Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

Aim : Implement an application with CRUD operation using Postgres with NodeJS.

A social media website xyz.com wants to manage its users records and this task is given to you. They are seeking functionalities :

Practical 19.1: New joiner data should be stored in DB.

Practical 19.2: The admin can see the whole DB

Practical 19.3: The admin can filter out the DB.

Practical 19.4: The user can update their information, which should be updated in the DB.

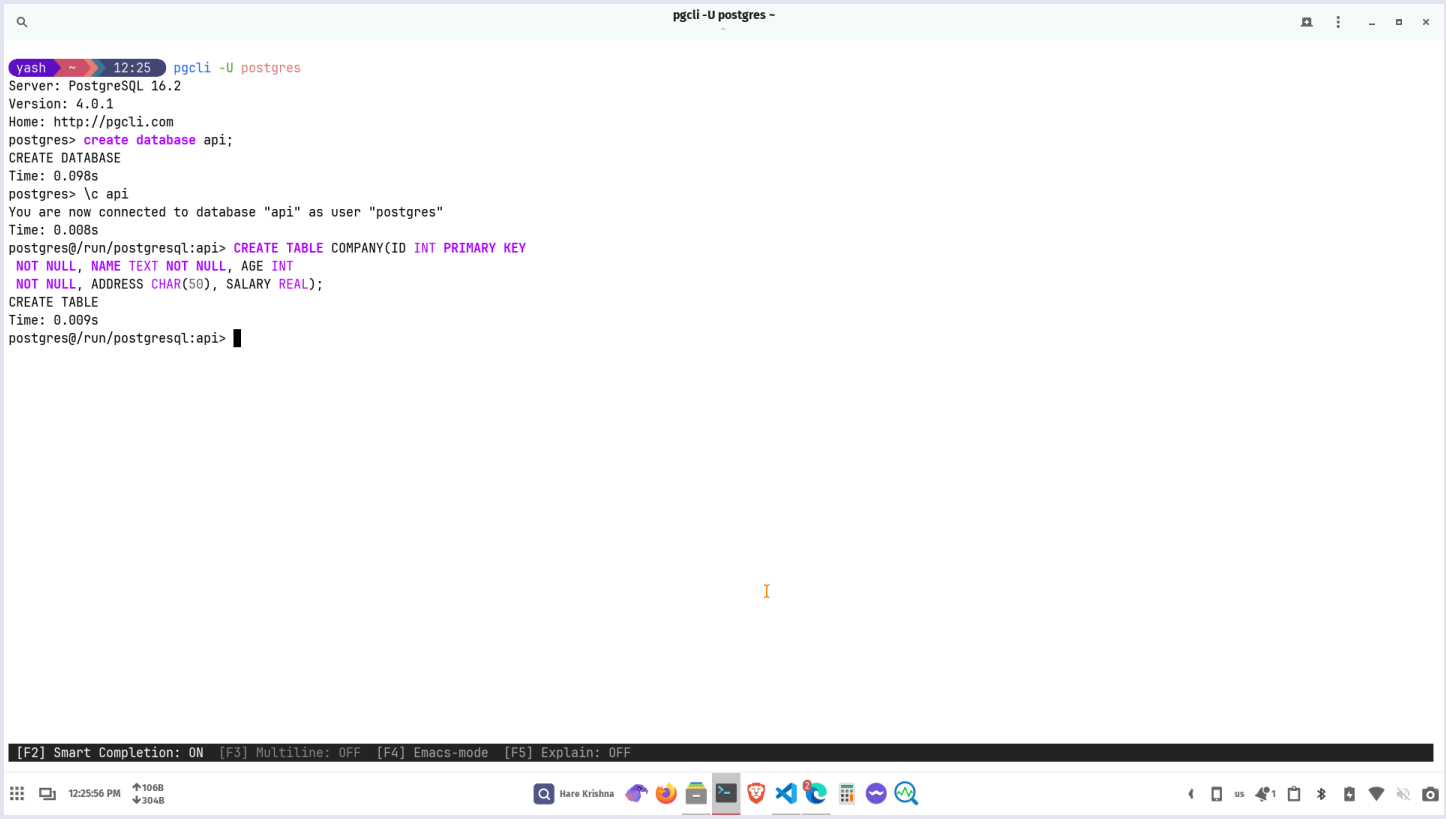
Practical 19.5: The admin can delete the record from DB.

Practical 19.6: Integrate your Nodejs application with the IBM cloud service-Databases for Postgresql and check for the database status using pg admin.

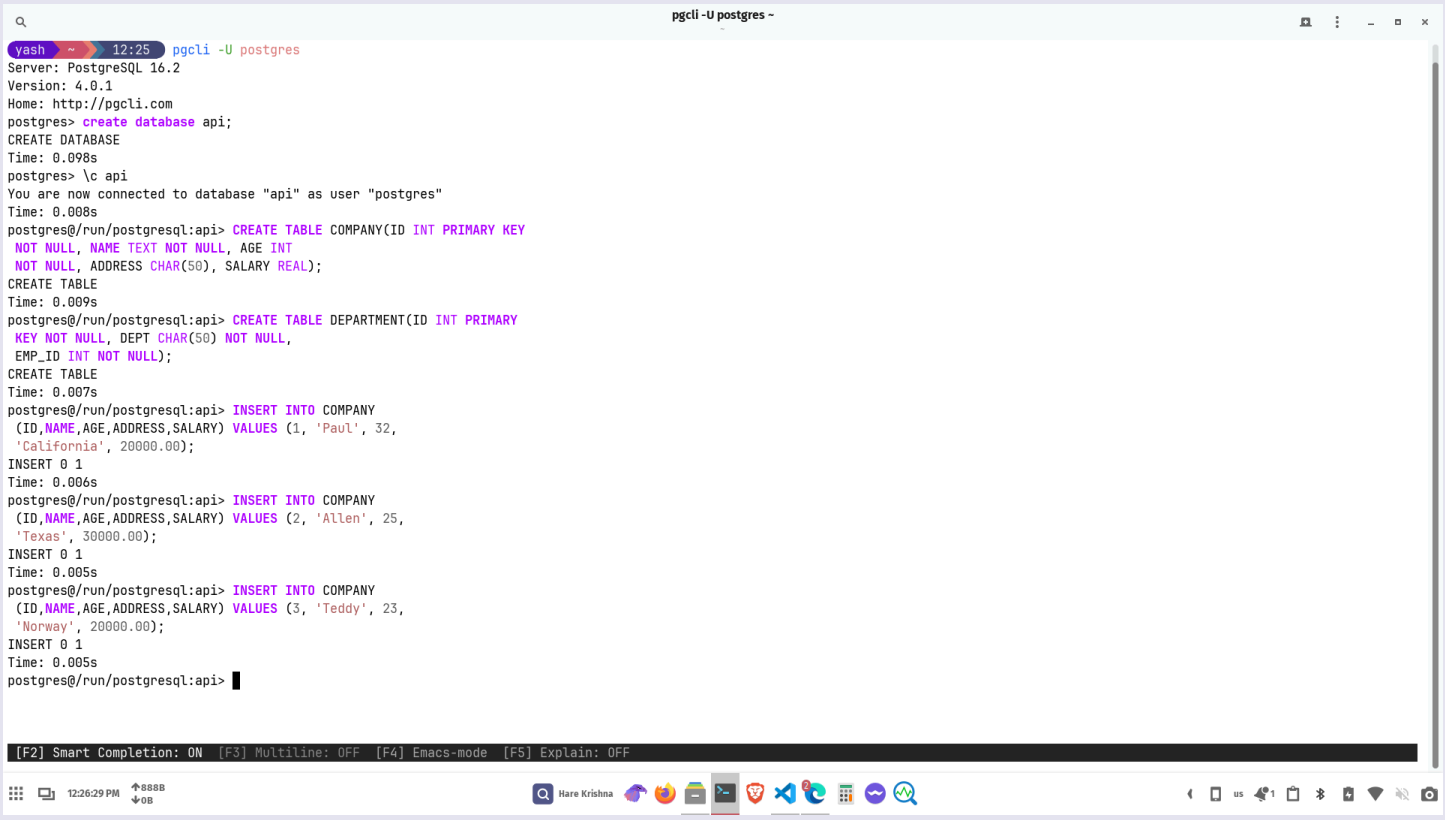
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

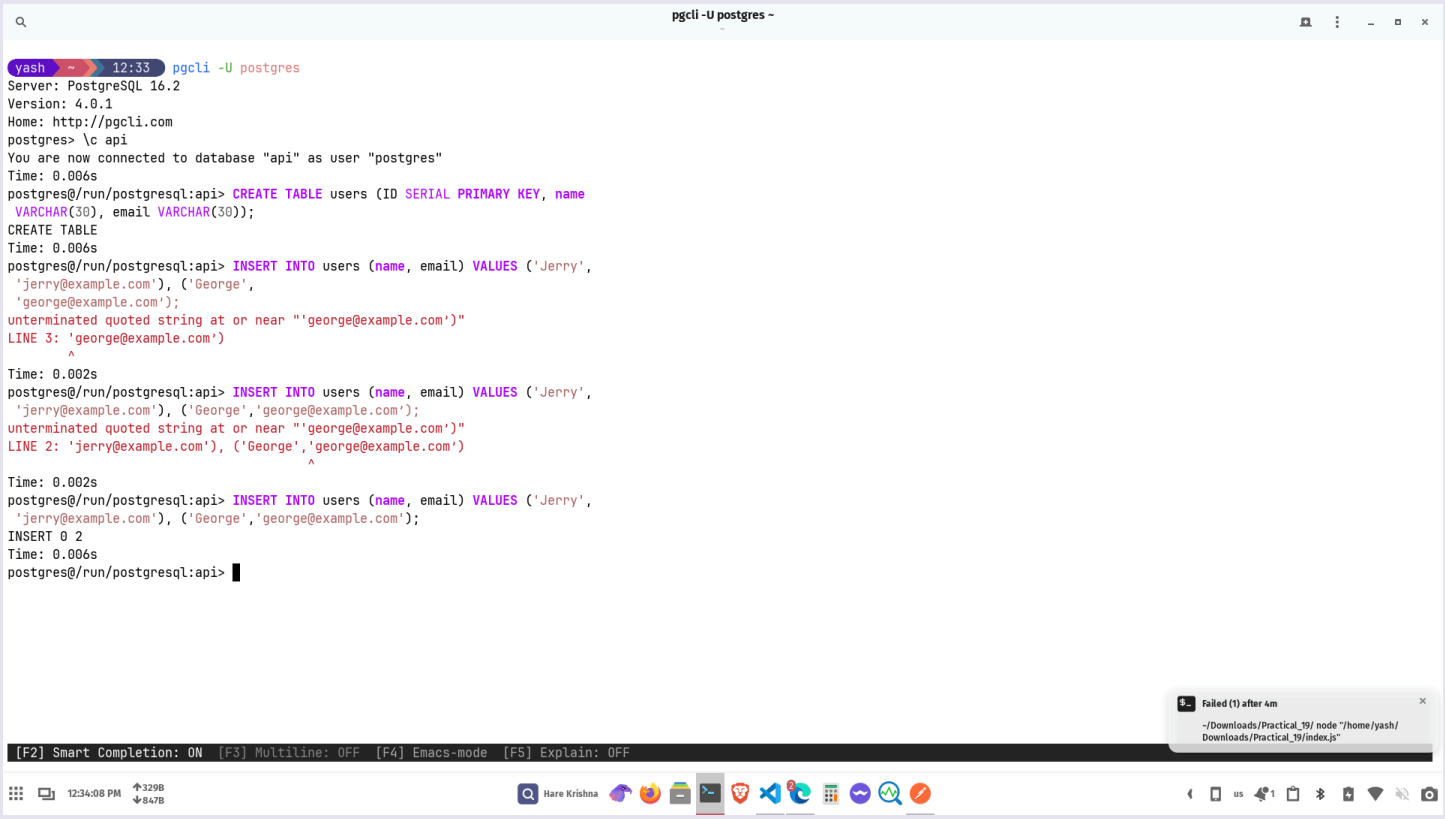
Steps and Screenshots **:**

1. Install and setup postgresql and pgadmin in local machine and try creating database, table and insert values in it



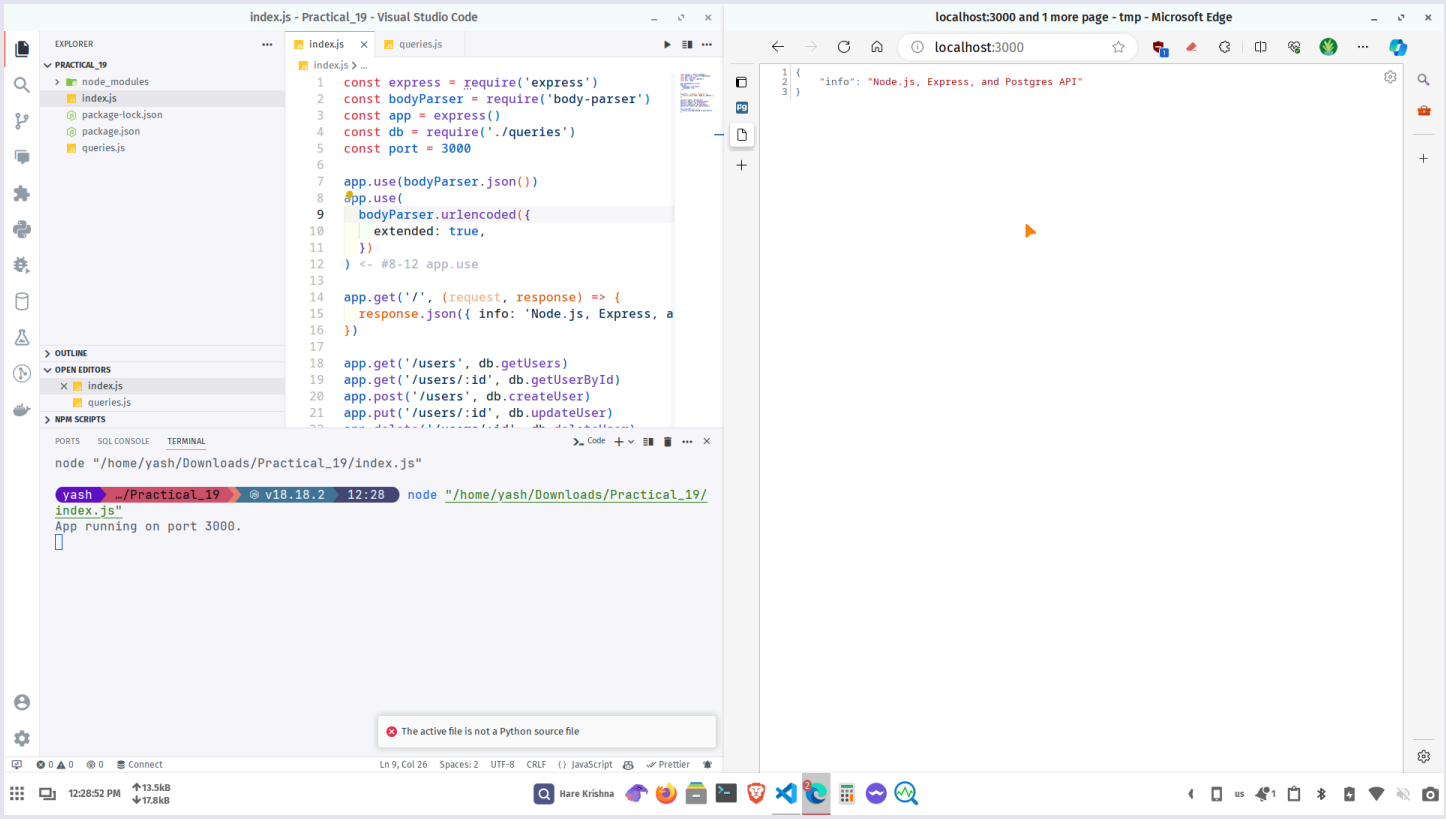
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19





Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

2. Now run locally the node server to manage data in postgres database



Code (queries.js) :

```
const Pool = require('pg').Pool
var fs = require('fs');
const pool = new Pool({
 user: 'postgres',
 host: 'localhost',
 database: 'api',
 password: 'krsna',
 port: 5432,
 /*ssl: {
   rejectUnauthorized: false,
   cert: fs.readFileSync('./cert.pem').toString(),
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
}*/
})


const getUsers = (request, response) => {
   pool.query('SELECT * FROM users ORDER BY id ASC', (error, results) => {
     if (error) {
       throw error
     }
     response.status(200).json(results.rows)
   })
 }

 const getUserById = (request, response) => {
   const id = parseInt(request.params.id)
    pool.query('SELECT * FROM users WHERE id = $1', [id], (error, results)
=> {
     if (error) {
       throw error
     }
     response.status(200).json(results.rows)
   })
 }

 const createUser = (request, response) => {
   const { name, email } = request.body
    pool.query('INSERT INTO users (name, email) VALUES ($1, $2) RETURNING
*', [name, email], (error, results) => {
     if (error) {
       throw error
     }
     response.status(201).send(`User added with ID:
${results.rows[0].id}`)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
  })
}
const updateUser = (request, response) => {
  const id = parseInt(request.params.id)
  const { name, email } = request.body
   pool.query(
    'UPDATE users SET name = $1, email = $2 WHERE id = $3',
    [name, email, id],
    (error, results) => {
      if (error) {
        throw error
      }
      response.status(200).send(`User modified with ID: ${id}`)
    }
  )
}
const deleteUser = (request, response) => {
  const id = parseInt(request.params.id)
   pool.query('DELETE FROM users WHERE id = $1', [id], (error, results)
=> {
    if (error) {
      throw error
    }
    response.status(200).send(`User deleted with ID: ${id}`)
  })
}

module.exports = {
getUsers,
getUserById,
createUser,
updateUser,
deleteUser,
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
}
```

Code (index.js) :

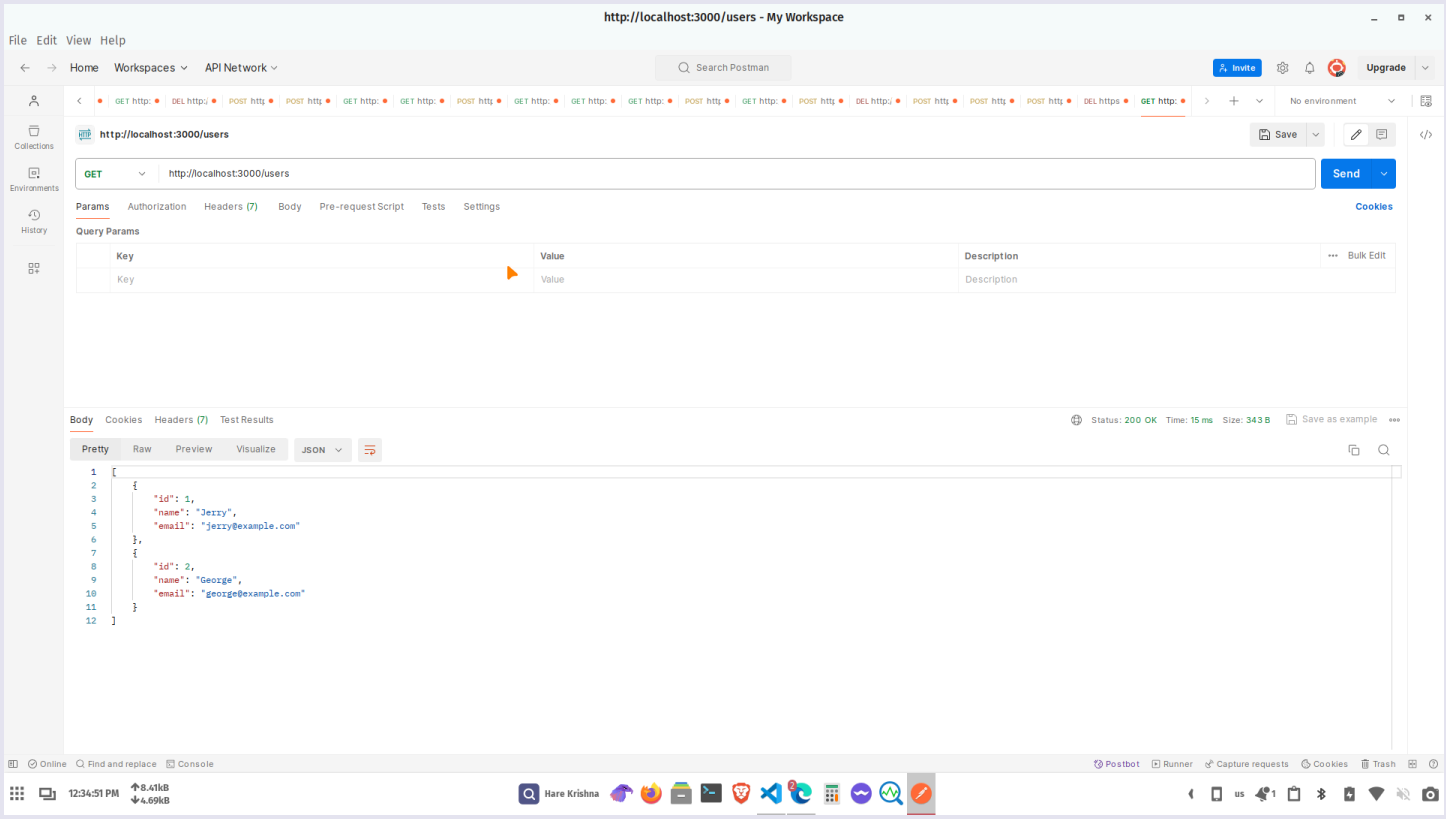
```
const express = require('express')
const bodyParser = require('body-parser')
const app = express()
const db = require('./queries')
const port = 3000

app.use(bodyParser.json())
app.use(
 bodyParser.urlencoded({
   extended: true,
 })
)

app.get('/', (request, response) => {
 response.json({ info: 'Node.js, Express, and Postgres API' })
})

app.get('/users', db.getUsers)
app.get('/users/:id', db.getUserById)
app.post('/users', db.createUser)
app.put('/users/:id', db.updateUser)
app.delete('/users/:id', db.deleteUser)

app.listen(port, () => {
 console.log(`App running on port ${port}.`)
})
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

3. As the local server is running, use Postman to send requests for performing operations :

    a) To get users table content in json format

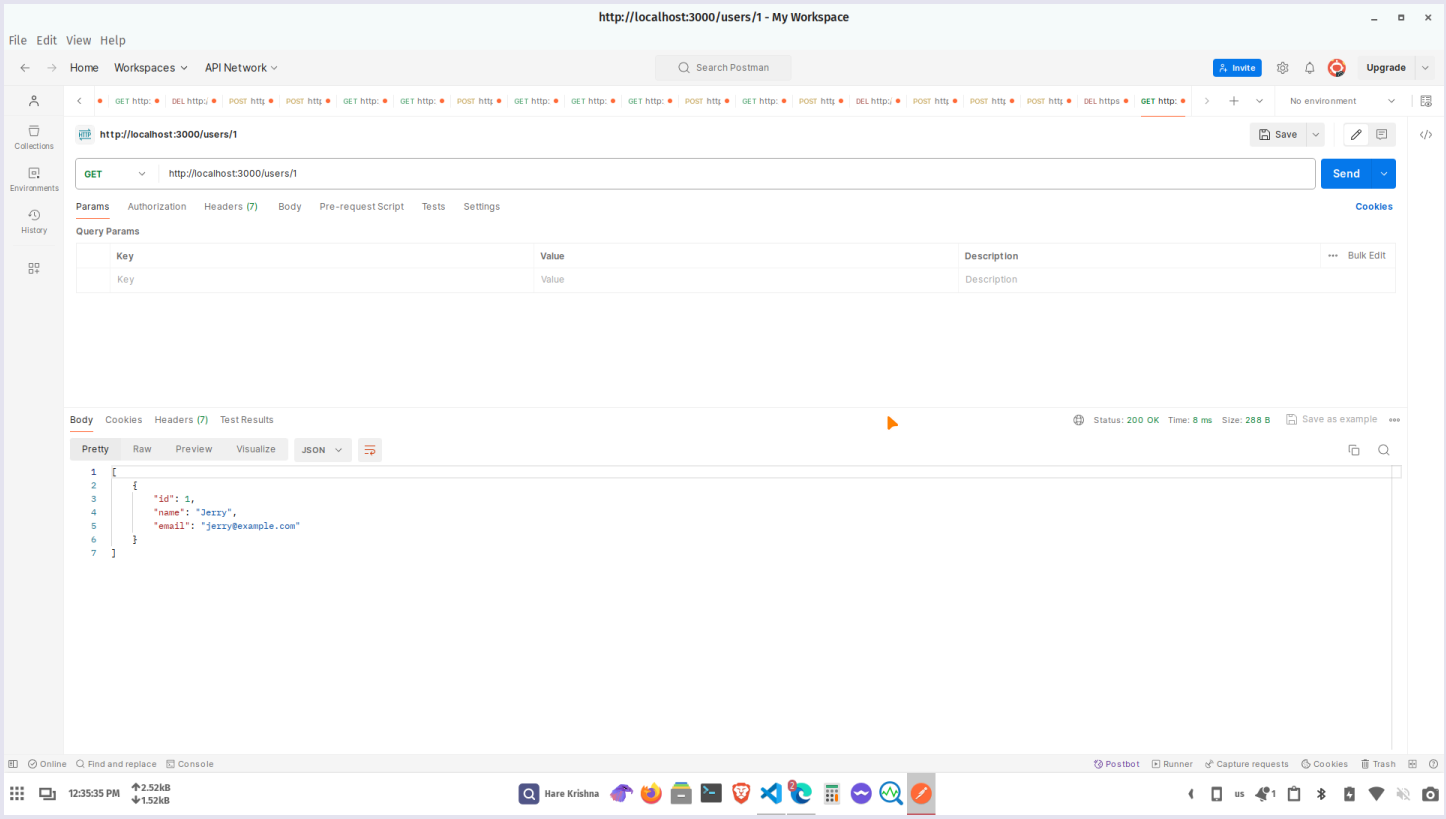
    Request : GET
    URL : http://localhost:*port*/users



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

b) Get details of specific user using ID

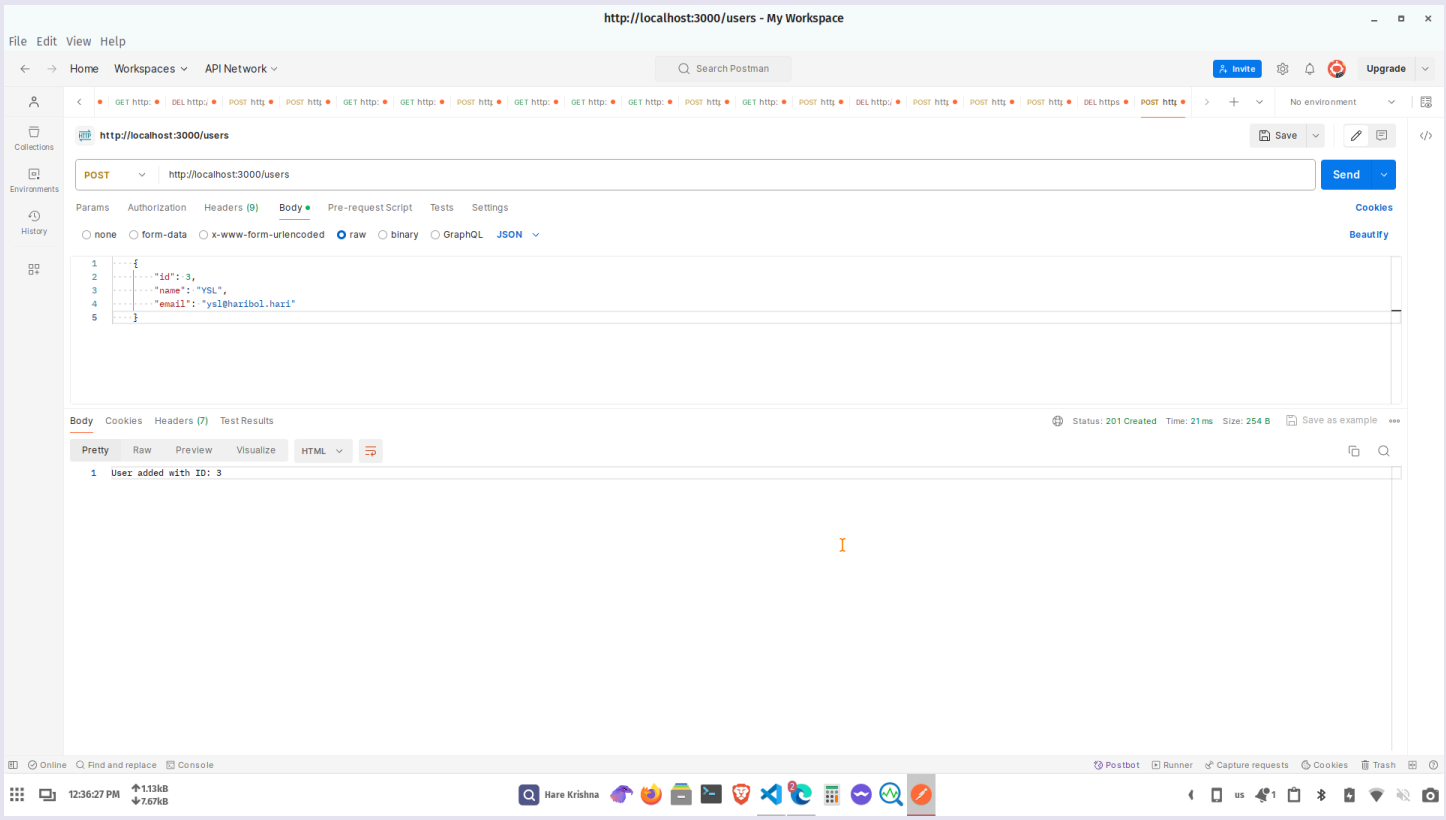
Request : GET
URL : http://localhost:*port*/users/:id



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

c) To add record of a user in users table

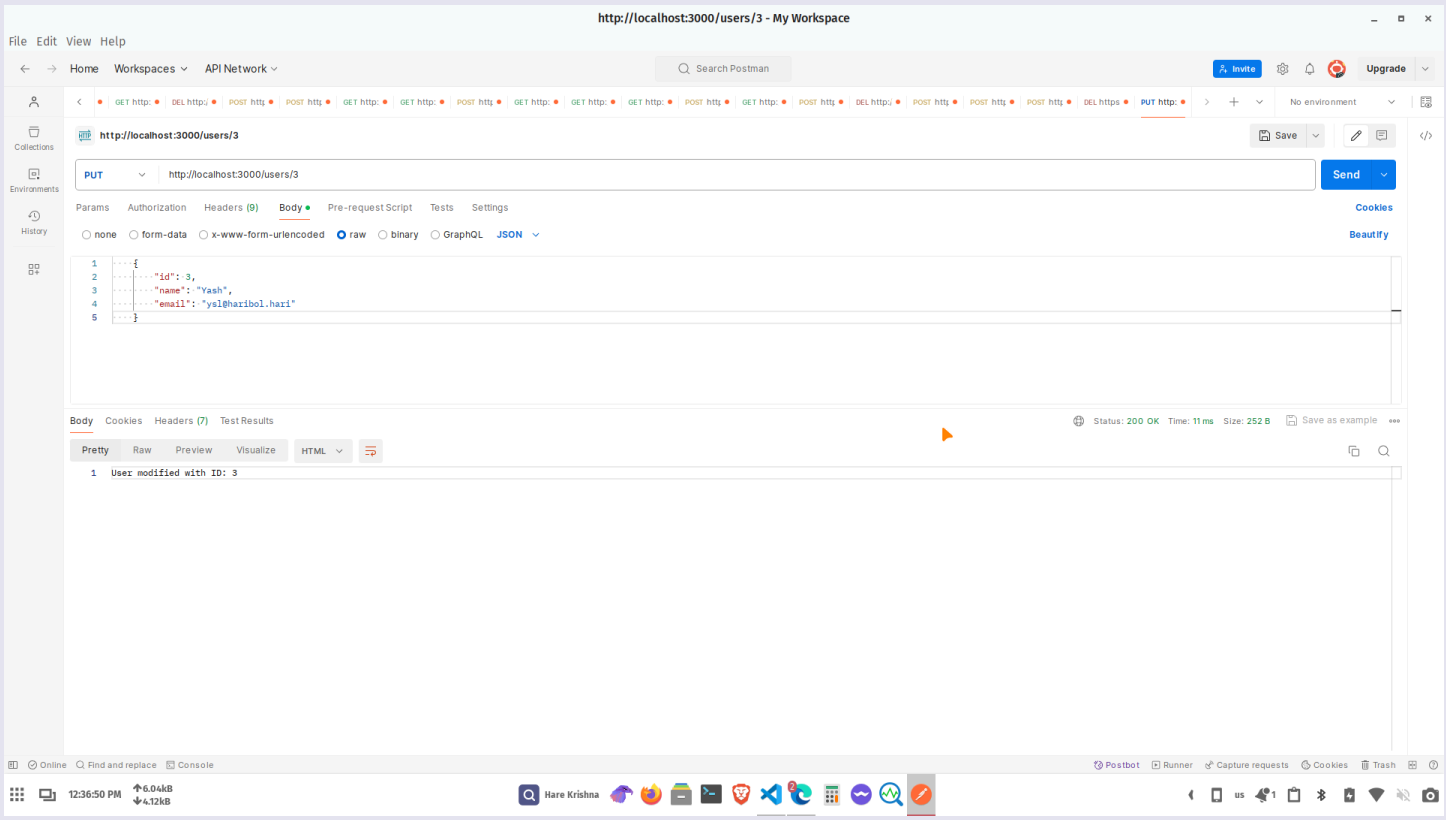
Request : POST
URL : http://localhost:*port*/users
Body : json object of user data



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

d) Update the detail(s) of a user using ID

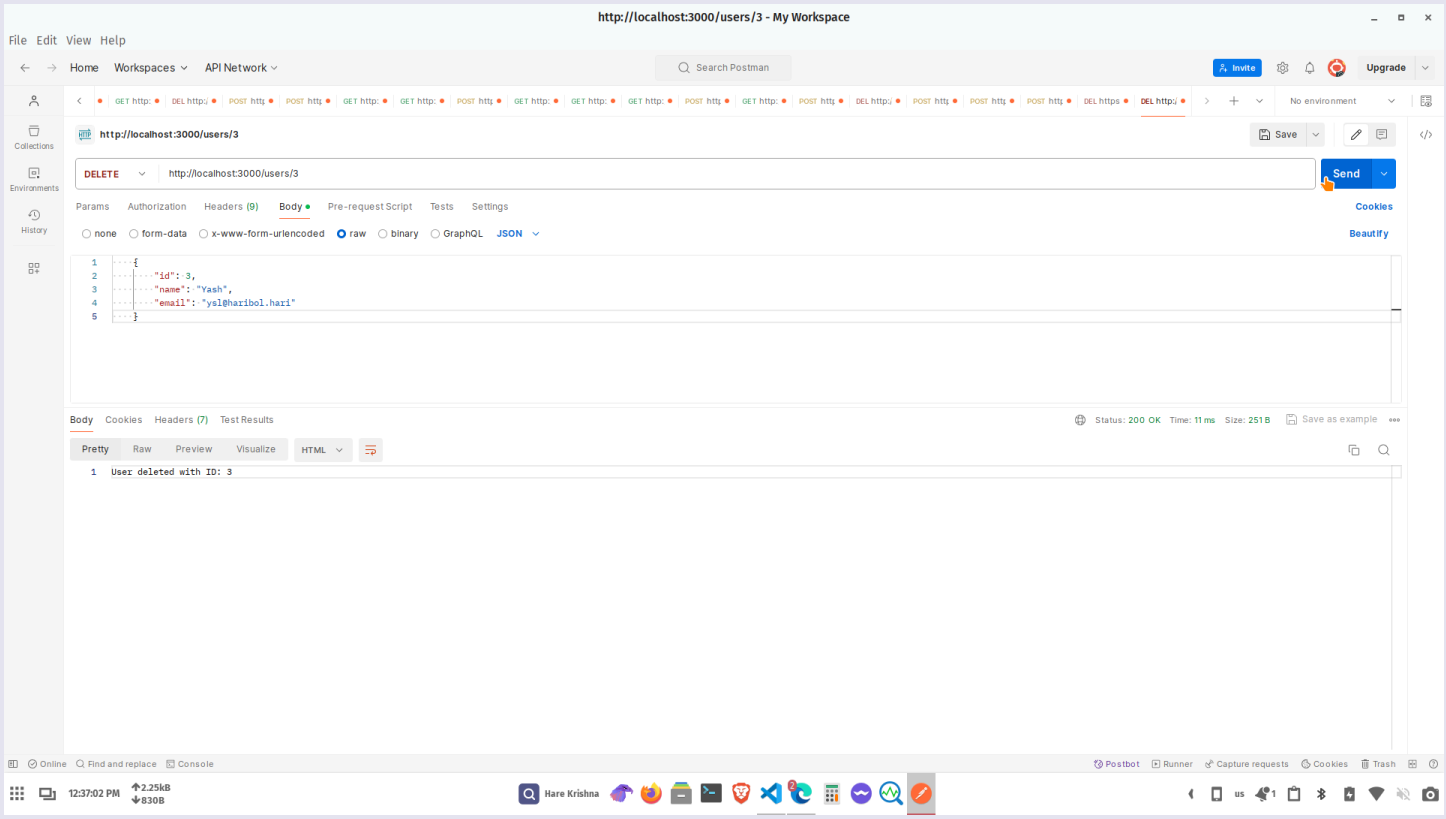
Request : PUT
URL : http://localhost:*port*/users/:id
Body : json object of updated user details



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

e) Delete the record of a user using ID

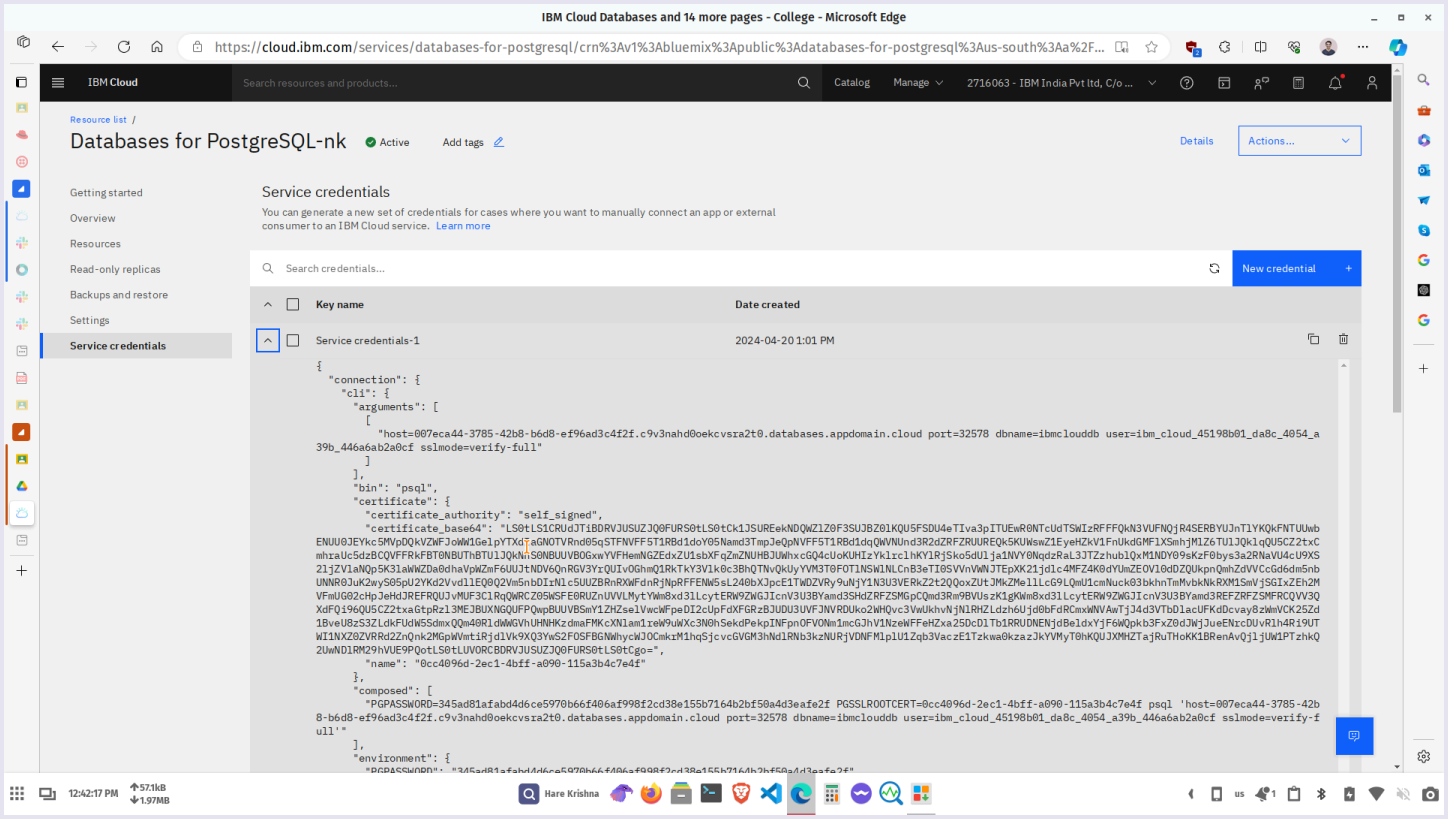
Request : DELETE
URL : http://localhost:*port*/users/:id



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

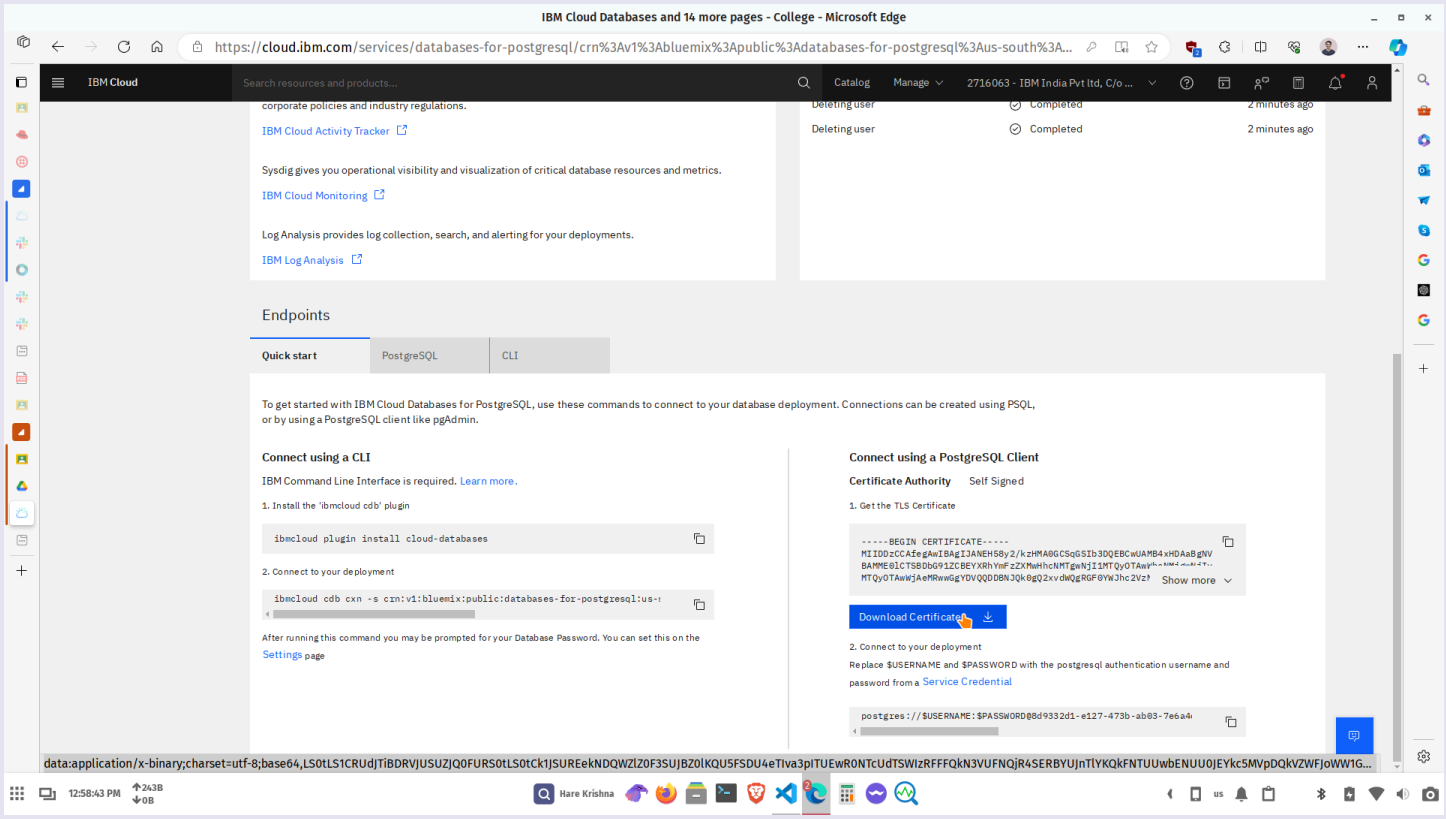
Integration with IBM Cloud :

1. Create if not exists, the PostgreSQL service on IBM Cloud and create if not exist its service credentials



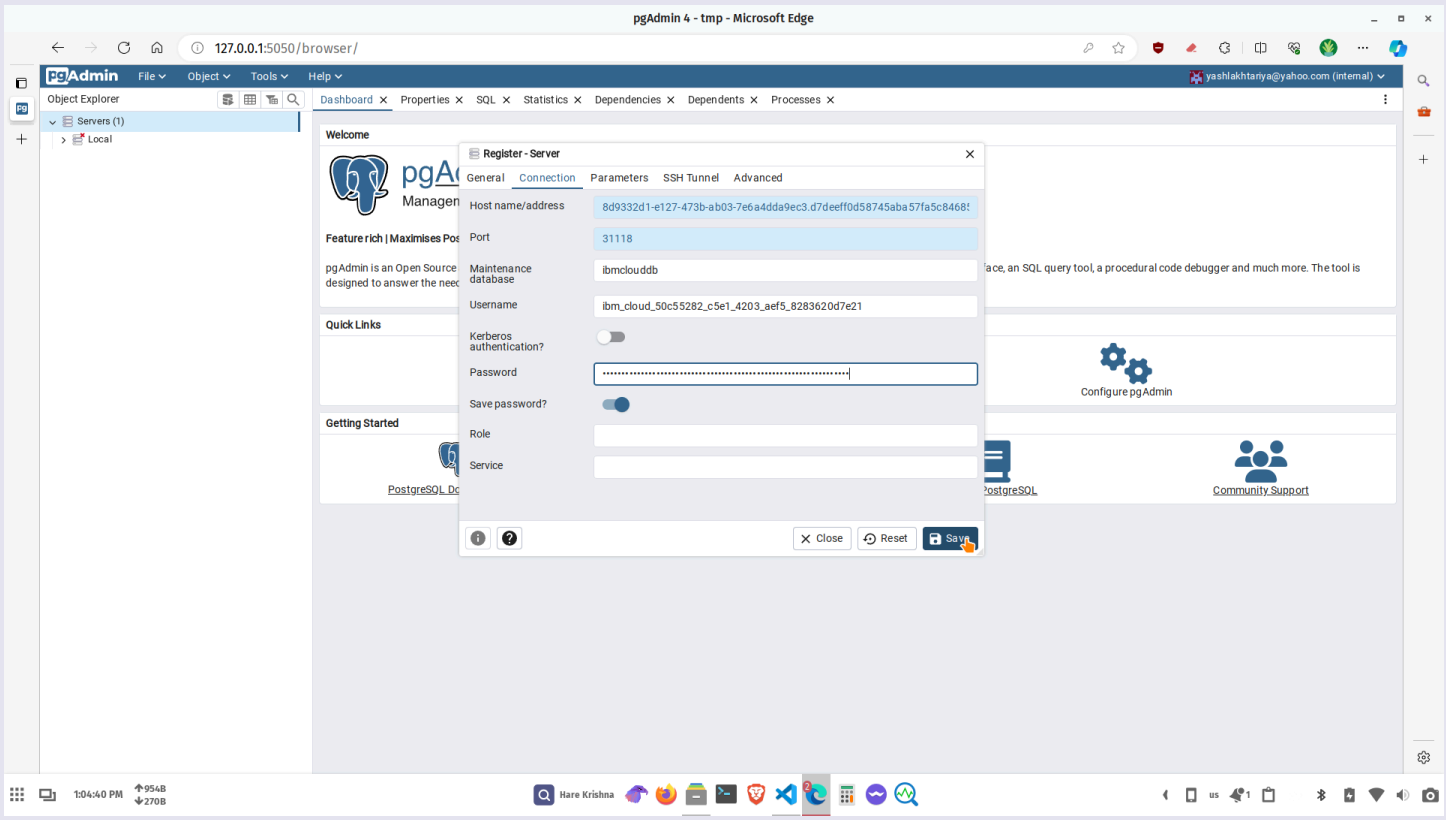
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

2. Download the TLS certificate to ensure authenticity

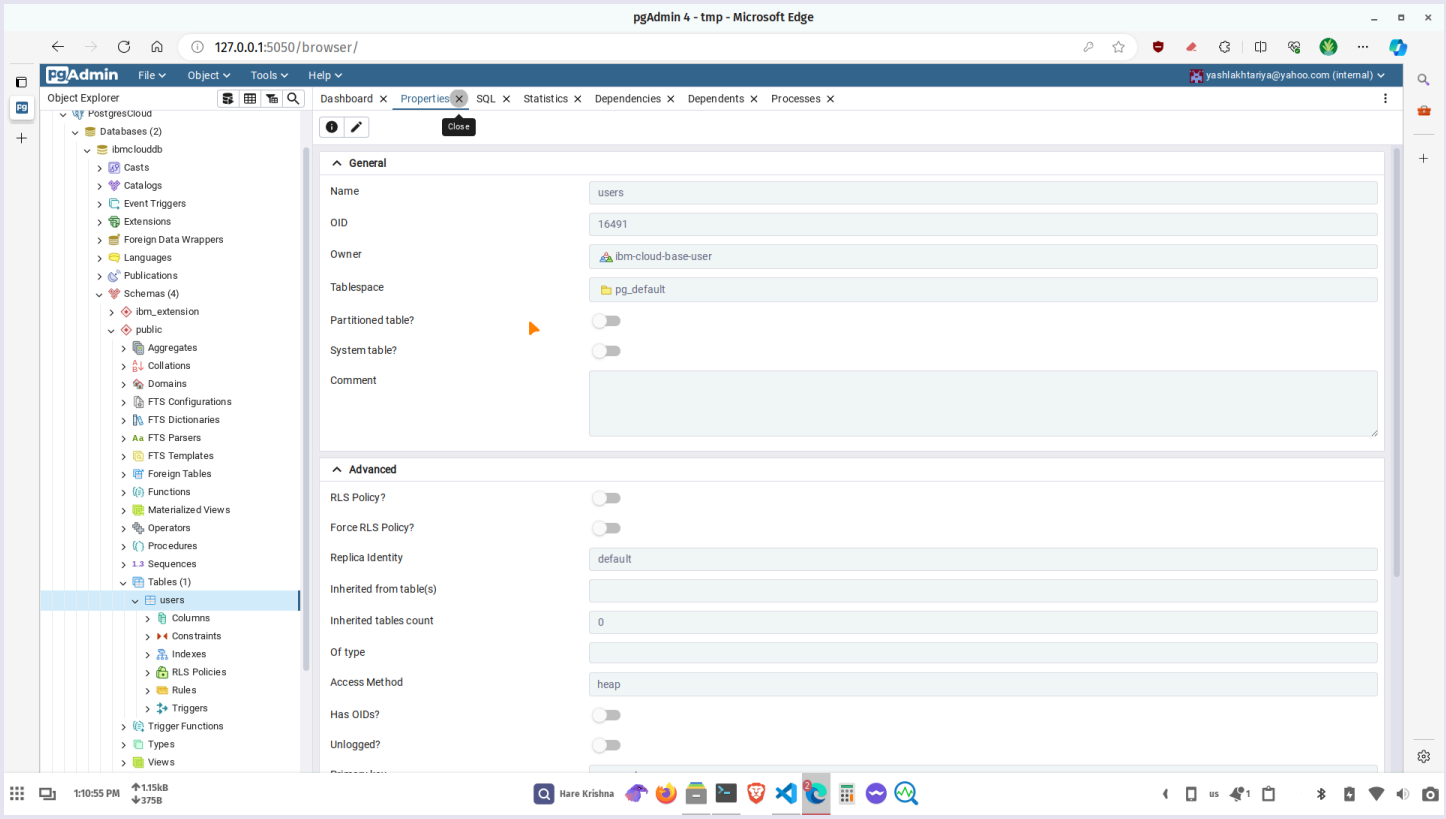


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

3. Connect the cloud PostgreSQL database with local pgadmin using service credentails : username, password, host, database name and port



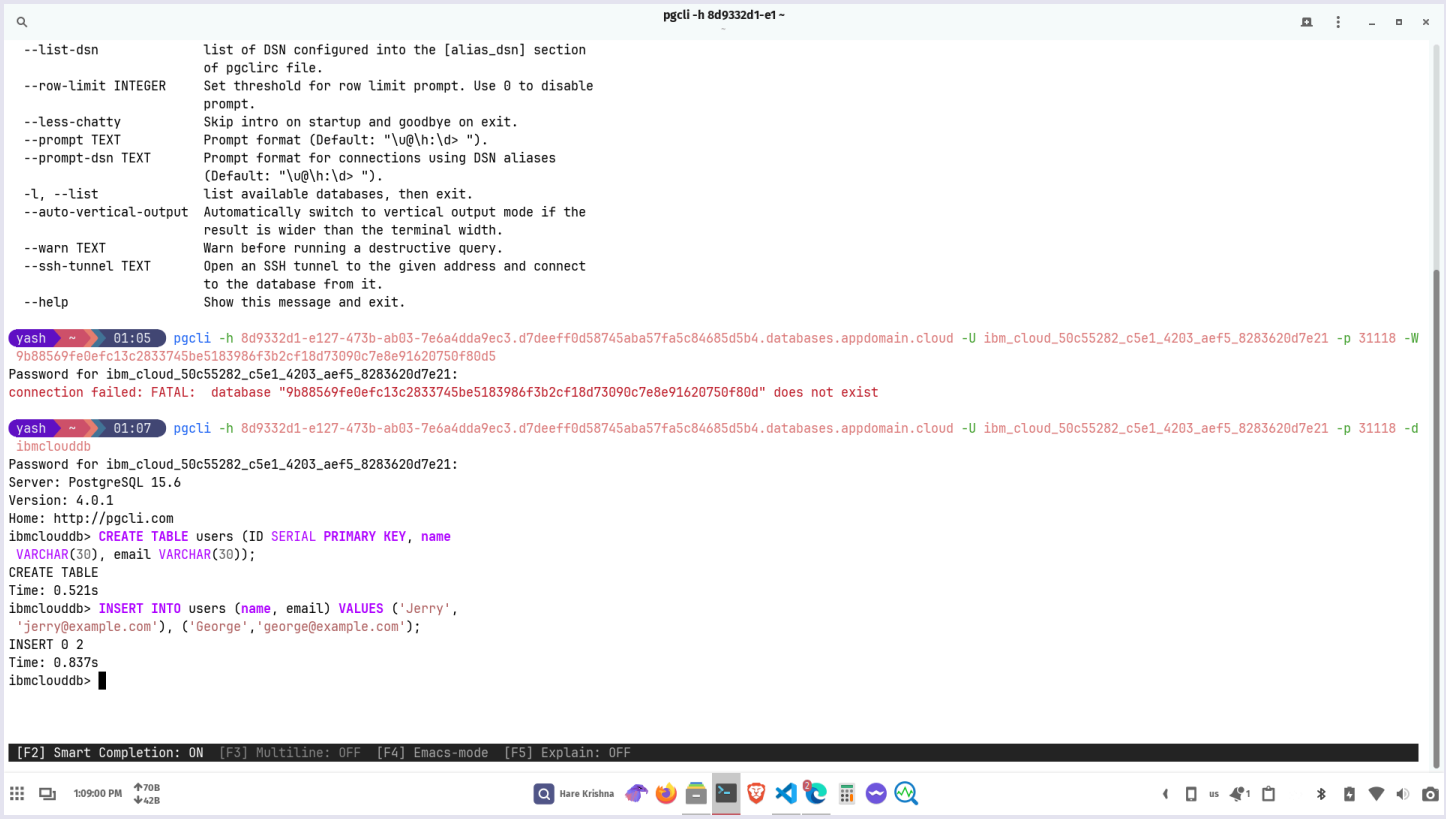
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

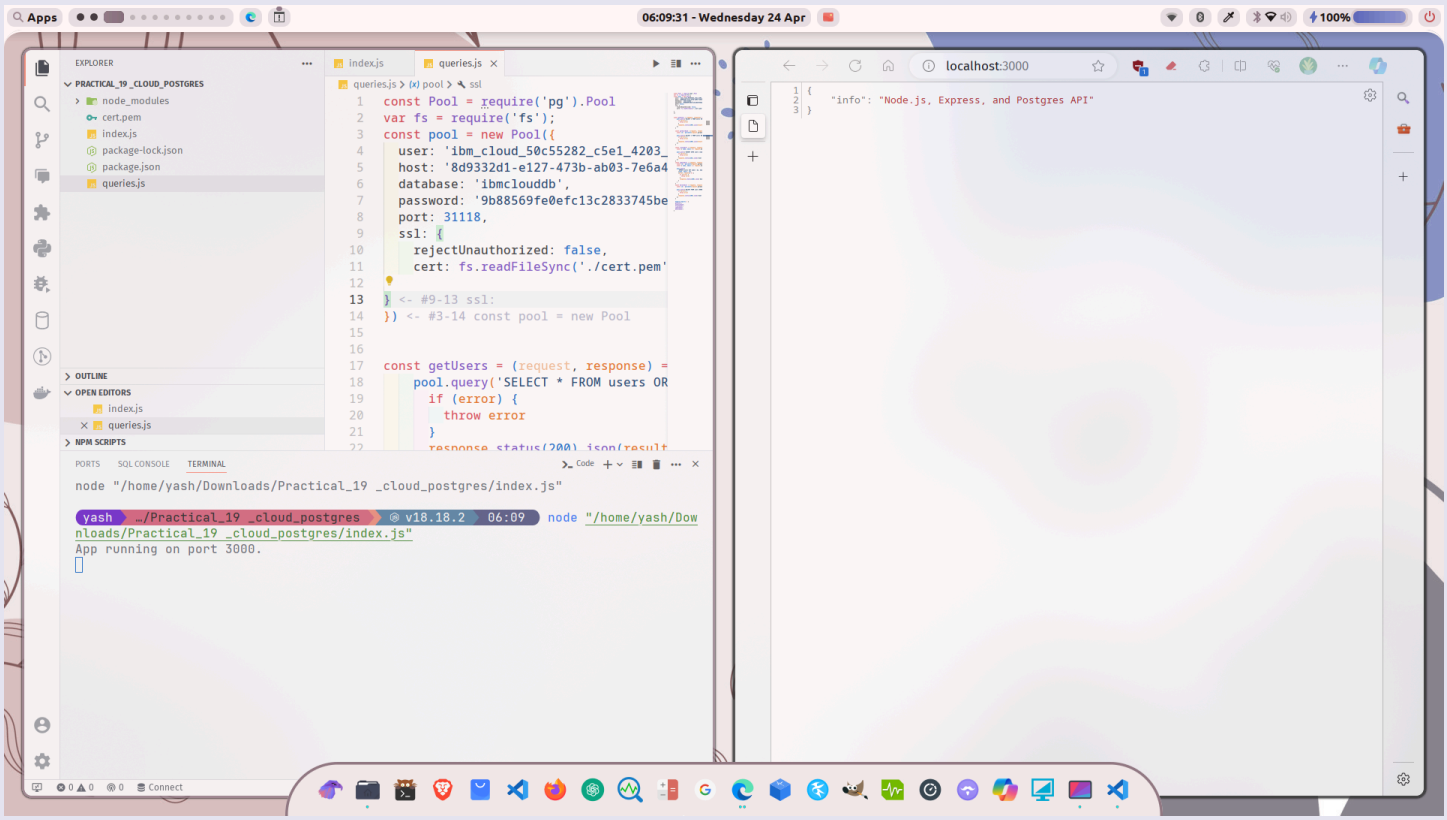
4. Also, it can be connected to local psql command line using the same credentials

(*Here pgcli is used instead of psql, which has same syntax like psql but with features like autocomplete, syntax highlighting, and much more)*



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

5. Now, integrate it with nodejs application and try running the same Postman requests



Code (queries.js) :

```
const Pool = require('pg').Pool
var fs = require('fs');
const pool = new Pool({
 user: 'ibm_cloud_50c55282_c5e1_4203_aef5_8283620d7e21',
 host:
'8d9332d1-e127-473b-ab03-7e6a4dda9ec3.d7deeff0d58745aba57fa5c84685d5b4.databases.appdomain.cloud',
 database: 'ibmclouddb',
 password:
'9b88569fe0efc13c2833745be5183986f3b2cf18d73090c7e8e91620750f80d5',
 port: 31118,
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
 ssl: {
   rejectUnauthorized: false,
   cert: fs.readFileSync('./cert.pem').toString(),

}
})


const getUsers = (request, response) => {
   pool.query('SELECT * FROM users ORDER BY id ASC', (error, results) => {
     if (error) {
       throw error
     }
     response.status(200).json(results.rows)
   })
 }

 const getUserById = (request, response) => {
   const id = parseInt(request.params.id)
    pool.query('SELECT * FROM users WHERE id = $1', [id], (error, results)
=> {
     if (error) {
       throw error
     }
     response.status(200).json(results.rows)
   })
 }

 const createUser = (request, response) => {
   const { name, email } = request.body
    pool.query('INSERT INTO users (name, email) VALUES ($1, $2) RETURNING
*', [name, email], (error, results) => {
     if (error) {
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
        throw error
      }
      response.status(201).send(`User added with ID:
${results.rows[0].id}`)
    })
 }
 const updateUser = (request, response) => {
   const id = parseInt(request.params.id)
   const { name, email } = request.body
    pool.query(
     'UPDATE users SET name = $1, email = $2 WHERE id = $3',
     [name, email, id],
     (error, results) => {
       if (error) {
         throw error
       }
       response.status(200).send(`User modified with ID: ${id}`)
     }
   )
 }
 const deleteUser = (request, response) => {
   const id = parseInt(request.params.id)
    pool.query('DELETE FROM users WHERE id = $1', [id], (error, results)
=> {
     if (error) {
       throw error
     }
     response.status(200).send(`User deleted with ID: ${id}`)
   })
 }

 module.exports = {
 getUsers,
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
 getUserById,
 createUser,
 updateUser,
 deleteUser,
}
```

Code (index.js) :

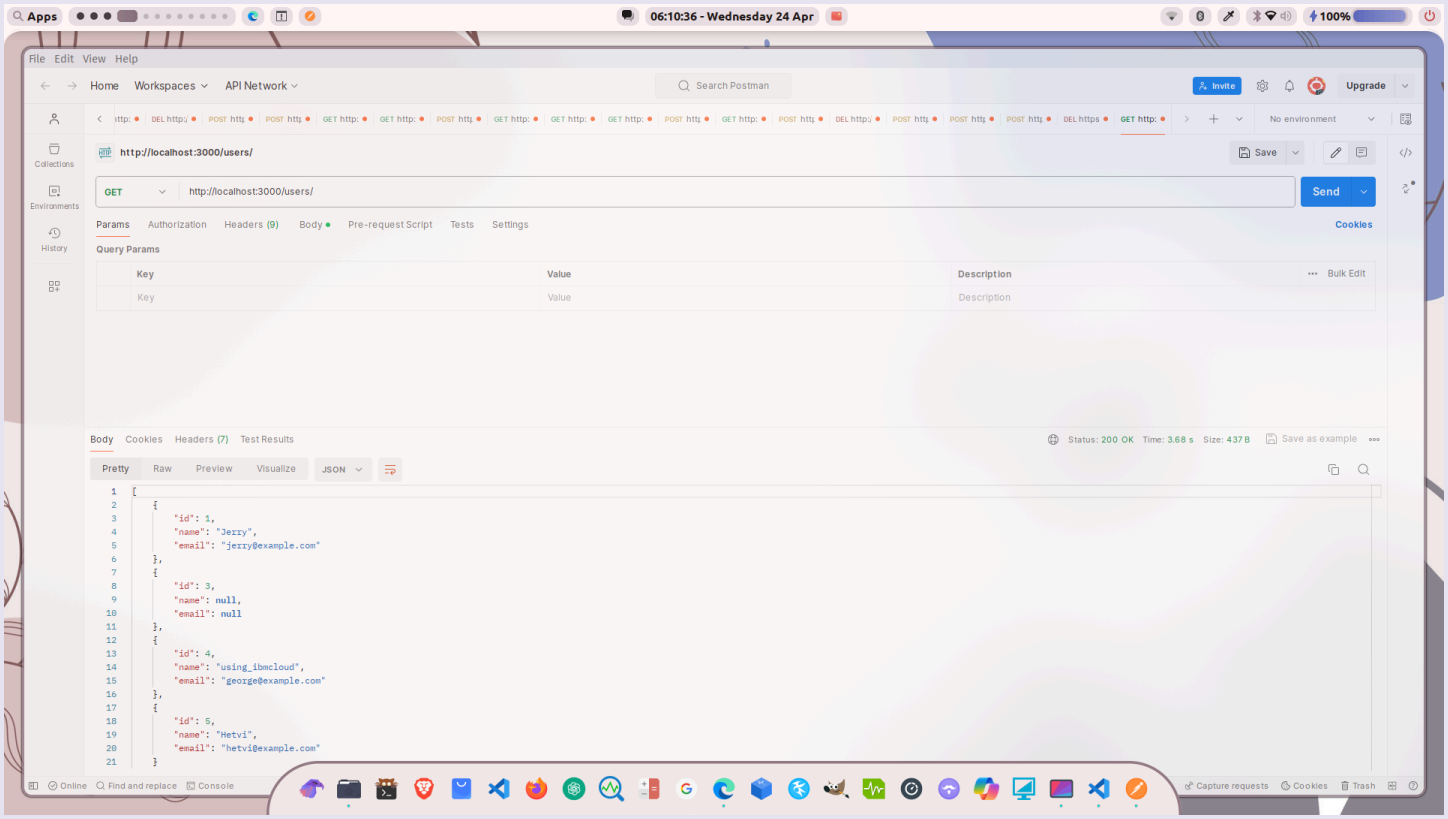
```
const express = require('express')
const bodyParser = require('body-parser')
const app = express()
const db = require('./queries')
const port = 3000

app.use(bodyParser.json())
app.use(
 bodyParser.urlencoded({
   extended: true,
 })
)

app.get('/', (request, response) => {
 response.json({ info: 'Node.js, Express, and Postgres API' })
})

app.get('/users', db.getUsers)
app.get('/users/:id', db.getUserById)
app.post('/users', db.createUser)
app.put('/users/:id', db.updateUser)
app.delete('/users/:id', db.deleteUser)

app.listen(port, () => {
 console.log(`App running on port ${port}.`)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

```
})
```

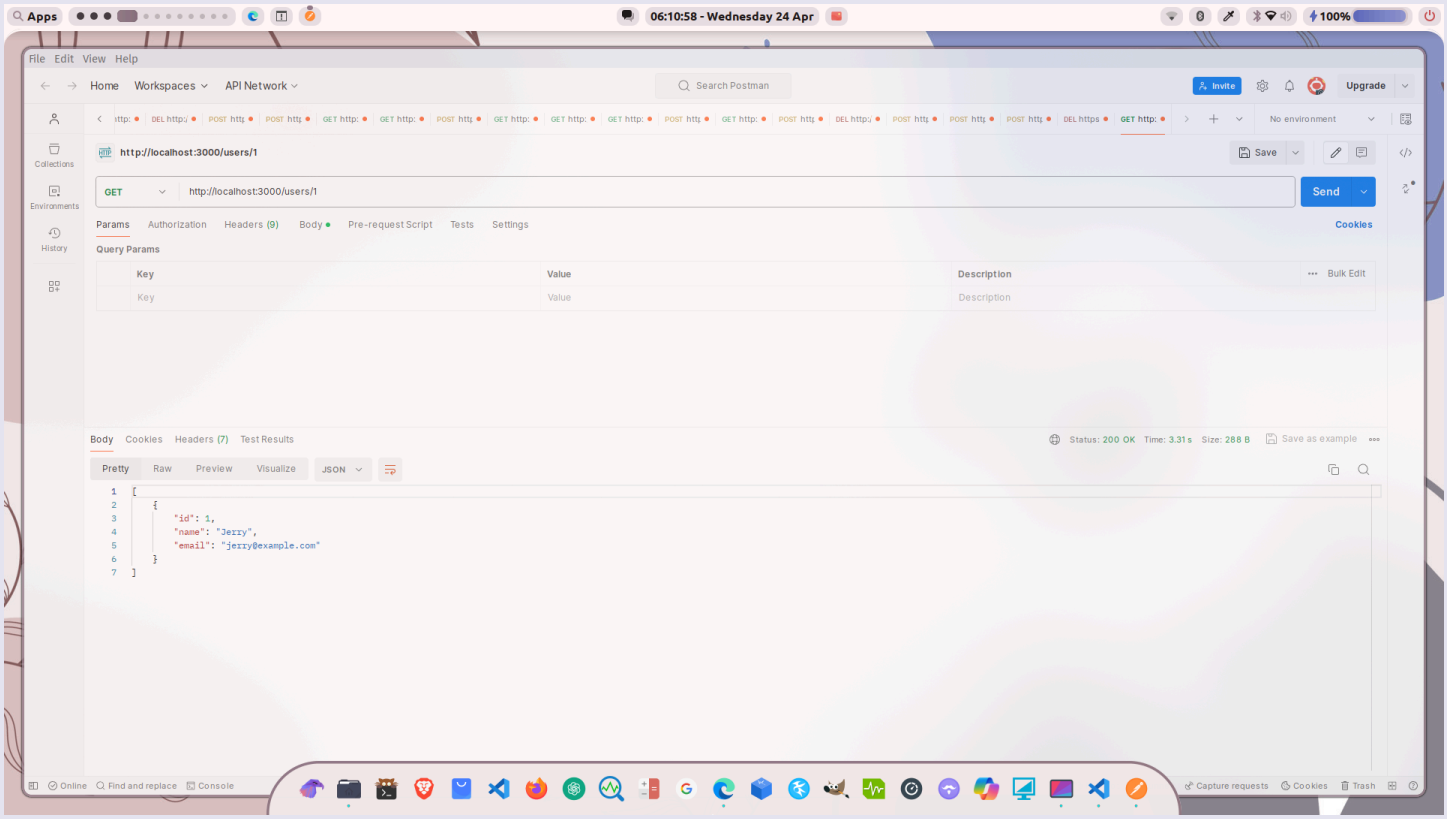
→ Requests same as above, in Postman, URL, body, same, because changes to integrate with cloud database is made in nodejs code, our API endpoints and output are same :

a) GET all users



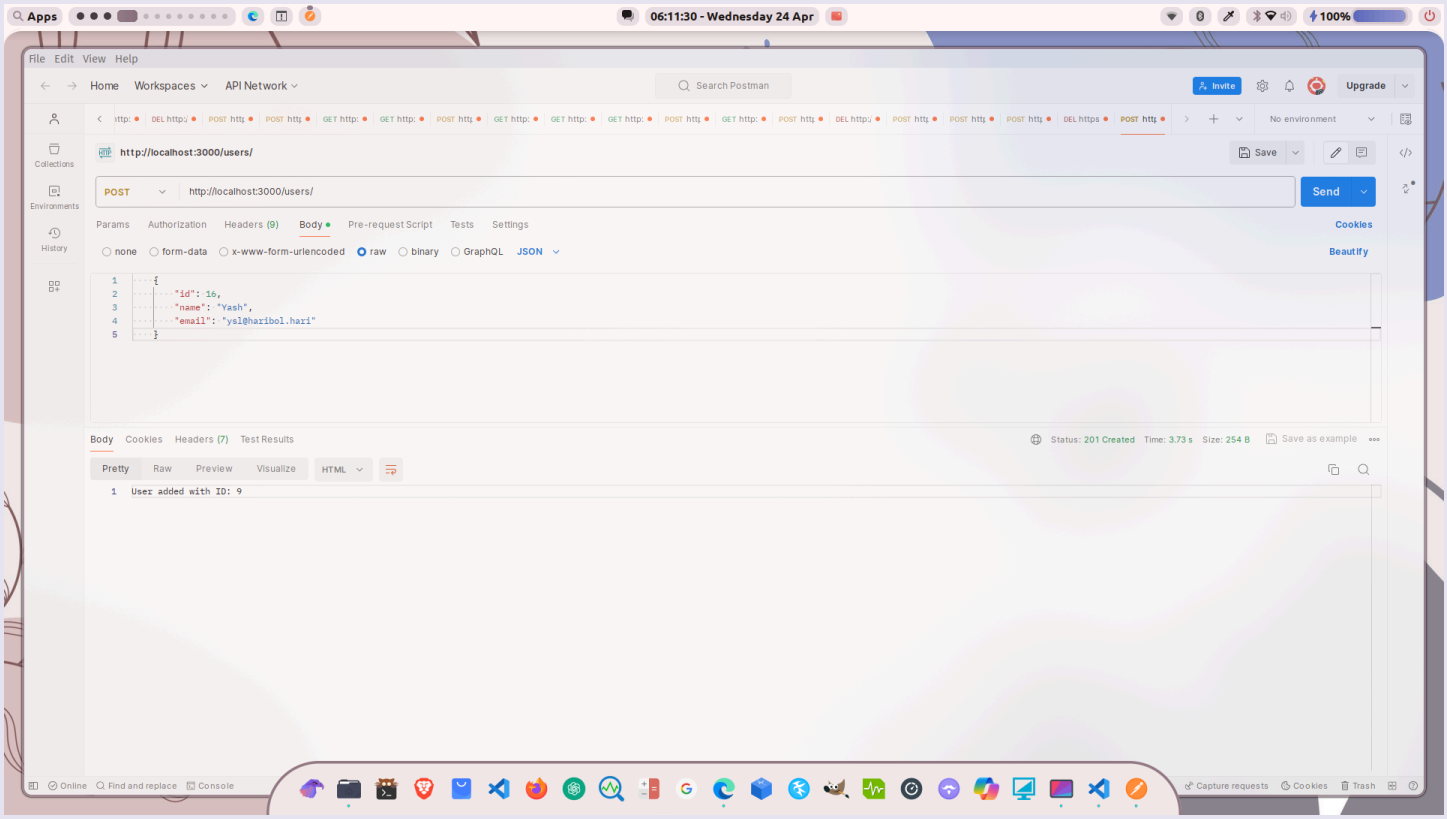
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

b) GET details of a user using ID



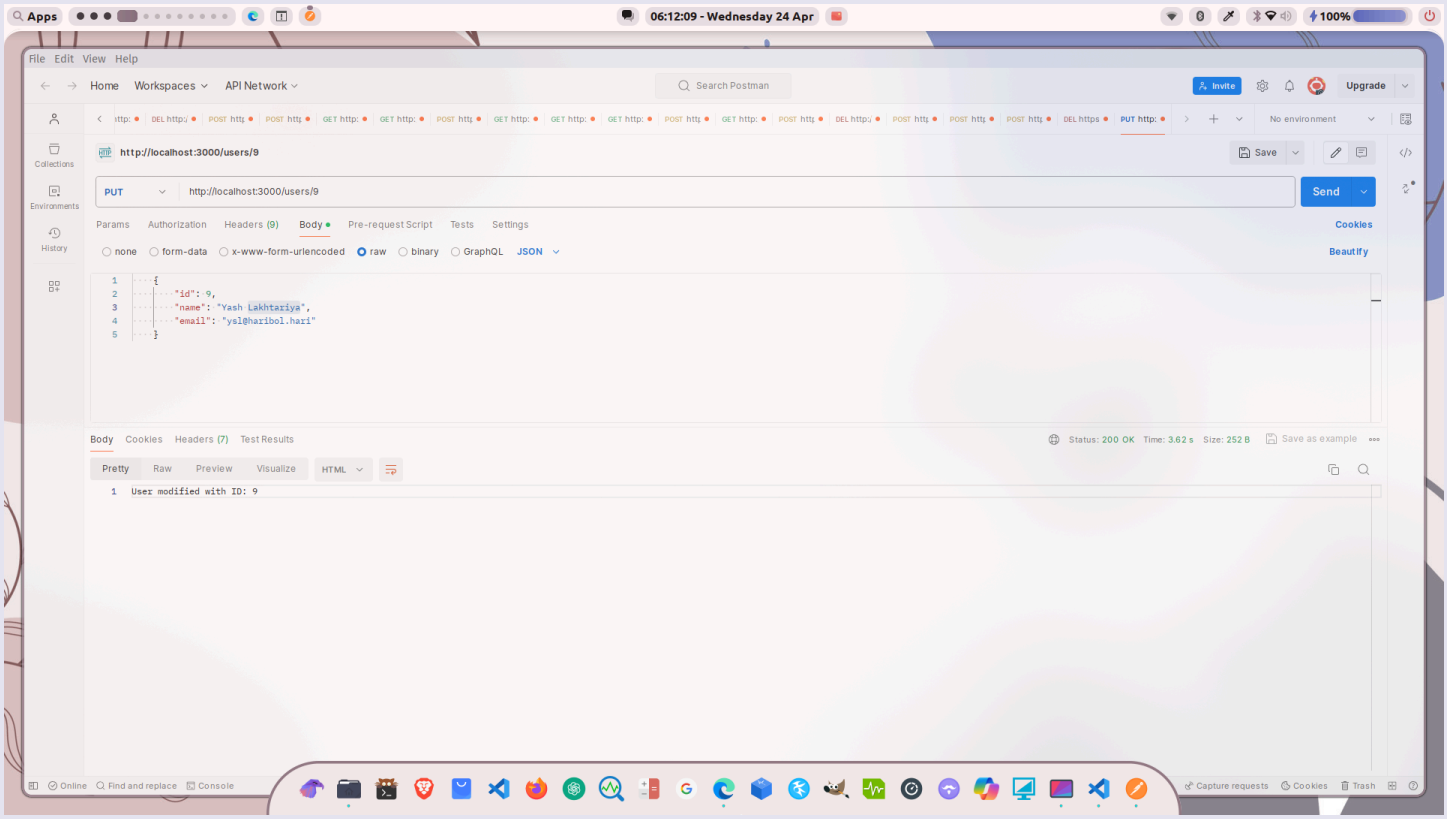
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

c) POST or add a user



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

d) PUT or update a user



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 19

e) DELETE a user