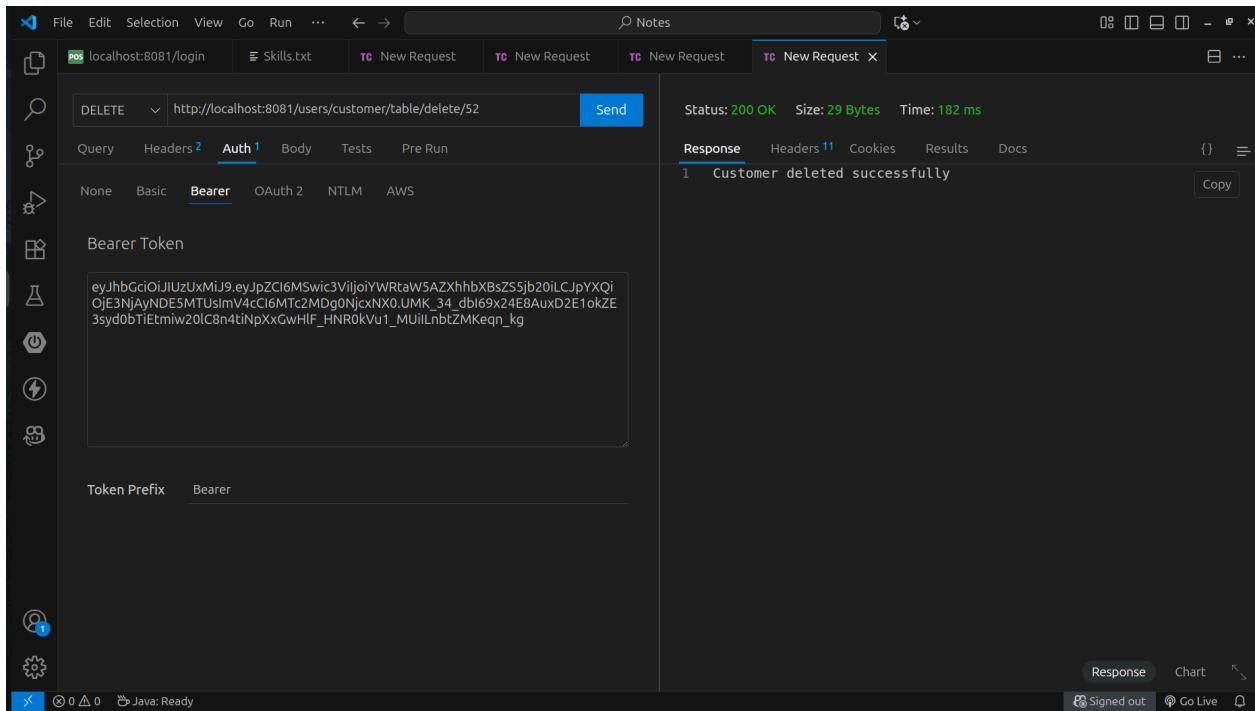
- <http://localhost:8080/users/customers/table/block-unblock/{userId}>

The screenshot shows the Postman application interface. A PUT request is being made to `http://localhost:8081/users/customers/table/block-unblock/52`. The 'Auth' tab is selected, showing a Bearer token. The response status is 200 OK, size 43 Bytes, and time 63 ms. The response body contains the message: "1 Users status changed from block to unblock." A copy button is available for the response body.

This screenshot shows another instance of the Postman application. A PUT request is being made to the same endpoint: `http://localhost:8081/users/customers/table/block-unblock/52`. The 'Auth' tab is selected, showing a Bearer token. The response status is 200 OK, size 43 Bytes, and time 187 ms. The response body contains the message: "1 Users status changed from unblock to block." A copy button is available for the response body.

- Delete a particular customer

```
- http://localhost:8080/users/customer/table/delete/{userId}
```



## ii. Service Provider

- Graph data for an overview of service providers (Four types of graph data are returned: a. service provider growth based on months, b. service provider growth in the current month, c. service provider growth in the last 5 years d. How many service providers are from each city)
  - <http://localhost:8080/users/service-providers/graphs>

The screenshot shows the Postman interface with a successful API call to `localhost:8081/login`. The response body is a JSON object:

```

1  {
2    "usersThisYearMonthlyTrend": [
3      {
4        "month": "JAN",
5        "count": 0
6      },
7      {
8        "month": "FEB",
9        "count": 0
10     },
11     {
12       "month": "MAR",
13       "count": 0
14     },
15     {
16       "month": "APR",
17       "count": 0
18     },
19     {
20       "month": "MAY",
21       "count": 0
22     },
23     {
24       "month": "JUN",
25       "count": 0
26     }
27   }

```

- Table data for service provider details(image link must be clickable when admin clicks on that link, one page is open that displays the image stored in Cloudinary)
- `http://localhost:8080/users/service-providers/table`

- Default data(start date is 2025-01-01, end date is the current date, and sort by user ID)
- Data based on keyword(Search by email or phone)
- Data based on start and end date(We can specify either start date or end date only, or we can also specify both)
- We can also apply sort by filter along with a date filter. This allows for sorting based on either the join date or the user ID.
- All the above filters function in the same manner as the customer's filter.

GET <http://localhost:8081/users/service-providers/table>

Status: 200 OK Size: 778 Bytes Time: 72 ms

```

1 [
2   {
3     "userId": 152,
4     "serviceProviderId": 1,
5     "firstName": "John",
6     "lastName": "Doe",
7     "phone": "+9876543210",
8     "email": "john@example.com",
9     "joinedAt": "2025-10-12T11:52:37.822527",
10    "businessName": "FreshSpin Laundry",
11    "businessLicenseNumber": "LIC123456",
12    "gstNumber": "GSTIN27AACF1234A1ZB",
13    "profilePhoto": "https://cdn.example.com/images/freshspin_profile.jpg",
14    "aadharCardPhoto": "https://cdn.example.com/images/freshspin_aadhar.jpg",
15    "panCardPhoto": "https://cdn.example.com/images/freshspin_pan.jpg",
16    "businessUtilityBillPhoto": "https://cdn.example.com/images/freshspin_bill.jpg",
17    "bankName": "HDFC Bank",
18    "ifscCode": "HDFC0000456",
19    "bankAccountNumber": "123456789012",
20    "accountHolderName": "Ramesh Patel",
21    "addresses": [
22      {
23        "name": "Home",
24        "areaName": "MG Road".
25      }
26    ]
27  }
28]

```

- Block/Unblock(toggle) particular service provider
- <http://localhost:8080/users/service-providers/table/block-unblock/{userId}>

PUT <http://localhost:8081/users/service-providers/table/block-unblock/102>

Status: 200 OK Size: 43 Bytes Time: 57 ms

```
1 Users status changed from unblock to block.
```

The screenshot shows the Postman application interface. A PUT request is made to `http://localhost:8081/users/service-providers/table/block-unblock/102`. The response status is 200 OK, size is 43 Bytes, and time is 34 ms. The response body contains the message: "1 Users status changed from block to unblock." The "Auth" tab is selected, showing a Bearer token.

- Delete a particular service provider

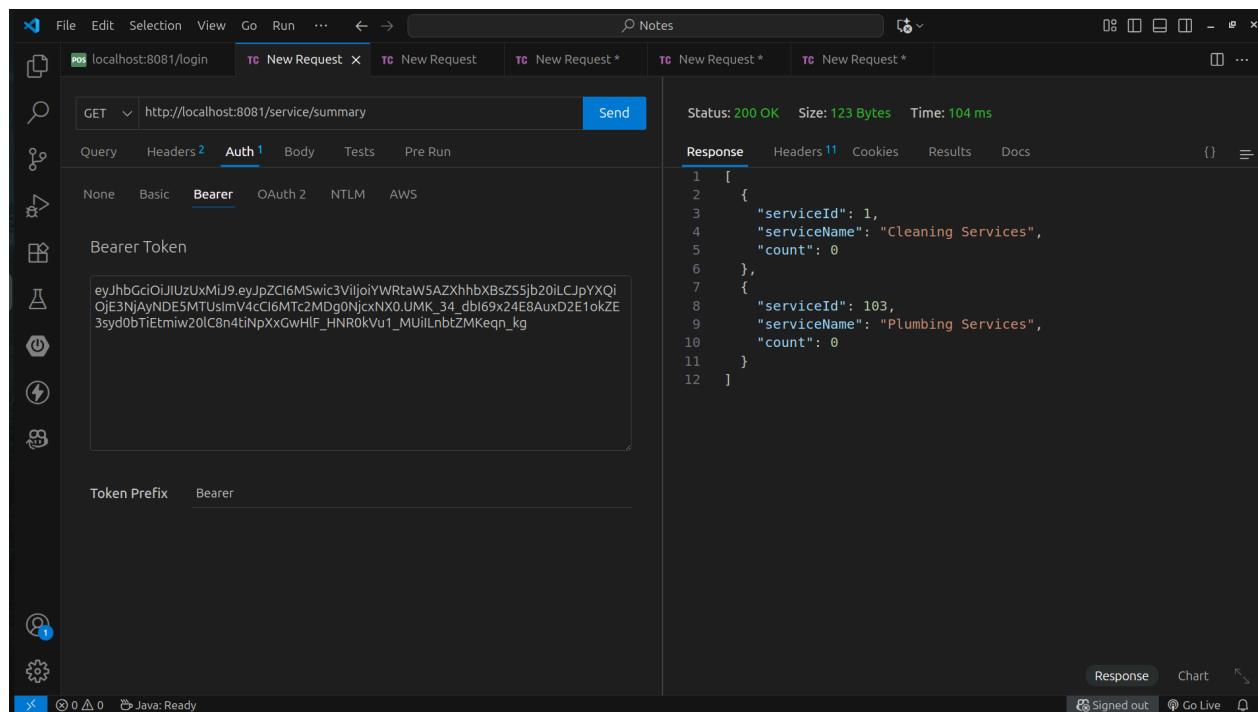
- `http://localhost:8080/users/service-providers/table/delete/{providerId}`

The screenshot shows the Postman application interface. A DELETE request is made to `http://localhost:8081/users/service-providers/table/delete/1`. The response status is 200 OK, size is 37 Bytes, and time is 64 ms. The response body contains the message: "1 Service provider deleted successfully." The "Auth" tab is selected, showing a Bearer token.

## 4. Service Listing

- Get the count of services

```
- http://localhost:8080/service/summary
```



The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. In the center, a request is being made to `http://localhost:8080/service/summary`. The method is set to `GET`. Under the `Auth` tab, `Bearer` is selected, and a token is entered: `eyJhbGciOiJIUzIwMiJ9.eyJpZCI6MSwi...kg`. The response tab shows the following JSON data:

```
[{"serviceId": 1, "serviceName": "Cleaning Services", "count": 0}, {"serviceId": 103, "serviceName": "Plumbing Services", "count": 0}]
```

- Add new service

```
- http://localhost:8080/service/add-services
```

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/service/add-services`. The request body contains the JSON object `{"serviceName": "Cleaning Services"}`. The response status is **200 OK**, size is **26 Bytes**, and time taken is **163 ms**. The response body is `1 Service added successfully`.

```
POST http://localhost:8081/service/add-services
{
  "serviceName": "Cleaning Services"
}
1 Service added successfully
```

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/service/add-services`. The request body contains the JSON object `{"serviceName": "A"}`. The response status is **400 Bad Request**, size is **112 Bytes**, and time taken is **56 ms**. The response body is a JSON object with `"status": 400`, `"error": "Bad Request"`, and `"message": "serviceName: Service name must be between 3 and 100 characters."`.

```
POST http://localhost:8081/service/add-services
{
  "serviceName": "A"
}
{
  "status": 400,
  "error": "Bad Request",
  "message": "serviceName: Service name must be between 3 and 100 characters."
}
```

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/service/add-services`. The request body contains the following JSON:

```
1 {
2     "serviceName": "A@*123"
3 }
```

The response status is **400 Bad Request**, with a size of **103 Bytes** and a time of **54 ms**. The response body is:

```
1 {
2     "status": 400,
3     "error": "Bad Request",
4     "message": "serviceName: Service name contains invalid characters."
5 }
```

## - Get services

```
- http://localhost:8080/service/get-services
```

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8081/service/get-services`. The request includes an **Auth** header set to **Bearer**. The response status is **200 OK**, with a size of **41 Bytes** and a time of **50 ms**. The response body is:

```
1 [
2     "Cleaning Services",
3     "Plumbing Services"
4 ]
```

- Add new sub service

- `http://localhost:8080/service/add-subservices`

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/service/add-subservices`. The request body is JSON, containing:

```
1 {
2   "subServiceName": "Leak Repair",
3   "serviceName": "Plumbing Services"
4 }
```

The response status is **200 OK**, size is **30 Bytes**, and time is **423 ms**. The response body is:

```
1 Sub-service added successfully
```

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/service/add-subservices`. The request body is JSON, containing:

```
1 {
2   "subServiceName": "A",
3   "serviceName": "Plumbing Services"
4 }
```

The response status is **400 Bad Request**, size is **119 Bytes**, and time is **81 ms**. The response body is:

```
1 {
2   "status": 400,
3   "error": "Bad Request",
4   "message": "subServiceName: Sub Service name must be between 3 and 100 characters."
5 }
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. In the center, a request is being built: method POST, URL http://localhost:8081/service/add-subservices, and a JSON body:

```
1 {
2     "subServiceName": "A@#_123",
3     "serviceName": "Plumbing Services"
4 }
```

The response tab shows a status of 400 Bad Request, with a size of 110 Bytes and a time of 30 ms. The response body is:

```
1 {
2     "status": 400,
3     "error": "Bad Request",
4     "message": "subServiceName: Sub Service name contains invalid
5     characters."
```

This screenshot shows another instance of the Postman application. The request setup is identical to the one above, but the response is different. The status is 500 Internal Server Error, with a size of 84 Bytes and a time of 42 ms. The response body is:

```
1 {
2     "status": 500,
3     "error": "Internal Server Error",
4     "message": "Service is not available."
5 }
```

- Get sub-service based on service name

```
- http://localhost:8080/service/get-subservices/{serviceName}
```

File Edit Selection View Go Run ... ← → Notes

localhost:8081/login TC New Request X

GET http://localhost/service/get-subservices/Plumbing Services Send

Query Headers 2 Auth 1 Body Tests Pre Run

None Basic Bearer OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzUxMiJ9eyJpZCI6MSwi3ViIjoiYWRtaW5AZXhhbXBsZSSjb20iLCJpYXQiOjE3NjAyNDE5MTUsImV4cCI6MTc2MDg0NjcxN00.UMK_34_db169x24E8AuxD2E1okZE3syd0bTiEtmiw20lC8n4tiNpxGwHf_HNR0kvU_MUiLnbtZMKeqn_kg
```

Token Prefix Bearer

Status: 200 OK Size: 102 Bytes Time: 77 ms

Response Headers 11 Cookies Results Docs

```
[{"Service": "Faucet/ Tap Replacement"}, {"Service": "Leak Repair"}, {"Service": "Pipe Installation"}, {"Service": "Tap Replacement"}, {"Service": "Water Heater Service"}]
```

Copy Response Chart ↻

Signed out Go Live

## 5. Configurations

- Add revenue-breakdown
- `http://localhost:8080/configurations/revenue-breakdown`

Oct 10 20:13

File Edit Selection View Go Run ... ← → Notes

POST http://localhost:8081/configurations/revenue-breakdown Send

Query Headers 2 Auth 1 Body! Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "serviceProviderRevenue": 30.0,
3   "message": "Revenue is set as Service Provider : 30.0%"
4 }
```

Status: 200 OK Size: 86 Bytes Time: 186 ms

Response Headers Cookies Results Docs

Copy

Signed out Go Live

- Get revenue breakdown details

- `http://localhost:8080/configurations/revenue-breakdown/history`

Oct 10 20:14

File Edit Selection View Go Run ... ← → Notes

GET http://localhost:8081/configurations/revenue-breakdown/history Send

Query Headers 2 Auth 1 Body Tests Pre Run

None Basic Bearer OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzUxMiJ9eyJpZCI6MSwiCi3ViljoiYWRtaW5AZXhhbXBsZS5jb20iLCJpYXQiOjE3NjAxMDczNTMsImV4cCI6MTc2MDcxMjE1M30.92x43JUj1ocMBv5PKEn85nNmijzsEQTfxarwi0jq9YQYw5zBMNCPyhsNgvZh-VGW-ET5JT98XQfxApknIPCw
```

Token Prefix Bearer

Status: 200 OK Size: 144 Bytes Time: 351 ms

Response Headers Cookies Results Docs

Copy

Signed out Go Live

- Change the status of the existing revenue breakdown
- `http://localhost:8080/configurations/revenue-breakdown/history/{id}`

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. The main area has tabs for 'Welcome', 'localhost:8081/login', 'New Request' (which is selected), and 'New Request'. The 'New Request' tab has a sub-tab 'Auth' which is also selected. Under 'Auth', 'Bearer' is chosen, and a token is pasted into the input field: `eyJhbGciOiJIUzUxMiJ9eyJpc3MiJ9.eyJ0eXAiOiYWRtaW5AZXhbXbsZS5jb20iLCJpYXQiOjE3NjAxMDcZNMTsImV4cCI6MTc2MDcxMjE1M30.92x43JuJ1ocMBv5PKEn85nNmyiZsEQTFXarwi0Jq9JYQYw5zBMNCPyhsNgvZh-VGW-ET5JT98XQfxiApknIPcw`. The 'Send' button is visible. To the right, the response section shows a status of 200 OK, size of 35 bytes, and time of 152 ms. The response body contains the message: `Status of lis changed successfully.`. There are tabs for Response, Headers, Cookies, Results, and Docs. At the bottom, there are buttons for Response, Chart, Signed out, Go Live, and other settings.

- Add State
- `http://localhost:8080/configurations/add-state`

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/configurations/add-state`. The request body contains the JSON object 

```
1 {  
2   "stateName": "Uttar Pradesh"  
3 }
```

. The response status is **200 OK**, size is **40 Bytes**, and time taken is **27 ms**. The response body is `1 State : Uttar Pradesh added successfully`.

## - Add City

- `http://localhost:8080/configurations/add-city`

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/configurations/add-city`. The request body contains the JSON object 

```
1 {  
2   "cityName": "Ahmedabad",  
3   "stateName": "Gujarat"  
4 }
```

. The response status is **200 OK**, size is **35 Bytes**, and time taken is **54 ms**. The response body is `1 City : Ahmedabad added successfully`.

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8081/configurations/add-city`. The request body contains the following JSON:

```
1 {
2   "cityName": "Ahmedabad",
3   "stateName": "Mumbai"
4 }
```

The response status is **400 Bad Request**, with a message: "State not exist".

- Get all states

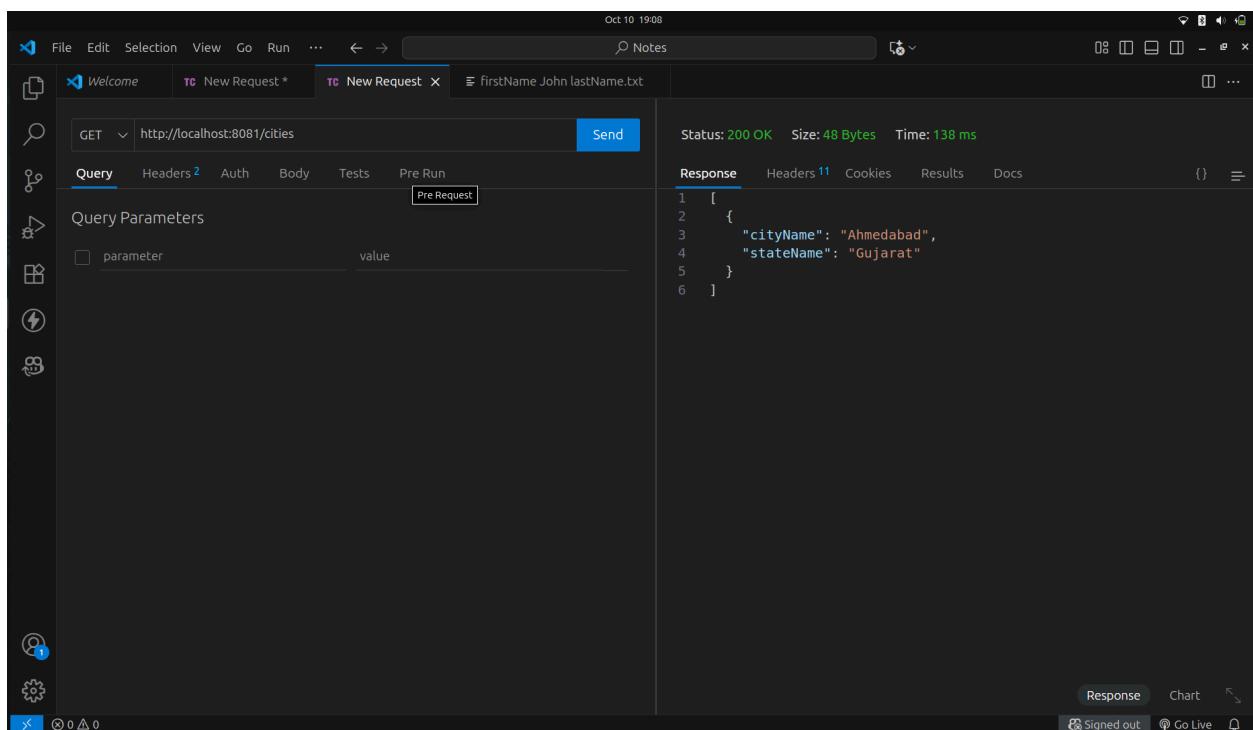
- `http://localhost:8080/states`

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8081/states`. The response status is **200 OK**, and the response body is:

```
1 [
2   {
3     "stateName": "Gujarat"
4   },
5   {
6     "stateName": "Maharashtra"
7   },
8   {
9     "stateName": "Uttar Pradesh"
10 }
11 ]
```

- Get all cities

- http://localhost:8080/cities



Oct 10 19:08

File Edit Selection View Go Run ... ← → Notes

TC New Request \* TC New Request x firstName John lastName.txt

GET http://localhost:8080/cities Send

Query Headers 2 Auth Body Tests Pre Run Response Headers 11 Cookies Results Docs

Status: 200 OK Size: 48 Bytes Time: 138 ms

Response

```
1 [
2   {
3     "cityName": "Ahmedabad",
4     "stateName": "Gujarat"
5   }
6 ]
```

Pre Request

Query Parameters

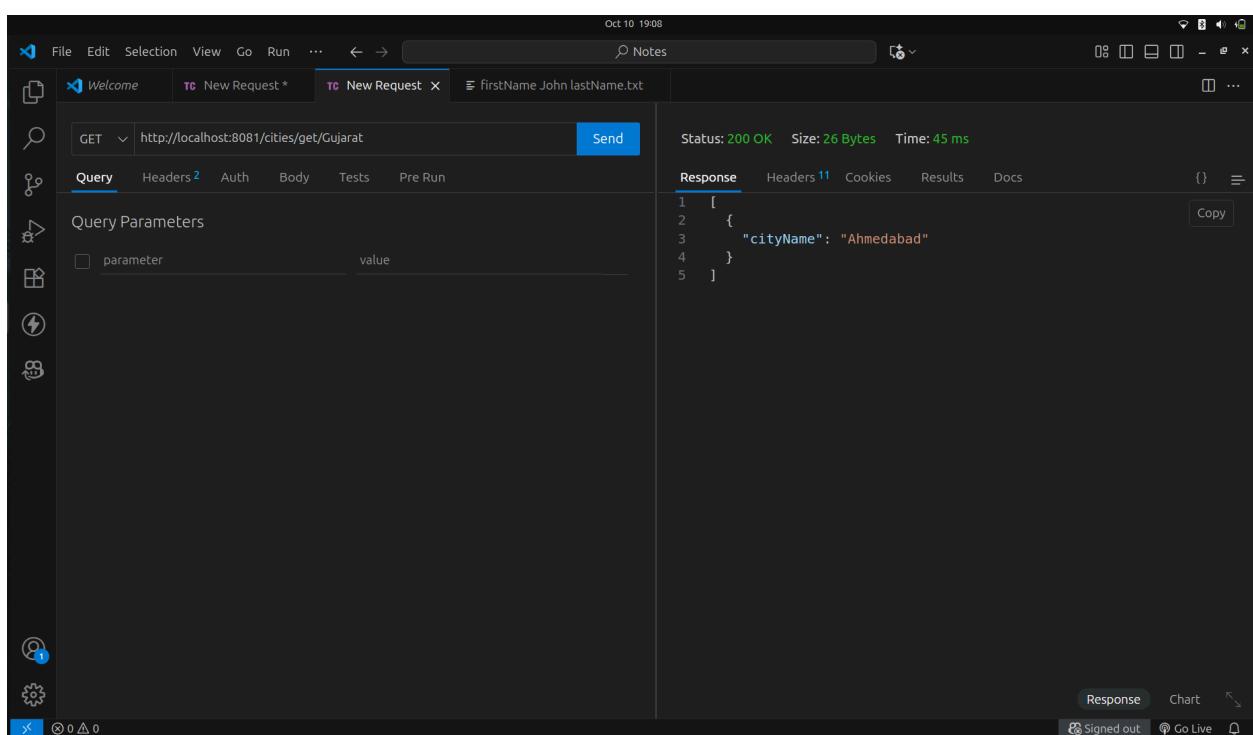
parameter value

Response Chart Copy

Signed out Go Live

## - Get cities by state name

- http://localhost:8080/cities/get/{stateName}



Oct 10 19:08

File Edit Selection View Go Run ... ← → Notes

TC New Request \* TC New Request x firstName John lastName.txt

GET http://localhost:8080/cities/get/Gujarat Send

Query Headers 2 Auth Body Tests Pre Run Response Headers 11 Cookies Results Docs

Status: 200 OK Size: 26 Bytes Time: 45 ms

Response Headers 11 Cookies Results Docs

```
1 [
2   {
3     "cityName": "Ahmedabad"
4   }
5 ]
```

Copy

Query Parameters

parameter value

Response Chart Copy

Signed out Go Live

## 6. Profile Page

### - Get data

```
- http://localhost:8080/profile/view
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons. In the center, a 'New Request' tab is selected. The request URL is set to `http://localhost:8081/profile/view`. Under the 'Auth' tab, 'Bearer' is selected, and a token is pasted into the field: `eyJhbGciOiJIUzUxMiJ9eyJpc3VljoYWRTaW5AZXhhbXBsZS5jb20lLCJpYXQiOjE3NjAwOTkYNjAsInV4cI6MTc2MDcvnDA2MH0.XzFx9MTEvmLSeoJVVFzyPOUBXv4idHv7gzQ2X_JZL0rYGgJ1BH3BjayLshnQ1u9zDlq0TfcON5Wddi1cQMSQfg`. The response status is **200 OK**, size is **106 Bytes**, and time is **346 ms**. The response body is a JSON object:

```
1 {  
2   "firstName": "Dummy",  
3   "lastName": "Admin",  
4   "phone": "9999999999",  
5   "email": "admin@example.com",  
6   "addresses": null  
7 }
```

### - Edit data

```
- http://localhost:8080/profile/edit
```

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'Welcome', 'New Request' (which is active), 'localhost:8081/register', and 'localhost:8081/login'. A search bar contains the text 'firstName John lastName.txt'. The main area shows a 'PUT' request to 'http://localhost:8081/profile/edit'. The 'Body' tab is selected, showing JSON content:

```
1 {
2   "address" : {
3     "name": "Home",
4     "areaName": "Krishnanagar",
5     "pincode": 382345,
6     "cityName": "Ahmedabad"
7   }
8 }
```

The response status is '200 OK', size is '29 Bytes', and time is '28 ms'. The response body is: 'Profile updated successfully.'

## - Change Password

```
- http://localhost:8080/profile/change-password
```

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'Welcome', 'localhost:8081/register', 'localhost:8081/login' (which is active), and 'New Request'. A search bar contains the text 'firstName John lastName.txt'. The main area shows a 'PUT' request to 'http://localhost:8081/profile/change-password'. The 'Body' tab is selected, showing JSON content:

```
1 {
2   "oldPassword": "admin123",
3   "newPassword": "Mypassword@123",
4   "confirmPassword": "Mypassword@123"
5 }
```

The response status is '200 OK', size is '30 Bytes', and time is '157 ms'. The response body is: 'Password changed successfully.'

Oct 10 19:23

File Edit Selection View Go Run ... ← → Notes

Welcome localhost:8081/register localhost:8081/login New Request x firstName John lastName.txt

PUT http://localhost:8081/profile/change-password Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "oldPassword": "admin123",
3   "newPassword": "Mypassword123",
4   "confirmPassword": "Mypassword123"
5 }
```

Status: 400 Bad Request Size: 193 Bytes Time: 96 ms

Response Headers 11 Cookies Results Docs

```
1 {
2   "status": 400,
3   "error": "Bad Request",
4   "message": "Password must be at least 8 characters long, contain
5     at least one uppercase letter, one lowercase letter, one
       number, and one special character."
```

Copy Response Chart ↴ Signed out Go Live

Oct 10 19:24

File Edit Selection View Go Run ... ← → Notes

Welcome localhost:8081/register localhost:8081/login New Request x firstName John lastName.txt

PUT http://localhost:8081/profile/change-password Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "oldPassword": "Mypassword@123",
3   "newPassword": "Mypassword@123",
4   "confirmPassword": "Mypassword@123"
5 }
```

Status: 400 Bad Request Size: 98 Bytes Time: 119 ms

Response Headers 11 Cookies Results Docs

```
1 {
2   "status": 400,
3   "error": "Bad Request",
4   "message": "New password must be different from old password."
5 }
```

Copy Response Chart ↴ Signed out Go Live