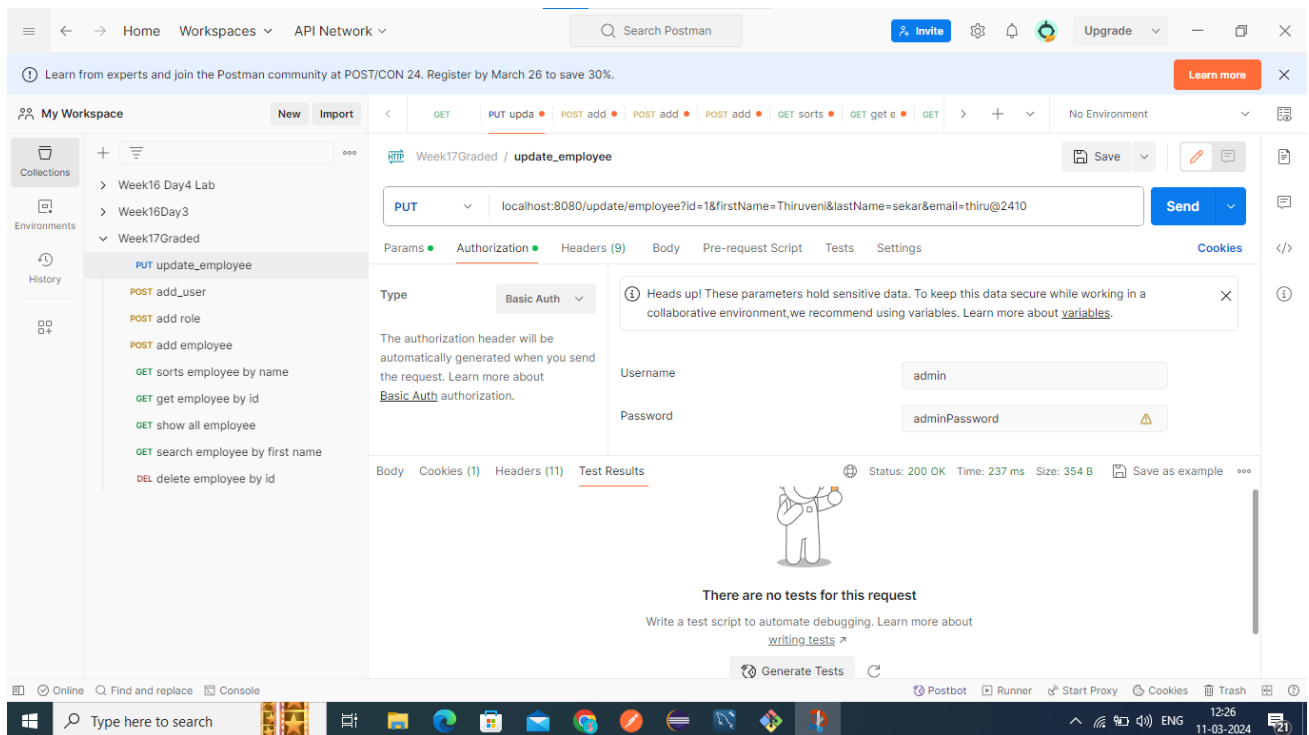


WEEK 17 ASSIGNMENT SCREENSHOTS (ESSVARAN)

- ❖ I have given screen shots for the login with the admin role and login with the user role
 1. First 10 screen shots is of admin role
 2. Last 10 screen shot is of user role

The username = admin , password = adminPassword has role of 'admin'



ENDPOINTS WITH THE ROLE 'ADMIN'

Adding role

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The 'Week17Graded' environment is selected, showing a list of endpoints. The main panel displays a POST request to 'localhost:8080/add/role?id=3&role=manager'. The 'Query Params' section is expanded, showing a table with the following data:

Key	Value	Description
id	3	
role	manager	
Key	Value	Description

The 'Body' tab is selected, showing a status of 200 OK, Time: 277 ms, and Size: 352 B. The response is displayed in the 'Text' view as 'added successfully'.

Adding user

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The 'Week17Graded' environment is selected, showing a list of endpoints. The main panel displays a POST request to 'localhost:8080/add/user?id=3&username=Kodeesh&email=kodesenthil@gmail.com&password=kodee&authorities=user,admin'. The 'Params' section is expanded, showing a table with the following data:

Key	Value	Description
id	3	
username	Kodeesh	
email	kodesenthil@gmail.com	
password	kodee	
authorities	user,admin	
Key	Value	Description

The 'Body' tab is selected, showing a status of 200 OK, Time: 1229 ms, and Size: 344 B. The response is displayed in the 'Text' view as 'user added'.

Adding employee

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The 'Week17Graded' environment is selected, showing a list of API endpoints. The main panel displays a POST request to 'localhost:8080/add/employee?id=1&firstName=Pugal&lastName=sekar&email=pugal53540@gmail.com'. The request body is a JSON object with fields: id (1), firstName (Pugal), lastName (sekar), and email (pugal53540@gmail.com). The response status is 200 OK, and the response body is 'Employee Added successfully'.

POST localhost:8080/add/employee?id=1&firstName=Pugal&lastName=sekar&email=pugal53540@gmail.com

Params

Key	Value	Description
id	1	
firstName	Pugal	
lastName	sekar	
email	pugal53540@gmail.com	

Body

1 Employee Added successfully

Delete employee by ID

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The 'Week17Graded' environment is selected, showing a list of API endpoints. The main panel displays a DELETE request to 'localhost:8080/delete-emp-by-id?id=2'. The response status is 200 OK, and the response body is 'deleted successfully'.

DELETE localhost:8080/delete-emp-by-id?id=2

Query Params

Key	Value	Description
id	2	

Body

1 deleted successfully

Get employee by id

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The main panel displays a GET request to 'localhost:8080/show/employee-by-id?id=1'. The 'Query Params' table shows 'id' with value '1'. The response is a JSON object with employee details.

Key	Value	Description
id	1	

```
{
  "id": 1,
  "firstName": "Thiruvani",
  "lastName": "sekar",
  "email": "thiru@2410"
}
```

Status: 200 OK Time: 259 ms Size: 406 B

Search employee by first name

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections and environments. The main panel displays a GET request to 'localhost:8080/search/employee-by-firstname?name=Thiru'. The 'Query Params' table shows 'name' with value 'Thiru'. The response is a JSON object with employee details.

Key	Value	Description
name	Thiru	

```
{
  "id": 1,
  "firstName": "Thiruvani",
  "lastName": "sekar",
  "email": "thiru@2410"
}
```

Status: 200 OK Time: 318 ms Size: 408 B

Show all employee

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists several collections, with 'Week17Graded' expanded to show a list of API endpoints. The main panel displays a GET request to 'localhost:8080/show-all/employee'. The response is a JSON array of two employee objects, sorted by ID. The status is 200 OK, and the response size is 481 B.

```
1 {
2   {
3     "id": 1,
4     "firstName": "Thiruveni",
5     "lastName": "sekar",
6     "email": "thiru@2410"
7   },
8   {
9     "id": 2,
10    "firstName": "essvaran",
11    "lastName": "sekar",
12    "email": "essvar@2410"
13  }
14 }
```

Sort employee by first name:

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists several collections, with 'Week17Graded' expanded to show a list of API endpoints. The main panel displays a GET request to 'localhost:8080/sort/employee-by-name?dir=asc'. The response is a JSON array of two employee objects, sorted by first name. The status is 200 OK, and the response size is 481 B.

```
1 {
2   {
3     "id": 2,
4     "firstName": "essvaran",
5     "lastName": "sekar",
6     "email": "essvar@2410"
7   },
8   {
9     "id": 1,
10    "firstName": "Thiruveni",
11    "lastName": "sekar",
12    "email": "thiru@2410"
13  }
14 }
```

Update Employee

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections under 'Week17Graded', with 'PUT update_employee' selected. The main panel displays the details of this request:

- Method:** PUT
- URL:** localhost:8080/update/employee?id=1&firstName=Thiruveni&lastName=sekar&email=thiru@2410
- Params:** A table with 3 columns: Key, Value, Description. It contains 5 entries: id (1), firstName (Thiruveni), lastName (sekar), email (thiru@2410), and a blank row for Key/Value/Description.
- Body:** The 'Test Results' tab is active, showing a single line: '1 updated successfully'.
- Status:** 200 OK, Time: 237 ms, Size: 354 B.

The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 12:16 on 11-03-2024.

All the endpoints

The screenshot shows the Postman application interface displaying the 'Week17Graded' collection. The left sidebar shows the collection is selected. The main panel shows the collection's documentation:

- Collection Name:** Week17Graded
- Version:** CURRENT
- Language:** JavaScript - Fetch
- Endpoints:** A list of endpoints is shown on the right, including PUT update_employee, POST add_user, POST add role, POST add employee, GET sorts employee by name, GET get employee by id, GET show all employee, GET search employee by first name, and DEL delete employee by id.
- PUT update_employee:** A specific endpoint is highlighted with its URL: localhost:8080/update/employee?id=2&firstName=essvaran&lastName=sekar&email=essvar@2410.

The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 12:12 on 11-03-2024.

ENDPOINTS WITH THE ROLE 'USER'

Username = asraf, password=ali with the role of 'user'

The screenshot shows the Postman interface with a PUT request to `localhost:8080/update/employee?id=1&firstName=Thiruveni&lastName=sekar&email=thiru@2410`. The request is configured with Basic Auth (Username: asraf, Password: ali). The response status is 200 OK, Time: 237 ms, Size: 354 B. The response body is empty, and the headers show 11 items. A message states: "There are no tests for this request".

Can't able to add roles(access denied)

The screenshot shows the Postman interface with a POST request to `localhost:8080/add/role?id=3&role=manager`. The request is configured with Basic Auth (Username: asraf, Password: ali). The response status is 403 Forbidden, Time: 177 ms, Size: 12.03 KB. The response body is a JSON object with the following structure:

```
1 {
2   "timestamp": "2024-03-10T19:01:46.834+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\\r\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.attemptAuthorization (AuthorizationManagerBeforeMethodInterceptor.java:266)\\r\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke(AuthorizationManagerBeforeMethodInterceptor..."
```

Can't able to add user (access denied)

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Week17Graded" collection is selected, and the "add_user" endpoint is being tested. The request is a POST to `localhost:8080/add/user?id=4&username=Asraf&email=asrafi@gmail.com&password=ali&authorities=user`. The parameters are:

Key	Value	Description
id	4	
username	Asraf	
email	asrafi@gmail.com	
password	ali	
authorities	user	

The response status is 403 Forbidden. The response body is:

```
{
  "timestamp": "2024-03-10T19:01:05.872+00:00",
  "status": 403,
  "error": "Forbidden",
  "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\\n\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.attemptAuthorization\\n\\n\\tat org.springframework.security.authorization.AuthorizationManagerBeforeMethodInterceptor.java:256\\n\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke(AuthorizationManagerBeforeMethodInterceptor..."
```

Can't able to add employee (access denied)

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Week17Graded" collection is selected, and the "add employee" endpoint is being tested. The request is a POST to `localhost:8080/add/employee?id=3&firstName=Preeti&lastName=Das&email=pd@gmail.com`. The parameters are:

Key	Value	Description
id	3	
firstName	Preeti	
lastName	Das	
email	pd@gmail.com	

The response status is 403 Forbidden. The response body is:

```
{
  "timestamp": "2024-03-10T19:02:29.739+00:00",
  "status": 403,
  "error": "Forbidden",
  "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\\n\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.attemptAuthorization\\n\\n\\tat org.springframework.security.authorization.AuthorizationManagerBeforeMethodInterceptor.java:256\\n\\n\\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke(AuthorizationManagerBeforeMethodInterceptor..."
```


Can't able to delete employee (access denied)

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections and environments. The main area displays a DELETE request to "localhost:8080/delete-emp-by-id?id=3". The request is configured with a query parameter "id" with value "3". The response is a JSON object indicating a 403 Forbidden status.

Request:

- Method: DELETE
- URL: localhost:8080/delete-emp-by-id?id=3
- Query Params: id=3

Response:

```
1 {
2   "timestamp": "2024-03-10T19:06:52.906+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\r\n\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.attemptAuthorization (AuthorizationManagerBeforeMethodInterceptor.java:256)\r\n\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke (AuthorizationManagerBeforeMethodInterceptor.java:256)"
}
```

Can't able to update employee (access denied)

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections and environments. The main area displays a PUT request to "localhost:8080/update/employee?id=1&firstName=Thiruveni&lastName=sekar&email=thiru@2410". The request is configured with query parameters "id", "firstName", "lastName", and "email". The response is a JSON object indicating a 403 Forbidden status.

Request:

- Method: PUT
- URL: localhost:8080/update/employee?id=1&firstName=Thiruveni&lastName=sekar&email=thiru@2410
- Query Params: id=1, firstName=Thiruveni, lastName=sekar, email=thiru@2410

Response:

```
1 {
2   "timestamp": "2024-03-10T19:00:04.430+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\r\n\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.attemptAuthorization (AuthorizationManagerBeforeMethodInterceptor.java:256)\r\n\tat org.springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke (AuthorizationManagerBeforeMethodInterceptor.java:256)"
}
```

Get Employee by id

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. [Learn more](#)

My Workspace | New | Import | GET | PUT update_employee | POST add_user | POST add role | POST add employee | GET sorts employee by name | **GET get employee by id** | GET show all employee | GET search employee by first name | DEL delete employee by id

Week17Graded / **get employee by id** | Save | Edit | Copy

GET | localhost:8080/show/employee-by-id?id=2 | Send

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

Query Params

Key	Value	Description
<input checked="" type="checkbox"/> id	2	
Key	Value	Description

Body | Cookies (1) | Headers (11) | Test Results | Status: 200 OK | Time: 215 ms | Size: 405 B | Save as example

Pretty | Raw | Preview | Visualize | JSON | Copy

```
1 {
2   "id": 2,
3   "firstName": "Loga",
4   "lastName": "Priya",
5   "email": "loga@gmail.com"
6 }
```

Online | Find and replace | Console | Postbot | Runner | Start Proxy | Cookies | Trash | 12:34 11-03-2024

Search by first name

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. [Learn more](#)

My Workspace | New | Import | GET sorts employee by name | GET get employee by id | GET show all employee | **GET search employee by first name** | DEL delete employee by id

Week17Graded / **search employee by first name** | Save | Edit | Copy

GET | localhost:8080/search/employee-by-firstname?name=Loga | Send

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

Query Params

Key	Value	Description
<input checked="" type="checkbox"/> name	Loga	
Key	Value	Description

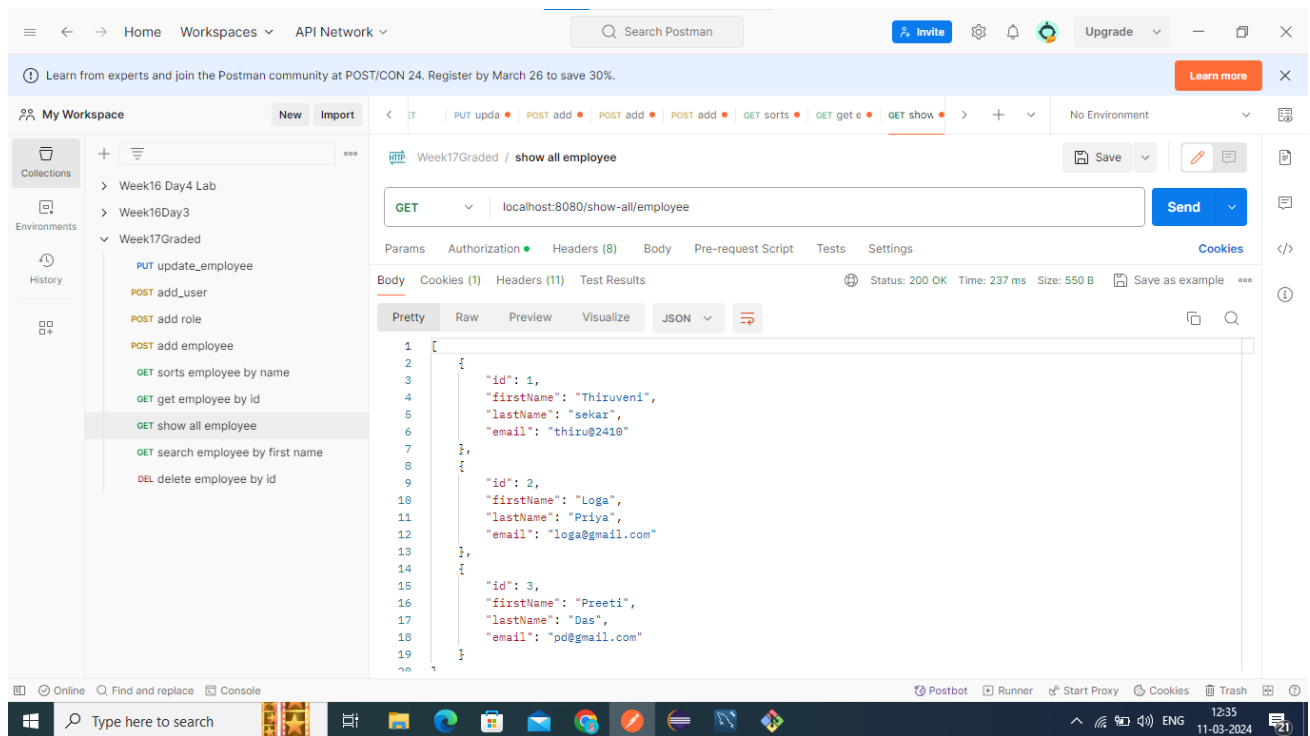
Body | Cookies (1) | Headers (11) | Test Results | Status: 200 OK | Time: 198 ms | Size: 407 B | Save as example

Pretty | Raw | Preview | Visualize | JSON | Copy

```
1 {
2   "id": 2,
3   "firstName": "Loga",
4   "lastName": "Priya",
5   "email": "loga@gmail.com"
6 }
```

Online | Find and replace | Console | Postbot | Runner | Start Proxy | Cookies | Trash | 12:36 11-03-2024

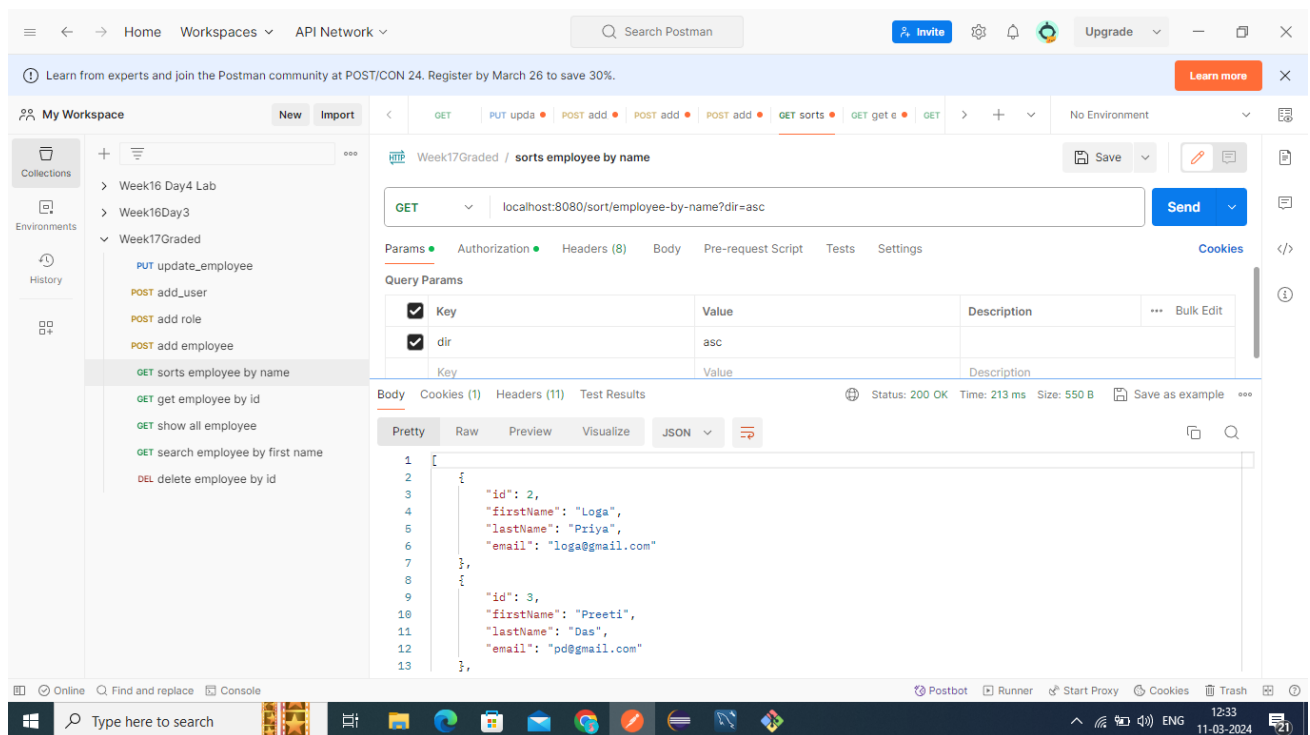
Show all employee



Postman interface showing the 'show all employee' endpoint. The URL is `localhost:8080/show-all/employee`. The response is a JSON array of three employee objects.

```
1 {
2   {
3     "id": 1,
4     "firstName": "Thiruvani",
5     "lastName": "sekar",
6     "email": "thiru@2410"
7   },
8   {
9     "id": 2,
10    "firstName": "Loga",
11    "lastName": "Priya",
12    "email": "loga@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Preeti",
17    "lastName": "Das",
18    "email": "pd@gmail.com"
19  }
20 }
```

Sort by name



Postman interface showing the 'sorts employee by name' endpoint. The URL is `localhost:8080/sort/employee-by-name?dir=asc`. The response is a JSON array of two employee objects, sorted by name.

Key	Value	Description
<input checked="" type="checkbox"/>	dir	asc

```
1 {
2   {
3     "id": 2,
4     "firstName": "Loga",
5     "lastName": "Priya",
6     "email": "loga@gmail.com"
7   },
8   {
9     "id": 3,
10    "firstName": "Preeti",
11    "lastName": "Das",
12    "email": "pd@gmail.com"
13  }
14 }
```

Thank you...