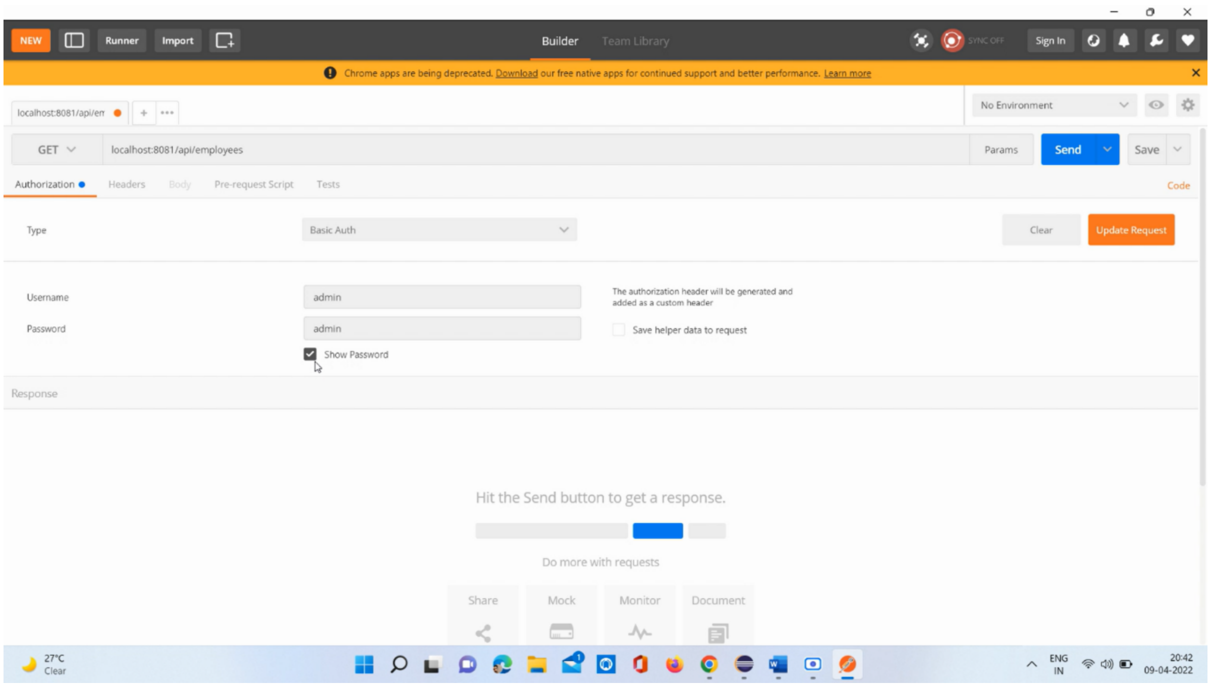
Snapshots Graded Assignment

The Screenshot below shows that all the api's has a security layer as we have to insert basic authentication i.e. username and password to execute the apis.



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

GET localhost:8081/api/employees

Type: Basic Auth

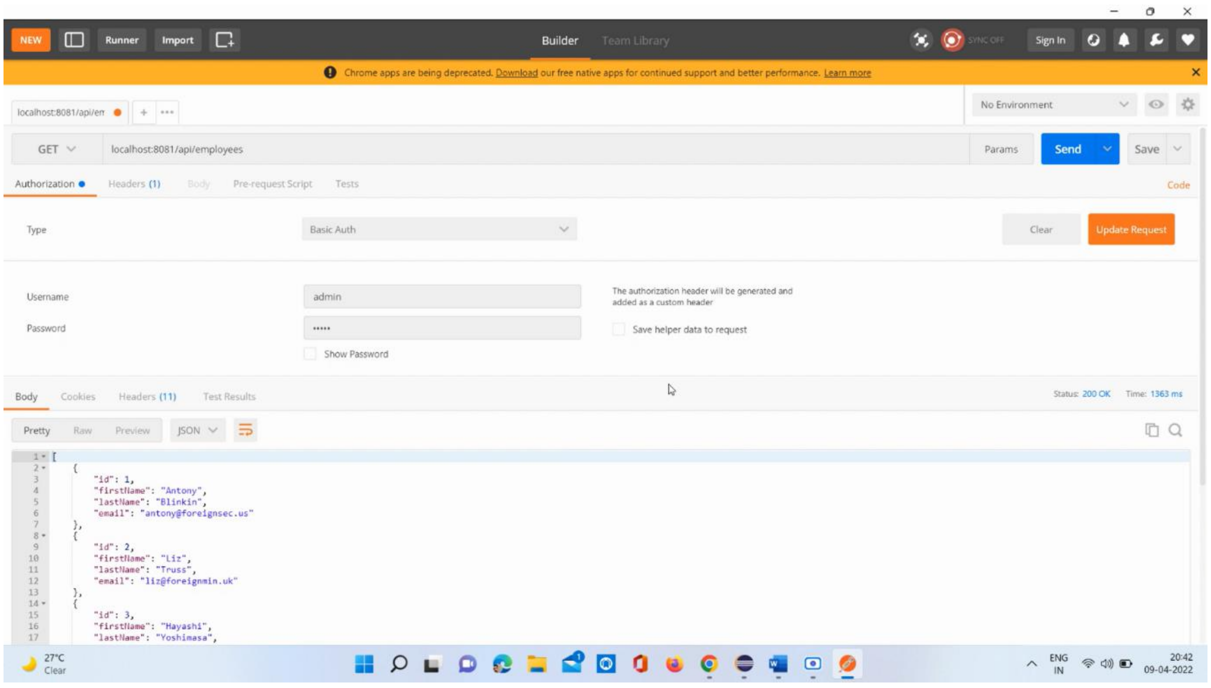
Username: admin

Password: admin

The authorization header will be generated and added as a custom header

Hit the Send button to get a response.

After input of username and password we were able to execute the "get" request to get the list of employees as shown below.



GET localhost:8081/api/employees

Type: Basic Auth

Username: admin

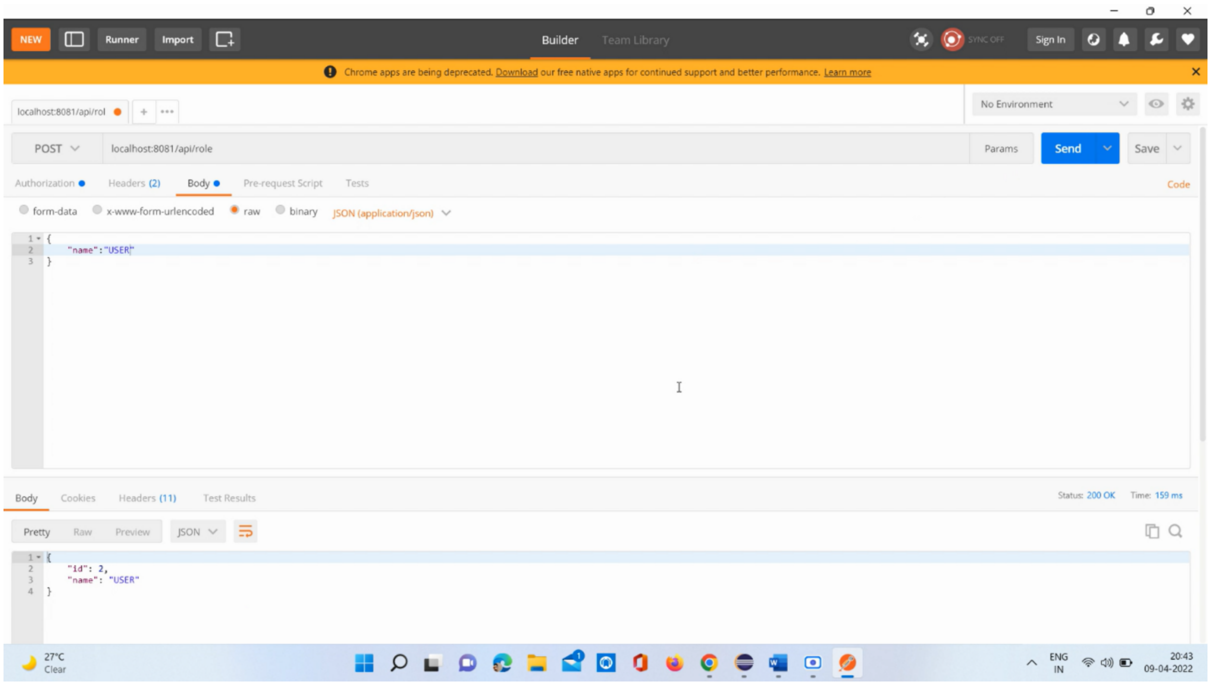
Password: •••••

Status: 200 OK Time: 1363 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Liz",
        "lastName": "Truss",
        "email": "liz@foreignmin.uk"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
```

SOLUTION Q1:

Now, let's start with the assignment solution, The first question was regarding adding a role dynamically. That can be seen as below:



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

POST localhost:8081/api/role

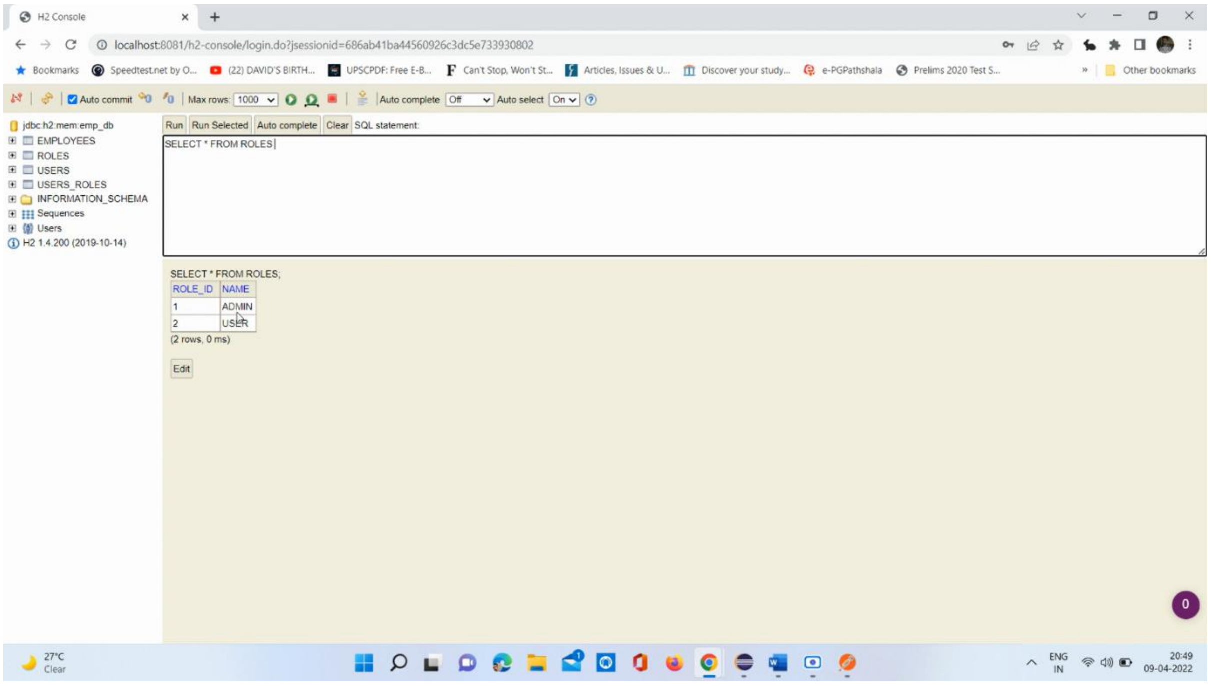
form-data x-www-form-urlencoded [✓ raw] binary JSON (application/json)

```
{
    "name":"USER"
}
```

Status: 200 OK Time: 159 ms

```
{
    "id": 2,
    "name": "USER"
}
```

Upon executing the "post request" we were able to add the role in roles table which can also be seen in the embedded database h2 as below:



localhost:8081/h2-console/login.do?jsessionid=686ab41ba44560926c3dc5e733930802

jdbc:h2:mem:emp_db
EMPLOYEES
ROLES
USERS
USERS_ROLES
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.200 (2019-10-14)

```
SELECT * FROM ROLES
```

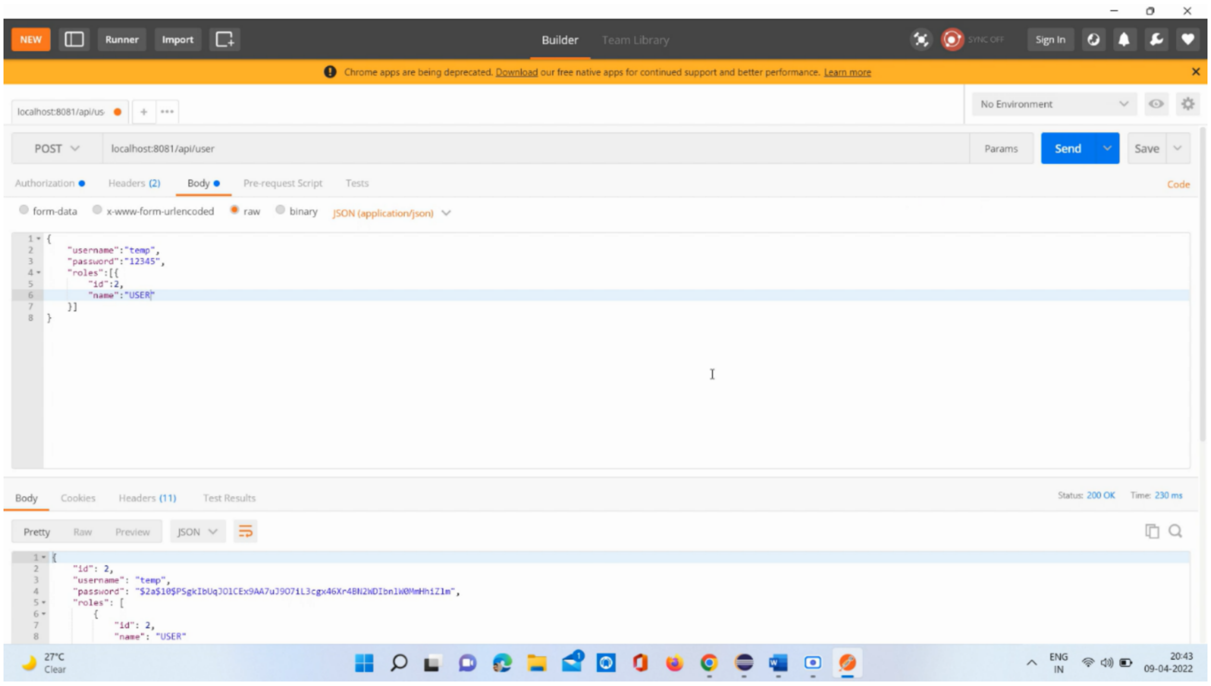
SELECT * FROM ROLES;

ROLE_ID	NAME
1	ADMIN
2	USER

(2 rows, 0 ms)

SOLUTION Q2:

Now, let’s add a user dynamically so that it can later be authenticated to execute the apis. This can be seen as:


POST localhost:8081/api/user

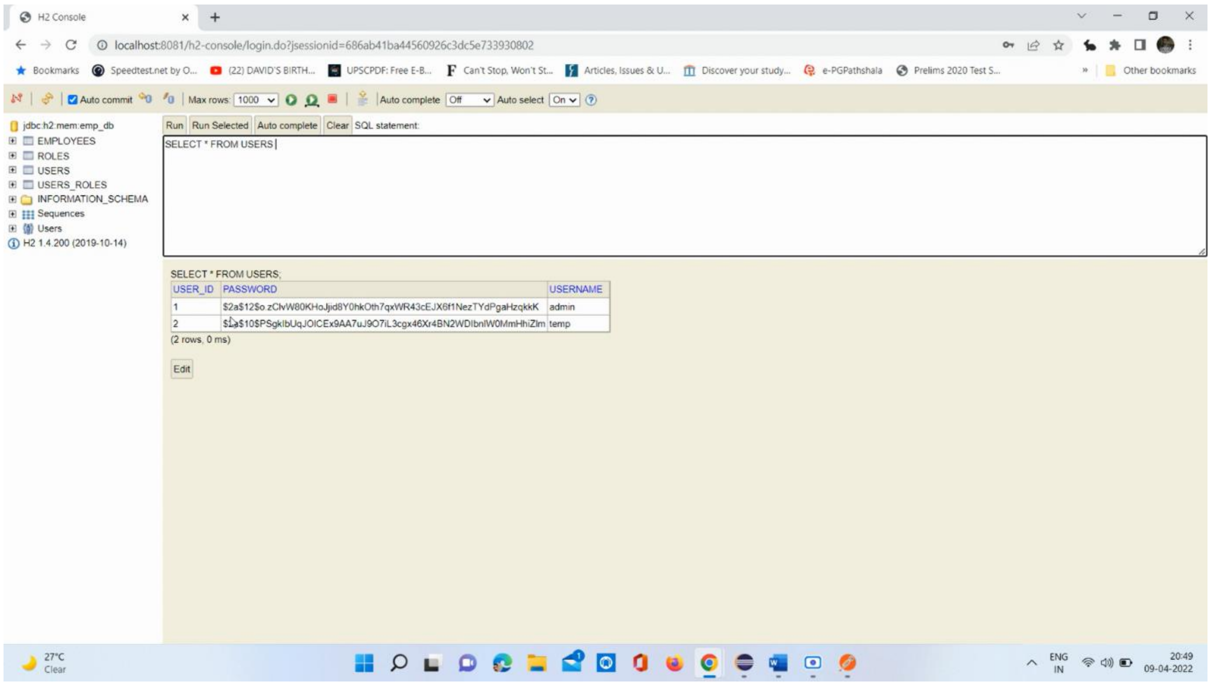
Body: raw, JSON (application/json)

```
{
    "username":"temp",
    "password":"12345",
    "roles":[{
        "id":2,
        "name":"USER"
    }]
}
```

Status: 200 OK Time: 230 ms

```
{
    "id": 2,
    "username": "temp",
    "password": "$2a$10$PSgkIbUqJOlCEx9AA7uJ9O7iL3cgx46Xr4BN2WDIbnlW0MmHhiZlm",
    "roles": [
        {
            "id": 2,
            "name": "USER"
```

To check whether the user was added dynamically or not as well as linked to the join table of users and roles, below screen shot from embedded h2 explains it very well:


localhost:8081/h2-console/login.do?jsessionid=686ab41ba44560926c3dc5e733930802

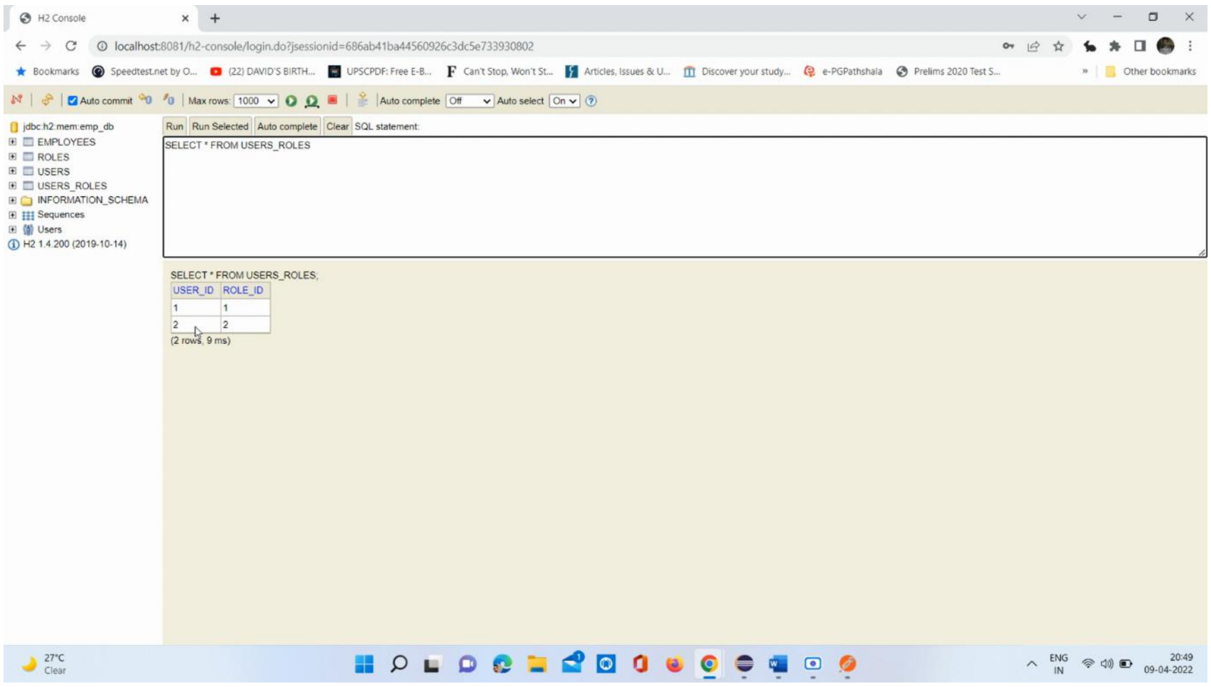
jdbc:h2:mem:emp_db
- EMPLOYEES
- ROLES
- USERS
- USERS_ROLES
- INFORMATION_SCHEMA
- Sequences
- Users
- H2 1.4.200 (2019-10-14)

```
SELECT * FROM USERS
```

SELECT * FROM USERS;

USER_ID	PASSWORD	USERNAME
1	$2a$12$o.zClvW80KHoJjid8Y0hkOth7qxWR43cEJX6f1NezTYdPgaHzqkkK	admin
2	$2a$10$PSgkIbUqJOICEx9AA7uJ9O7iL3cgx46Xr4BN2WDIbnIW0MmHhiZlm	temp

(2 rows, 0 ms)


jdbc:h2:mem:emp_db

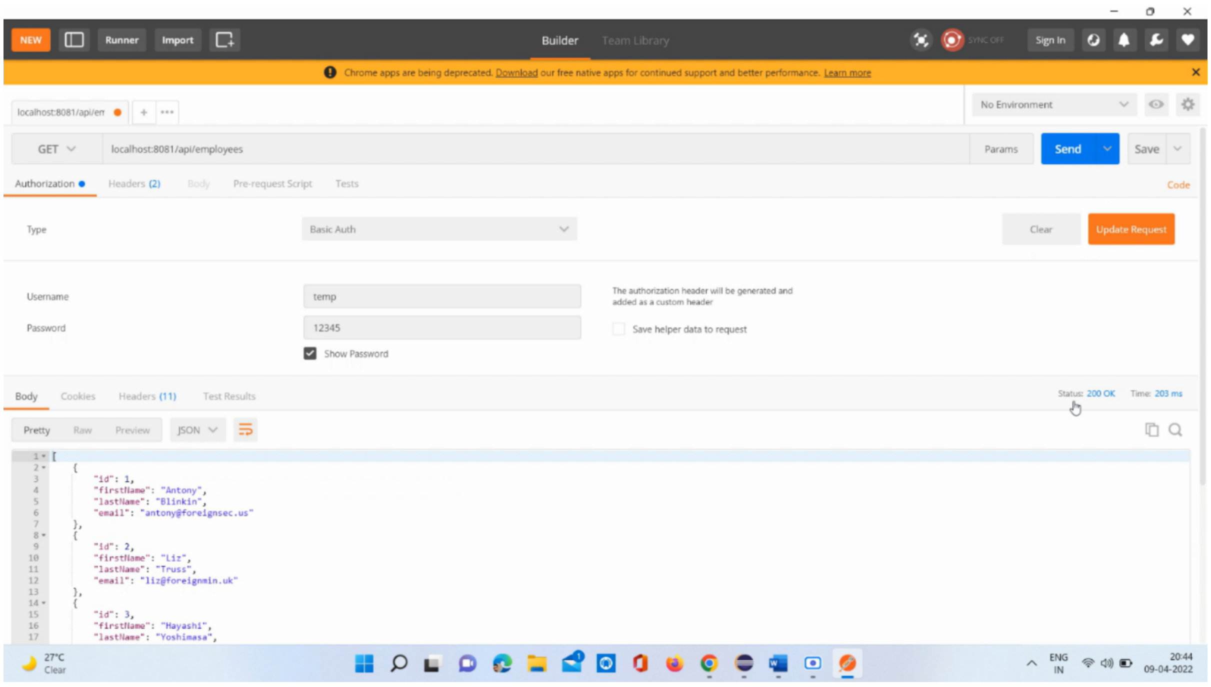
```
SELECT * FROM USERS_ROLES
```

SELECT * FROM USERS_ROLES;

USER_ID	ROLE_ID
1	1
2	2

(2 rows, 9 ms)

Let’s now try and execute the “get” request using the newly added user.


GET localhost:8081/api/employees

Type: Basic Auth

Username: temp

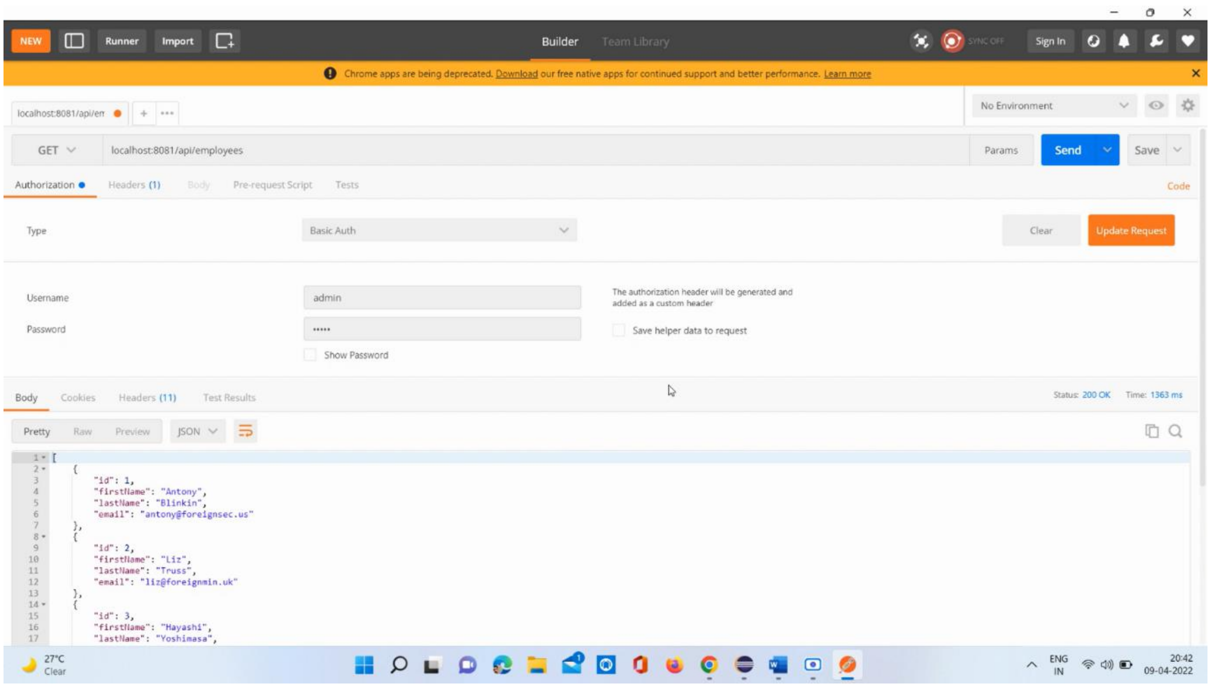
Password: 12345

Status: 200 OK Time: 203 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Liz",
        "lastName": "Truss",
        "email": "liz@foreignmin.uk"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
```

As you can see with inserting the username and password of the added user role we can get the list of all employees. Now the assignment mentions that only “ADMIN” can execute the below api requests.

Okay, so the above screenshots fulfil the 1st and 2nd requirement of the assignment. Onto next, The below screenshot is the working of "get" api to get the list of employees using the link localhost:8081/api/employees



GET localhost:8081/api/employees

Type: Basic Auth

Username: admin

Password: •••••

The authorization header will be generated and added as a custom header

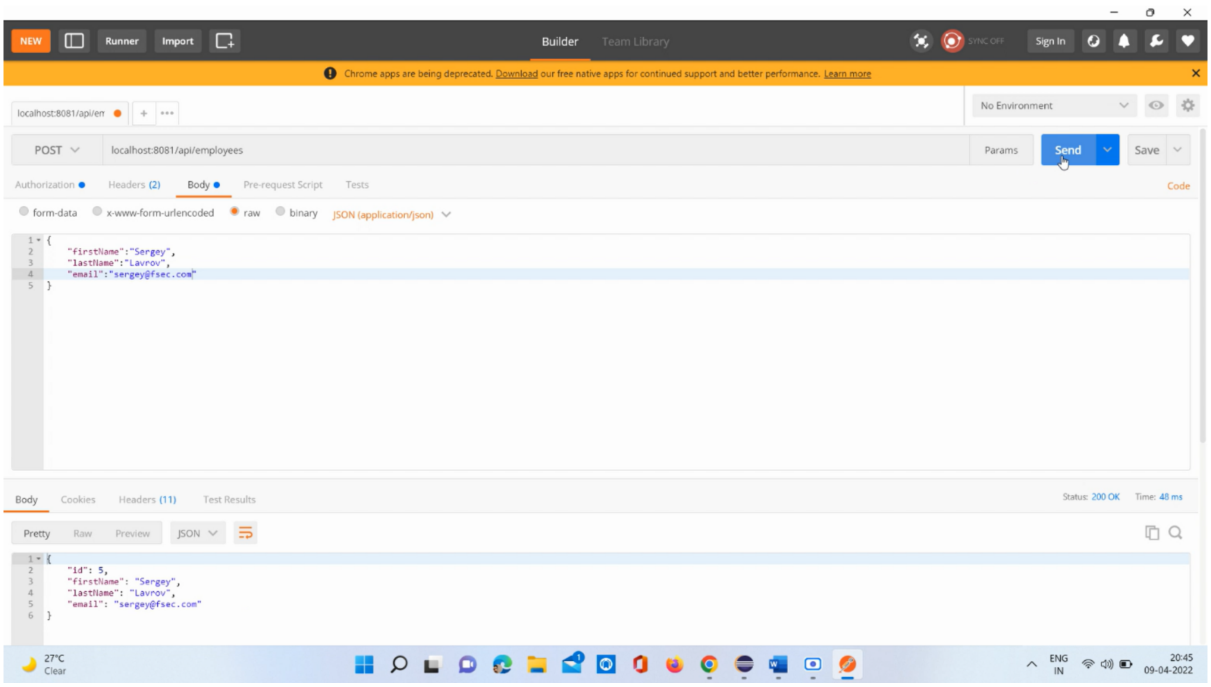
Save helper data to request

Status: 200 OK Time: 1363 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Liz",
        "lastName": "Truss",
        "email": "liz@foreignmin.uk"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
```

SOLUTION Q3:

Now, let's add a employee using "Post" Request and in this we need to add the body containing the fields of the employee.



POST localhost:8081/api/employees

raw — JSON (application/json)

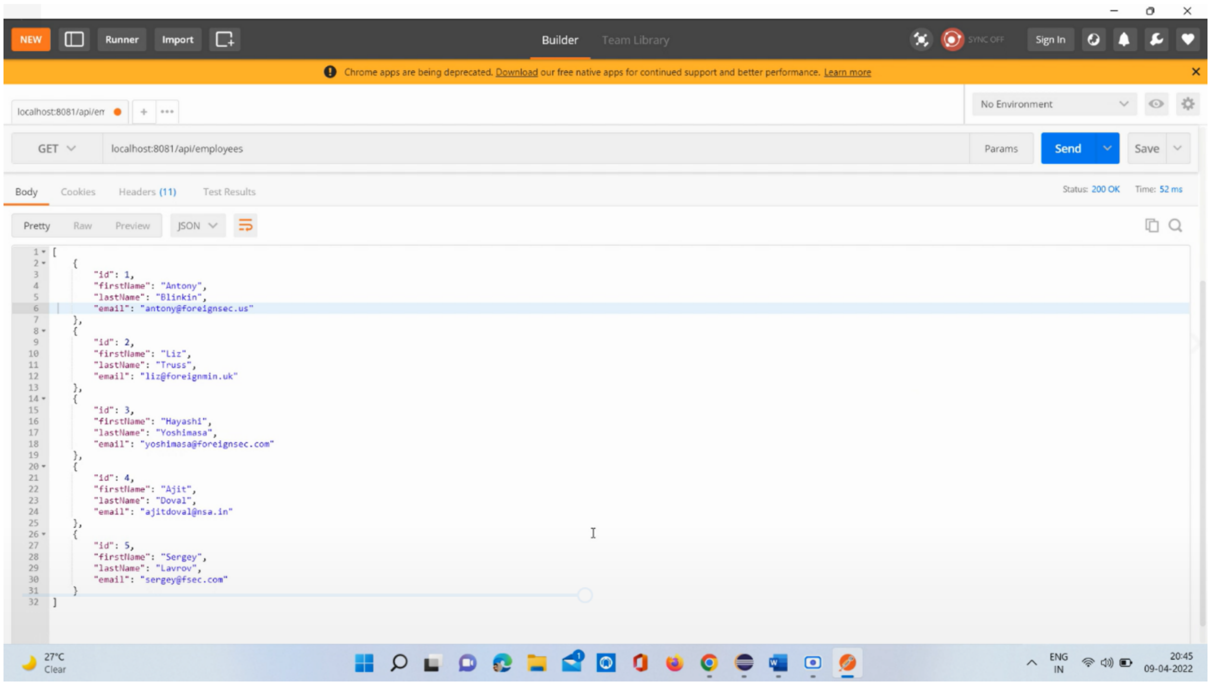
```
{
    "firstName":"Sergey",
    "lastName":"Lavrov",
    "email":"sergey@fsec.com"
}
```

Status: 200 OK Time: 48 ms

```
{
    "id": 5,
    "firstName": "Sergey",
    "lastName": "Lavrov",
    "email": "sergey@fsec.com"
}
```

SOLUTION Q4:

As you can see we have added a new employee with id = 5. Here’s a look at the updated list along with getting all the employees in the list

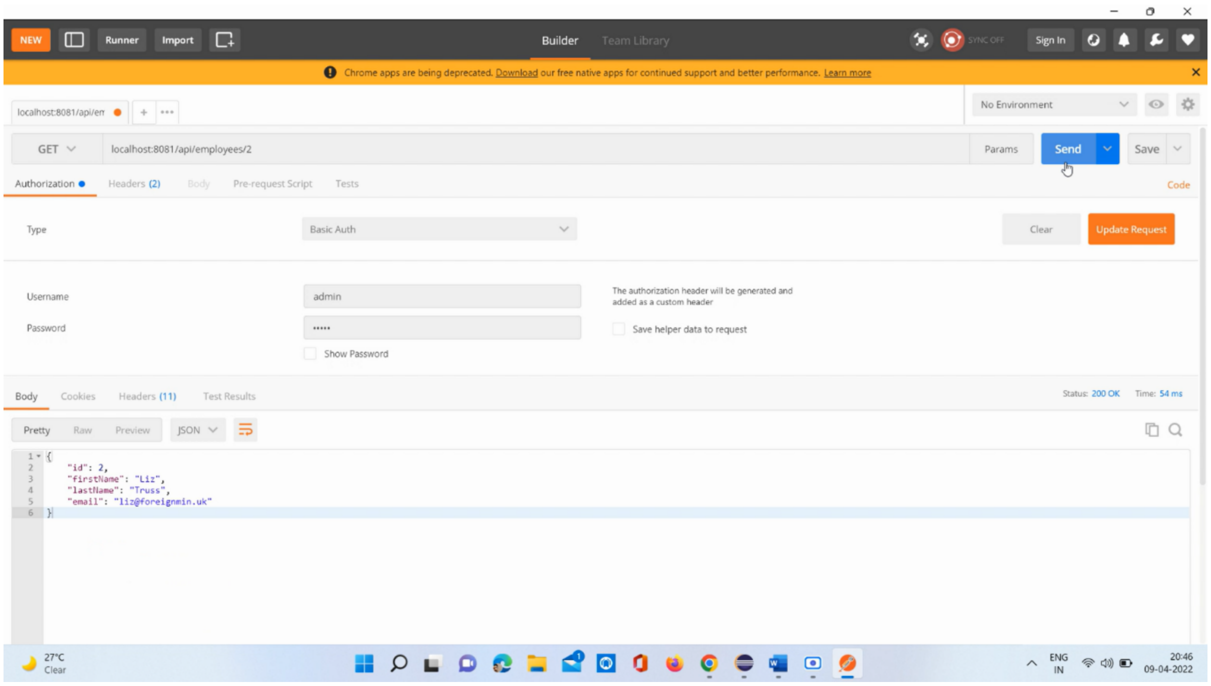

GET localhost:8081/api/employees

Status: 200 OK Time: 52 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Liz",
        "lastName": "Truss",
        "email": "liz@foreignmin.uk"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    },
    {
        "id": 5,
        "firstName": "Sergey",
        "lastName": "Lavrov",
        "email": "sergey@fsec.com"
    }
]
```

SOLUTION Q5:

Now it was required to get employee by id, the below screenshot provides it:


GET localhost:8081/api/employees/2

Type: Basic Auth

Username: admin

Password: •••••

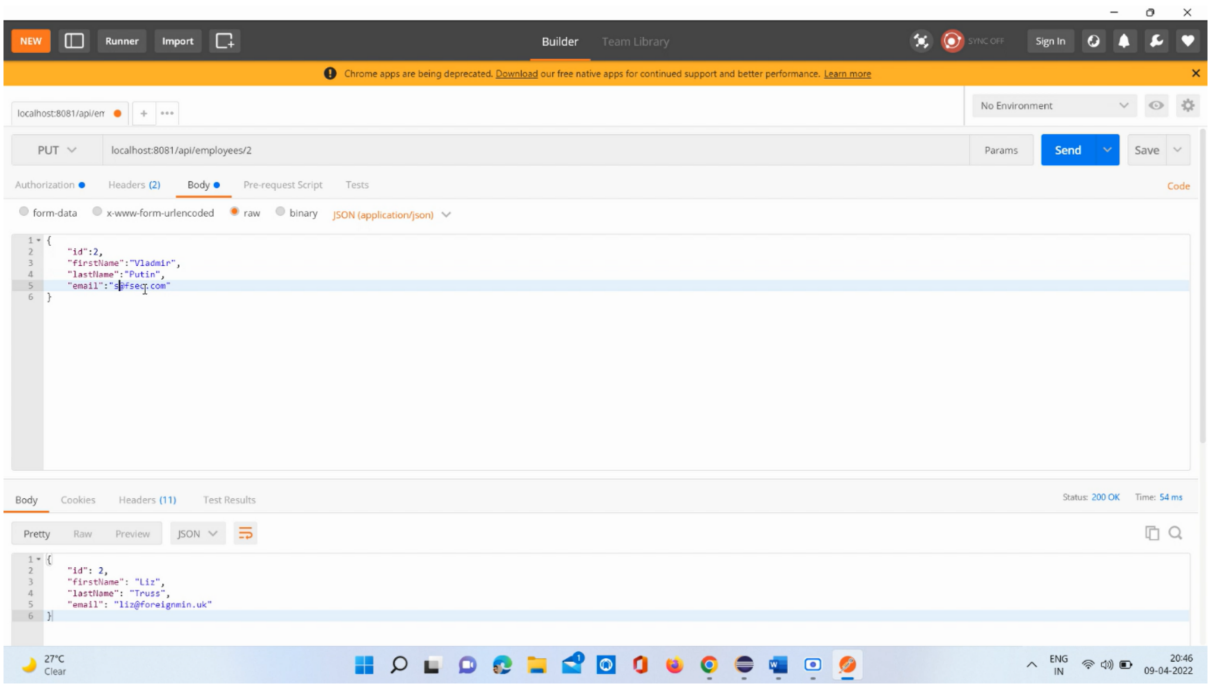
Status: 200 OK Time: 54 ms

```
{
    "id": 2,
    "firstName": "Liz",
    "lastName": "Truss",
    "email": "liz@foreignmin.uk"
}
```

In the “get” request we wanted employee with id = 2 and that works fine.

SOLUTION Q6:

Now, let’s update an existing employee was the next requirement.


PUT localhost:8081/api/employees/2

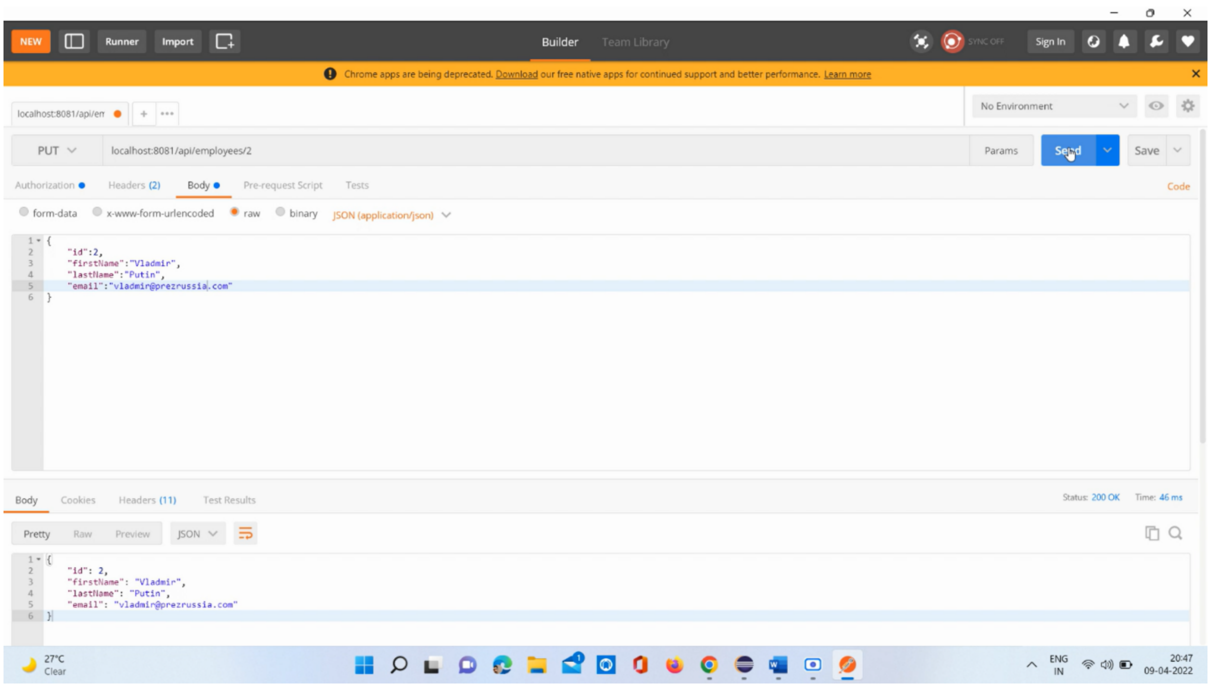
Body: raw, JSON (application/json)

```
{
    "id":2,
    "firstName":"Vladmir",
    "lastName":"Putin",
    "email":"s@fsec.com"
}
```

Status: 200 OK Time: 54 ms

```
{
    "id": 2,
    "firstName": "Liz",
    "lastName": "Truss",
    "email": "liz@foreignmin.uk"
}
```

As you can see in the response body the first name, last name and email of the employee with id = 2 and we are updating the employee details with employee id = 2. The below screenshot is the after the execution of updating the employee detail.


PUT localhost:8081/api/employees/2

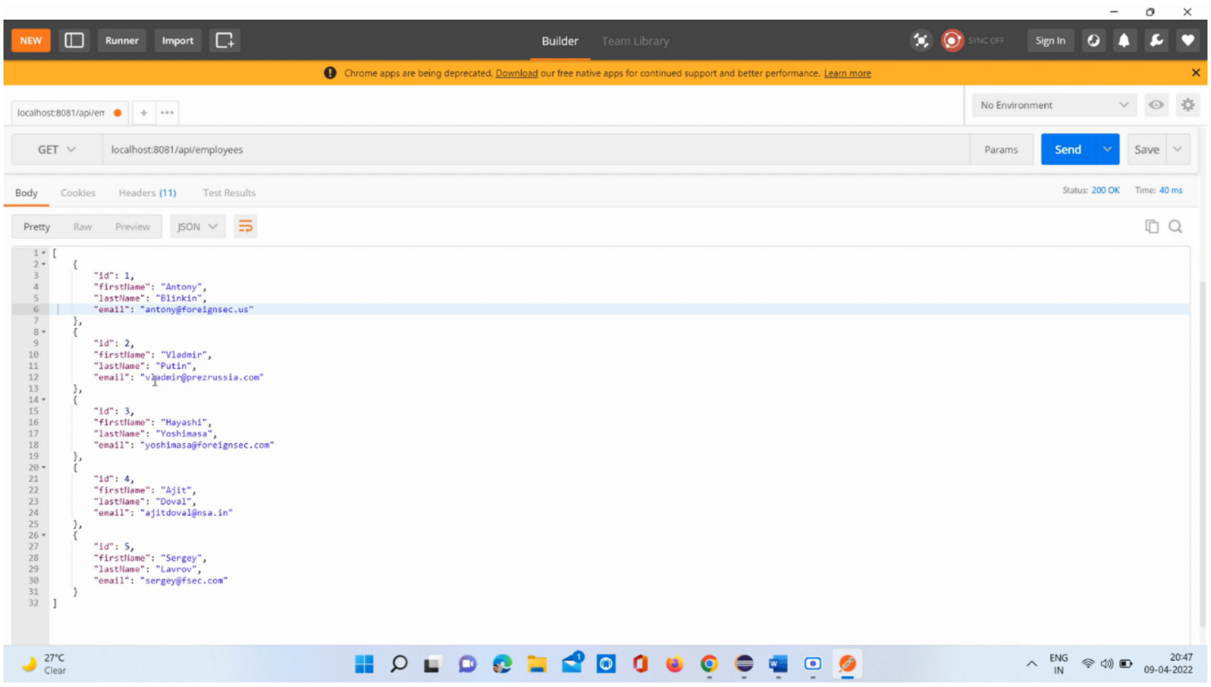
Body: raw, JSON (application/json)

```
{
    "id":2,
    "firstName":"Vladmir",
    "lastName":"Putin",
    "email":"vladmir@prezrussia.com"
}
```

Status: 200 OK Time: 46 ms

```
{
    "id": 2,
    "firstName": "Vladmir",
    "lastName": "Putin",
    "email": "vladmir@prezrussia.com"
}
```

Below is the updated list of employees



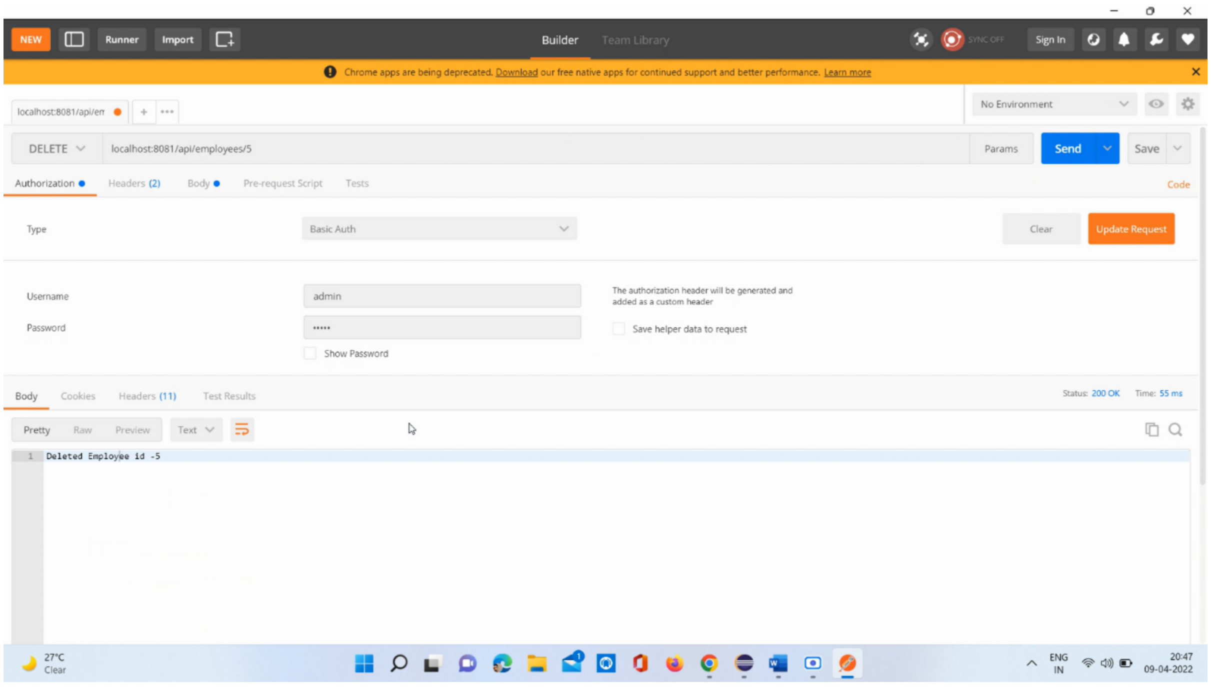
GET localhost:8081/api/employees

Status: 200 OK Time: 40 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    },
    {
        "id": 5,
        "firstName": "Sergey",
        "lastName": "Lavrov",
        "email": "sergey@fsec.com"
    }
]
```

SOLUTION Q7:

Now, let’s perform the delete operation which would delete an employee from the list



DELETE localhost:8081/api/employees/5

Type: Basic Auth

Username: admin

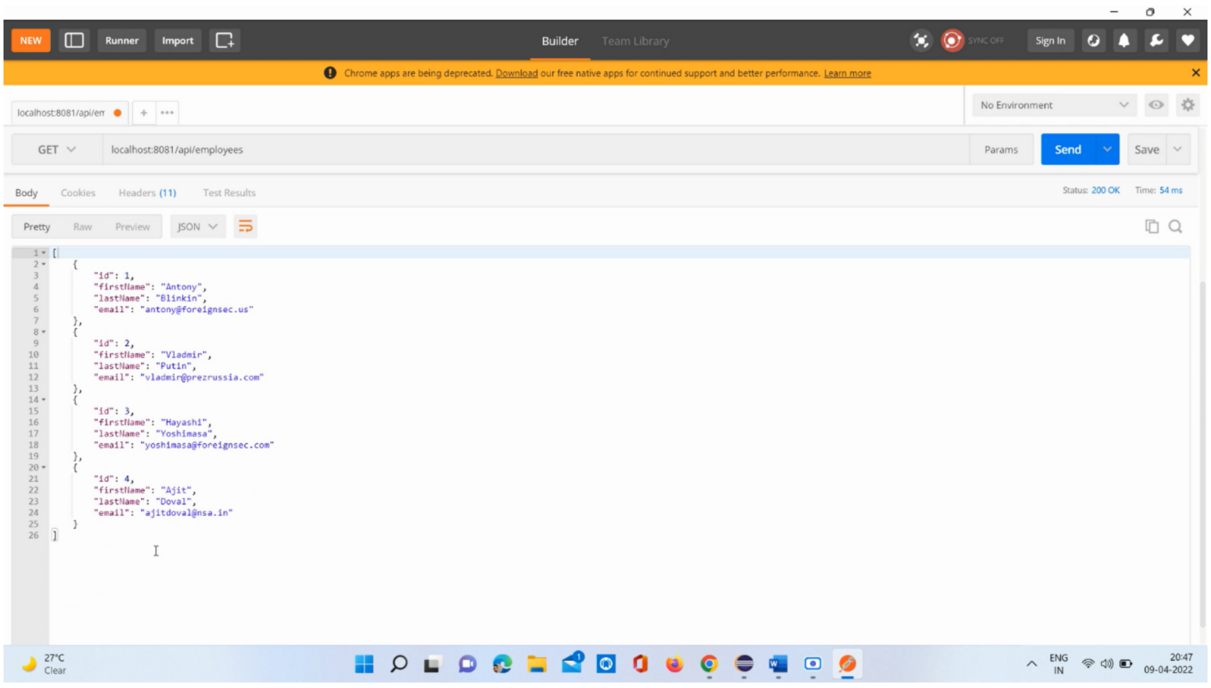
Password: •••••

Status: 200 OK Time: 55 ms

```
Deleted Employee id -5
```

As you can see after executing the api request we got a response message like “deleted employee id =5”, So let’s check with the list of employees whether it was deleted or not.

Yes, it was deleted as the updated list of employees below doesn’t show employee with id =5 as earlier list of employees did.

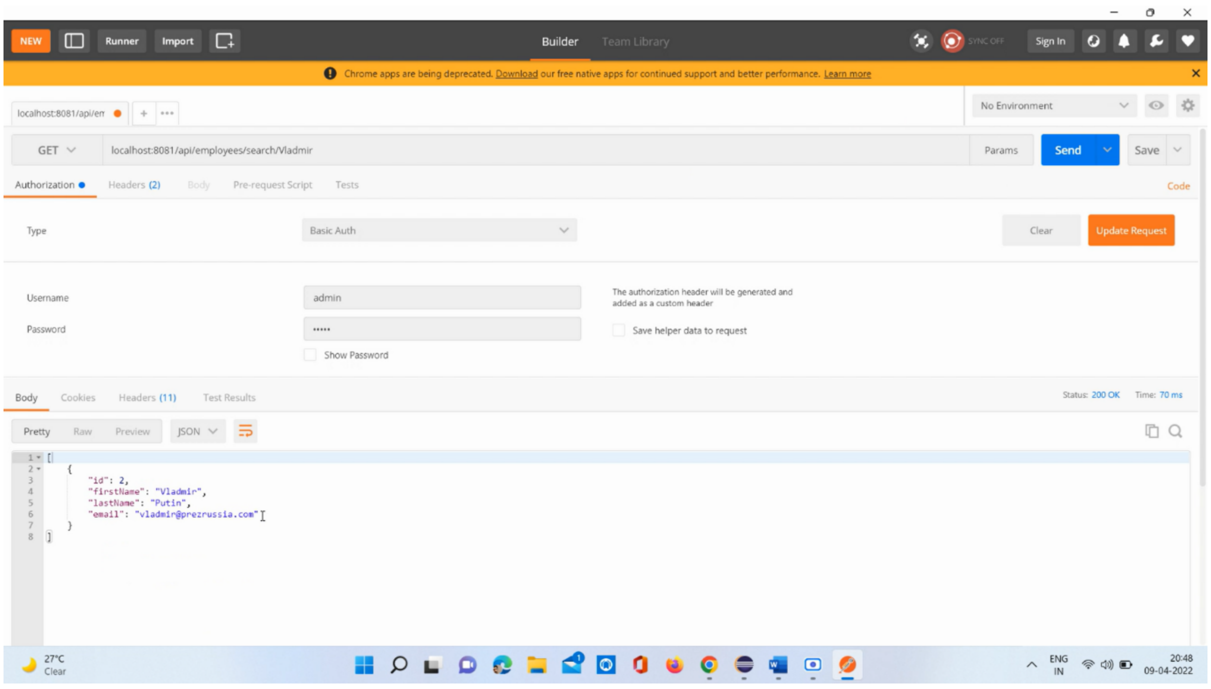

GET localhost:8081/api/employees

Status: 200 OK Time: 54 ms

```
[
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    }
]
```

SOLUTION Q8:

Now, the next requirement was to perform search operation using first name, so below screenshot proves that this api is also working.


GET localhost:8081/api/employees/search/Vladmir

Type: Basic Auth

Username: admin

Password: •••••

The authorization header will be generated and added as a custom header

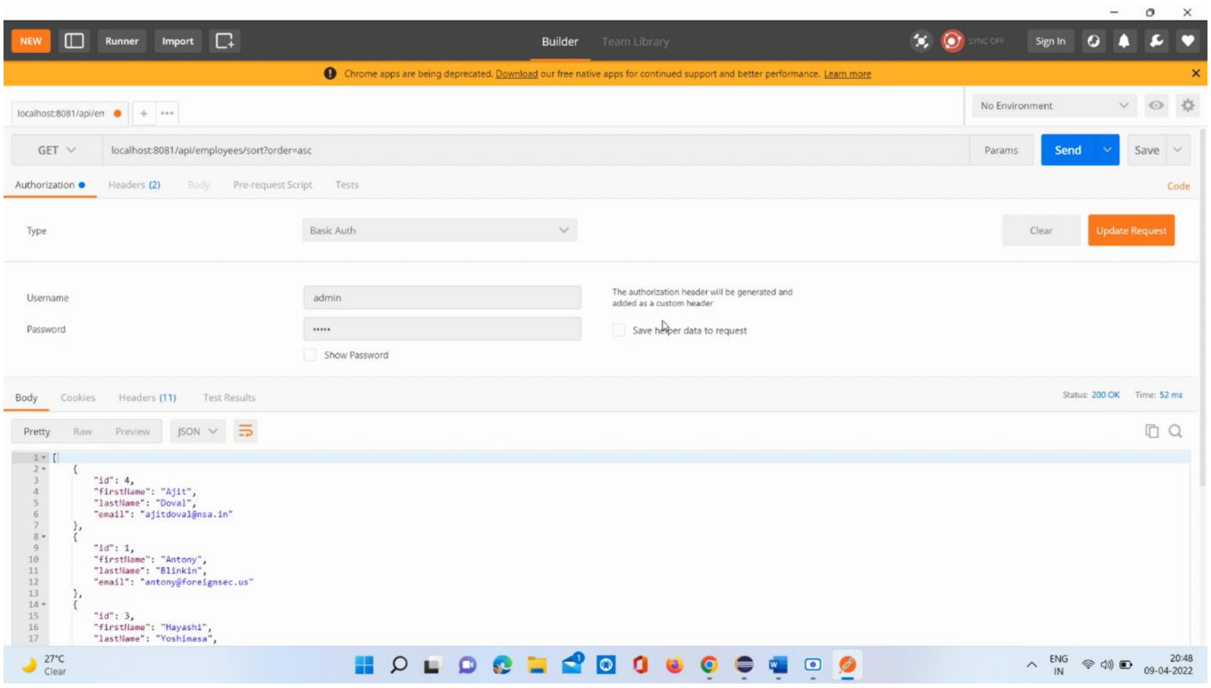
Status: 200 OK Time: 70 ms

```
[
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    }
]
```

Here as you can see in the url localhost:8081/api/employees/search/Vladmir, we are searching the employee with the first name as "Vladmir" and it works fine as seen in the response body.

SOLUTION Q9(I):

Next is to sort the list of employees in ascending order by first name.



GET localhost:8081/api/employees/sort?order=asc

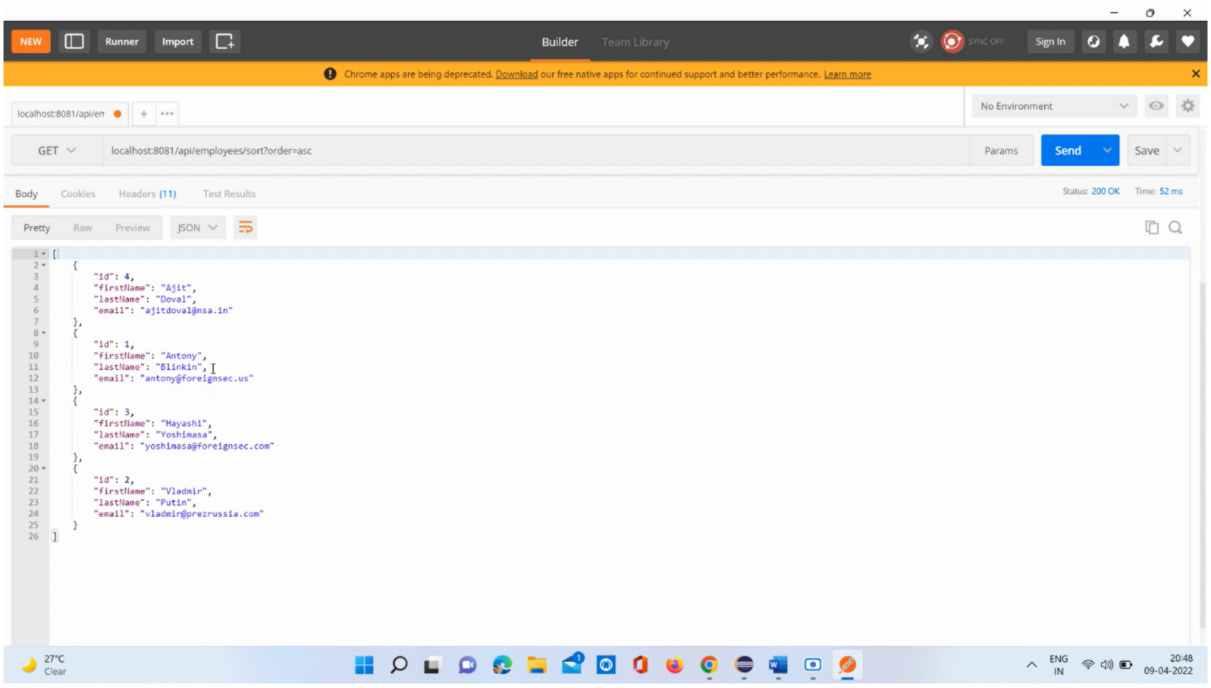
Authorization: Type: Basic Auth
Username: admin
Password: •••••

Status: 200 OK Time: 52 ms

```
[
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    },
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
```

As you can see the url localhost:8081/api/employees/sort?order=asc shows the list of employees in ascending order starting from “A”.

Here’s a better look:



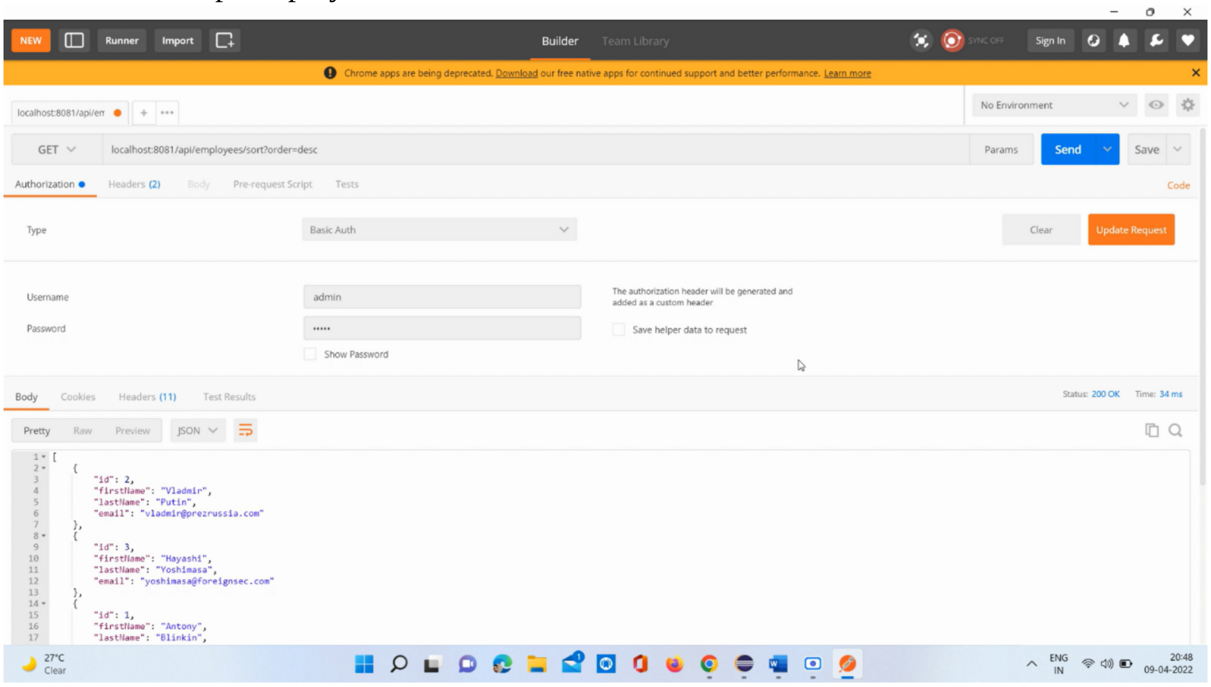
GET localhost:8081/api/employees/sort?order=asc

Status: 200 OK Time: 52 ms

```
[
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    },
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    }
]
```

SOLUTION Q9(II)

Now, next and the final question was regarding sorting the list of employees in descending order and as you can see in the url we have changed the order localhost:8081/api/employees/sort?order=desc from asc



GET localhost:8081/api/employees/sort?order=desc

Type: Basic Auth

Username: admin

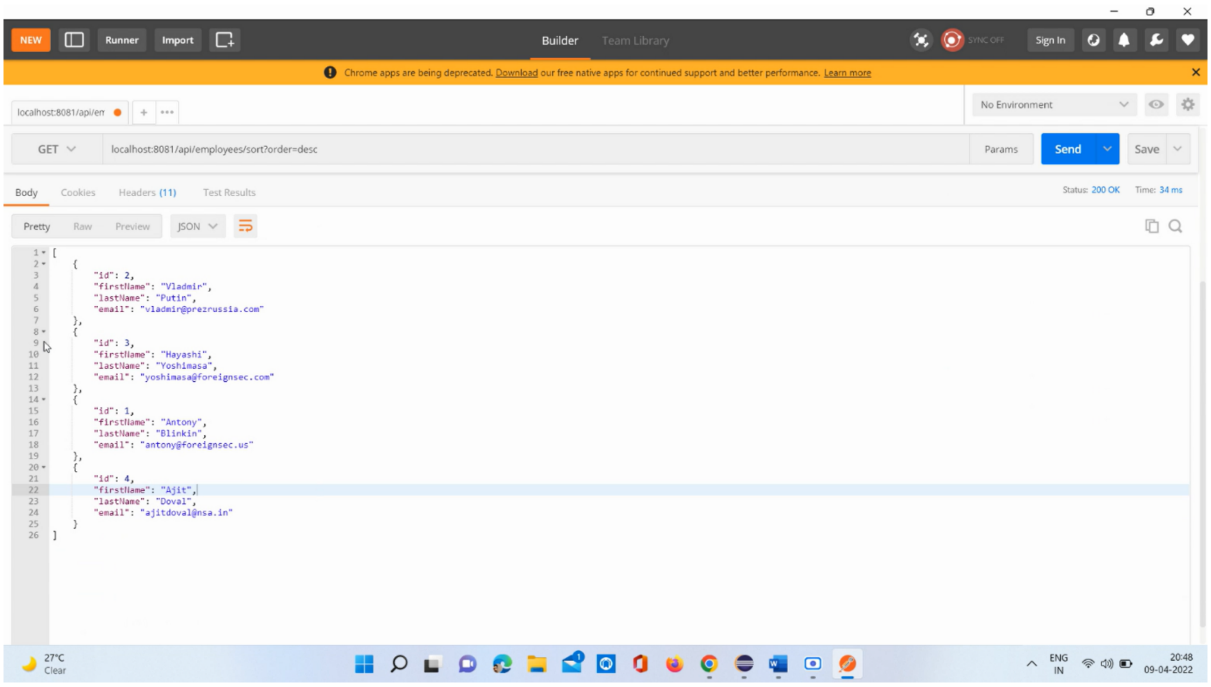
Password: •••••

The authorization header will be generated and added as a custom header

Status: 200 OK Time: 34 ms

```
[
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
```

Here's a better look:

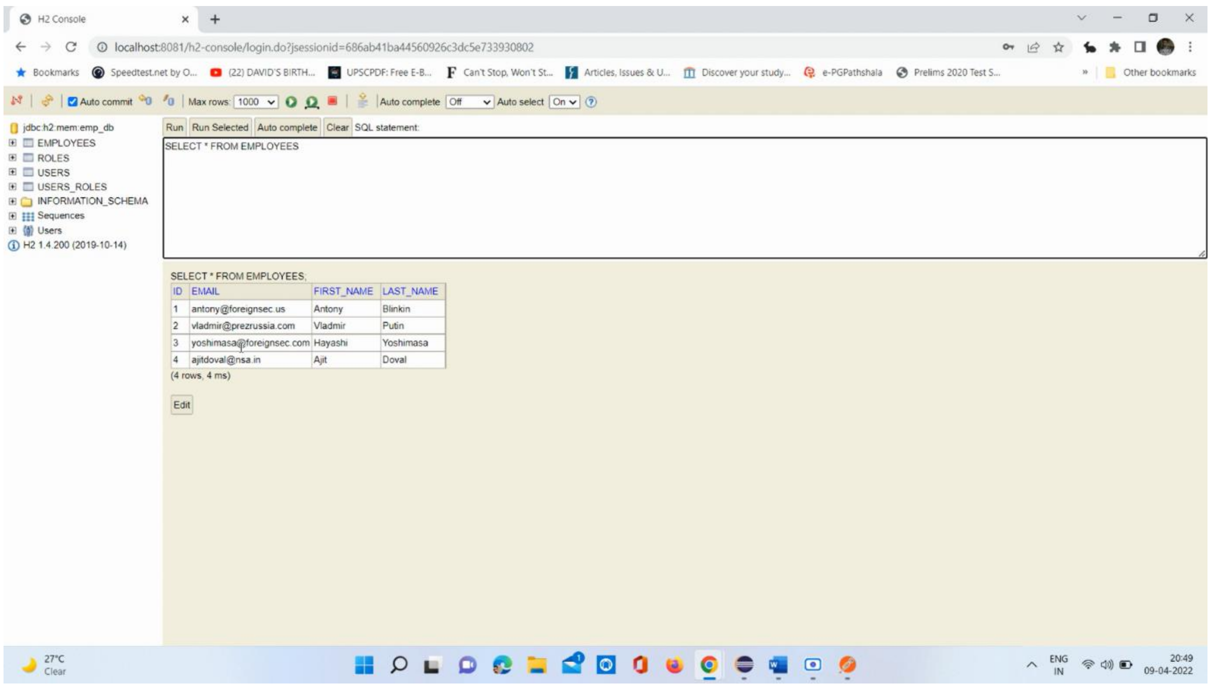


GET localhost:8081/api/employees/sort?order=desc

Status: 200 OK Time: 34 ms

```
[
    {
        "id": 2,
        "firstName": "Vladmir",
        "lastName": "Putin",
        "email": "vladmir@prezrussia.com"
    },
    {
        "id": 3,
        "firstName": "Hayashi",
        "lastName": "Yoshimasa",
        "email": "yoshimasa@foreignsec.com"
    },
    {
        "id": 1,
        "firstName": "Antony",
        "lastName": "Blinkin",
        "email": "antony@foreignsec.us"
    },
    {
        "id": 4,
        "firstName": "Ajit",
        "lastName": "Doval",
        "email": "ajitdoval@nsa.in"
    }
]
```

The list is sorted reverse alphabetic as required, let's also check the list of employees from the embedded database.



```
SELECT * FROM EMPLOYEES
```

SELECT * FROM EMPLOYEES;

ID	EMAIL	FIRST_NAME	LAST_NAME
1	antony@foreignsec.us	Antony	Blinkin
2	vladmir@prezrussia.com	Vladmir	Putin
3	yoshimasa@foreignsec.com	Hayashi	Yoshimasa
4	ajitdoval@nsa.in	Ajit	Doval

(4 rows, 4 ms)

NOTE: All the requests were executed in Postman and I've used Embedded Database H2.