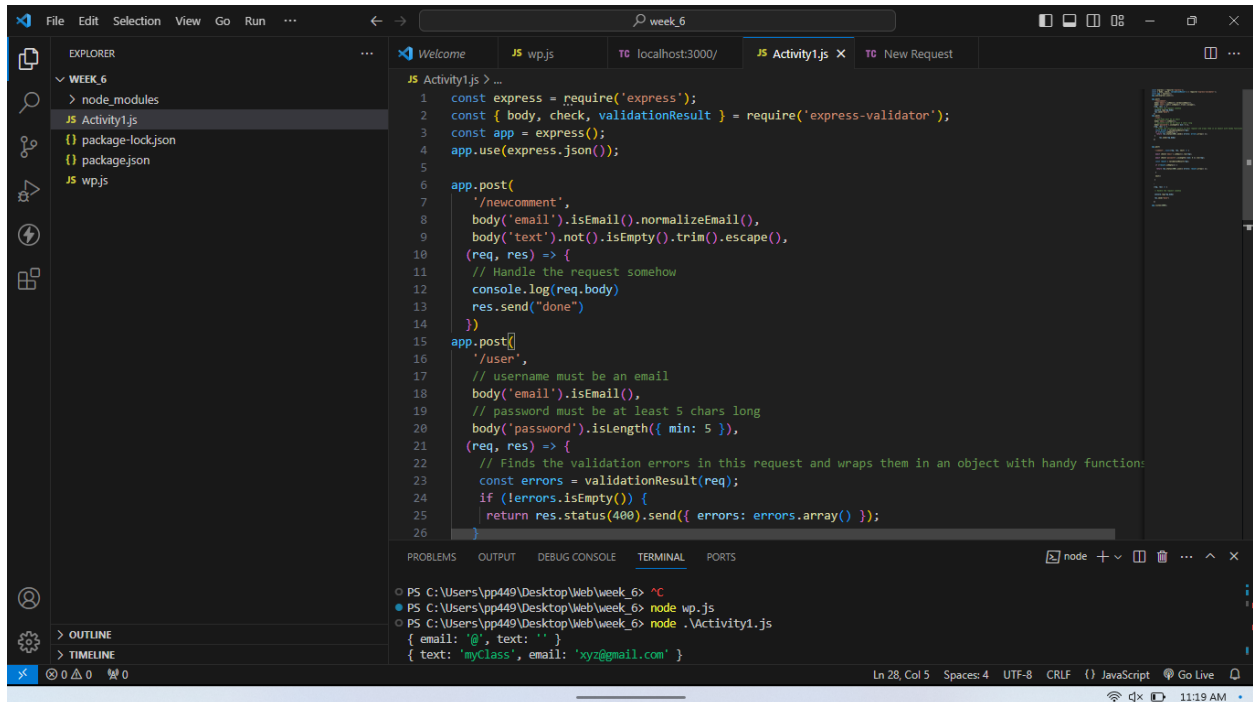


Activity1:

The `.body()` function is a middleware provided by the `express-validator` package in Node.js

The `body` method contains validation rules for email and password if the user enters the valid information then it will get the data.

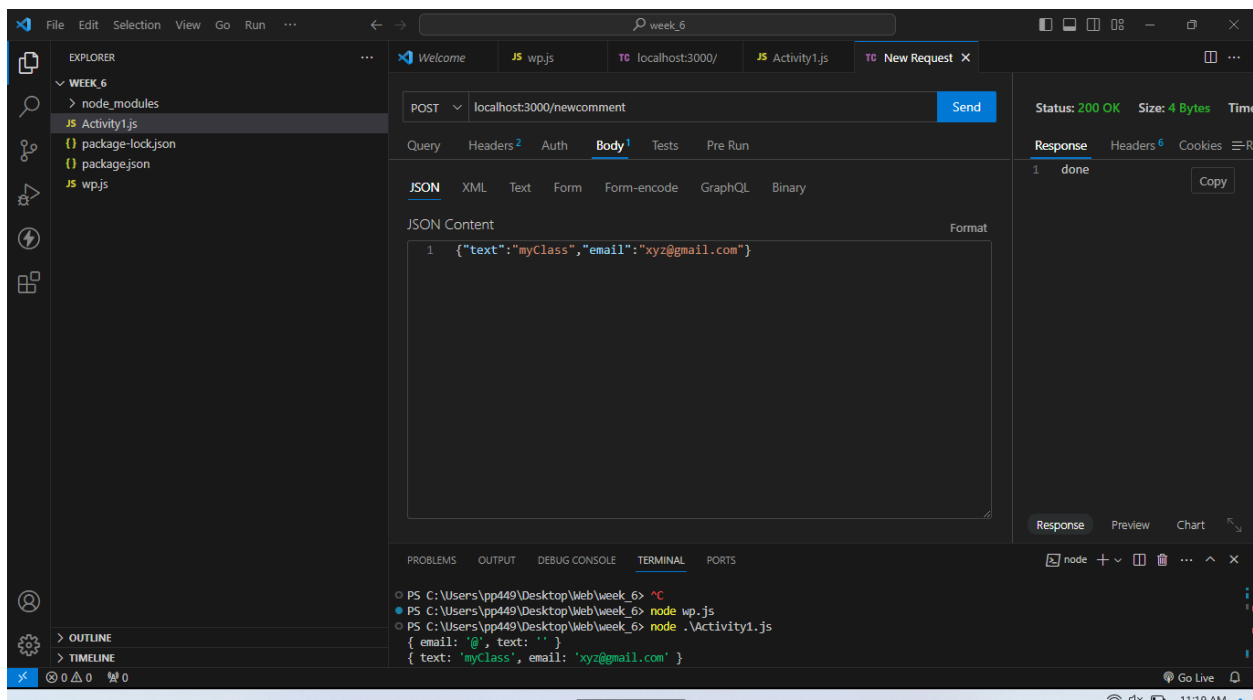


This screenshot shows the source code of `Activity1.js` in VS Code. The code sets up an Express application with `express-validator` for validation. It defines two POST routes: `/newcomment` and `/user`. The `/user` route uses `body()` to validate email and password. The terminal shows the command `node wp.js` being executed, and the output displays the request body for a POST request to `/user`.

```
1 const express = require('express');
2 const { body, check, validationResult } = require('express-validator');
3 const app = express();
4 app.use(express.json());
5
6 app.post(
7   '/newcomment',
8   body('email').isEmail().normalizeEmail(),
9   body('text').not().isEmpty().trim().escape(),
10  (req, res) => {
11    // Handle the request somehow
12    console.log(req.body)
13    res.send("done")
14  })
15 app.post(
16   '/user',
17   // username must be an email
18   body('email').isEmail(),
19   // password must be at least 5 chars long
20   body('password').isLength({ min: 5 }),
21  (req, res) => {
22    // Finds the validation errors in this request and wraps them in an object with handy functions
23    const errors = validationResult(req);
24    if (!errors.isEmpty()) {
25      return res.status(400).send({ errors: errors.array() });
26    }
27  })
```

Terminal Output:

```
PS C:\Users\pp449\Desktop\Web\week_6> ^C
PS C:\Users\pp449\Desktop\Web\week_6> node wp.js
PS C:\Users\pp449\Desktop\Web\week_6> node .\Activity1.js
{ email: '@', text: '' }
{ text: 'myClass', email: 'xyz@gmail.com' }
```



This screenshot shows the REST Client interface in VS Code. A POST request is configured to `localhost:3000/newcomment` with a JSON body. The response is `200 OK` with the text `done`. The terminal shows the command `node wp.js` being executed, and the output displays the request body for a POST request to `/user`.

Request Configuration:

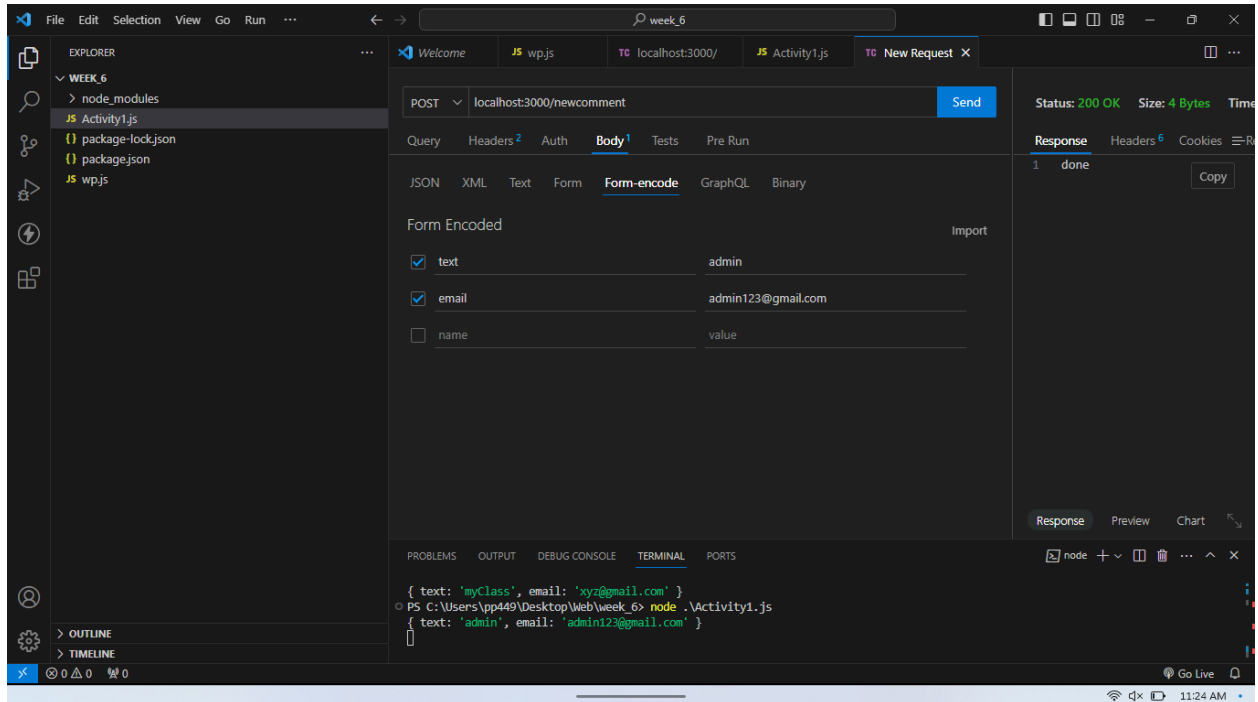
- Method: POST
- URL: localhost:3000/newcomment
- Body: `{ "text": "myClass", "email": "xyz@gmail.com" }`

Response:

```
1 done
```

Terminal Output:

```
PS C:\Users\pp449\Desktop\Web\week_6> ^C
PS C:\Users\pp449\Desktop\Web\week_6> node wp.js
PS C:\Users\pp449\Desktop\Web\week_6> node .\Activity1.js
{ email: '@', text: '' }
{ text: 'myClass', email: 'xyz@gmail.com' }
```

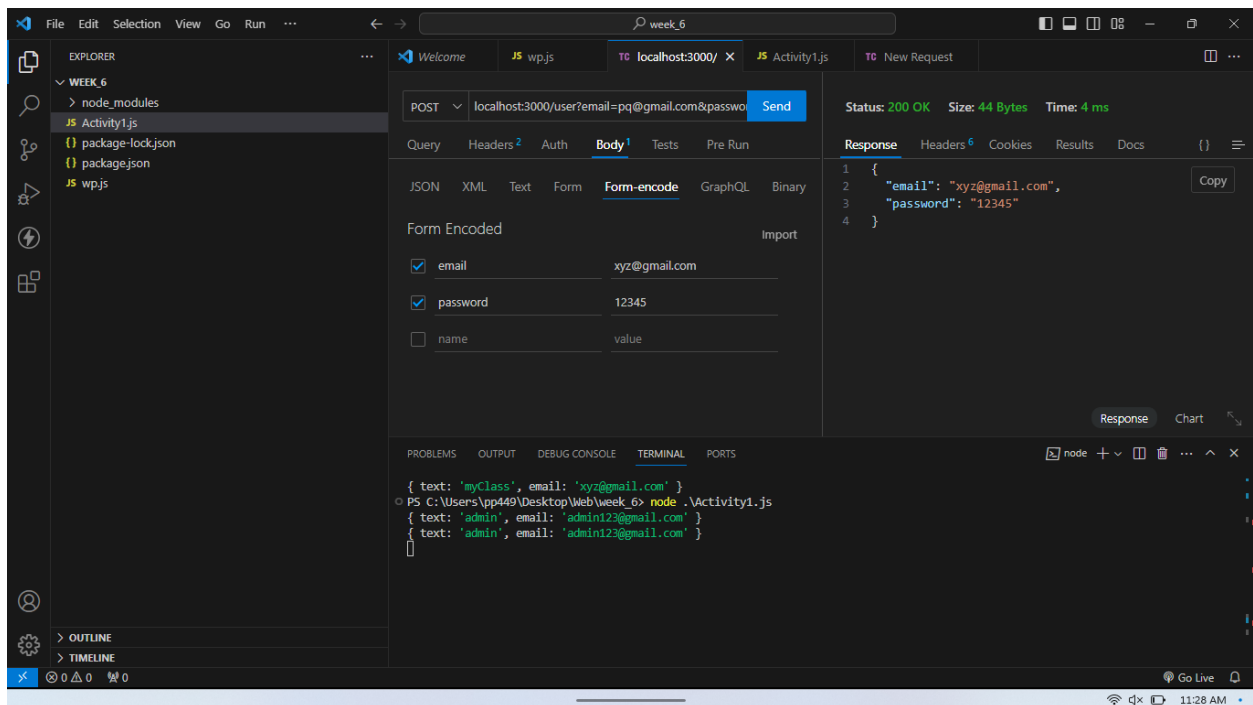


B. `express.urlencoded({ extended: true })` is used for parsing incoming requests payloads, enabling parsing of objects .now we cannot pass the json data.we can pass strings and arrays.

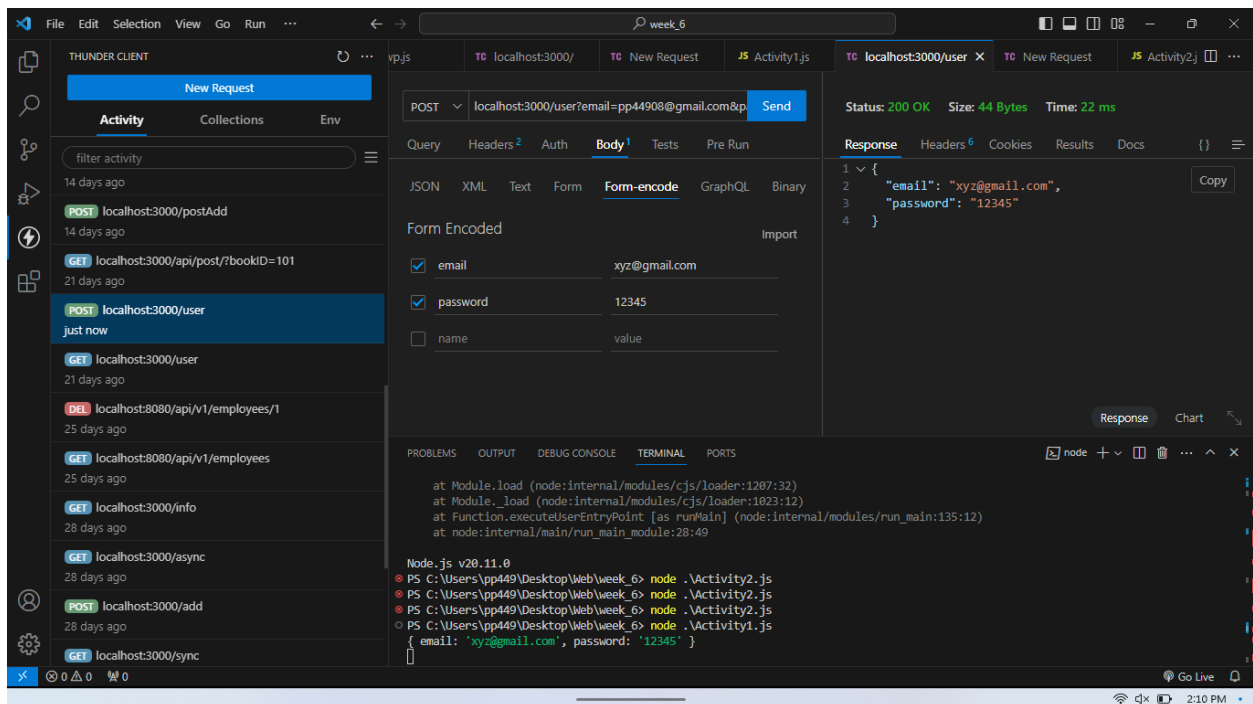
```
const express = require('express');
const { body, check, validationResult } = require('express-validator');
const app = express();
app.use(express.json());

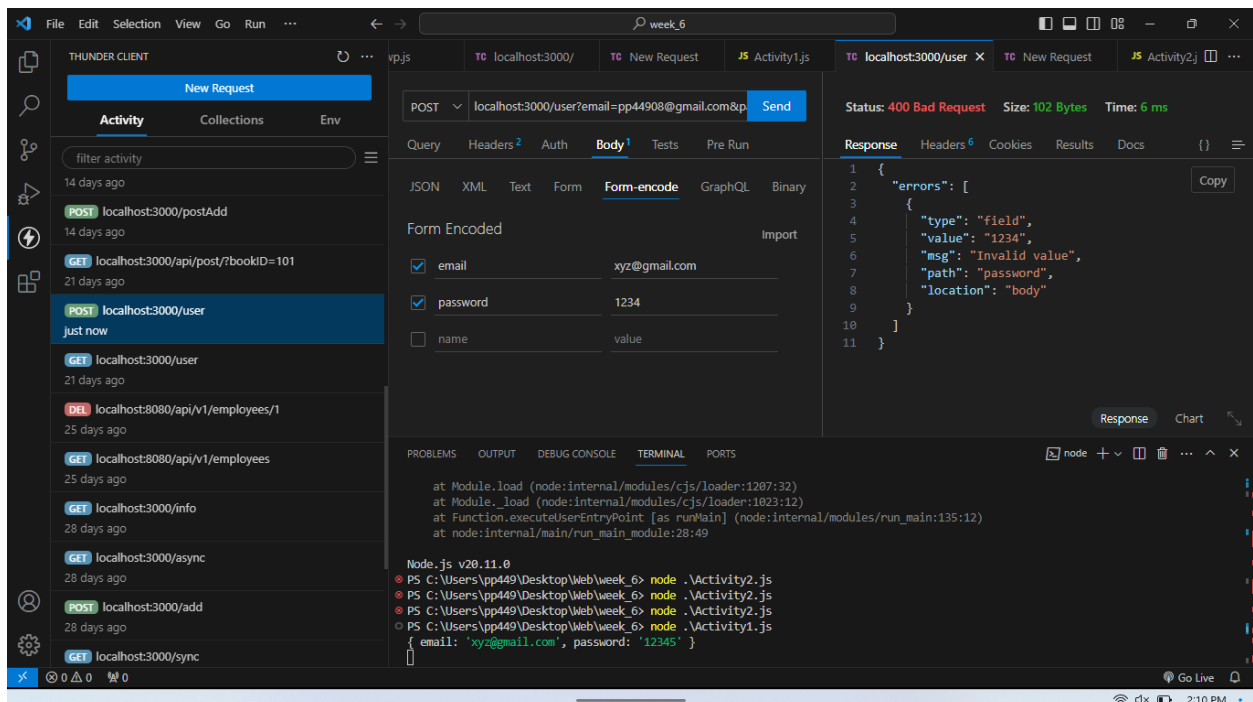
app.use(express.urlencoded({ extended: true }));

app.post(
  '/newcomment',
  body('email').isEmail().normalizeEmail(),
  body('text').not().isEmpty().trim().escape(),
  (req, res) => {
    // Handle the request somehow
    console.log(req.body)
    res.send("done")
  })
```



Step2:in the this url /user first of all the body is checking is email and password is valid or not if not it throws the error of 400

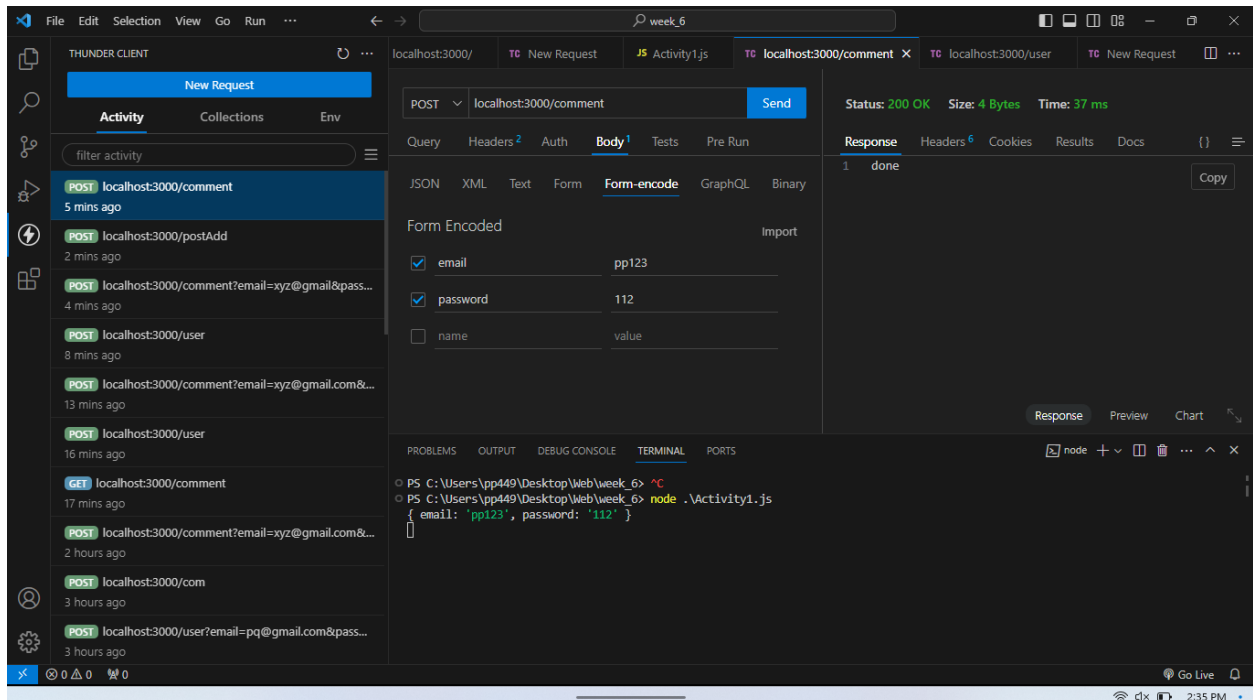




3) Try the 3rd route (/comment) in ThunderClient. Identify the middleware that performs validation. Explain how they work (.check?, .validationResult()) ?

What happens if you removed "async" and "await" . What is the role of "async" and "await"?

If we remove aysns and await the validation will not work and the data will be passed without validation.



4) Based on 1,2,3, compare the ".body" vs ".check" and elaborate the purpose/role of each

body and check can be used for request validation, .body() is specifically used for request body parameters, while .check() offers more flexibility and can be used for validating parameters from different parts of the request.

5) Based on 1,2,3, compare the "validation" vs "sanitization". Explain your answer

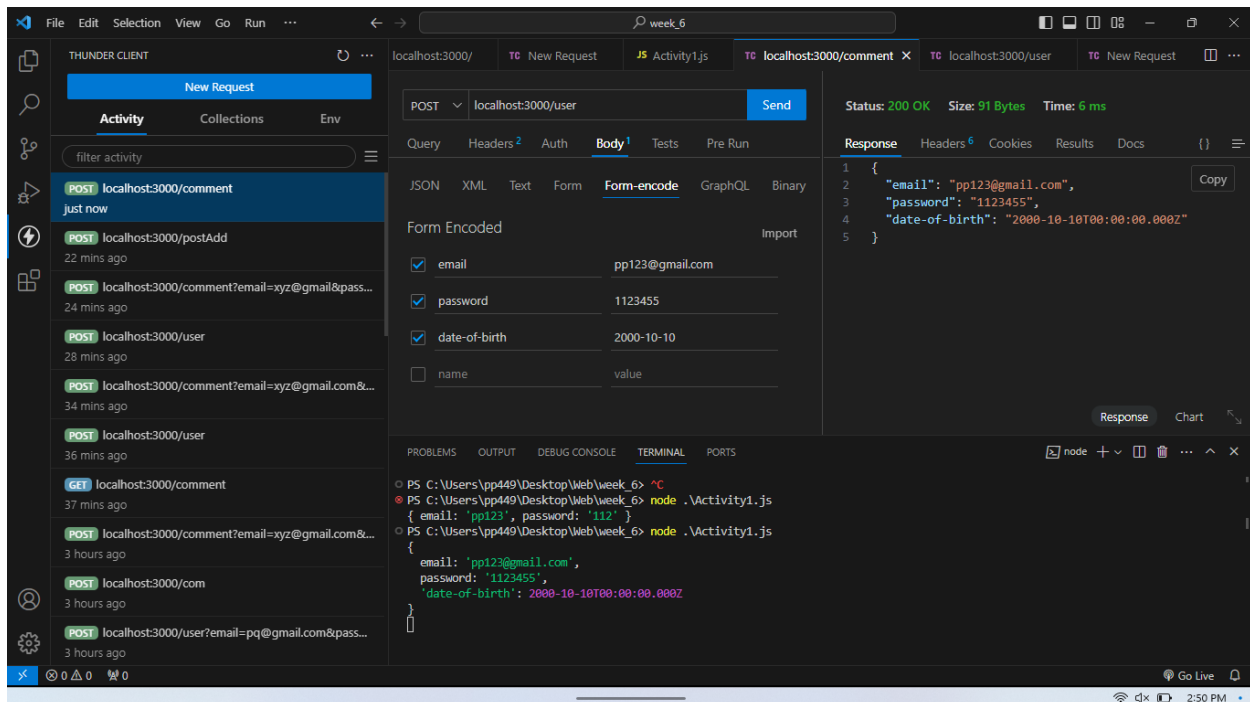
Validation is the process of checking whether the provided data meets certain criteria or constraints, while Sanitization is the process of cleaning, filtering, or modifying input data to ensure that it is safe and free from any potentially harmful content.

Validation results in either valid data or validation errors, while sanitization results in sanitized (cleaned) data that is safe for further processing.

6) Based on your observation, what would be the purpose/use case of the following two commands?

a) `check('date-of-birth').isISO8601().toDate()`

b) `body('date-of-birth').isISO8601().toDate()`



Activity2:

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

```
const { body, check, validationResult } = require('express-validator');
const app = express();
app.use(express.json());

const exphb = require('express-handlebars');

// add middleware -- initialize template

app.engine('.hbs', exphb.engine({ extname: '.hbs' }))
app.set('view engine', 'hbs')

app.get("/getData", function (req, res) {
  var someData = {
    name: "John",
    age: 23,
    occupation: "developer",
    company: "Scotiabank"
  };
  res.json(someData);
});

// This will return the JSON-formatted string:

// { "name": "John", "age": 23, "occupation": "developer", "company":
"Scotiabank" }

//The question is What if we want to return a valid HTML5 page to the client that
actually references some data stored on the server? One solution would be to
build out a string that contains both HTML code and values, ie:
app.get("/viewData", function (req, res) {
  var someData = [{
    name: "John",
    age: 23,
    occupation: "developer",
    company: "Scotiabank",
    fulltime:true
  },{
    name: "cena",
    age: 24,
    occupation: "developer",
    company: "Cibcbank",
    fulltime:false
  }
}
```

```
];
```

```
res.render('viewData', {
  data: someData,
  layout: false
})
//   var htmlString = "<!doctype html>" +
//     "<html>" +

//     "<head>" +

//     "<title>" + "View Data" + "</title>" +

//     "</head>" +

//     "<body>" +

//     "<table border='1'>" +

//     "<tr>" +

//     "<th>" + "Name" + "</th>" +

//     "<th>" + "Age" + "</th>" +

//     "<th>" + "Occupation" + "</th>" +

//     "<th>" + "Company" + "</th>" +

//     "</tr>" +

//     "<tr>" +

//     "<td>" + someData.name + "</td>" +

//     "<td>" + someData.age + "</td>" +

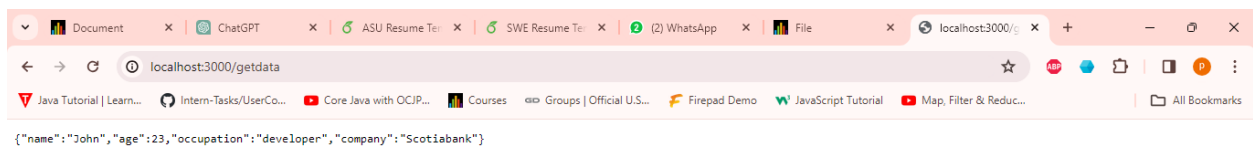
//     "<td>" + someData.occupation + "</td>" +

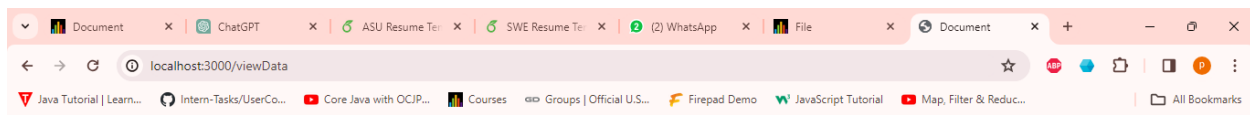
//     "<td>" + someData.company + "</td>" +

//     "</tr>" +

//     "</table>" +
```

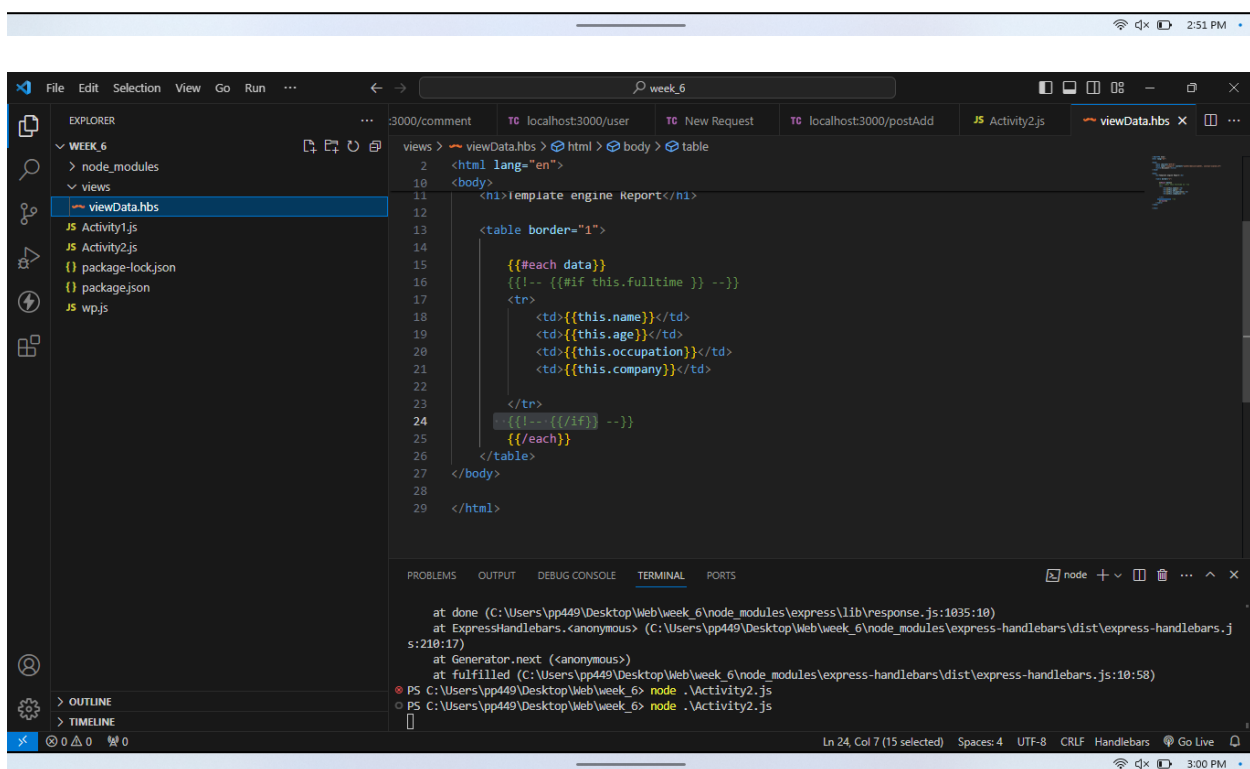
```
//      "</body>" +  
  
//      "</html>";  
  
//      res.send(htmlString);  
});  
app.listen(3000);
```

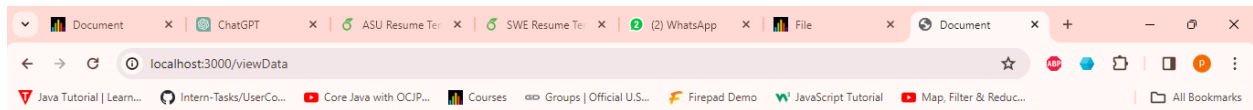
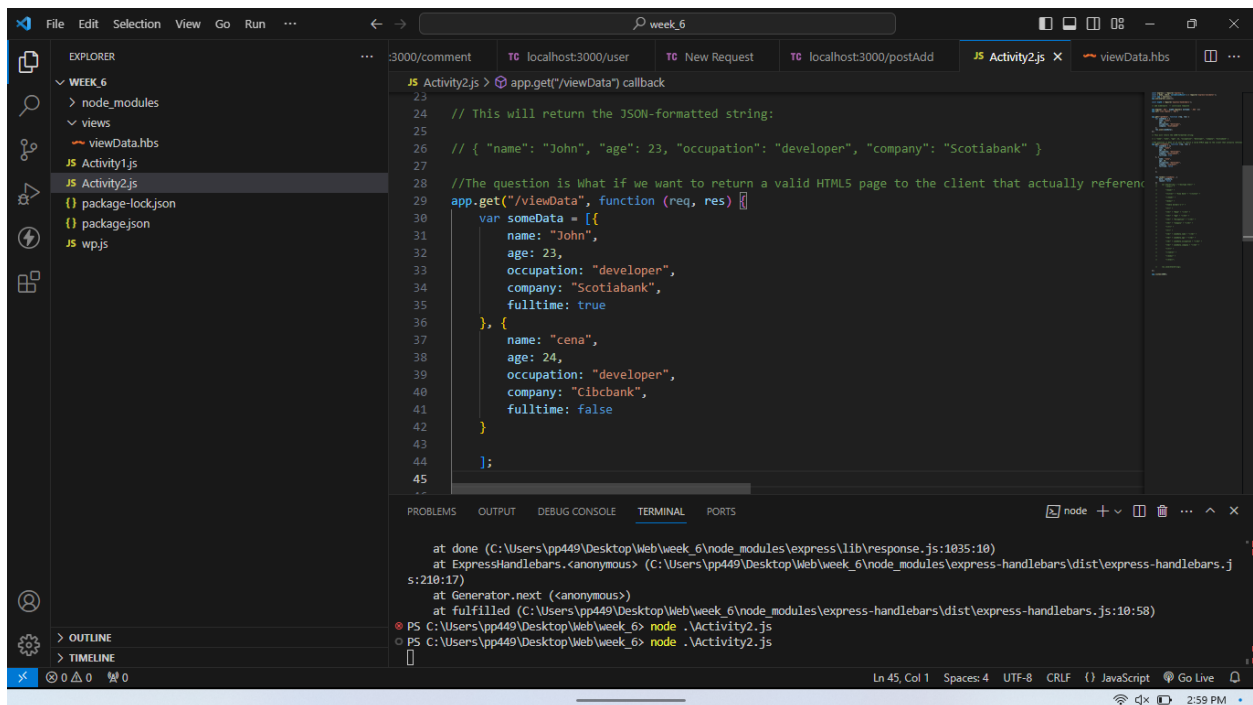




Template engine Report

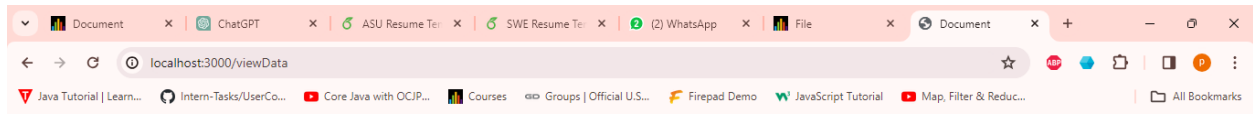
John 23 developer Scotiabank



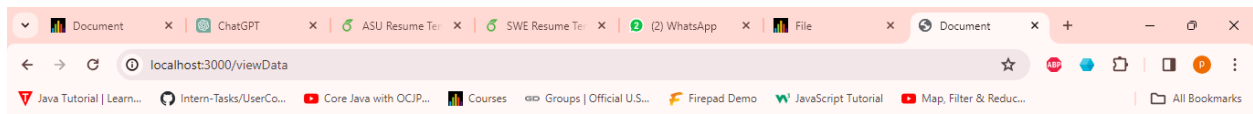
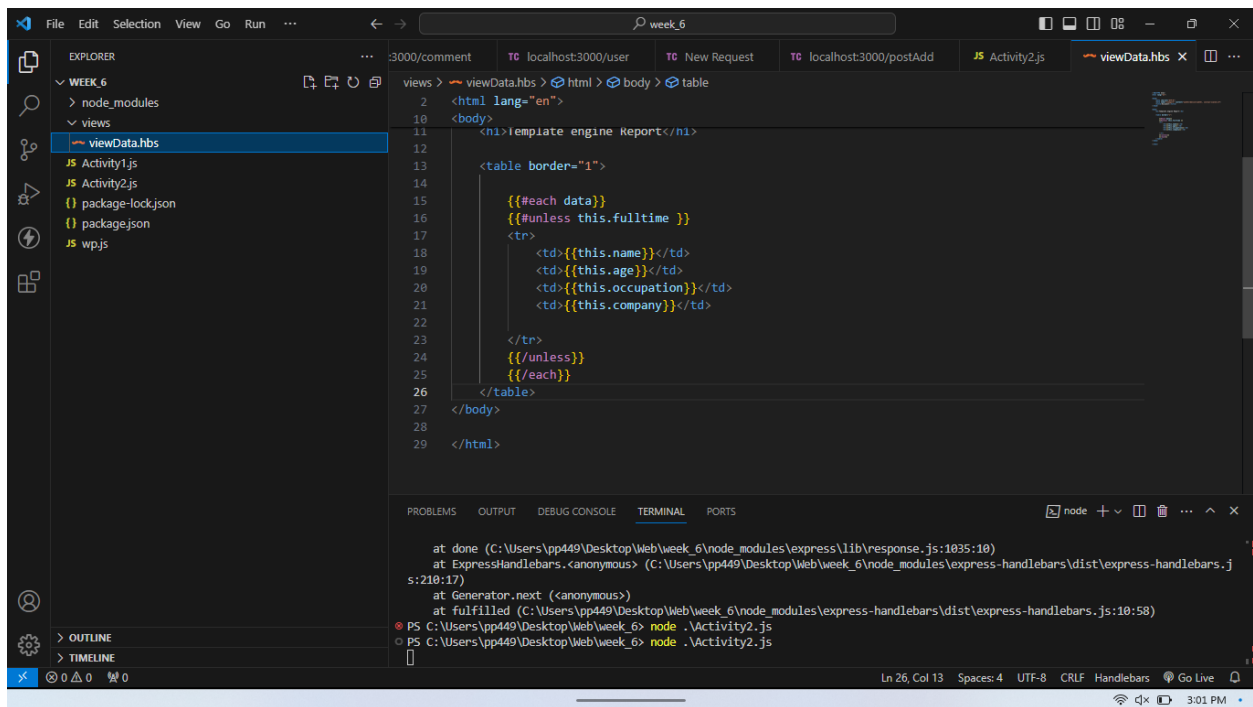


Template engine Report

John	23	developer	Scotiabank
cena	24	developer	Cibcbank



John	23	developer	Scotiabank
------	----	-----------	------------



Template engine Report

cena	24	developer	Cibcbank
------	----	-----------	----------