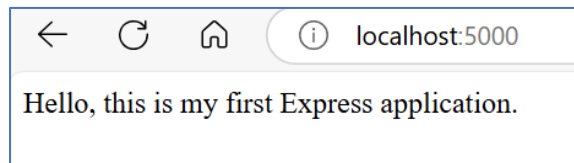


Week 4

Exercise 1:

Output:

```
Server is running on port 5000
```

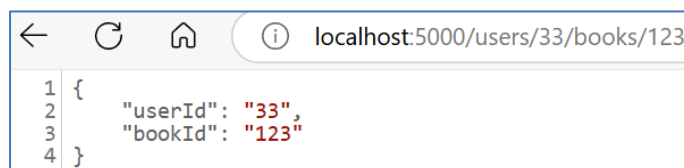
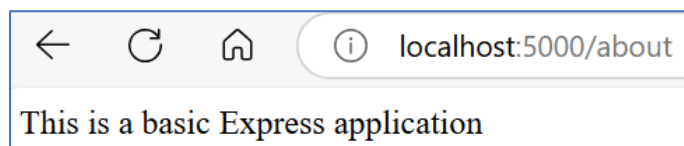


Explanation:

I established the default route and sent a string message using the `.get` route method and its parameter `res`.

Exercise 2:

Output:



Explanation:

I created additional routes: `/about` and `/users...`. The second route included parameters which I accessed with the use of `req.params`.

Exercise 3:

Output:



```
localhost:5000/Getstudent/1
1 {
2   "Name": "Jonhthon",
3   "Age": 33,
4   "Qualification": "BSC",
5   "Email": "std123@gm.com",
6   "Id": 1
7 }
```

```
localhost:5000/Getstudent/2
1 {
2   "Name": "David",
3   "Age": 23,
4   "Qualification": "HNC",
5   "Email": "abc@gm.com",
6   "Id": 2
7 }
```

```
localhost:5000/Getstudent/3
1 {
2   "Name": "Emily",
3   "Age": 25,
4   "Qualification": "A-level",
5   "Email": "email@gm.com",
6   "Id": 3
7 }
```

```
localhost:5000/Getstudent/4
1 {
2   "Status": true,
3   "Status Code": 200,
4   "Requested URL": "/Getstudent/4",
5   "Requested Method": "GET",
6   "Student Data": {
7     "Student1": {
8       "Name": "Jonhthon",
9       "Age": 33,
10      "Qualification": "BSC",
11      "Email": "std123@gm.com",
12      "Id": 1
13    },
14    "Student2": {
15      "Name": "David",
16      "Age": 23,
17      "Qualification": "HNC",
18      "Email": "abc@gm.com",
19      "Id": 2
20    },
21    "Student3": {
22      "Name": "Emily",
23      "Age": 25,
24      "Qualification": "A-level",
25      "Email": "email@gm.com",
26      "Id": 3
27    }
28  }
29 }
```

Explanation:

I created another route that reads and parses a JSON file and sends it to the browser using the response object. I then created another route with an id parameter which filtered out the record with the matching id. The last screenshot shows all records if that particular id does not match an existing record.

Exercise 4:

Output:

The image shows two browser screenshots. The top screenshot is of a form at `localhost:5000/studentinfo` with fields for First Name (Anakin), Last Name (Skywalker), Email (sand@gmail.com), Age (21), Gender (Male selected), and Qualifications (GCSE, A-level, Foundation Degree, Bachelor Degree, Master Degree selected). A Submit button is at the bottom. The bottom screenshot is of the response at `localhost:5000/submit-data`, showing a JSON object with status, message, and data fields.

Student Details

First Name:

Last Name:

Email:

Age:

Please select your gender:

☒ Male
☐ Female
☐ Other

Qualifications:

☒ GCSE
☒ A-level
☐ Higher National Certificate
☒ Foundation Degree/HND/DipHE/Level 5
☒ Bachelor Degree/Graduate Diploma or Certificate/ Level 6
☒ Master Degree/PGCE/Level 7
☐ PhD/Level 8

```
1 {
2   "Status": true,
3   "Message": "form details",
4   "data": {
5     "Name": "AnakinSkywalker ",
6     "Age": "21 Gender: male ",
7     "Qualifications": "Qualificaiton: gcse,a-level,hnd,b.sc,masterDegree"
8   }
9 }
```

Explanation:

I created an html document, sent it to a route, and enabled data submission with the use of the post method.

Reflection:

This week, I learned the theory behind creating an Express server, including the concepts of middleware, request/response cycle, and HTTP headers. I was able to create an Express server using an instance of the express module which I called app. I used .use, .get, and .post to set routes, filter results by URL parameters, send and manipulate response and request objects, and submit JSON data via a HTTP form.