# Name :Rahav Patidar

# Email : [raghavpatidar997764@gmail.com](mailto:raghavpatidar997764@gmail.com)

# ***Probo Assingment***

I uses the **express** library to handle HTTP requests, and an array **jobList** to store the **<JobID, JobValue>** entities. The API methods are implemented as handlers for the **POST** and **GET** requests.

The **/add** API takes the **jobValue** and **jobid** parameters from the query string, adds them to the **jobList** array, and returns a JSON response with **stat: 'ok'**.

The **/all** API filters the **jobList** array based on the **jobValue** parameter, sorts the resulting array by **jobValue**, and returns it as a JSON response.

The **/remove** API removes the job with the specified **jobid** from the **jobList** array and returns a JSON response with **stat: 'ok'**. If the job is not found, it returns a **404** response with an error message.

I am using express-session to store data in local storage

In the **POST /add** endpoint, we store the updated **jobs** object in the session after adding a new job.

In the **GET /all** and **POST /remove** endpoints, we retrieve and manipulate the **jobs** object from the session instead of the global **jobs** variable.

**Code in app.js file :**

const express = require('express');

const session = require('express-session');

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(

    session({

        secret: 'secret-key',

        resave: false,

        saveUninitialized: false,

    })

);

let jobs = {};

app.post('/add', (req, res) => {

    const jobValue = req.query.jobValue || req.body.jobValue;

    const jobId = req.query.jobid || req.body.jobid;

    if (!jobValue || !jobId) {

        return res.status(400).json({ error: 'jobValue and jobId are required parameters' });

    }

    if (isNaN(jobValue)) {

        return res.status(400).json({ error: 'jobValue must be a number' });

    }

    jobs[jobId] = parseInt(jobValue, 10);

    req.session.jobs = jobs;

    res.status(200).json({ stat: 'ok' });

});

app.get('/all', (req, res) => {

    const startValue = req.query.jobValue;

    let filteredJobs = startValue

        ? Object.entries(req.session.jobs).filter(([jobId, jobValue]) => jobValue >= startValue)

        : Object.entries(req.session.jobs);

    const sortedJobs = filteredJobs.sort(([, a], [, b]) => a - b);

    res.json(sortedJobs);

});

app.post('/remove', (req, res) => {

    const jobId = req.query.jobid || req.body.jobid;

    if (!jobId) {

        return res.status(400).json({ error: 'jobId is a required parameter' });

    }

    if (!req.session.jobs[jobId]) {

        return res.status(404).json({ error: 'Job not found' });

    }

    delete req.session.jobs[jobId];

    res.status(200).json({ stat: 'ok' });

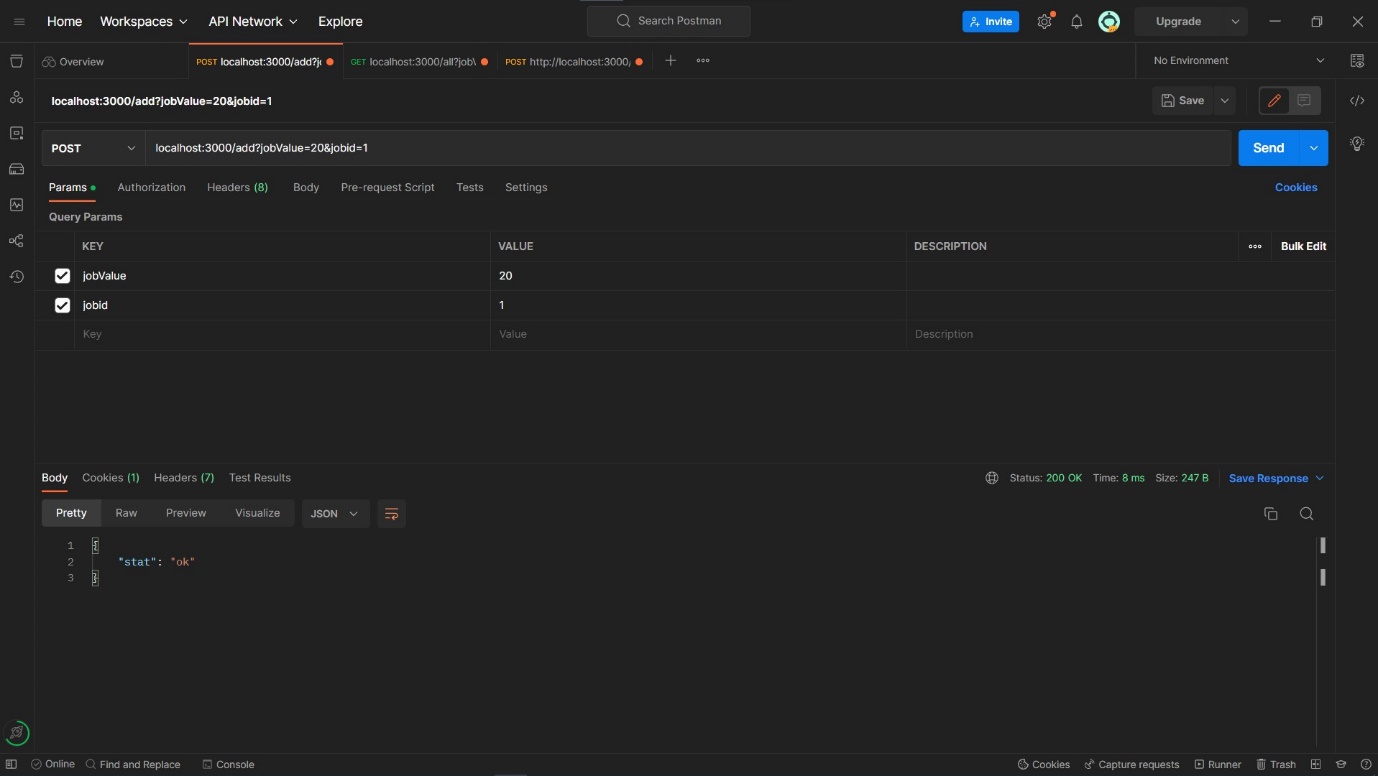
});

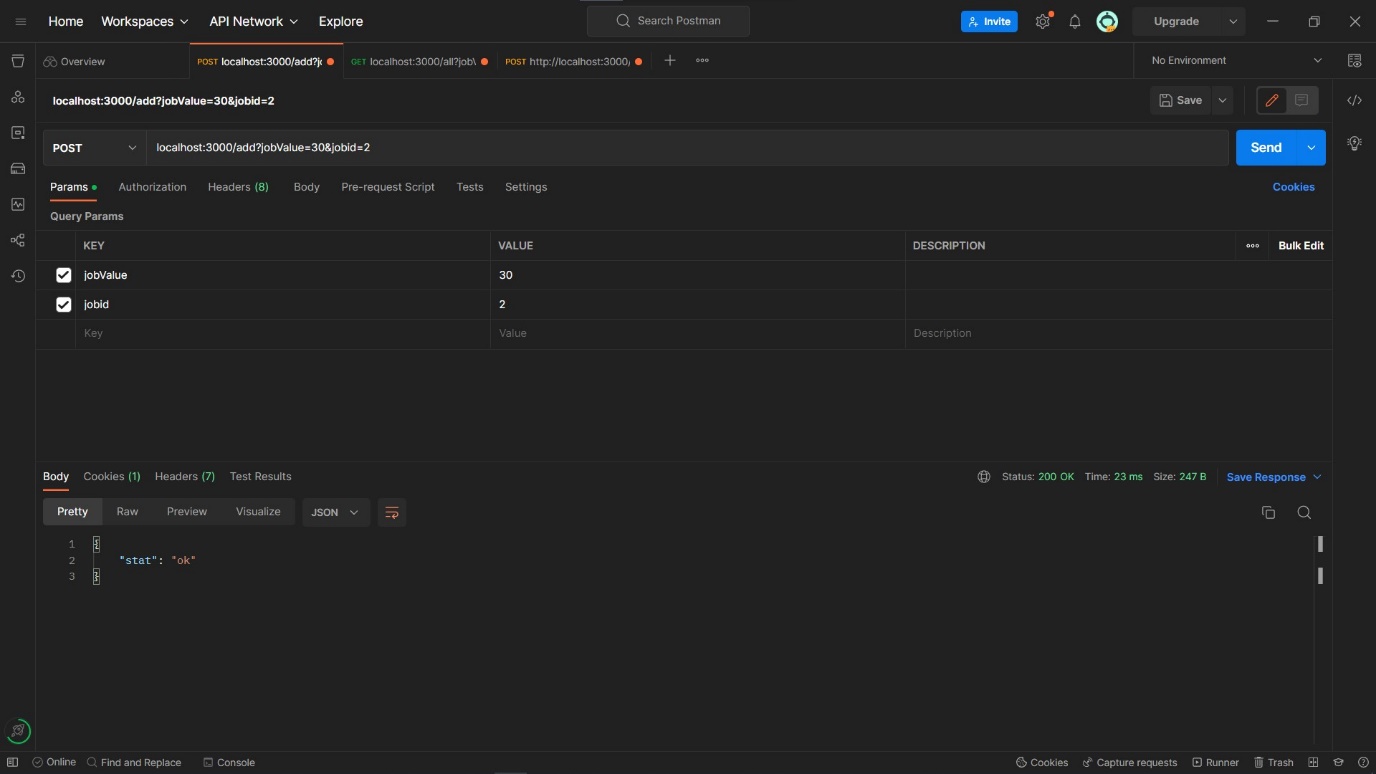
app.listen(3000, () => {

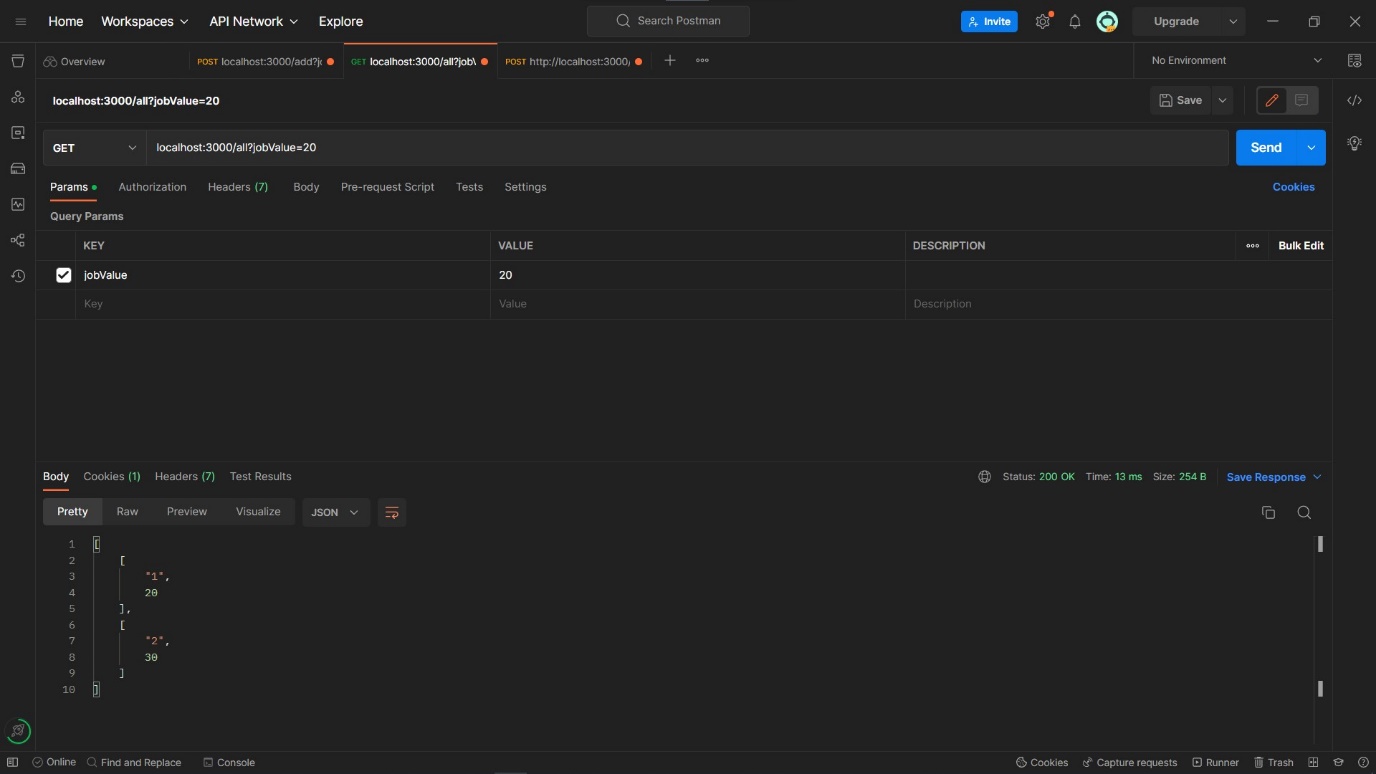
    console.log('Server started on http://localhost:3000');

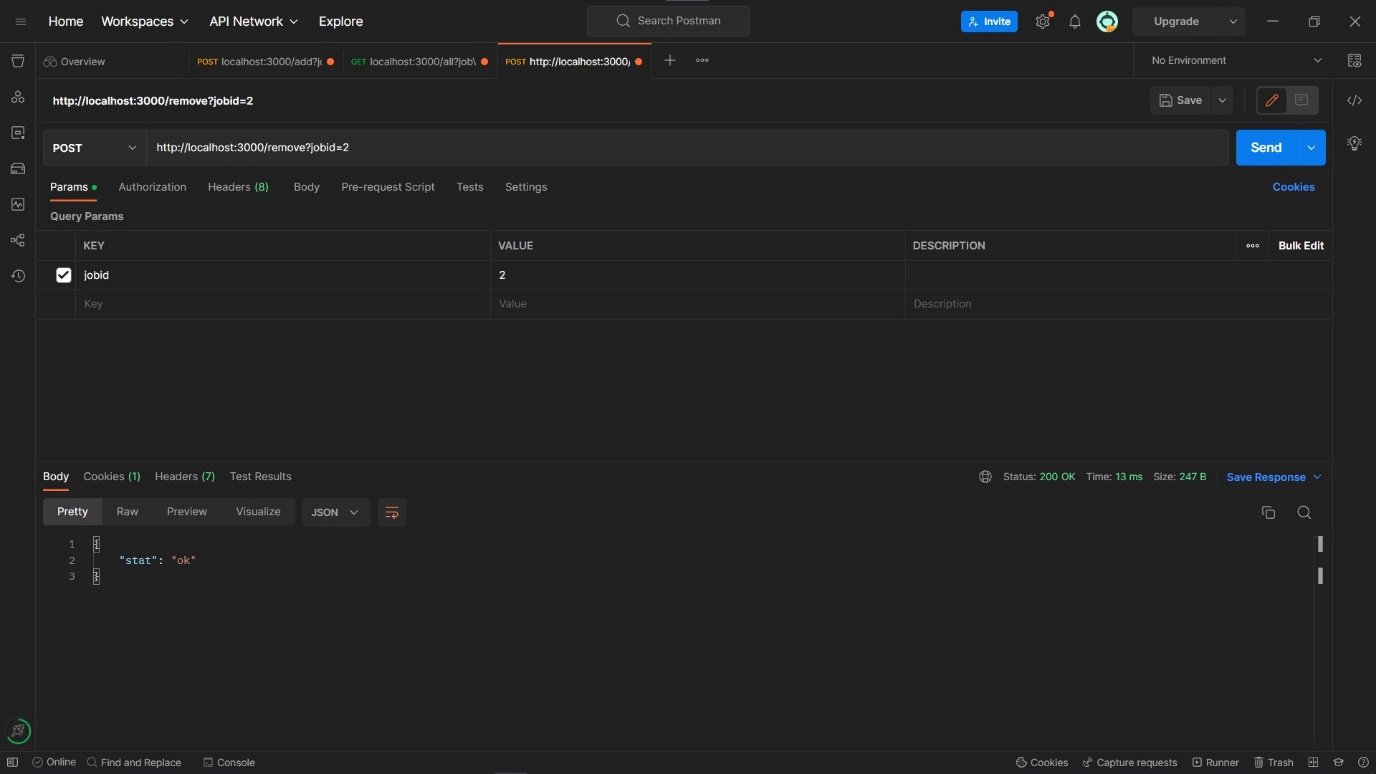
});

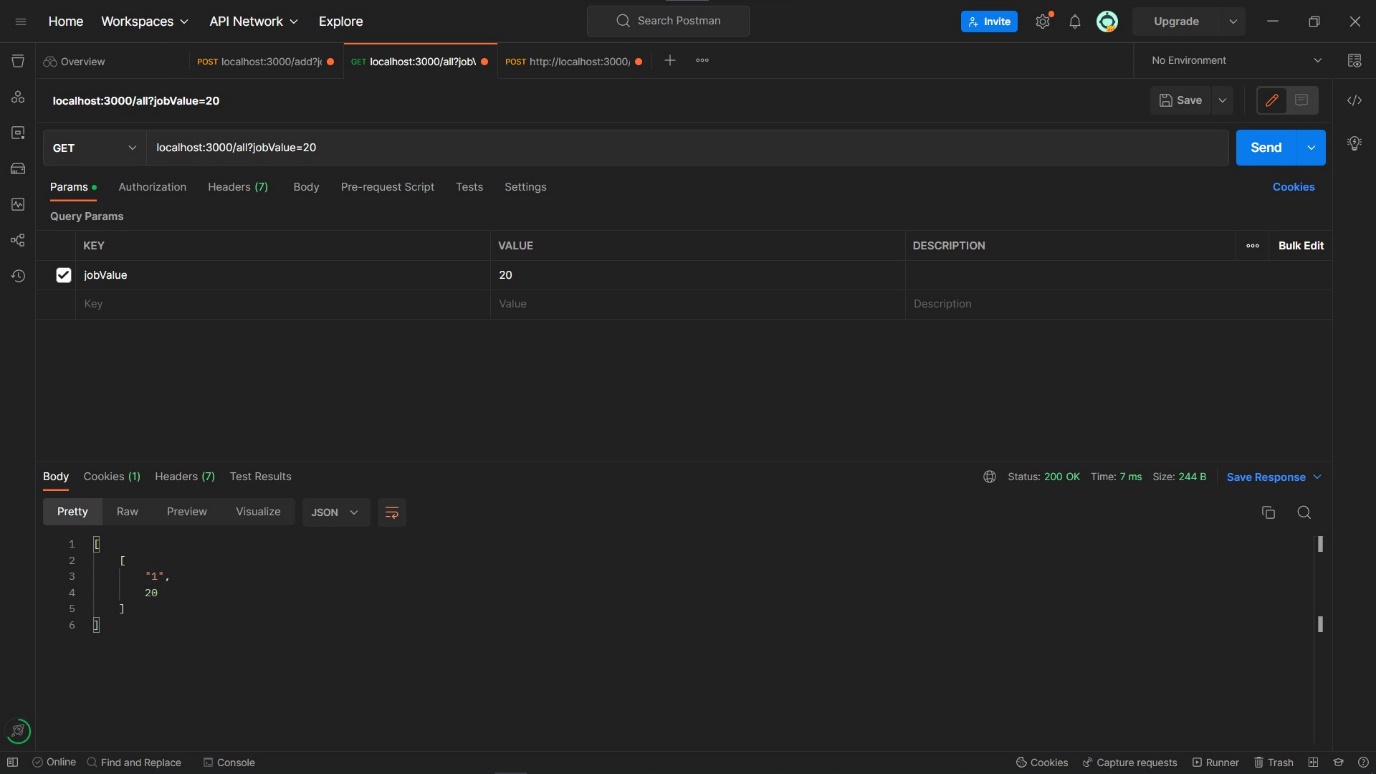
Output

**Screenshot1 : localhost:3000/add?jobValue=20&jobid=1**

**Screenshot2: localhost:3000/add?jobValue=30&jobid=2**

**Screenshot3: localhost:3000/all?jobValue=20 **

**Screenshot4: localhost:3000/remove?jobid=2**

**Screenshot5: localhost:3000/all?jobValue=20**

**So all three api endpoints are working fine**

**Github repo :-**