SOFTWARE ENGINEERING 1 – ASSIGNMENT 04

Name: Manasvini Nittala

NET ID: mn777

1. CODE:

File path:

'/Users/manasvininittala/Desktop/Rutgers/SUBJECTS/FALL_2023/SE/firstJasmineProject/jasmine/jasmine-standalone-5.1.1/spec/homework_four.js'

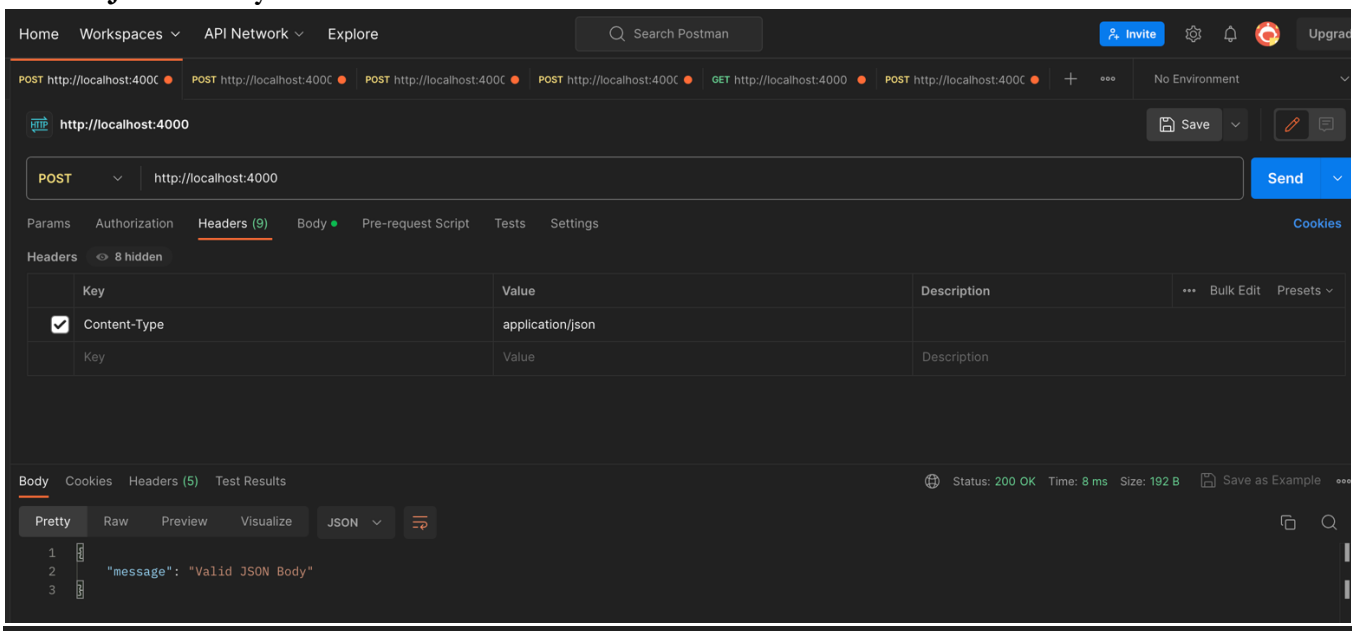
```
const http = require('http');


const supportedContentTypes = ['application/json', 'text/plain'];


const server = http.createServer((req, res) => {
  let requestBody = '';
  const contentType = req.headers['content-type'];

  if (req.url !== '/') {
    res.writeHead(400, { 'Content-Type': 'text/plain' });
    res.end('Incorrect URI. This server only supports http://localhost:4000.');
    console.log(
      'Sending Response: Incorrect URI. This server only supports http://localhost:4000.'
    );
    return;
  }

  if (req.method !== 'POST') {
    console.log('Unsupported Method');
    res.writeHead(405, { 'Content-Type': 'text/plain' });
    console.log('Sending Response: Method not allowed');
    res.end('Method not allowed. Use HTTP POST for this endpoint.');
    return;
  }

  if (!contentType) {
    console.log('Missing Content-Type Header');
    res.writeHead(415, { 'Content-Type': 'text/plain' });
    console.log('Sending Response: Missing Content-Type Header');
    res.end(
      'Missing Content-Type Header. Supported types are: application/json, text/plain'
    );
    return;
  }
  if (!supportedContentTypes.includes(contentType)) {
    console.log('Unsupported Content-Type');
    res.writeHead(415, { 'Content-Type': 'text/plain' });
```

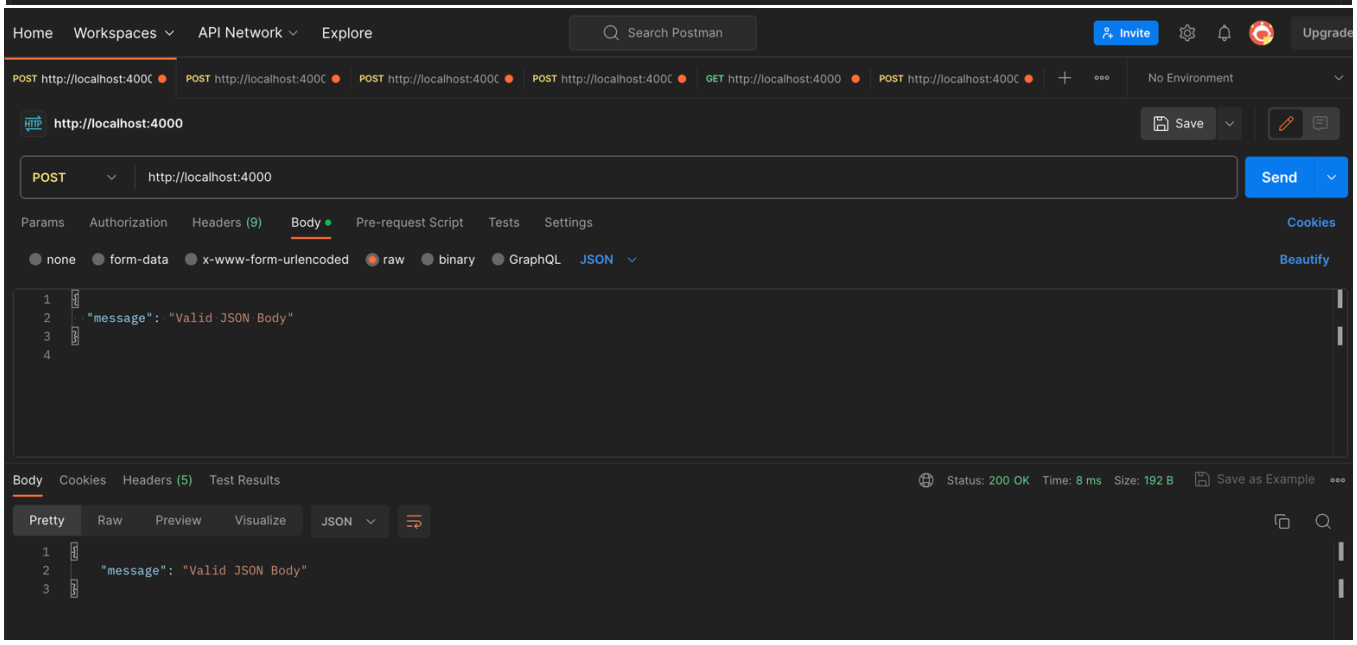
```
        console.log('Sending Response: Unsupported Content-Type');
        res.end(
            `Unsupported Content-Type. Supported types are: ${supportedContentTypes.join(
                ', '
            )}`
        );
        return;
    }

    req.on('data', (chunk) => {
        requestBody += chunk;
    });

    req.on('end', () => {
        if (requestBody.trim() === '') {
            console.log('Empty Request Body');
            res.writeHead(400, { 'Content-Type': 'text/plain' });
            console.log('Sending Response: Bad Request - Empty Request Body');
            res.end('Bad Request - Empty Request Body.');
            return;
        }

        try {
            let parsedBody;
            if (contentType === 'application/json') {
                parsedBody = JSON.parse(requestBody);
            } else if (contentType === 'text/plain') {
                parsedBody = requestBody;
            }

            console.log('Received Request Body:', parsedBody);

            res.writeHead(200, { 'Content-Type': contentType });
            console.log('Sending Response:', parsedBody);
            res.end(JSON.stringify(parsedBody));
        } catch (error) {
            console.log('Failed to parse the request body:', error);
            res.writeHead(400, { 'Content-Type': 'text/plain' });
            console.log(
                'Sending Response: Bad Request - Failed to parse the request body'
            );
            res.end('Bad Request - Failed to parse the request body.');
        }
    });
});

const PORT = process.env.PORT || 4000;
if (PORT !== 4000) {
```

```
  console.error('Incorrect port number provided. Server not started.');
} else {
  server.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
  });
}
```

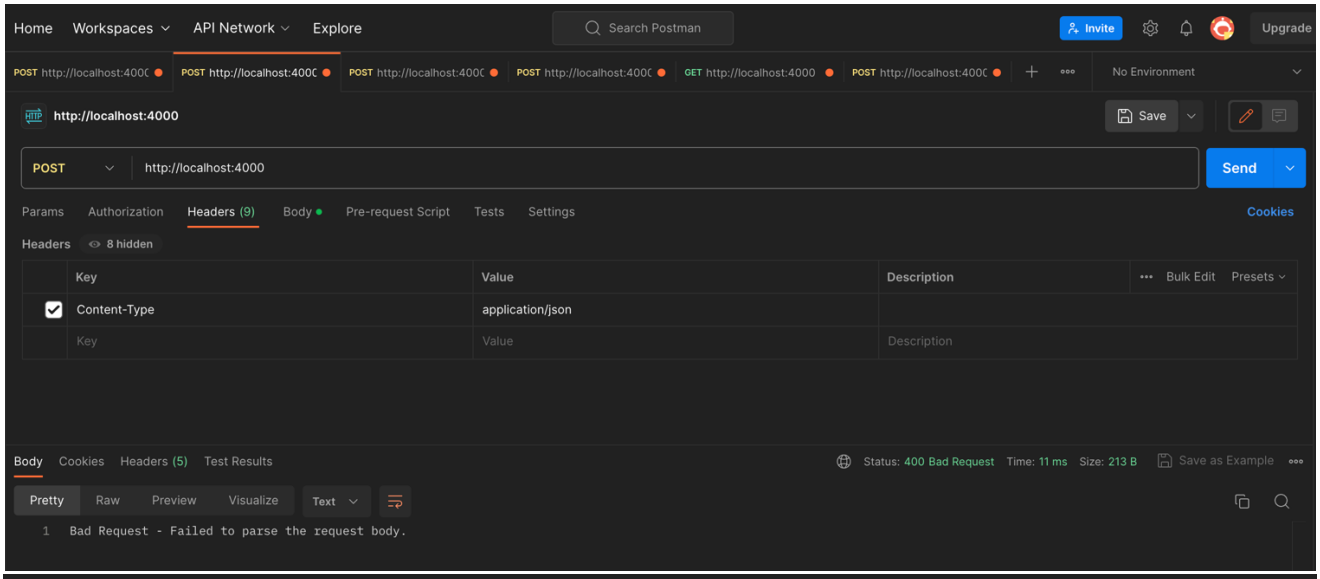
2. POSTMAN

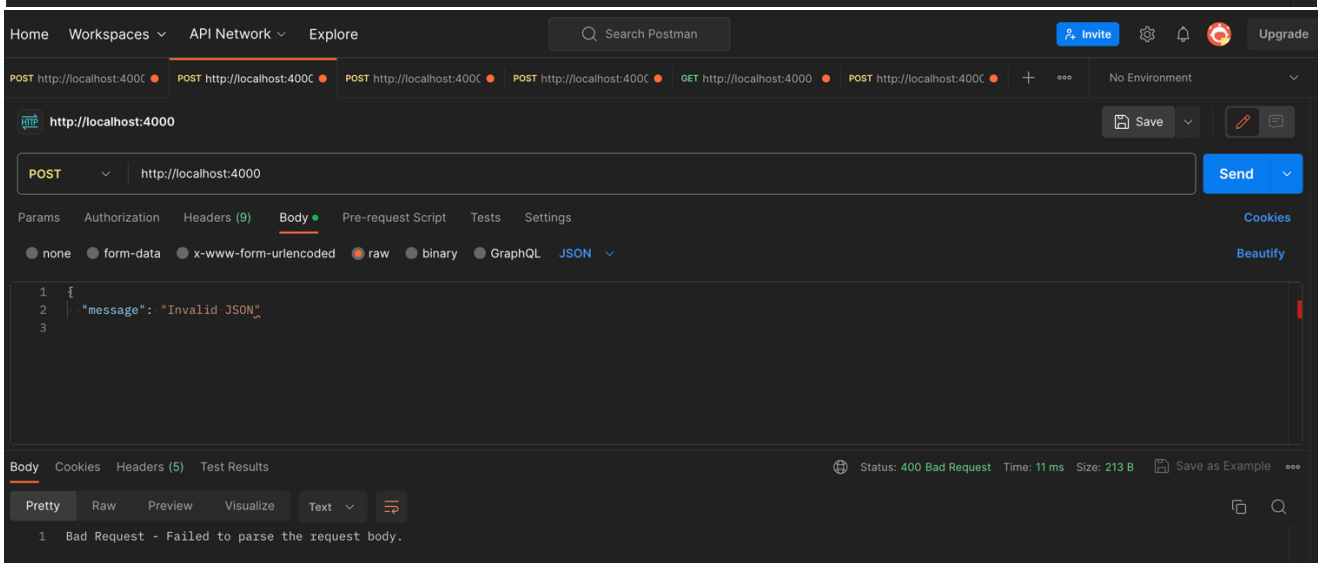
1. Valid JSON Body:



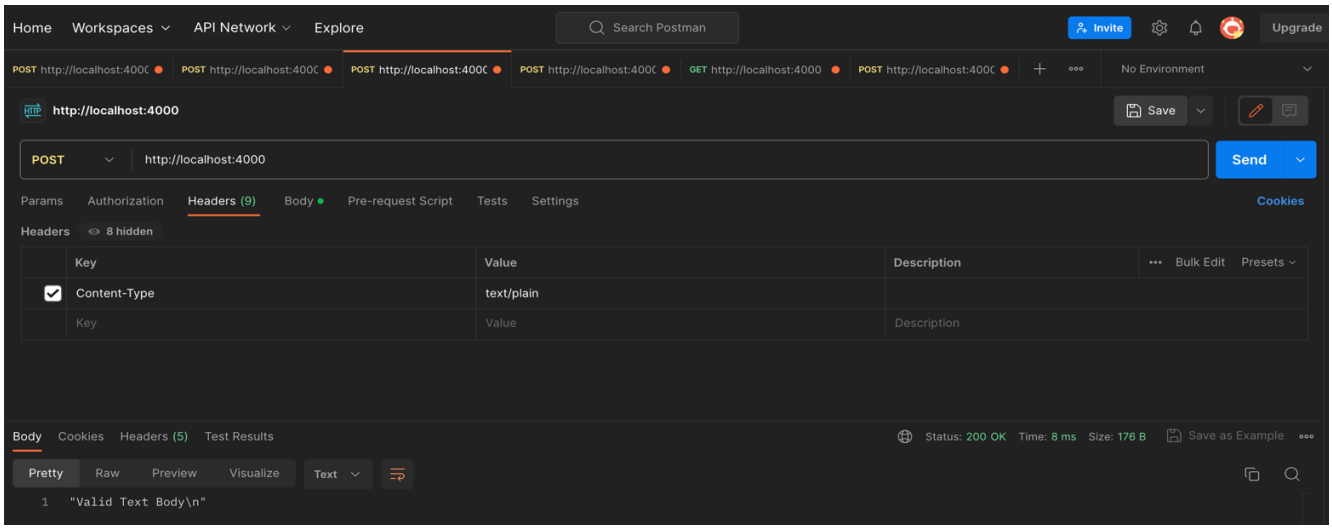


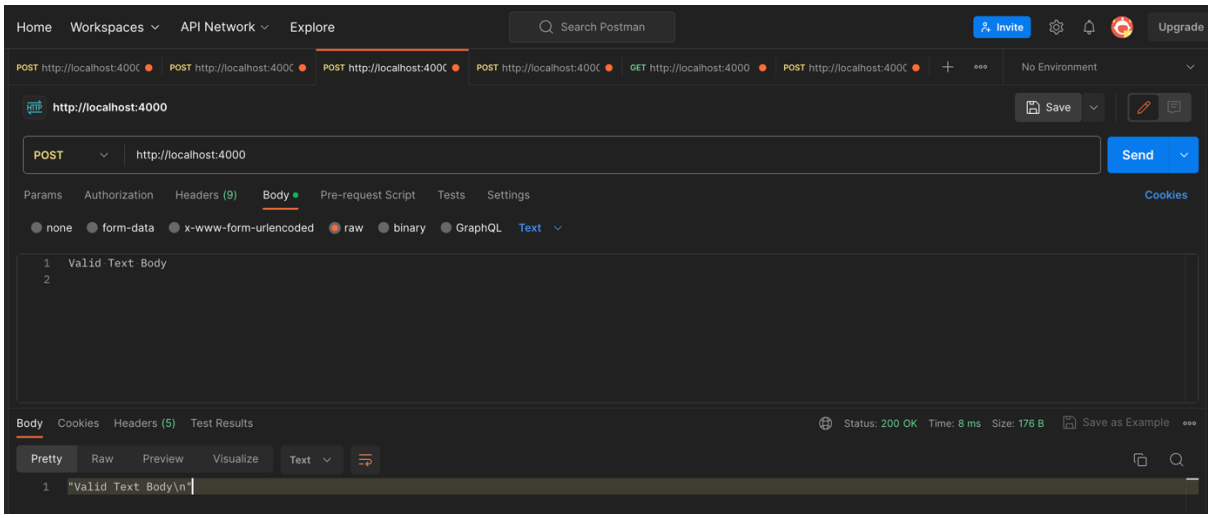
2. Invalid JSON Body:



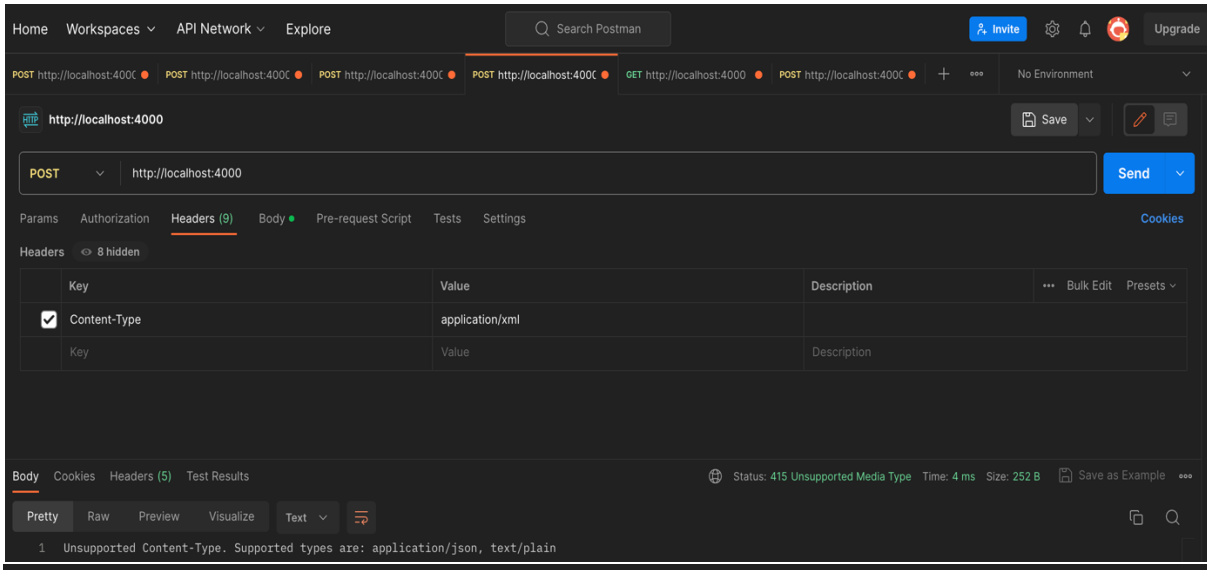


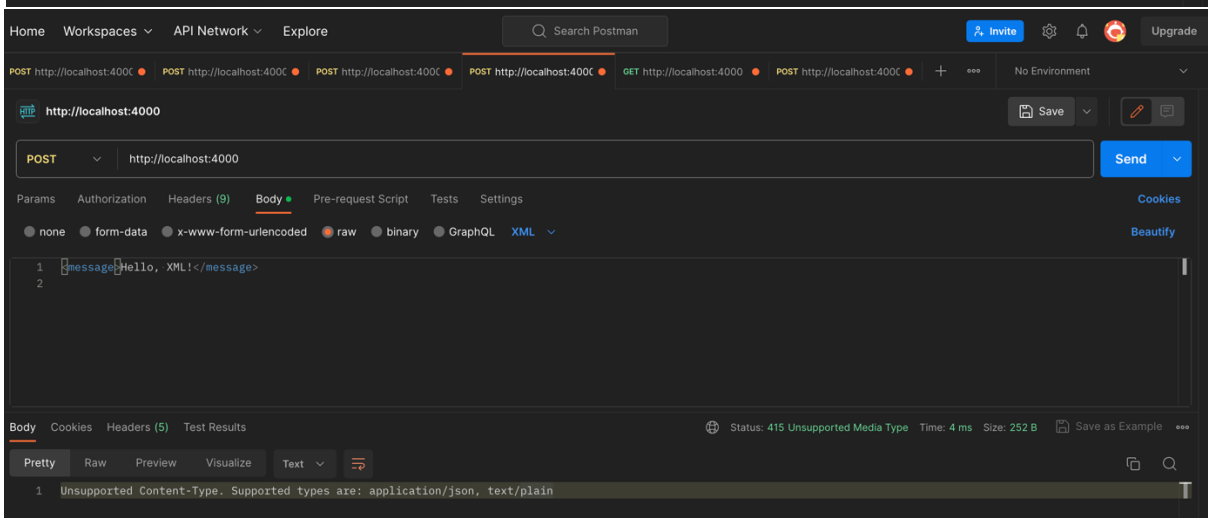
3. Valid Text Body



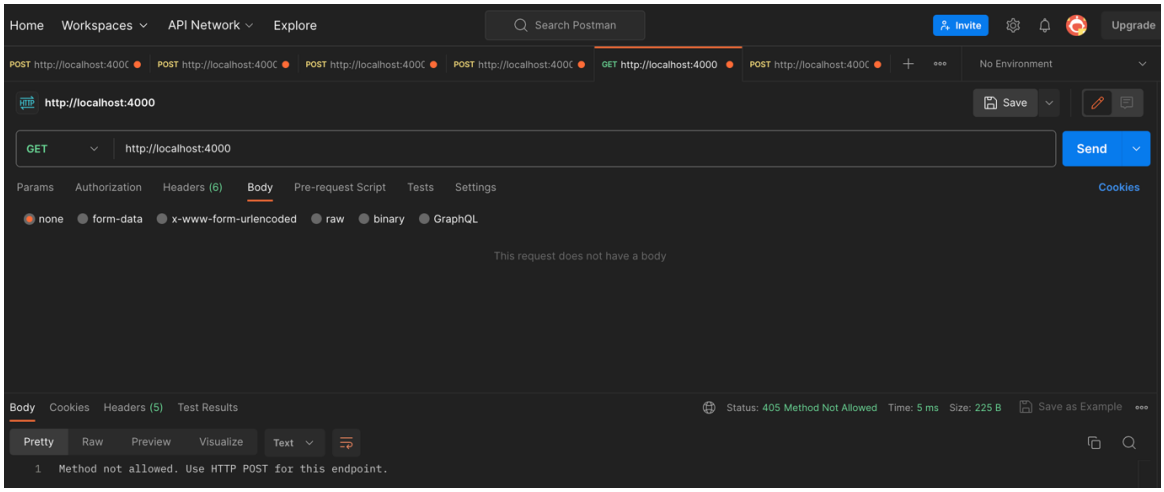

Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
GET http://localhost:4000
POST http://localhost:400
No Environment
http://localhost:4000
Save
POST
http://localhost:4000
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Text
Valid Text Body
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 8 ms
Size: 176 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
"Valid Text Body\n"

4. XML Body:

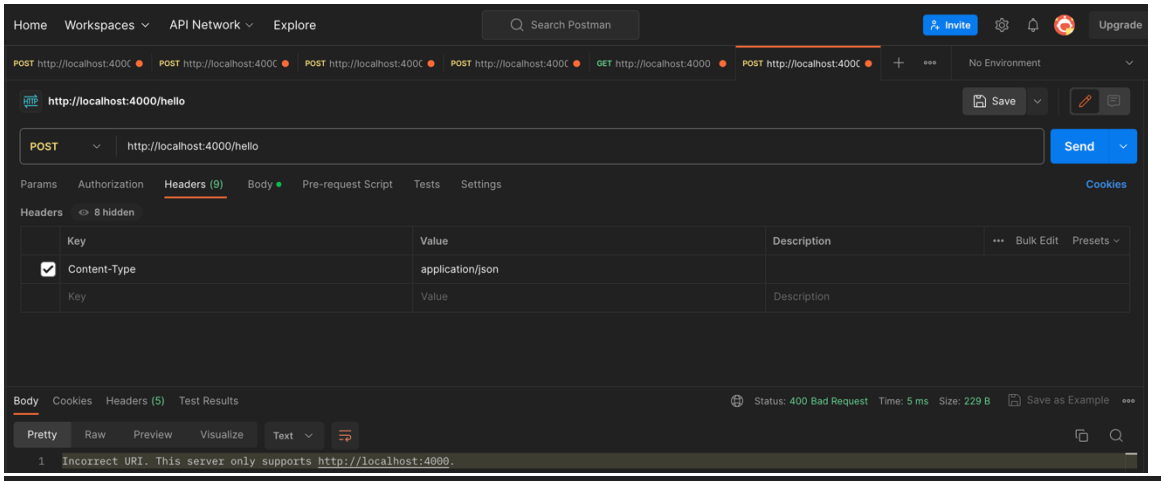

Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
GET http://localhost:4000
POST http://localhost:400
No Environment
http://localhost:4000
Save
POST
http://localhost:4000
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Headers
8 hidden
Key
Value
Description
Bulk Edit
Presets
Content-Type
application/xml
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 415 Unsupported Media Type
Time: 4 ms
Size: 252 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
Unsupported Content-Type. Supported types are: application/json, text/plain

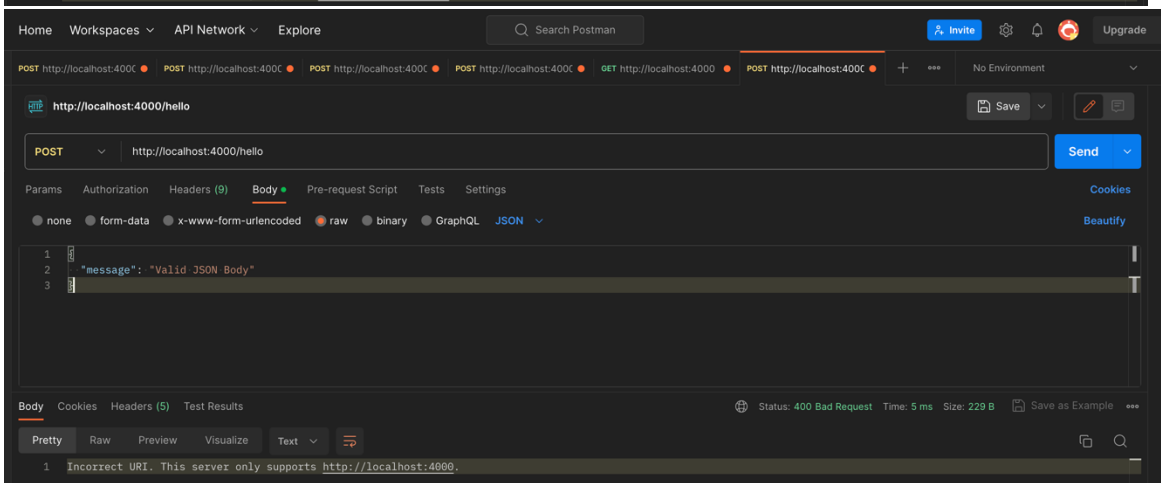

Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
POST http://localhost:400
GET http://localhost:4000
POST http://localhost:400
No Environment
http://localhost:4000
Save
POST
http://localhost:4000
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
XML
Beautify
<message>Hello, XML!</message>
Body
Cookies
Headers (5)
Test Results
Status: 415 Unsupported Media Type
Time: 4 ms
Size: 252 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
Unsupported Content-Type. Supported types are: application/json, text/plain

5. GET Request:



6. Incorrect URI:





7. Console:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                                    Code

manasvininittala@nbp-136-98 SE % node "/Users/manasvininittala/Desktop/Rutgers/SUBJECTS/FALL_2023/SE/firstJasmineProject/jasmine/jasmine-standalone-5.1.1/spec/homework_four.js"
Server is running on port 4000
Received Request Body: { message: 'Valid JSON Body' }
Sending Response: { message: 'Valid JSON Body' }
Failed to parse the request body: SyntaxError: Expected ',' or '}' after property value in JSON at position 30
    at JSON.parse (<anonymous>)
    at IncomingMessage.<anonymous> (/Users/manasvininittala/Desktop/Rutgers/SUBJECTS/FALL_2023/SE/firstJasmineProject/jasmine/jasmine-standalone-5.1.1/spec/homework_four.js:63:23)
    at IncomingMessage.emit (node:events:514:28)
    at endReadableNT (node:internal/streams/readable:1408:12)
    at process.processTicksAndRejections (node:internal/process/task_queues:82:21)
Sending Response: Bad Request - Failed to parse the request body
Received Request Body: Valid Text Body

Sending Response: Valid Text Body

Unsupported Content-Type
Sending Response: Unsupported Content-Type
Unsupported Method
Sending Response: Method not allowed
Sending Response: Incorrect URI. This server only supports http://localhost:4000.
^C
manasvininittala@nbp-136-98 SE %
```

3. JASMINE TESTING: USING AXIOS

CODE: '/Users/manasvininittala/Desktop/Rutgers/SUBJECTS/FALL_2023/SE/firstJasmineProj ect/jasmine/jasmine-standalone-5.1.1/spec/hfourSpec.js'

```
const axios = require('axios');
const { v4: uuidv4 } = require('uuid');
const server = require('./homework_four');
const PORT = process.env.PORT || 4000;

function generateLargePayload() {
  const largeData = [];
  for (let i = 0; i < 10000; i++) {
    largeData.push({ key: 'value' });
  }

  return JSON.stringify(largeData);
}
describe('Server', () => {
  it('should be running and respond to POST requests', async () => {
    console.log('Should be running and respond to POST requests');
    try {
      const response = await axios.post(`http://localhost:${PORT}`, {
        key: 'working!',
      });
      console.log('Response status:', response.status);
      expect(response.status).toBe(200);
      console.log(
        '--------------------------------------------------------------------------------'
      );
    } catch (error) {
      console.log('Request failed with error:', error);
      fail(`Request failed with error: ${error}`);
    }
  });
  it('should handle a POST request with a JSON payload and return the same JSON', async () => {
    console.log(
      'Should handle a POST request with a JSON payload and return the same JSON'
```

```
    );
    try {
        const requestData = { key: 'json-test' };
        const response = await axios.post(
            `http://localhost:${PORT}`,
            requestData,
            {
                headers: { 'Content-Type': 'application/json' },
            }
        );

        console.log('Request data:', requestData);
        console.log('Response data:', response.data);
        console.log('Response status:', response.status);
        expect(response.status).toBe(200);
        expect(response.data).toEqual(requestData);
        console.log(
            '--------------------------------------------------------------------------------'
        );
    } catch (error) {
        console.log('Request failed with error:', error);
        fail(`Request failed with error: ${error}`);
    }
});
it('should fail with status code 405 for GET requests', async () => {
    console.log('Should fail with status code 405 for GET requests');
    try {
        const response = await axios.get(`http://localhost:${PORT}`);

        fail('Expected a 405 status code for GET requests');
    } catch (error) {
        console.log(
            'GET request failed with status code:',
            error.response.status
        );
        expect(error.response.status).toBe(405);
        console.log(
            '--------------------------------------------------------------------------------'
        );
    }
});
it('should handle a POST request with plain text and return the same text', async () => {
    console.log(
        'Should handle a POST request with plain text and return the same text'
    );
    try {
        const requestData = 'Plain text data Test';
        const response = await axios.post(
```

```
            `http://localhost:${PORT}`,
            requestData,
            {
                headers: { 'Content-Type': 'text/plain' },
            }
        );
        console.log('Request data:', requestData);
        console.log('Response data:', response.data);
        console.log('Response status:', response.status);
        expect(response.status).toBe(200);
        expect(response.data).toBe(requestData);
        console.log(
            '--------------------------------------------------------------------------'
        );
    } catch (error) {
        console.log('Request failed with error:', error);
        fail(`Request failed with error: ${error}`);
    }
});
it('should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header', async () => {
    console.log(
        'Should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header'
    );
    try {
        const requestData = 'Plain text data';
        const response = await axios.post(
            `http://localhost:${PORT}`,
            requestData
        );
        fail(
            'Expected a 415 status code for POST requests without Content-Type header'
        );
    } catch (error) {
        console.log(
            'Failed for Unsupported Media Content for POST requests with status code:',
            error.response.status
        );
        expect(error.response.status).toBe(415);
        console.log(
            '--------------------------------------------------------------------------'
        );
    }
});
it('should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header', async () => {
    console.log(
        'Should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header'
    );
    try {
```

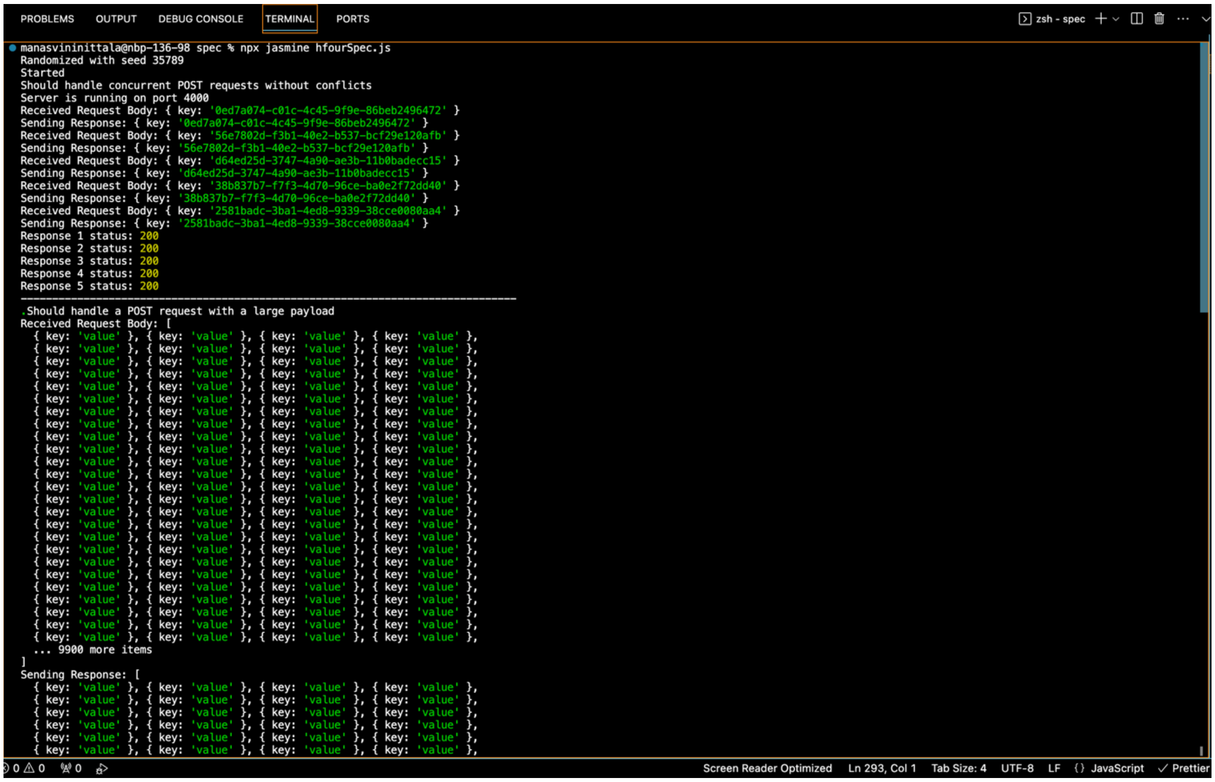
```
        const requestData = 'Plain text data';
        const response = await axios.post(
            `http://localhost:${PORT}`,
            requestData
        );
        fail(
            'Expected a 415 status code for POST requests without Content-Type header'
        );
    } catch (error) {
        console.log(
            'Failed for Unsupported Media Type for POST Requests with status code:',
            error.response.status
        );
        expect(error.response.status).toBe(415);
        console.log(
            '--------------------------------------------------------------------------'
        );
    }
});
it('should return a 400 Bad Request for POST requests with invalid JSON content', async () => {
    console.log(
        'Should return a 400 Bad Request for POST requests with invalid JSON content'
    );
    try {
        const requestData = '{ key value ';
        const response = await axios.post(
            `http://localhost:${PORT}`,
            requestData,
            {
                headers: { 'Content-Type': 'application/json' },
            }
        );
        console.log(
            'Should return a 400 Bad Request for POST requests with invalid JSON content'
        );
        console.log(
            '--------------------------------------------------------------------------'
        );
        expect(response.status).toBe(200);
    } catch (error) {
        console.log('Request failed with error:', error);
        fail(`Request failed with error: ${error}`);
    }
});
it('should handle concurrent POST requests without conflicts', async () => {
    console.log('Should handle concurrent POST requests without conflicts');
    try {
        const numRequests = 5;
```

```
        const promises = [];
        for (let i = 0; i < numRequests; i++) {
            const requestData = { key: uuidv4() };
            const promise = axios.post(`http://localhost:${PORT}`, requestData, {
                headers: { 'Content-Type': 'application/json' },
            });
            promises.push(promise);
        }

        const responses = await Promise.all(promises);
        responses.forEach((response, index) => {
            console.log(`Response ${index + 1} status:`, response.status);
            expect(response.status).toBe(200);
        });
        console.log(
            '--------------------------------------------------------------------------'
        );
    } catch (error) {
        console.log('Request failed with error:', error);
        fail(`Request failed with error: ${error}`);
    }
});
it('should handle a POST request with a large payload', async () => {
    console.log('Should handle a POST request with a large payload');
    try {
        const largePayload = generateLargePayload();
        const response = await axios.post(
            `http://localhost:${PORT}`,
            largePayload,
            {
                headers: { 'Content-Type': 'application/json' },
            }
        );
        console.log('Response status:', response.status);
        expect(response.status).toBe(200);
        console.log(
            '--------------------------------------------------------------------------'
        );
    } catch (error) {
        console.log('Request failed with error:', error);
        fail(`Request failed with error: ${error}`);
    }
});
it('should handle a POST request with an empty request body', async () => {
    console.log(
        'Should fail with status code 400 for a POST request with empty body'
    );
    try {
```

```
        await axios.post(`http://localhost:${PORT}`, '', {
            headers: { 'Content-Type': 'text/plain' },
        });
        fail('Expected a 400 status code for an empty request body');
    } catch (error) {
        console.log(
            'Failed for empty request body for POST Requests with status code:',
            error.response.status
        );
        expect(error.response.status).toBe(400);
        console.log(
            '-------------------------------------------------------------------------------'
        );
    }
});
it('should handle an incorrect port number', async () => {
    console.log('Should handle an incorrect port number');
    const incorrectPort = 3000;
    try {
        const response = await axios.get(`http://localhost:3000`);

        fail('Expected a different response');
    } catch (error) {
        console.log('Request failed: Incorrect PORT number', incorrectPort);
        expect(error.message).toMatch('');
        console.log(
            '-------------------------------------------------------------------------------'
        );
    }
});
it('should handle an incorrect URI', async () => {
    const incorrectURI = 'http://localhost:4000/hello';
    console.log('Should handle an incorrect URI');
    try {
        const response = await axios.get(incorrectURI);
        fail('Expected a different response');
    } catch (error) {
        console.log('Request failed with error: Incorrect URI:', incorrectURI);
        expect(error.message).toMatch('Request failed with status code 400');
        console.log(
            '-------------------------------------------------------------------------------'
        );
    }
});
it('should respond with the same JSON body', async () => {
    console.log('Should respond with the same JSON body');
    const jsonData = { key: 'working!' };
    const jsonDataString = JSON.stringify(jsonData);
```

```
        try {
            const response = await axios.post(`http://localhost:${PORT}`, jsonData, {
                headers: { 'Content-Type': 'application/json' },
            });
            expect(response.status).toBe(200);
            expect(response.headers['content-type']).toBe('application/json');
            res = JSON.stringify(response.data);
            expect(res).toEqual(jsonDataString);
            console.log('Response Headers:', response.headers);
            console.log('Response Data:', response.data);
            console.log('Response status:', response.status);
            console.log(
                '--------------------------------------------------------------------------------'
            );
        } catch (error) {
            fail(error);
        }
    });
});
```

OUTPUT:



```
manasvininittala@nbp-136-98 spec % npx jasmine hfourSpec.js
Randomized with seed 35789
Started
Should handle concurrent POST requests without conflicts
Server is running on port 4000
Received Request Body: { key: '0ed7a074-c01c-4c45-9f9e-86beb2496472' }
Sending Response: { key: '0ed7a074-c01c-4c45-9f9e-86beb2496472' }
Received Request Body: { key: '56e7802d-f3b1-40e2-b537-bcf29e120afb' }
Sending Response: { key: '56e7802d-f3b1-40e2-b537-bcf29e120afb' }
Received Request Body: { key: 'd64ed25d-3747-4a90-ae3b-11b0badecc15' }
Sending Response: { key: 'd64ed25d-3747-4a90-ae3b-11b0badecc15' }
Received Request Body: { key: '38b837b7-f7f3-4d70-96ce-ba0e2f72dd40' }
Sending Response: { key: '38b837b7-f7f3-4d70-96ce-ba0e2f72dd40' }
Received Request Body: { key: '2581badc-3ba1-4ed8-9339-38cce0080aa4' }
Sending Response: { key: '2581badc-3ba1-4ed8-9339-38cce0080aa4' }
Response 1 status: 200
Response 2 status: 200
Response 3 status: 200
Response 4 status: 200
Response 5 status: 200
--------------------------------------------------------------------------------
.Should handle a POST request with a large payload
Received Request Body: [
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  ... 9900 more items
]
Sending Response: [
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                                                    zsh - spec

  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  { key: 'value' }, { key: 'value' }, { key: 'value' }, { key: 'value' },
  ... 9900 more items
]
Response status: 200
--------------------------------------------------------------------------------
.Should handle a POST request with a JSON payload and return the same JSON
Received Request Body: { key: 'json-test' }
Sending Response: { key: 'json-test' }
Request data: { key: 'json-test' }
Response data: { key: 'json-test' }
Response status: 200
--------------------------------------------------------------------------------
.Should fail with status code 405 for GET requests
Unsupported Method
Sending Response: Method not allowed
GET request failed with status code: 405
--------------------------------------------------------------------------------
.Should be running and respond to POST requests
Received Request Body: { key: 'working!' }
Sending Response: { key: 'working!' }
Response status: 200
--------------------------------------------------------------------------------
.Should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header
Unsupported Content-Type
Sending Response: Unsupported Content-Type
Failed for Unsupported Media Type for POST Requests with status code: 415
--------------------------------------------------------------------------------
.Should return a 400 Bad Request for POST requests with invalid JSON content
Received Request Body: { key value
Sending Response: { key value
Should return a 400 Bad Request for POST requests with invalid JSON content
--------------------------------------------------------------------------------
.Should handle an incorrect port number
Request failed: Incorrect PORT number 3000
--------------------------------------------------------------------------------
.Should respond with the same JSON body
Received Request Body: { key: 'working!' }
Sending Response: { key: 'working!' }
Response Headers: Object [AxiosHeaders] {
  'content-type': 'application/json',
  date: 'Mon, 23 Oct 2023 21:55:42 GMT',
  connection: 'keep-alive',
  'keep-alive': 'timeout=5',
  'transfer-encoding': 'chunked'
}
Response Data: { key: 'working!' }
Response status: 200
--------------------------------------------------------------------------------
.Should handle a POST request with plain text and return the same text
Received Request Body: Plain text data Test
Sending Response: Plain text data Test
Request data: Plain text data Test
Response data: Plain text data Test
Response status: 200
⊗ 0 ⚠ 0   0                                  Screen Reader Optimized   Ln 293, Col 1   Tab Size: 4   UTF-8   LF   {} JavaScript   ✓ Prettier
```

```
--------------------------------------------------------------------------------
.Should handle an incorrect URI
Sending Response: Incorrect URI. This server only supports http://localhost:4000.
Request failed with error: Incorrect URI: http://localhost:4000/hello
--------------------------------------------------------------------------------
.Should fail with status code 415 Unsupported Media Type for POST requests without Content-Type header
Unsupported Content-Type
Sending Response: Unsupported Content-Type
Failed for Unsupported Media Content for POST requests with status code: 415
--------------------------------------------------------------------------------
.Should fail with status code 400 for a POST request with empty body
Empty Request Body
Sending Response: Bad Request - Empty Request Body
Failed for empty request body for POST Requests with status code: 400
--------------------------------------------------------------------------------
.


13 specs, 0 failures
Finished in 0.096 seconds
Randomized with seed 35789 (jasmine --random=true --seed=35789)
○ manasvininittala@nbp-136-98 spec %
⊗ 0 ⚠ 0   0                                  Screen Reader Optimized   Ln 293, Col 1   Tab Size: 4   UTF-8   LF   {} JavaScript   ✓ Prettier
```

NOTE:

- **RUN the "homework_four.js" file and the "hfourSpec.js" file separately. If they are run together a port already in use error occurs. I have called the "homework_four.js" in the "hfourSpec.js" too, so to check jasmine tests running the "hfourSpec.js" file is sufficient. Similarly, to check for postman "homework_four.js" file is sufficient.**
- **I have used "axios" instead of "supertest" as I am familiar with "axios". I have used 2 content types. "application/json" and "text/plain". This code satisfies all other content types too if they must be included in the content-types array. For demonstration and code purposes I have taken only 2 content types.**