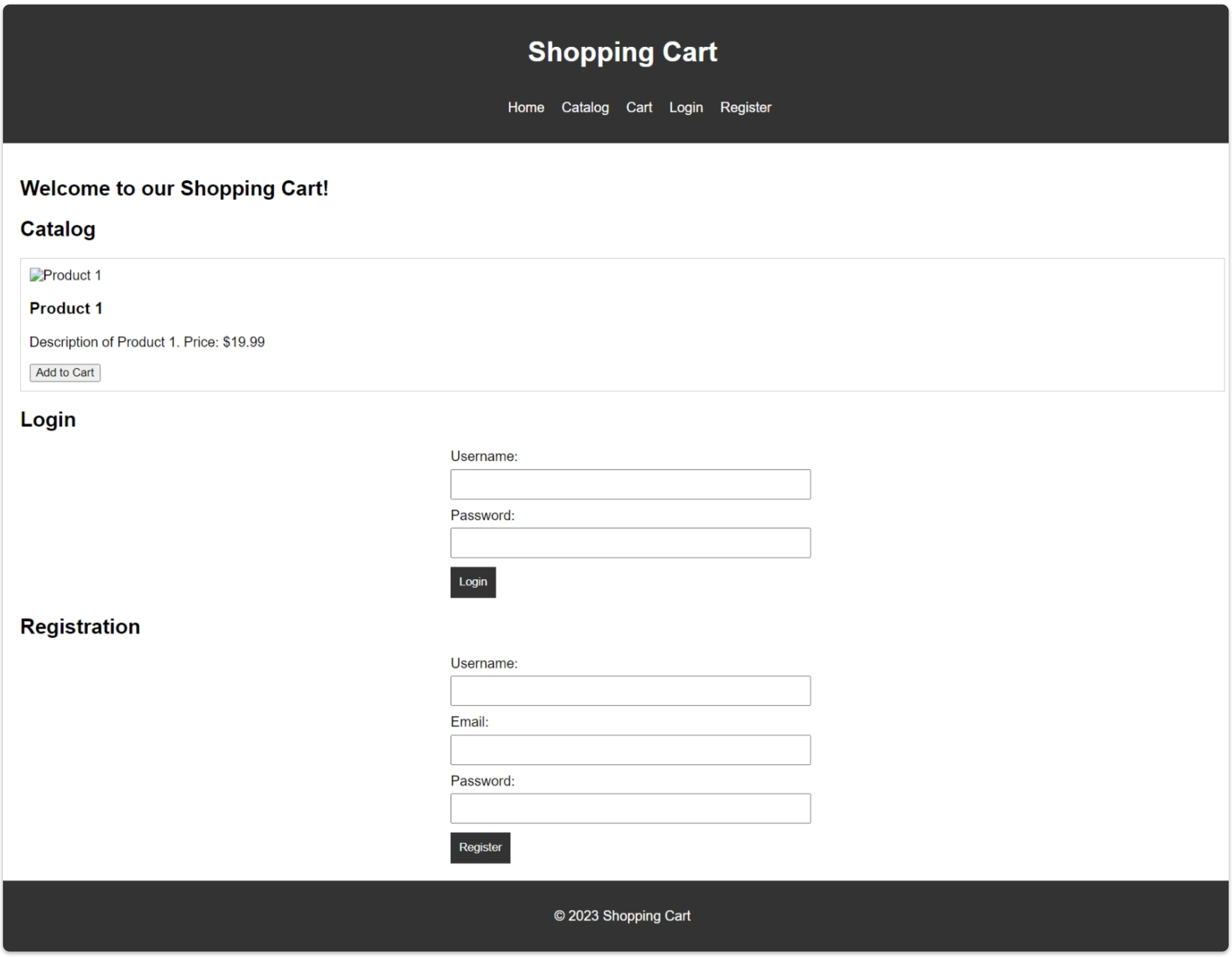
1. Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid

Output:

The shopping cart application's basic structure is displayed in index.html, with the catalog section containing dummy product data



Shopping Cart

Welcome to our Shopping Cart!

Catalog

Product 1

Description of Product 1. Price: $19.99

Login

Username:

Password:

Registration

Username:

Email:

Password:

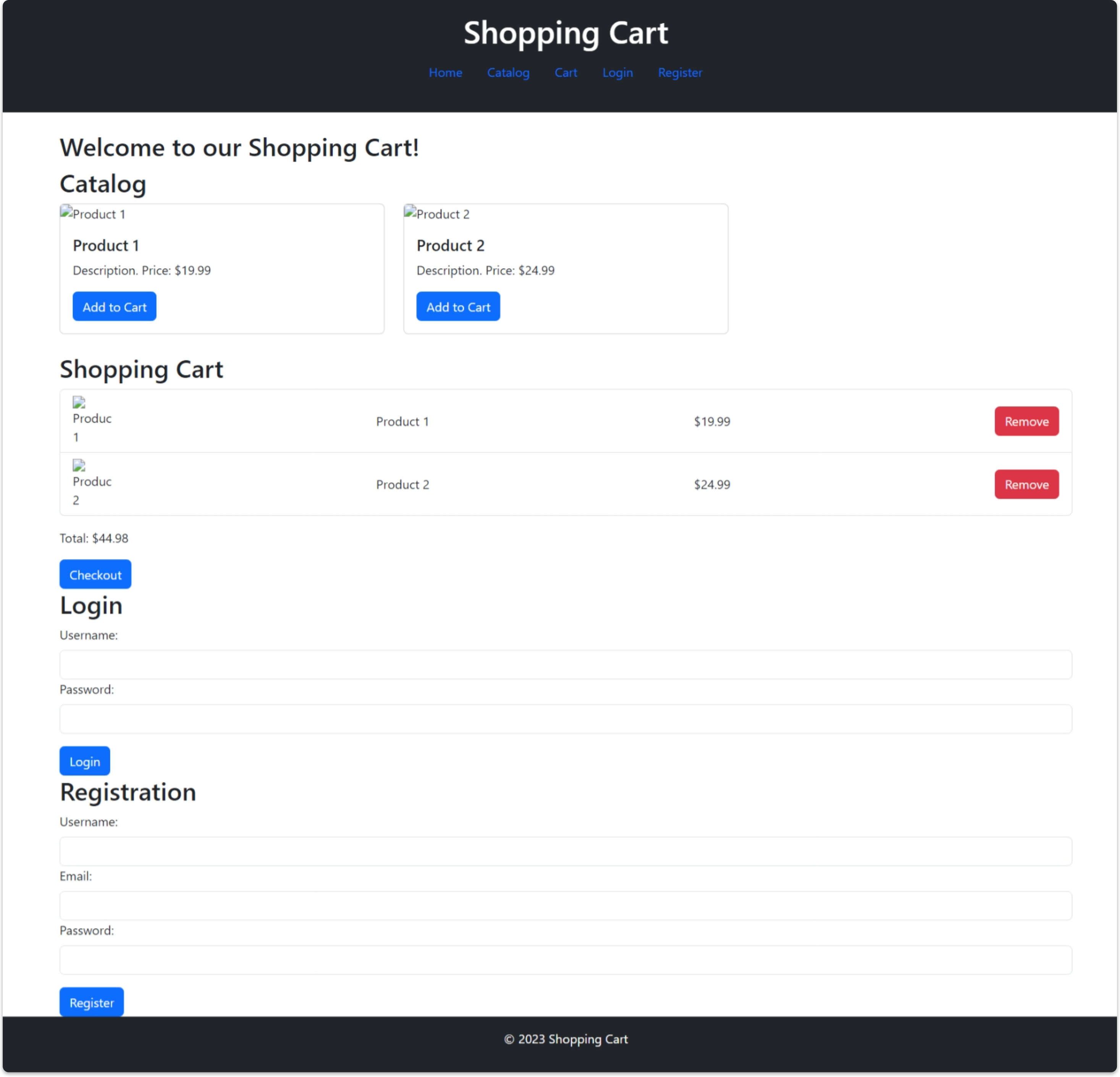
© 2023 Shopping Cart

2. Make the above web application responsive using Bootstrap framework

Output:

When you open `index.html` in a web browser, you'll see that the web application is now responsive. The Bootstrap framework takes care of making the layout adapt to different screen sizes, providing a more user-friendly experience on various devices.

Remember to test the responsiveness by resizing your browser or using different devices to see how the layout adjusts.


Shopping Cart

Home Catalog Cart Login Register

Welcome to our Shopping Cart!

Catalog

Product 1

Product 1

Description. Price: $19.99

Add to Cart

Product 2

Product 2

Description. Price: $24.99

Add to Cart

Shopping Cart

Produc 1	Product 1	$19.99	Remove
Produc 2	Product 2	$24.99	Remove

Total: $44.98

Checkout

Login

Username:

Password:

Login

Registration

Username:

Email:

Password:

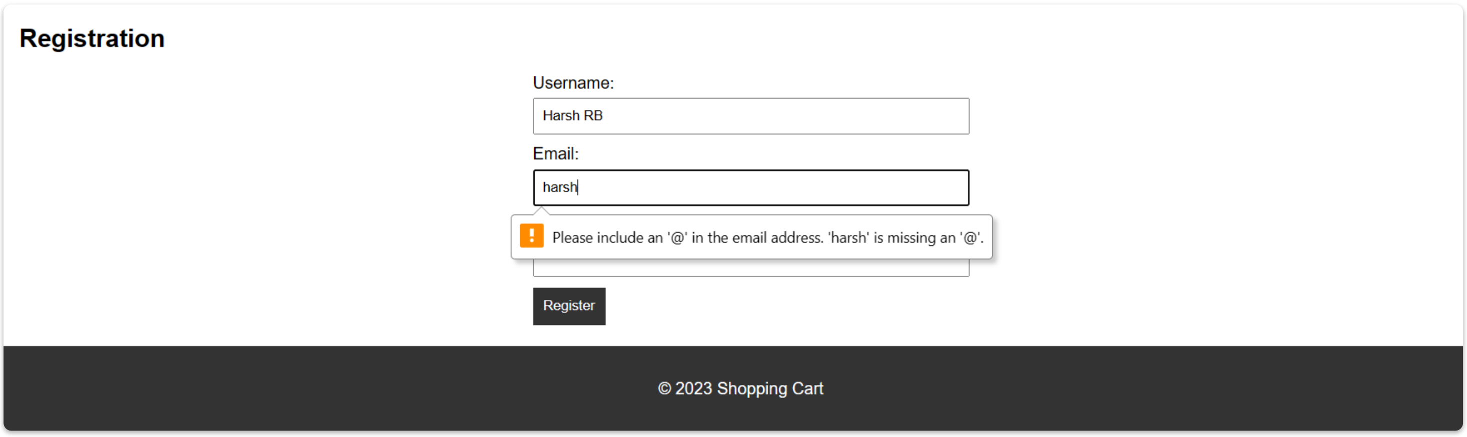
Register

© 2023 Shopping Cart

3. Use JavaScript for doing client–side validation of the pages implemented in experiment 1 and experiment 2

Outputs

Experiment 1:


Registration

Username: Harsh RB

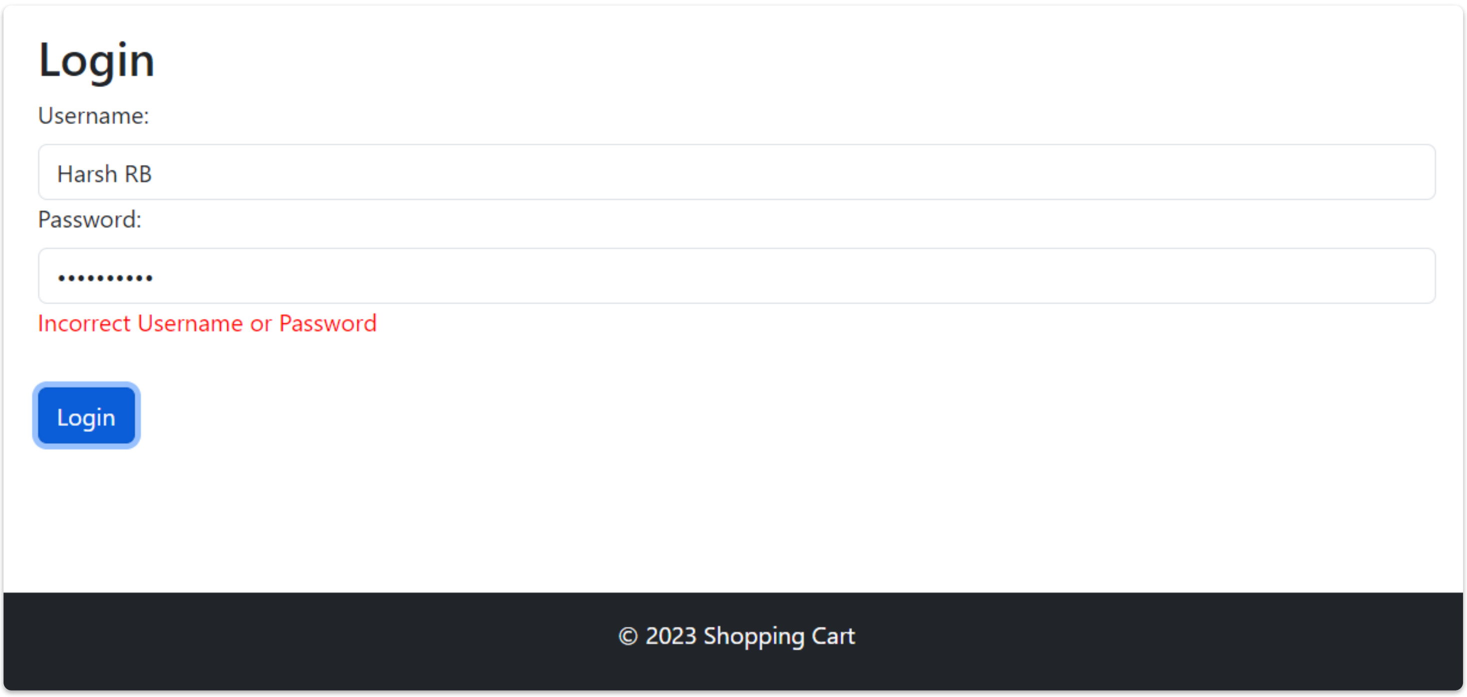
Email: harsh

Please include an '@' in the email address. 'harsh' is missing an '@'.

Register

© 2023 Shopping Cart

Experiment 2:



Login

Username:

Harsh RB

Password:

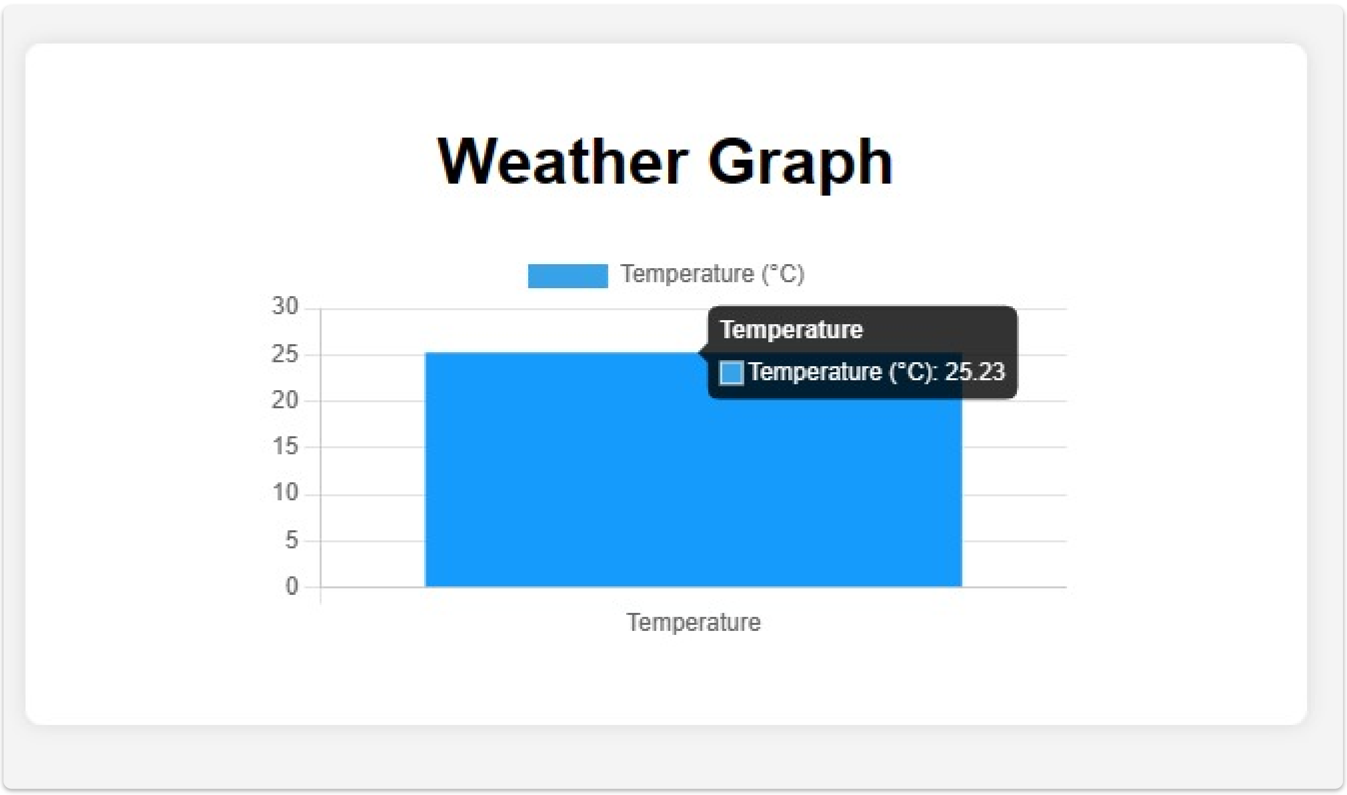
••••••••••

Incorrect Username or Password

© 2023 Shopping Cart

4. Explore the features of ES6 like arrow functions, callbacks, promises, async/await. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page

Outputs



5. Develop a Java stand-alone application that connects with the database (Oracle / MySQL) and perform the CRUD

operation on the database tables

Outputs

```
Record created successfully.

ID Name                 Salary
1  John Doe      50000

Record updated successfully.

ID Name                 Salary
1  John Updated  55000

Record deleted successfully.

ID   Name   Salary
```

6. Create an XML for the bookstore. Validate the same using both DTD and XSD

Outputs

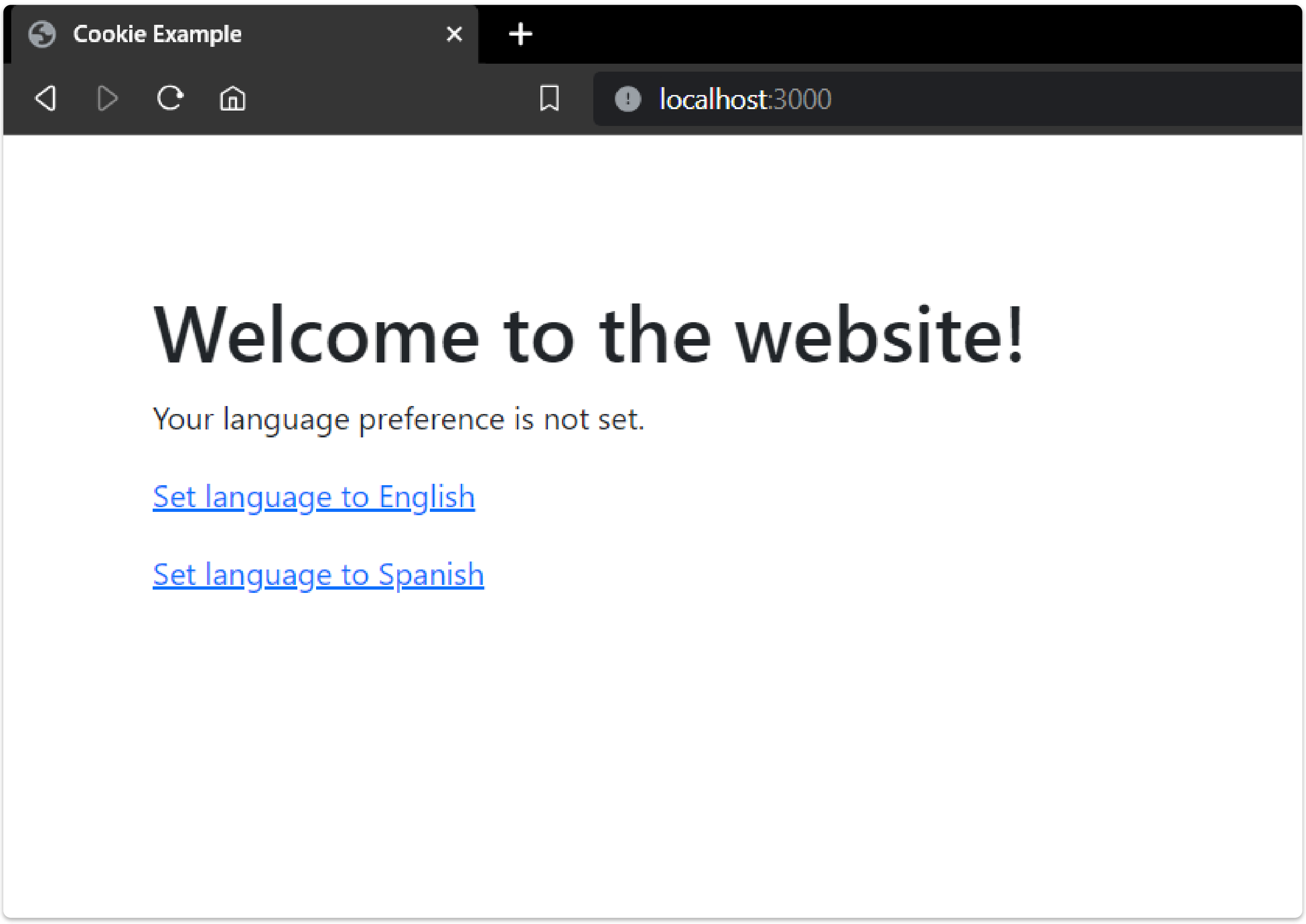
```
Validation with DTD successful.
Validation with XSD successful.

Book 1:
Title: Introduction to XML
Author: John Doe
Price: 29.99

Book 2:
Title: Web Development Basics
Author: Jane Smith
Price: 39.95
```

7. Design a controller with servlet that provides the interaction with the application developed in experiment 1 and the database created in experiment 5

8. Maintaining the transactional history of any user is very important. Explore the various session tracking mechanisms (Cookies, HTTP Session)

Outputs

1. Cookies:


Cookie Example
localhost:3000

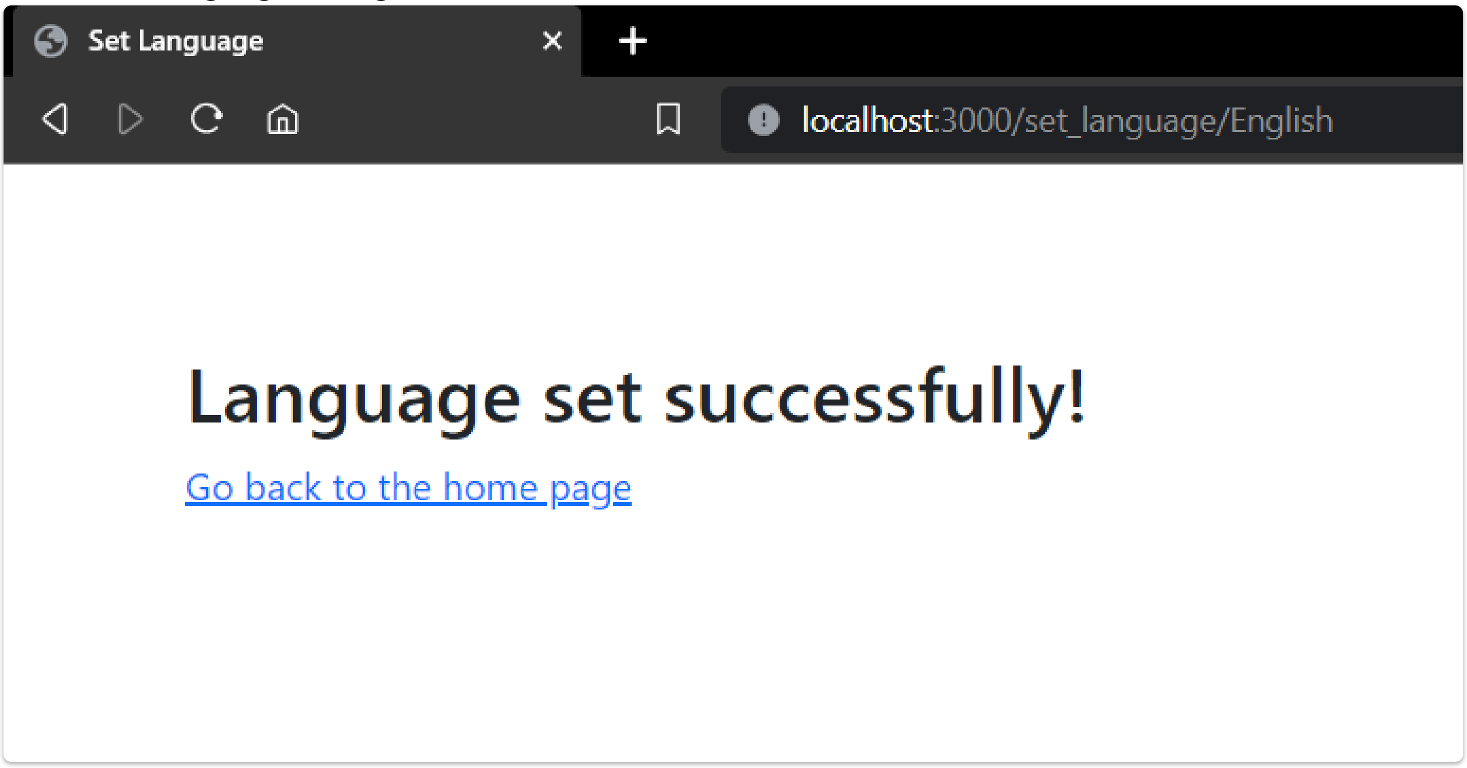
Welcome to the website!

Your language preference is not set.

Set language to English

Set language to Spanish

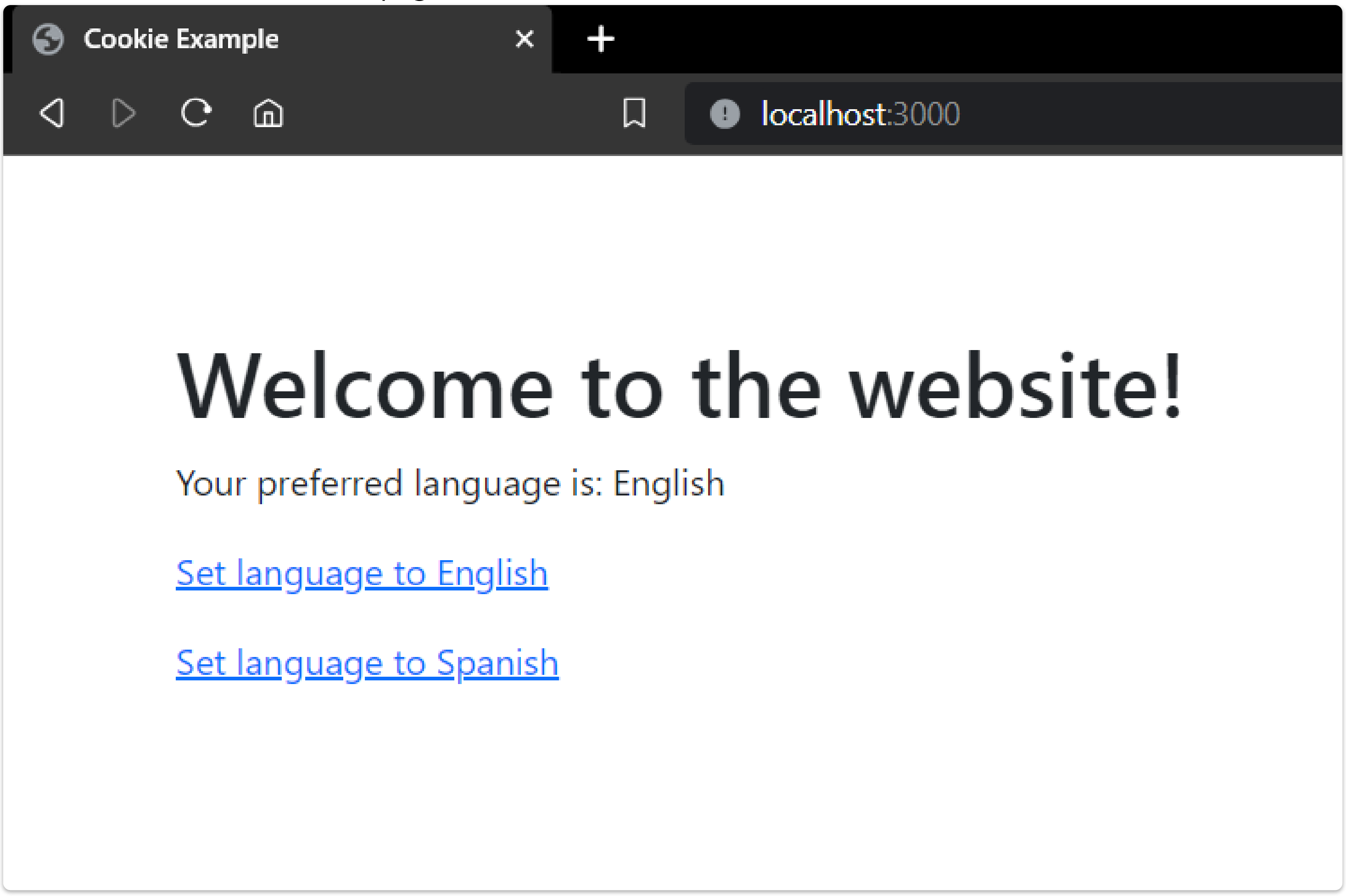
When "Set language to English" is clicked:


Set Language
localhost:3000/set_language/English

Language set successfully!

Go back to the home page

When "Go back to the home page" is clicked:



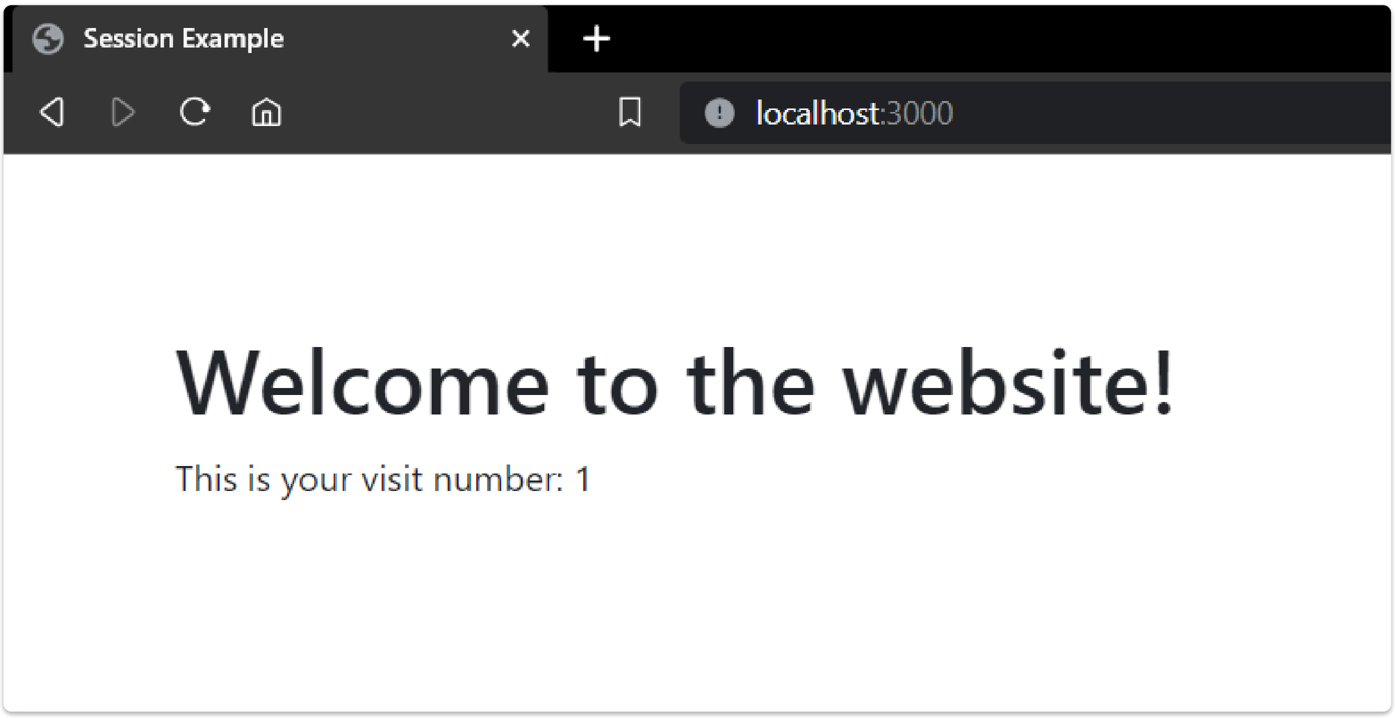
Welcome to the website!

Your preferred language is: English

Set language to English

Set language to Spanish

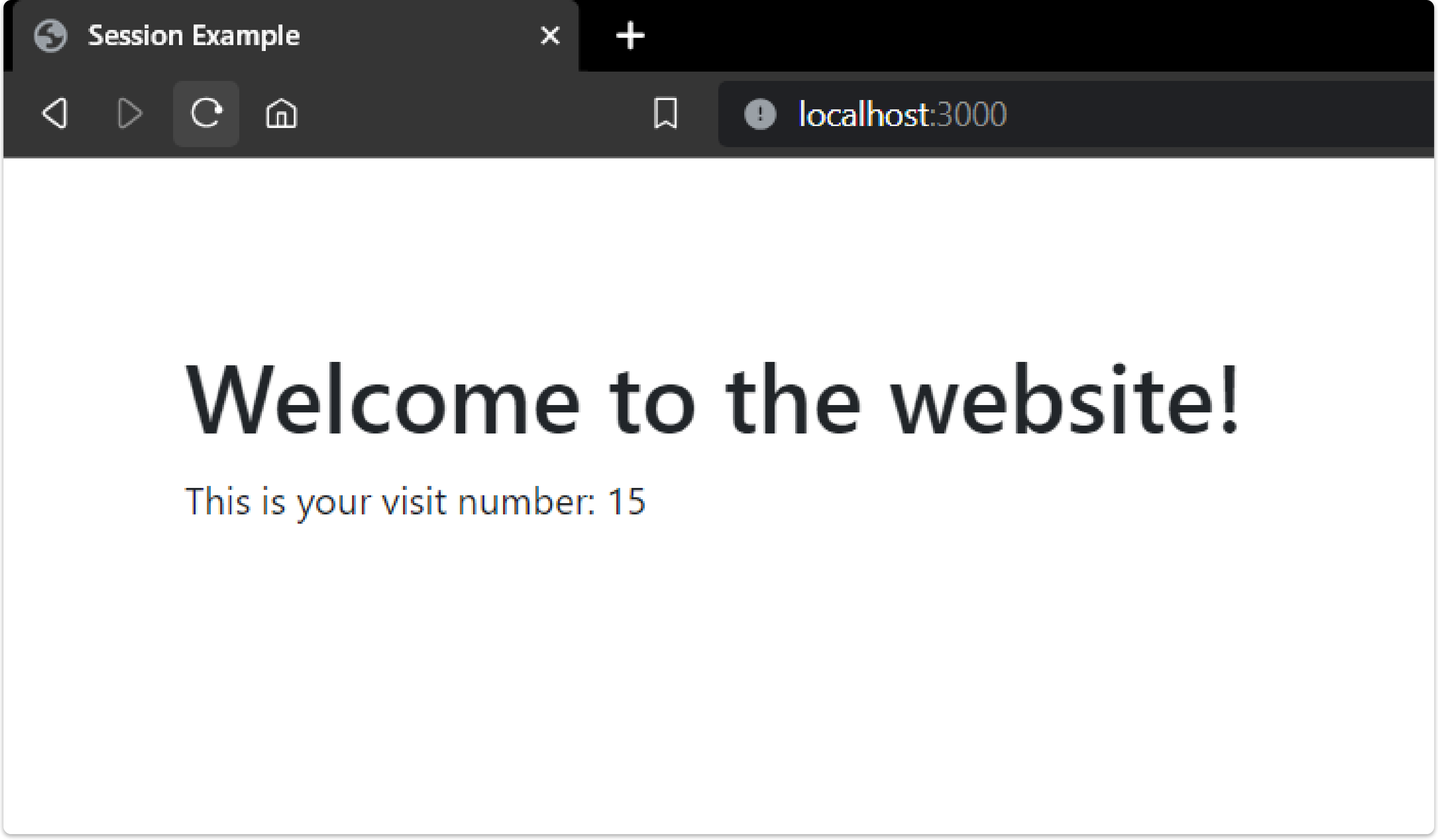
2. HTTP Session:



Welcome to the website!

This is your visit number: 1

When page is refreshed multiple times:

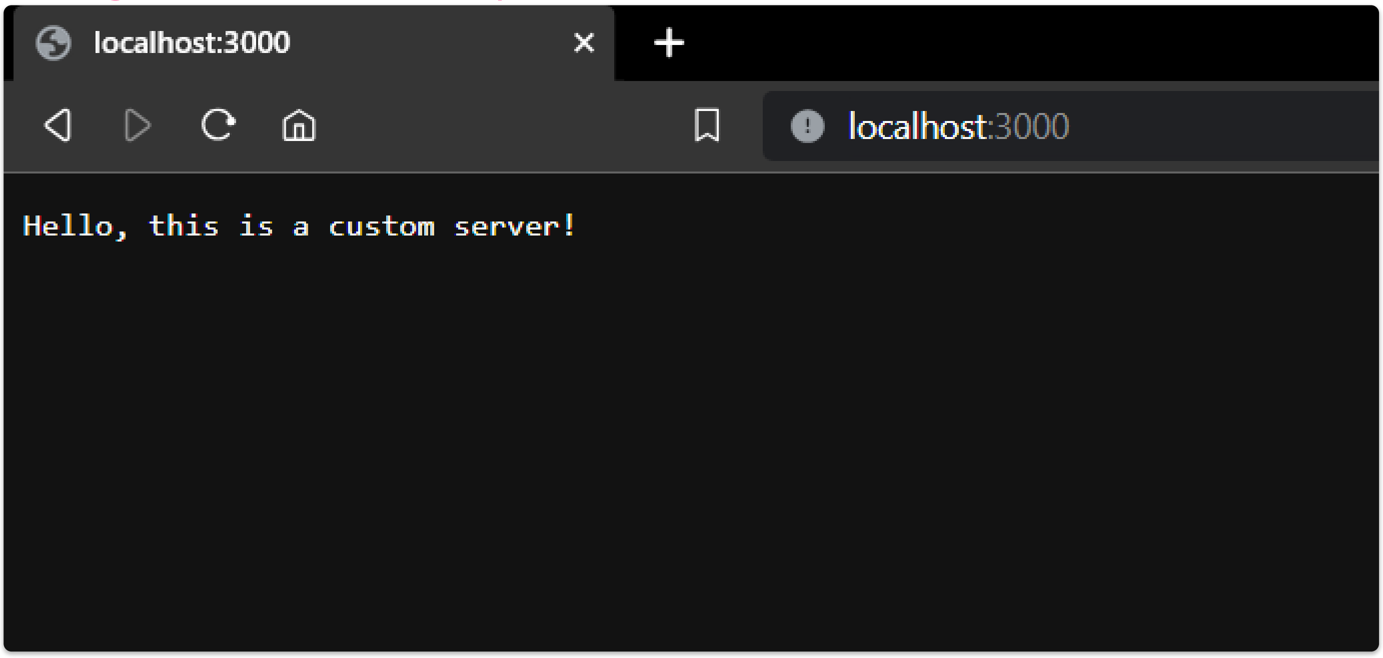

Welcome to the website!

This is your visit number: 15

9. Create a custom server using http module and explore the other modules of Node.js like OS, path, event

Outputs

1. Creating a Custom Server with http Module:


```
Hello, this is a custom server!
```

2. Exploring Node.js Modules:

   A. OS Module:

```
OS Platform: win32
OS Architecture: x64
Total Memory (in bytes): 8262680576
Free Memory (in bytes): 748179456
```

B. path Module:

```
File Name: file.txt
Directory Name: /path/to/some
File Extension: .txt
```

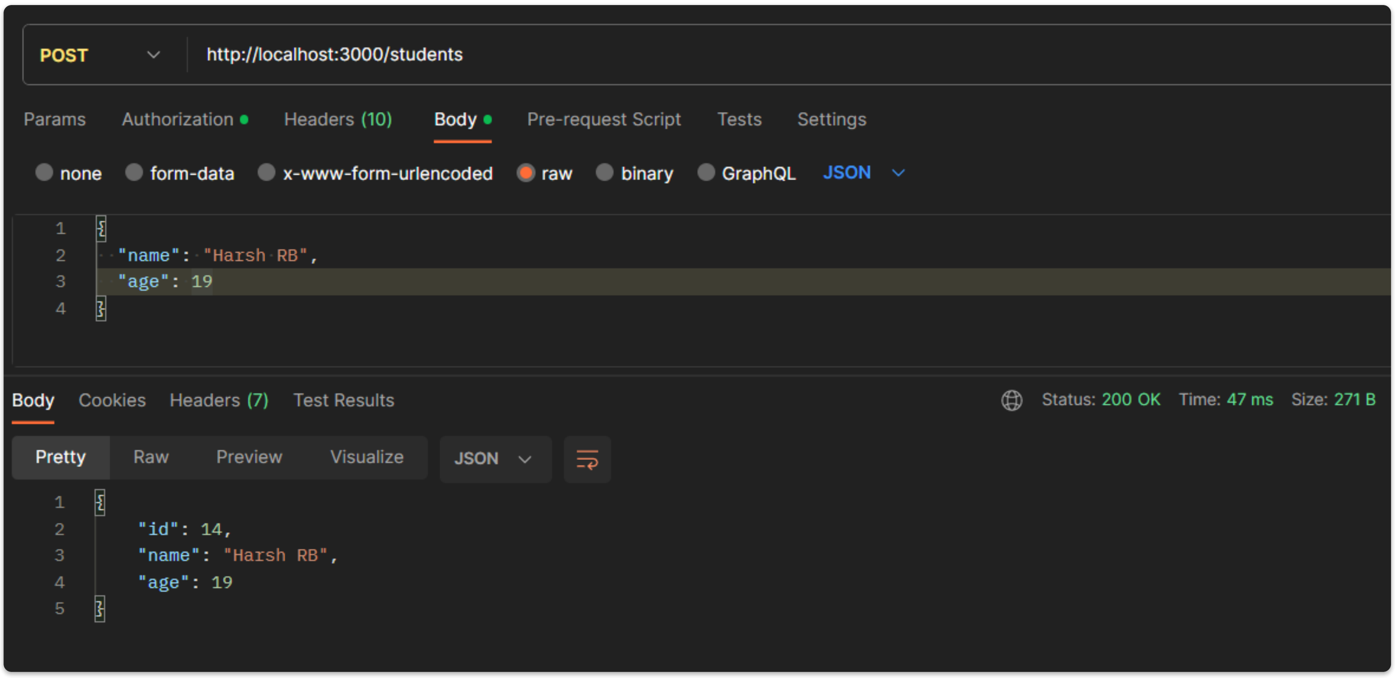
C. events Module:

```
Event triggered with argument: Hello, EventEmitter!
```

10. Develop an Express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)

Outputs

1. Create: Send a POST request to http://localhost:3000/students with JSON body containing student data.



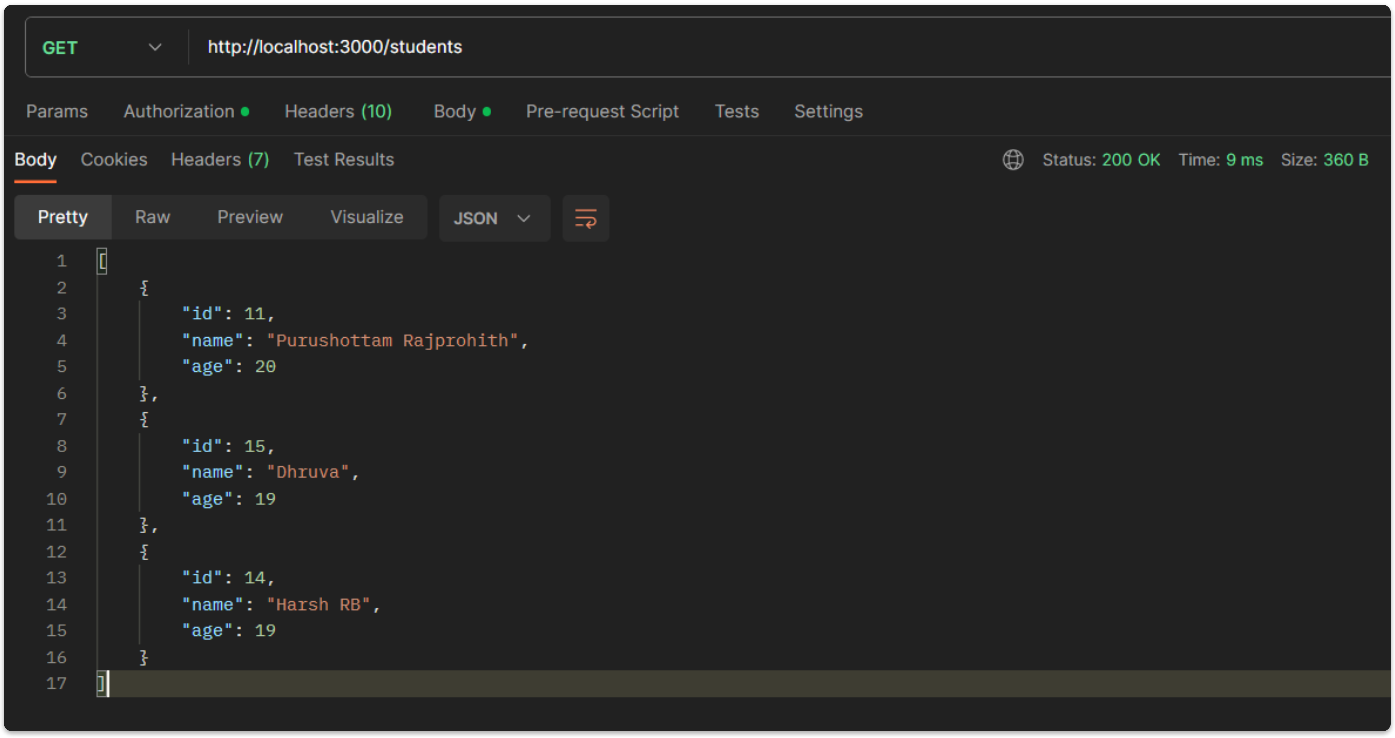
POST http://localhost:3000/students

```
{
  "name": "Harsh RB",
  "age": 19
}
```

Status: 200 OK Time: 47 ms Size: 271 B

```
{
    "id": 14,
    "name": "Harsh RB",
    "age": 19
}
```

2. Read (All): Send a GET request to http://localhost:3000/students.


GET http://localhost:3000/students

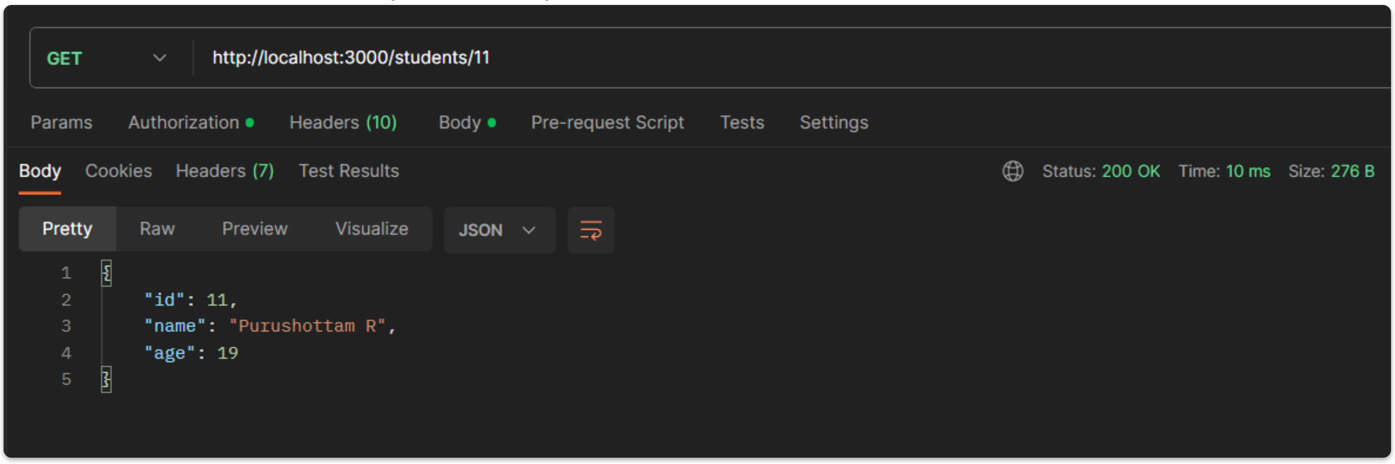
Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 9 ms Size: 360 B

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "id": 11,
        "name": "Purushottam Rajprohith",
        "age": 20
    },
    {
        "id": 15,
        "name": "Dhruva",
        "age": 19
    },
    {
        "id": 14,
        "name": "Harsh RB",
        "age": 19
    }
]
```

3. Read (One): Send a GET request to http://localhost:3000/students/{student_id}.


GET http://localhost:3000/students/11

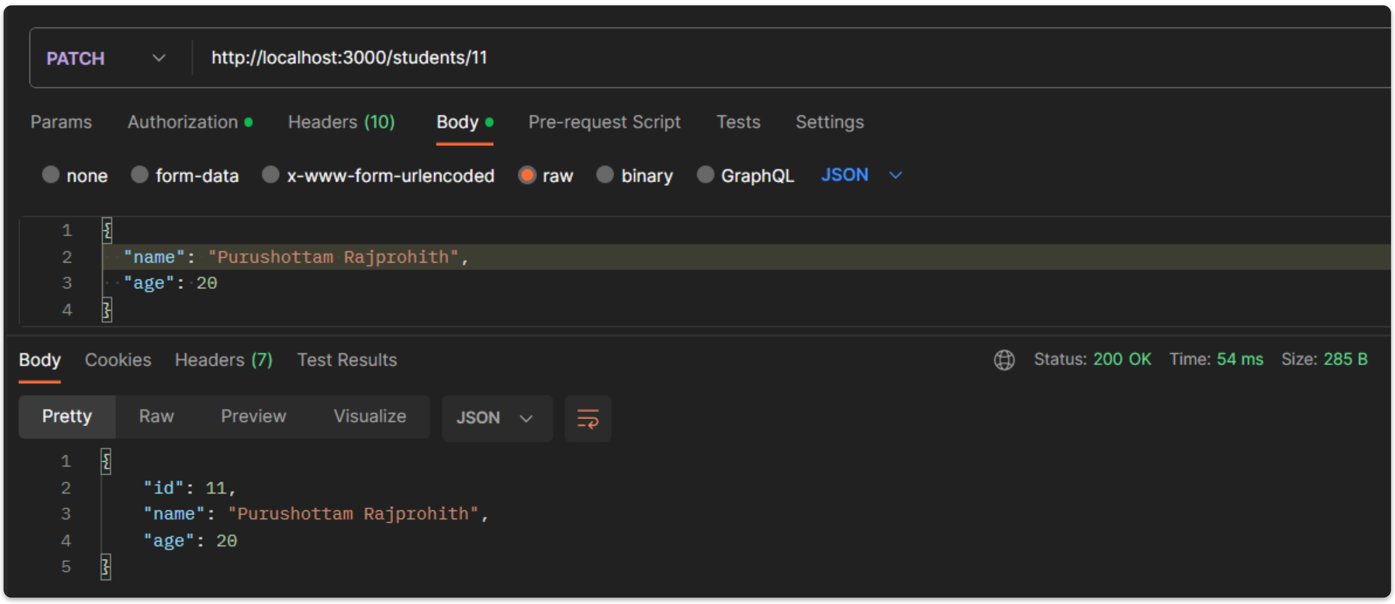
Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 10 ms Size: 276 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": 11,
    "name": "Purushottam R",
    "age": 19
}
```

4. Update: Send a PATCH request to http://localhost:3000/students/{student_id} with the updated data.


PATCH http://localhost:3000/students/11

Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

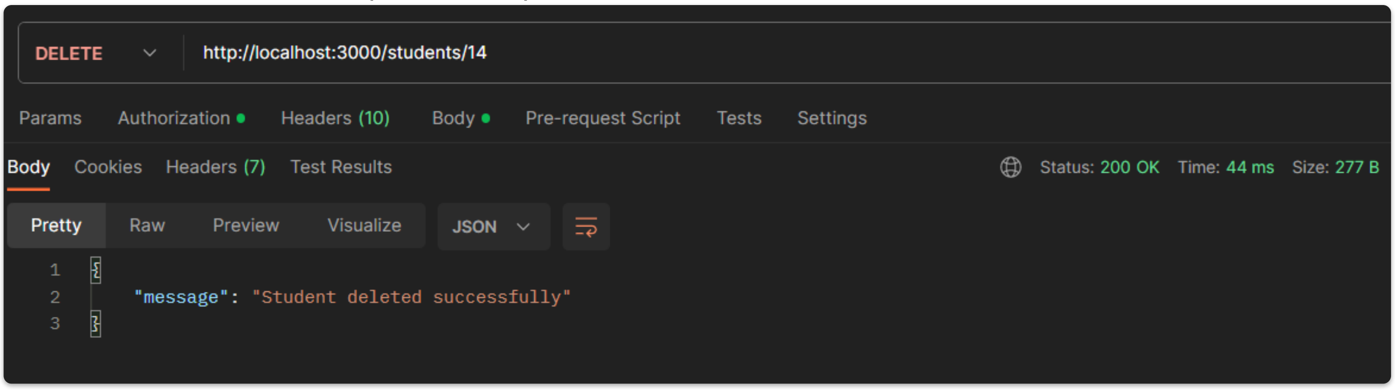
```
{
  "name": "Purushottam Rajprohith",
  "age": 20
}
```

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 54 ms Size: 285 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": 11,
    "name": "Purushottam Rajprohith",
    "age": 20
}
```

5. Delete: Send a DELETE request to http://localhost:3000/students/{student_id}.



DELETE http://localhost:3000/students/14

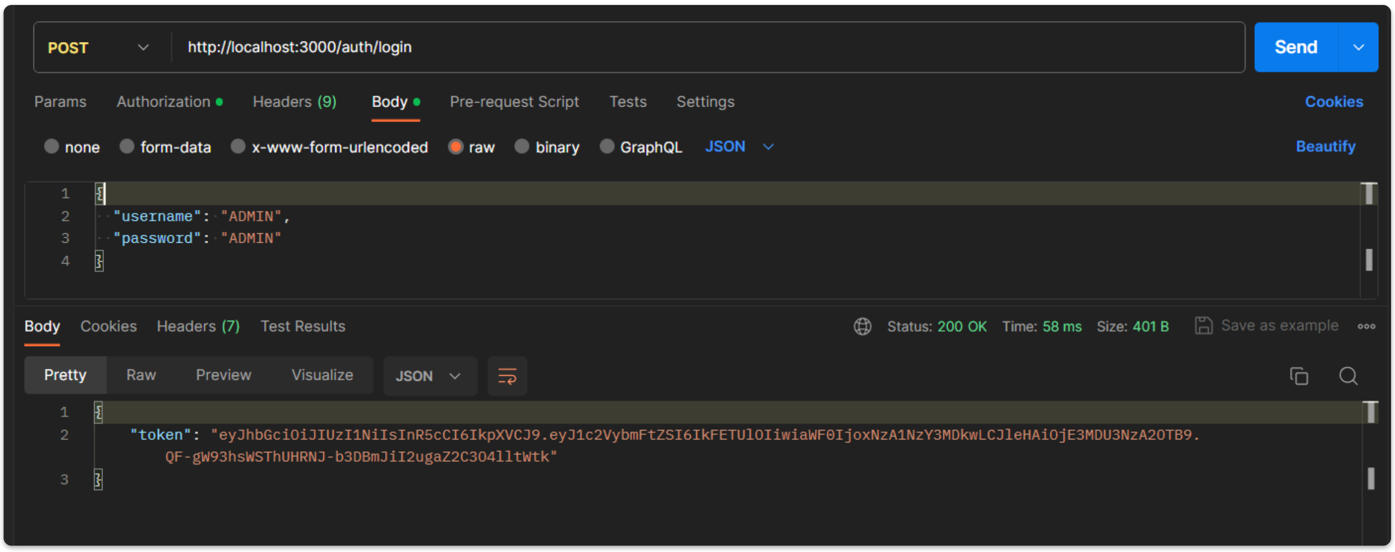
Status: 200 OK Time: 44 ms Size: 277 B

```
{
    "message": "Student deleted successfully"
}
```

11. For the above application, create authorized endpoints using JWT (JSON Web Token)

Outputs

1. Authentication:



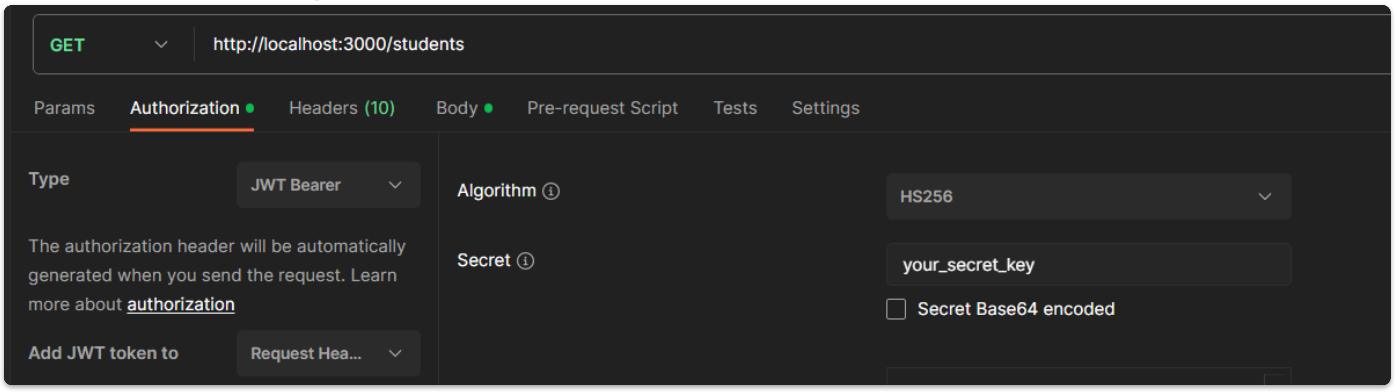
POST http://localhost:3000/auth/login

```
{
  "username": "ADMIN",
  "password": "ADMIN"
}
```

Status: 200 OK Time: 58 ms Size: 401 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IkFETUlOIiwiaWF0IjoxNzA1NzY3MDkwLCJleHAiOjE3MDU3NzA2OTB9.
        QF-gW93hsWSThUHRNJ-b3DBmJiI2ugaZ2C304lltWtk"
}
```

2. Access Protected Endpoints:



GET http://localhost:3000/students

Authorization

Type: JWT Bearer

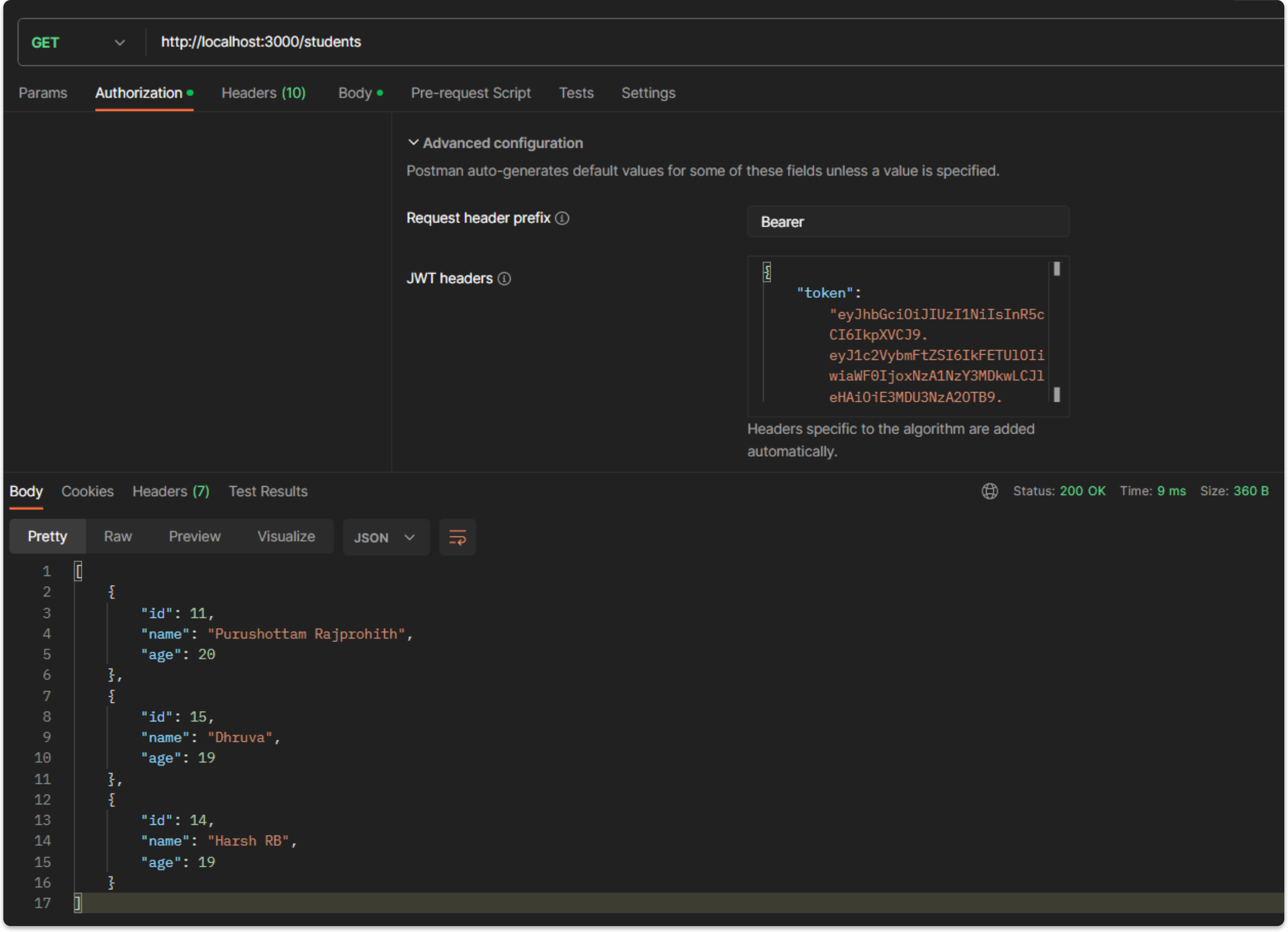
The authorization header will be automatically generated when you send the request. Learn more about authorization

Add JWT token to: Request Hea...

Algorithm: HS256

Secret: your_secret_key

☐ Secret Base64 encoded



http://localhost:3000/students

Advanced configuration

Postman auto-generates default values for some of these fields unless a value is specified.

Request header prefix: Bearer

JWT headers:

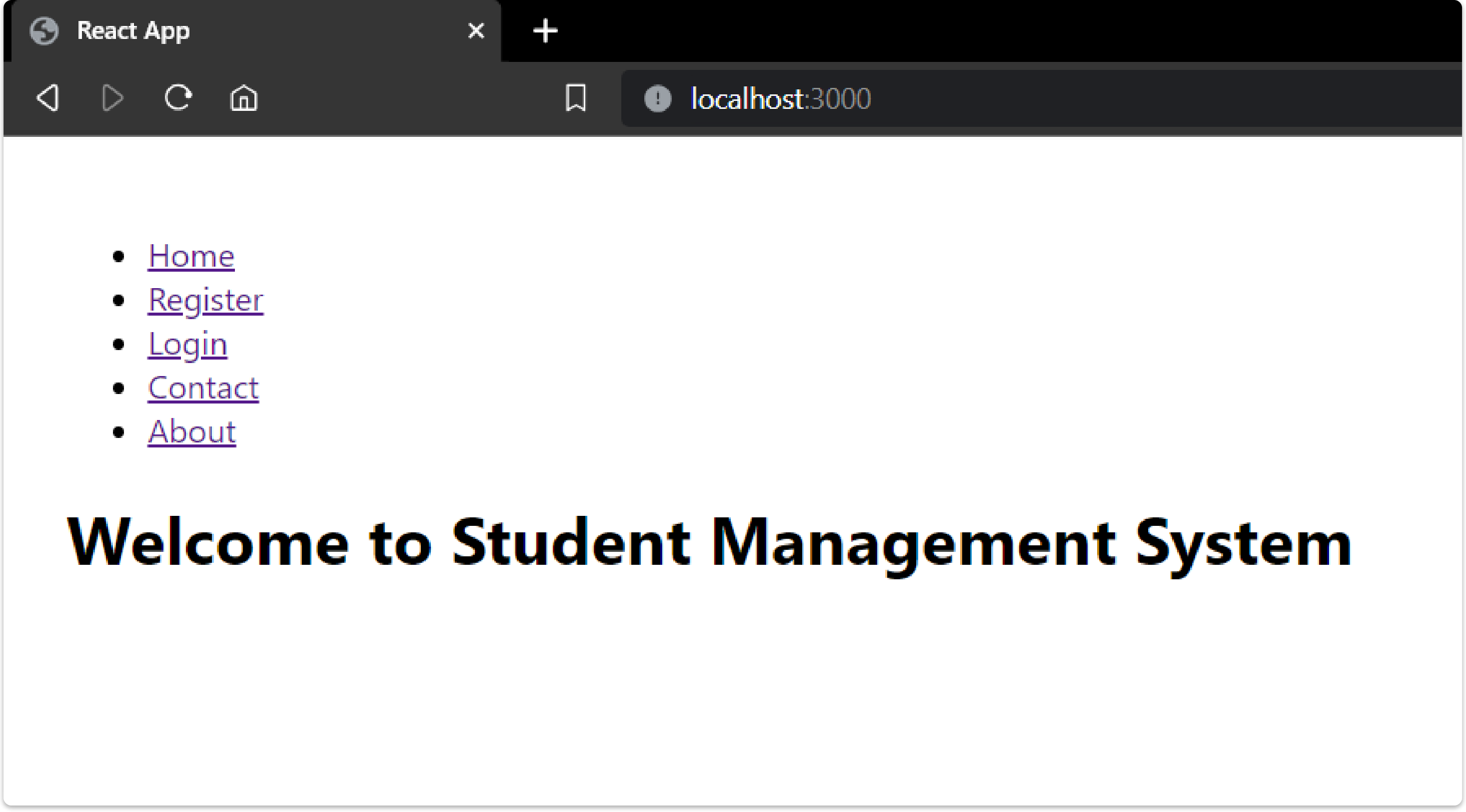
```
{
    "token":
        "eyJhbGciOiJIUzI1NiIsInR5c
        CI6IkpXVCJ9.
        eyJ1c2VybmFtZSI6IkFETUlOIi
        wiaWF0IjoxNzA1NzY3MDkwLCJl
        eHAiOiE3MDU3NzA2OTB9.
```

Headers specific to the algorithm are added automatically.

Status: 200 OK Time: 9 ms Size: 360 B

```
[
    {
        "id": 11,
        "name": "Purushottam Rajprohith",
        "age": 20
    },
    {
        "id": 15,
        "name": "Dhruva",
        "age": 19
    },
    {
        "id": 14,
        "name": "Harsh RB",
        "age": 19
    }
]
```

12. Create a React application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages

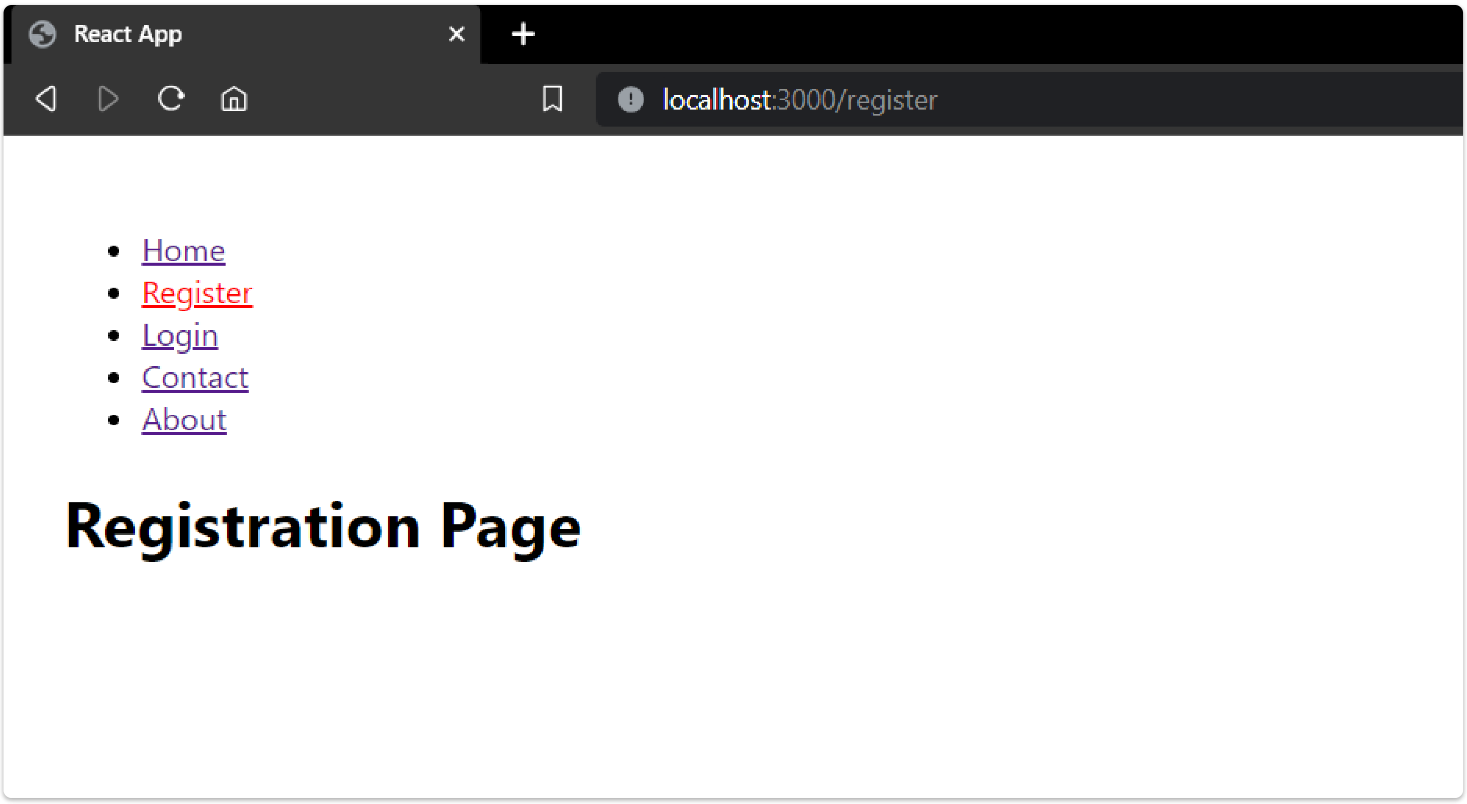


React App

localhost:3000

- Home
- Register
- Login
- Contact
- About

Welcome to Student Management System

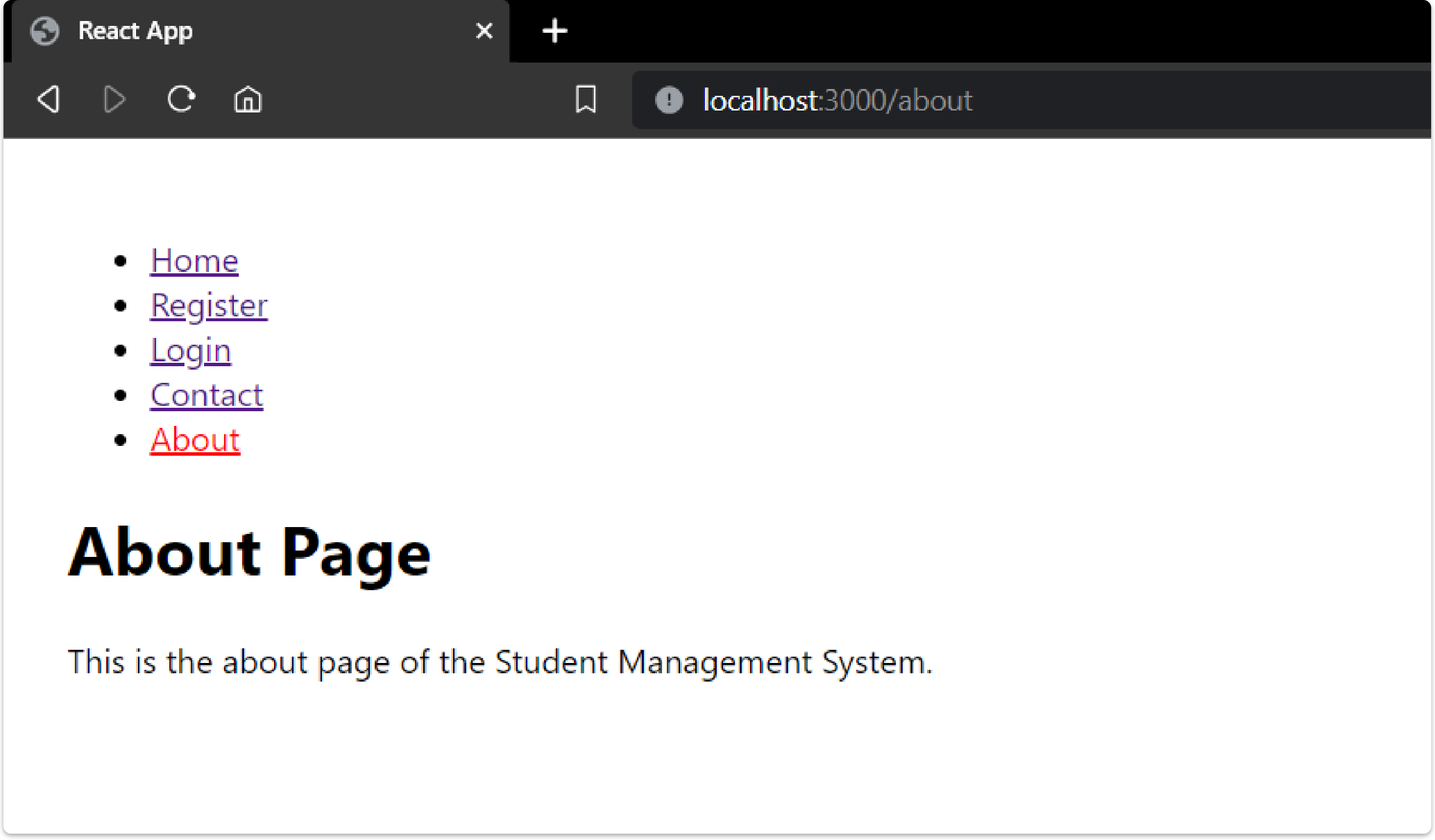


React App

localhost:3000/register

- Home
- Register
- Login
- Contact
- About

Registration Page



React App

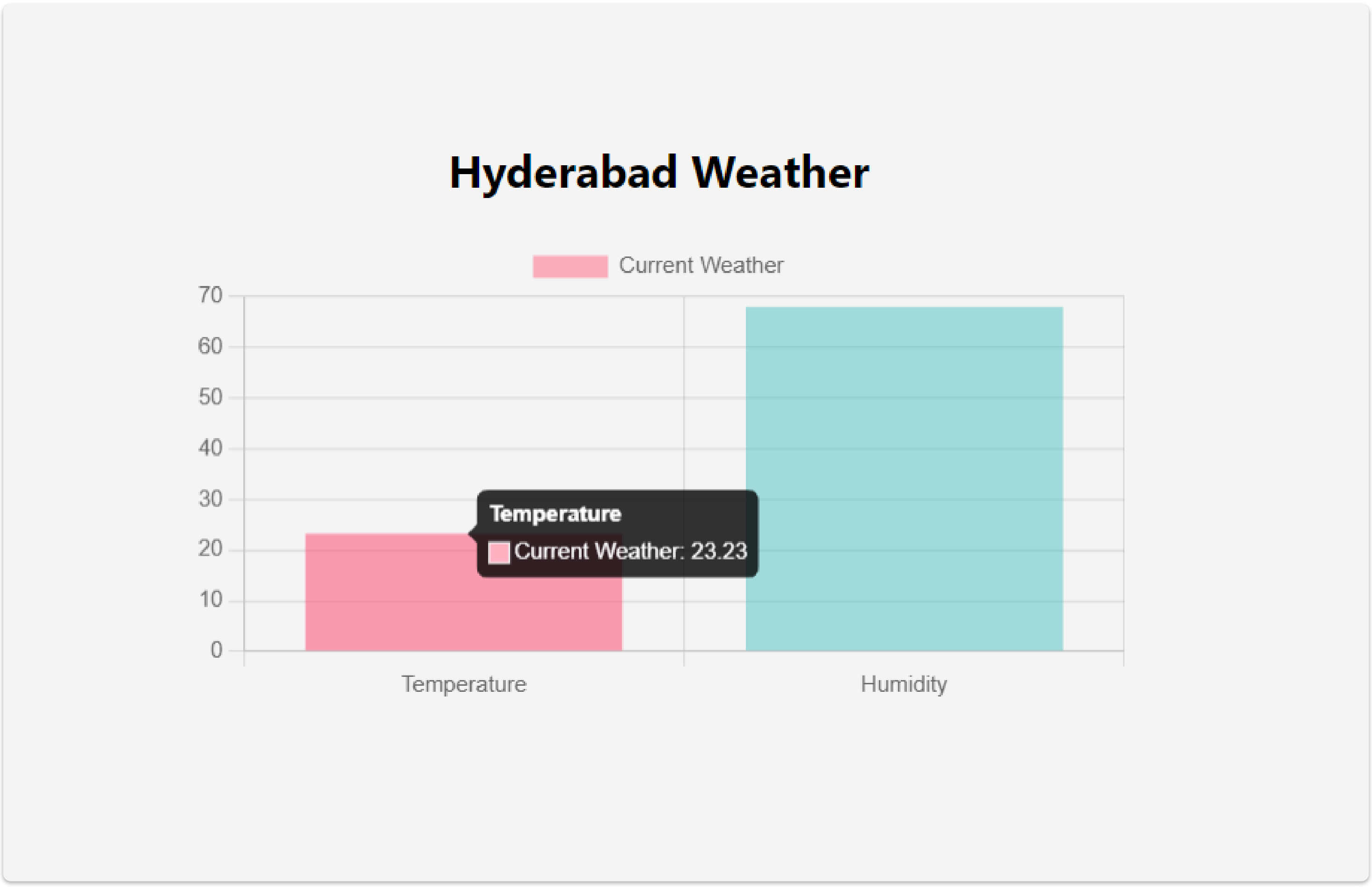
localhost:3000/about

- Home
- Register
- Login
- Contact
- About

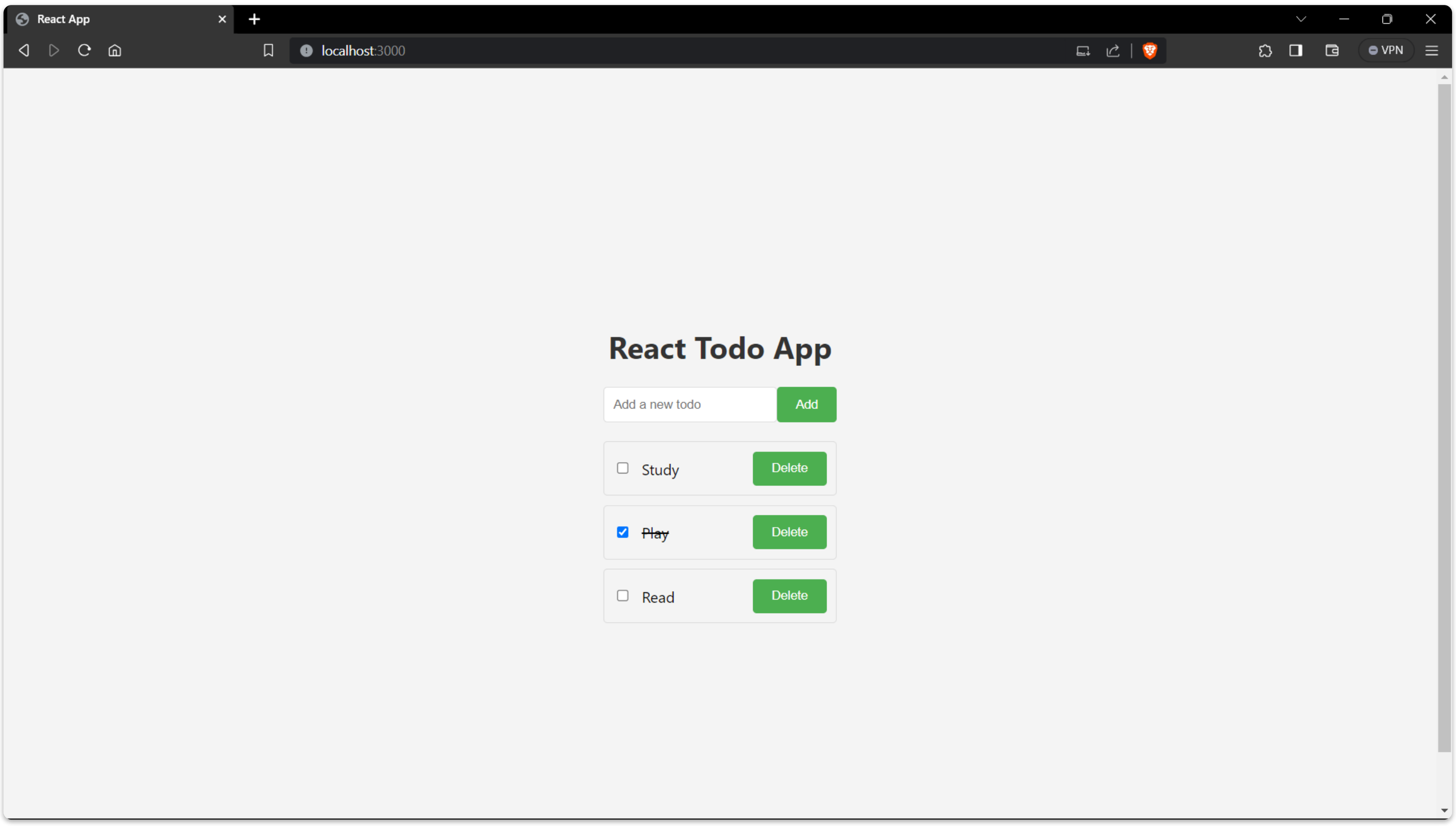
About Page

This is the about page of the Student Management System.

13. Create a service in React that fetches the weather information from openweathermap.org and then display the current and historical weather information using graphical representation using chart.js



14. Create a TODO application in React with necessary components and deploy it into GitHub



React Todo App

```
cd react-todo-app
git init
git add .
git commit -m "todo react app"
git remote add origin https://github.com/your-username/your-repo-name.git
git push -u origin master
```