WEB TECHNOLOGIES

Module 5:
Express JS

Aisha Begam
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS

INTRODUCTION TO WEB SERVICES AND REST API'S

Aisha Begam
Department of Computer Science and Engineering

WEB TECHNOLOGIES
Introductions to Web Services and REST API's

- Web Services
- REST
  - Resource Based
  - HTTP Methods as Actions

WEB TECHNOLOGIES
Introductions to Web Services

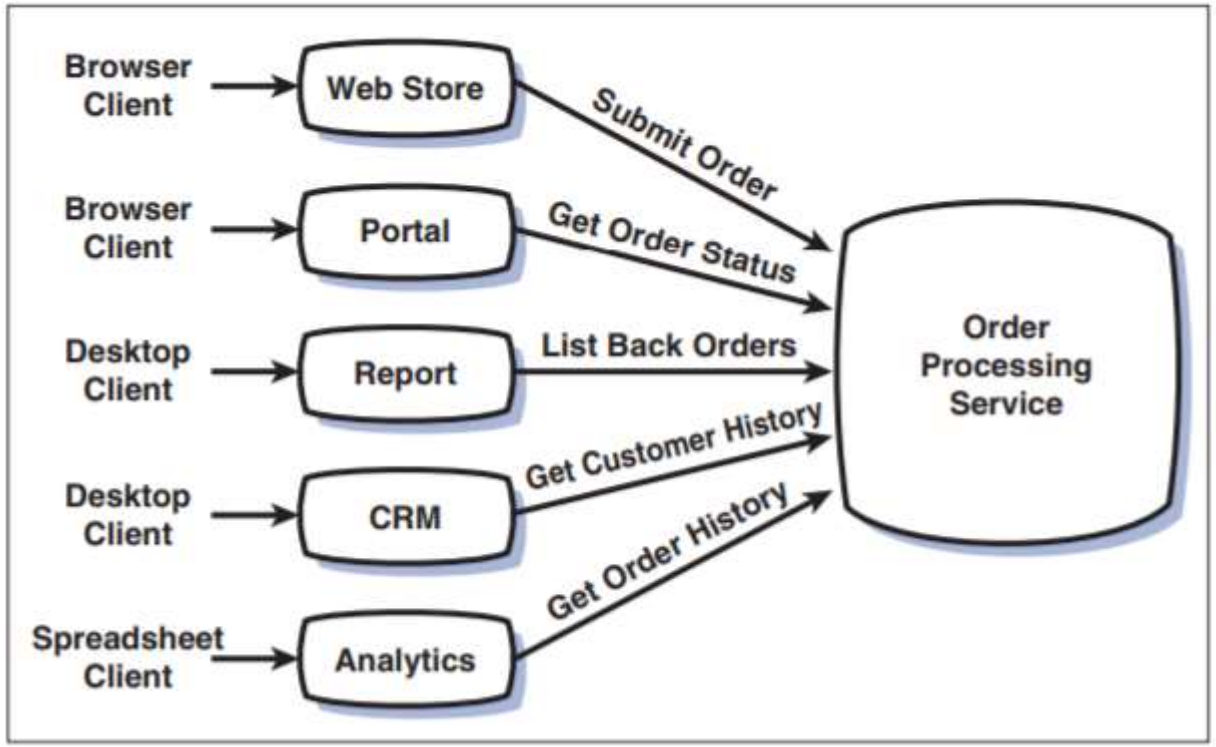
Defining "Web" and "Service"

WEB TECHNOLOGIES
Introductions to Web Services

- A Web service is a service that lives on the Web.
- The ***Web*** is a huge information space filled with interconnected resources.
- A ***service*** is an application that can be consumed by software.
- "Service" refers to the service-oriented architecture.
- An interface hides the complexities of the internal system.
- One service can support many applications.

Source: **Web Services: A Manager's Guide** By Anne Thomas Manes

WEB TECHNOLOGIES
Introductions to Web Services



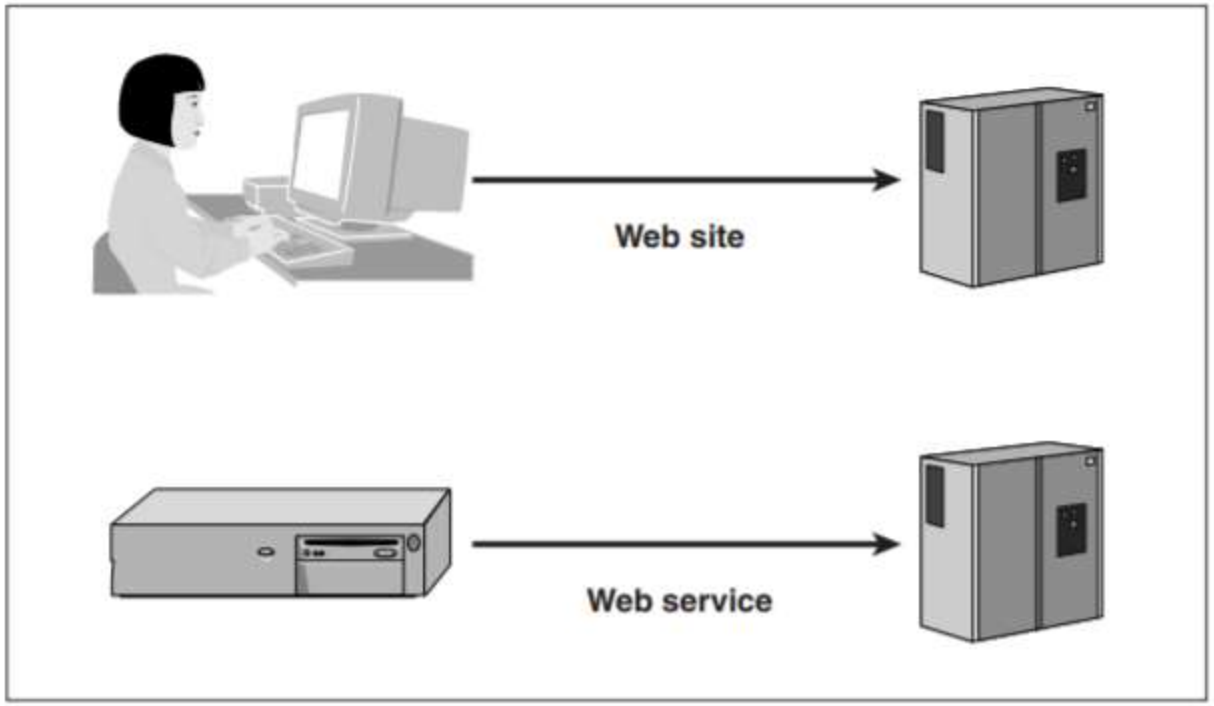
- A service can be shared by many different applications.

Source: Figure 2-1: **Web Services: A Manager's Guide** By Anne Thomas Manes

WEB TECHNOLOGIES
Introductions to Web Services

- A Web site is designed to be accessed by humans.

- A **Web service** is designed to be **accessed by applications**.



Source: Figure 2-1: **Web Services: A Manager's Guide** By Anne Thomas Manes

WEB TECHNOLOGIES
Introductions to Web Services

Defining web service?

WEB TECHNOLOGIES
Introductions to Web Services

- A programmable application component that is accessible via standard Internet protocols.

- **W3C definition**: *Software system designed to support interoperable machine-to-machine interaction over a network.*

- A Web service is an application with a Web API.

PES UNIVERSITY ONLINE

WEB TECHNOLOGIES
Introductions to Web Services

- **The key things to understand about Web services:**

- Designed for machine-to-machine (or application-to-application) interaction.

- Should be interoperable, Not platform dependent.

- Should allow communication over a network.

Why Web Services?

WEB TECHNOLOGIES
Introductions to Web Services

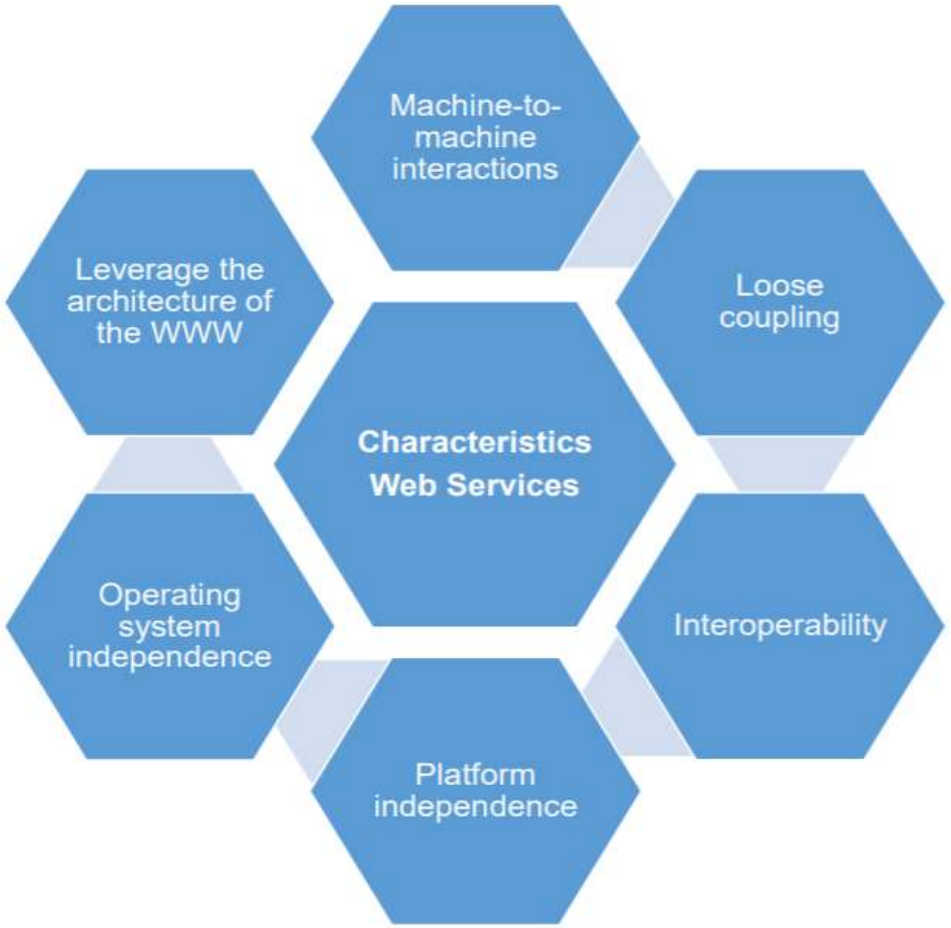
- Web services help you integrate applications.
- Web services support heterogeneous interoperability.
- Web services are inexpensive.
- Web services are flexible and adaptable.

Source: **Web Services: A Manager's Guide** By Anne Thomas Manes

WEB TECHNOLOGIES
Introductions to Web Services

"Characteristics of Web Services"

WEB TECHNOLOGIES
Introductions to Web Services



Source: https://www.slideshare.net/ramawadhp/an-overview-of-web-services-soap-and-rest

WEB TECHNOLOGIES
REST API'S

"REST API's"

WEB TECHNOLOGIES
REST API'S

- REST is about **resources** and how to **represent resources** in different ways.

- REST is about **client-server communication**.

- REST is about how to **manipulate resources**.

- REST offers a **simple**, **interoperable** and **flexible** way of writing web services that can be very different from other techniques.

WEB TECHNOLOGIES
REST API'S

REST is NOT!

- A protocol.
- A standard.
- A replacement for SOAP

WEB TECHNOLOGIES
REST API'S

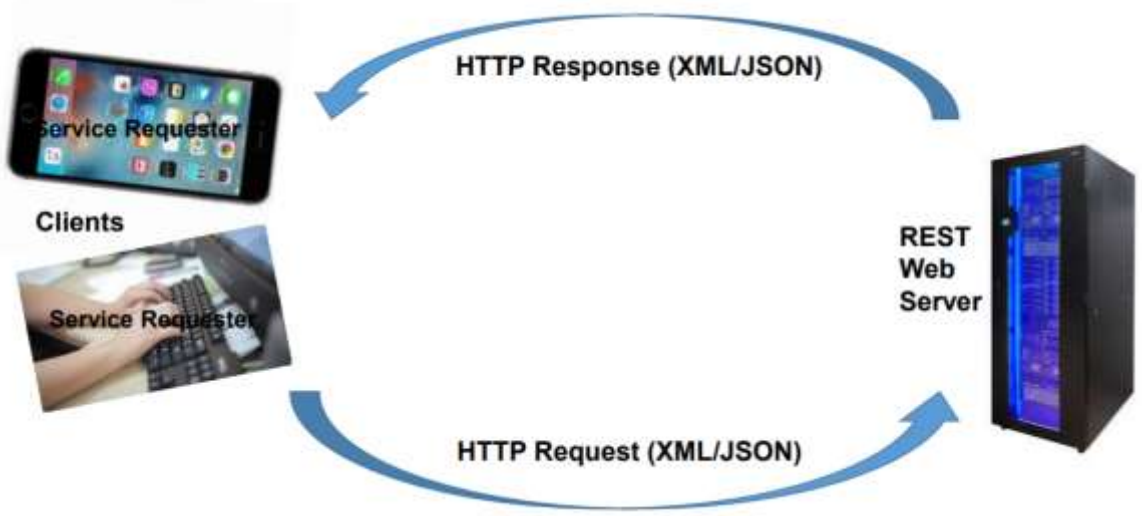
- **Re**presentational **S**tate **T**ransfer
- REST is an architecture all about the **Client-Server communication** .
- Guided by REST constraints (design rules).
- Based on **Resource Oriented Architecture**.
  - A network of web pages where the client progresses through an application by selecting links.
  - Requests/responses relate to representations of states of a resource.
  - When client traverses link, accesses new resource (i.e., transfers state).
- Uses simple **HTTP protocol**.

WEB TECHNOLOGIES
REST API'S

An Architectural Style

- It is an software architectural model which is used to describe distributed systems like **WWW** (**W**orld **W**ide **W**eb).

- It has been developed in parallel with **HTTP** protocol.



Source: https://www.slideshare.net/ramawadhp/an-overview-of-web-services-soap-and-rest

WEB TECHNOLOGIES
REST API'S

- Client **requests** a specific **resource** from the server.
- The server **responds** to that request by delivering the requested resource.
- Server does not have any information about any client.
- So, there is no difference between the two requests of the same client.
- A model which the representations of the resources are transferred between the client and the server.

•The Web as we know is already in this form!

WEB TECHNOLOGIES
REST API'S

Resources

- Resources are consistent mappings from an identifier [such as a URL path] to some set of views on server-side state.
- Every resource must be uniquely addressable via a URI.
- "If one view doesn't suit your needs, then feel free to create a different resource that provides a better view. "
- "These views need not have anything to do with how the information is stored on the server. They just need to be understandable (and actionable) by the recipient." *Roy T. Fielding*

WEB TECHNOLOGIES
REST API'S

RESTful Design Specifications (Constraints)

Client-Server
- Separation of concerns - user interface vs data storage
- Client and server are independent from eachother

Uniform Interface
- All resources are accessed with a generic interface (HTTP-based) which remains same for all clients.

Stateless
- Each request from client to server must contain all of the information
- No client session data or any context stored on the server

Layered System
- Allows an architecture to be composed of hierarchical layers
- Each component cannot "see" beyond the immediate layer.

Cacheable
- Specify data as cacheable or non cacheable
- HTTP responses must be cacheable by the clients

Code On-Demand
- REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts.

Source: https://www.slideshare.net/ramawadhp/an-overview-of-web-services-soap-and-rest

WEB TECHNOLOGIES
REST API'S

HTTP Methods

- GET – *safe, idempotent, cacheable*
- PUT – *idempotent*
- POST
- DELETE – *idempotent*
- HEAD
- OPTIONS

WEB TECHNOLOGIES
REST API'S

Table 5-1. *CRUD Mapping for Collections*

Operation	HTTP Method	Resource	Example	Remarks
Read – List	GET	Collection	GET /customers	Lists objects (additional query string can be used to filter)
Read	GET	Object	GET /customers/1234	Returns a single object (query string may be used to filter fields)
Create	POST	Collection	POST /customers	Creates an object, and the object is supplied in the body.
Update	PUT	Object	PUT /customers/1234	Replaces the object with the object supplied in the body.
Update	PATCH	Object	PATCH /customers/1234	Modifies some attributes of the object, specification in the body.
Delete	DELETE	Object	DELETE /customers/1234	Deletes the object

WEB TECHNOLOGIES
REST API'S

RESTful Design Considerations : Steps for designing RESTful Web Service

- Identifying resources the service will expose over the network.
- Designing the URI Templates: map URIs to resources
- Applying the Uniform HTTP Interface: options available on each resource for different user groups.
- Security Considerations: Authentication and authorization

WEB TECHNOLOGIES
REST API'S

RESTful Design Considerations : Steps for designing RESTful Web Service

- Designing the Resource Representations – XML/JSON.
- Supporting alternate Representations – XML or JSON based on filters
- Providing Resource Metadata – Ability to discover resources and options
- Avoiding RPC Tendencies

WEB TECHNOLOGIES
REST API'S

RESTful Design Considerations : RESTful Service Implementation Considerations

- Parse the incoming request to
  - Use URI to identify the resource.
  - Identify URI variables (and map them to resource variables)
  - HTTP method used in the request (and whether it's allowed for the resource).
  - Read the resource representation

- Authenticate and authorize the user.

WEB TECHNOLOGIES
REST API'S

RESTful Design Considerations : RESTful Service Implementation Considerations

- Use all of this information to perform the underlying service logic.

- Generate an appropriate HTTP response, including
  - Proper status code
  - Description
  - Outgoing resource representation in the response entity body (if any)

WEB TECHNOLOGIES
REST API'S

Requests & Responses

REQUEST

GET /news/ HTTP/1.1

Host: example.org

Accept-Encoding: compress, gzip

User-Agent: Python- httplib2

Here is a **GET** request to «http://example.org/news/»

Method = **GET**

Source: https://www.slideshare.net/hburakcetinkaya/rest-res-tful-web-services

WEB TECHNOLOGIES
REST API'S

Requests & Responses

And here is the response…

RESPONSE

HTTP/1.1 200 Ok

Date: Thu, 07 Aug 2008 15:06:24 GMT

Server: Apache

ETag: "85a1b765e8c01dbf872651d7a5"

Content-Type: text/html

Cache-Control: max-age=3600

<!DOCTYPE HTML>

Source: https://www.slideshare.net/hburakcetinkaya/rest-res-tful-web-services

WEB TECHNOLOGIES
REST API'S

Requests & Responses

- The request is to a resource identified by a **URI**
  - (**URI** = **U**nified **R**esource **I**dentifier).
- In this case, the resource is «http://example.org/news/»
- Resources, or addressability is very important.
- Every resource is URL-addressable.
- To change system state, simply change a resource.

WEB TECHNOLOGIES
REST API'S

JSON

•JSON is the preferred encoding for the data, both in request and response bodies.

•Some REST specifications allow the caller to specify the format (JSON, XML, CSV, etc.),

THANK YOU

Aisha Begam
Department of Computer Science Engineering

aisha.b@pes.edu

+91 9741907626

Express JS

Aruna S
Department of
Computer Science and Engineering

EXPRESS JS

Express JS Introduction

S. Aruna
Department of Computer Science and Engineering

EXPRESS JS Introduction

- ExpressJS is a web application framework that provides you with a simple API to build websites, web apps and back ends. With ExpressJS, we need not worry about low level protocols, processes, etc.
- Express provides a minimal interface to build our applications.
- It provides us the tools that are required to build our app.
- It is flexible as there are numerous modules available on **npm**, which can be directly plugged into Express.
- Express was developed by **TJ Holowaychuk** and is maintained by the Node.js foundation and numerous open source contributors.

EXPRESS JS

EXPRESS JS Features

Few of the most important features of Express.js:

- Express quickens the development pace of a web application.
- It also helps in creating mobile and web application of single-page, multi-page, and hybrid types
- Express can work with various templating engines such as Pug, Mustache, and EJS.
- Express follows the Model-View-Controller (MVC) architecture.
- It makes the integration process with databases such as MONGODB, Redis, MYSQL effortless.
- Express also defines an error-handling middleware.
- It helps in simplifying the configuration and customization steps for the application.

EXPRESS JS Installation

To install Express.js, first, you need to create a project directory and create a package.json file which will be holding the project dependencies.

Below is the code to perform the same:

npm init

To install it globally, you can use the below command:

npm install -g express

To install it locally into your project folder, you need to execute the below command:

npm install express --save

HTTP Common Methods

The most common HTTP Methods are

- **GET**

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.

- **POST**

The POST method creates a new object/entity of the resource identified by the URI.

- **PUT**

The PUT method modifies the existing object identified by the URI. If it does not exist then the PUT method should create one.

- **DELETE**

The DELETE method requests that the server delete the specified resource.

THANK YOU

Aruna S
Department of
Computer Science and Engineering
arunas@pes.edu

WEB TECHNOLOGIES

Module 5:
Express JS

Aisha Begam
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS

ROUTING and URL BINDING

Aisha Begam
Department of Computer Science and Engineering

WEB TECHNOLOGIES

Routing and URL Binding

Routing

- Routing refers to the mechanism for serving the client the content it has asked for.
- It is the most important aspects of your website or web services.
- Routing in Express is simple, flexible, and robust.
- A route specification consists of
  - An HTTP method (GET, POST, etc.),
  - A path specification that matches the request URI,
  - And the route handler.

WEB TECHNOLOGIES

Routing and URL Binding

Routing

•The handler is passed in a request object and a response object.

•The request object can be inspected to get the various details of the request, and

•The response object's methods can be used to send the response to the client

WEB TECHNOLOGIES

Routing and URL Binding

Routing

•Create an application using its root-level exported function

• ***const app = express();*** Set up routes using this app.

• To set up a route, use a function to indicate which HTTP method; for e.g., to handle the GET,PUT, DELETE etc.,

•To this function, pass the pattern to match and a function to deal with the request if it does match.

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES

Routing and URL Binding

Request Matching

- The request's method is matched with the route method (e.g., the get function was called on app, indicating it should match only GET HTTP methods)

- The request URL with the path spec matches ('/hello'), then the handler is called.

- In other words, "If you receive a GET request to the URL */hello*, then execute *this piece of code.*"

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES

Routing and URL Binding

Request Matching

- The method and the path need not be very specific. E.g., app.get(), app.post(), app.put(), etc.,

- If you want to say “any method,” you could use app.all().

- The path specification can also take regular expression-like patterns (like '/*.do') or regular expressions themselves.

- Regular expressions in paths are rarely used. Route *parameters in the path are used often.*

- They are named segments in the path specification that match a part of the URL.

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES

Routing and URL Binding

Route Parameters

•If a match occurs, the value in that part of the URL is supplied as a variable in the request object.

```
app.get('/customers/:customerId', ...
```

•The customer ID will be captured and supplied to the handler function as part of the request in req.params, with the name of the parameter as the key.

•req.params.customerId can have any value for each of these URLs and can have multiple parameters.

•For e.g., /customers/:customerId/orders/:orderId, to match *a customer's order.*

WEB TECHNOLOGIES

Routing and URL Binding

Route Lookup

•Multiple routes can be set up to match different URLs and patterns.

•The router tries to match all routes in the order in which they are installed.

•If two routes are possible matches to a request, it will use the first defined one.

•Can define routes in the order of priority.

WEB TECHNOLOGIES

Routing and URL Binding

Route Lookup

When you add patterns its recommended to add more generic pattern *after the specific paths.*

•*For e.g, if you want to match everything that* goes under /api/, that is, a pattern like /api/*,

•Add this route only *after all the* specific routes that handle paths such as /api/issues.

WEB TECHNOLOGIES

Routing and URL Binding

Handler Function

•Once a route is matched, the handler function is called.

•The parameters passed to the handler are a request object and a response object.

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

1. To access a parameter value we use ***req.params.*** **req.param(name [, defaultValue])**

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

2. ***req.query***: This holds a parsed query string.

- It's an object with keys: as the query string parameters and
- Values as the query string values.
- Multiple keys with the same name are converted to arrays, and
- Keys with a square bracket notation result in nested objects

(e.g., order[status]=closed can be accessed as req.query.order.status).

```
{ key: value }
```

```
?key1=value1&key2=value2&key3=value3
```

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

3. ***req.header, req.get(header)***:
   - The get method gives access to any header in the request.
   - The header property is an object with all headers stored as key-value pairs.

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

4. ***req.path:***

- The path for which the middleware function is invoked; can be any of:
  - A string representing a path.
  - A path pattern.
  - A regular expression pattern to match paths.
  - An array of combinations of any of the above.

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

5. ***req.url, req.originalURL***:
   - Contain the complete URL, including the query string.
   - If any middleware modifies the request URL, originalURL retains the original URL as it was received, *before the modification.*

WEB TECHNOLOGIES

Routing and URL Binding

Request Objects

6. ***req.body:***

- Contains key-value pairs of data submitted in the request body.
- By default, it is undefined, and is populated when you use body-parsing middleware is installed to read and optionally interpret or parse the body.

WEB TECHNOLOGIES

Routing and URL Binding

Response Objects

The ***res*** object represents the HTTP response that an Express app sends when it gets an HTTP request.

```
res.send('<p>some html</p>')
```

1. ***res.send(body)***: Sends the HTTP response.
   - If the body is an object or an array, it is automatically converted to a JSON string with an appropriate content type.

WEB TECHNOLOGIES

Routing and URL Binding

Response Objects

The ***res*** object represents the HTTP response that an Express app sends when it gets an HTTP request.

```
res.status(400).send('Bad Request')
```

2. ***res.status(code)***: This sets the response status code.
   - If not set, it is defaulted to 200 OK.
   - One common way of sending an error is by combining the status() and send() methods in a single call like res.status(403).send("Access Denied").

WEB TECHNOLOGIES

Routing and URL Binding

Response Objects

3. ***res.json(object)***: Sends a JSON response
   - This method sends a response (with the correct content-type) that is the parameter converted to a JSON string using JSON.stringify().
   - The parameter can be any JSON type, including object, array, string, Boolean, number, or null

```
res.json({ user: 'tobi' })
```

WEB TECHNOLOGIES

Routing and URL Binding

Response Objects

4. ***res.sendFile(path)***:
   - This responds with the contents of the file at path.
   - The content type of the response is guessed using the extension of the file.

```
res.sendFile('/uploads/' + uid + '/' + file)
```

WEB TECHNOLOGIES

Routing and URL Binding

```
var express = require('express');
var app = express();
app.get('/getPESLocation', function(req, res){
    res.send("At Bangalore!");
});
app.listen(8080);
```

• Run the app and go to http://localhost:8080/getPESLocation , the server receives a get request at route **"/getPESLocation"**, our Express app executes the **callback** function attached to this route and sends **"At Bangalore!"** as the response. Use app.all to set many methods for a single route.

WEB TECHNOLOGIES

Routing and URL Binding

- Defining routes explicitly are very tedious to maintain.
- To separate the routes from our main **index.js** file, we will use **Express.Router**

THANK YOU

Aisha Begam
Department of Computer Science Engineering

aisha.b@pes.edu

+91 9741907626

WEB TECHNOLOGIES

Module 5:
Express JS

Aisha Begam
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS

Aisha Begam
Department of Computer Science and Engineering

WEB TECHNOLOGIES
ERROR HANDLING

Express JS

- ***Error Handling*** refers to how Express catches and processes errors that occur both synchronously and asynchronously

- The success or failure of any REST API call is typically reflected in the HTTP status code.
  - E.g., 400 Bad Request status code or syntactically incorrect requests.
  - 422 Unprocessable Entity

WEB TECHNOLOGIES
ERROR HANDLING

Express JS

- The error message (a description of what went wrong) itself can be returned in the response body, using JSON string.

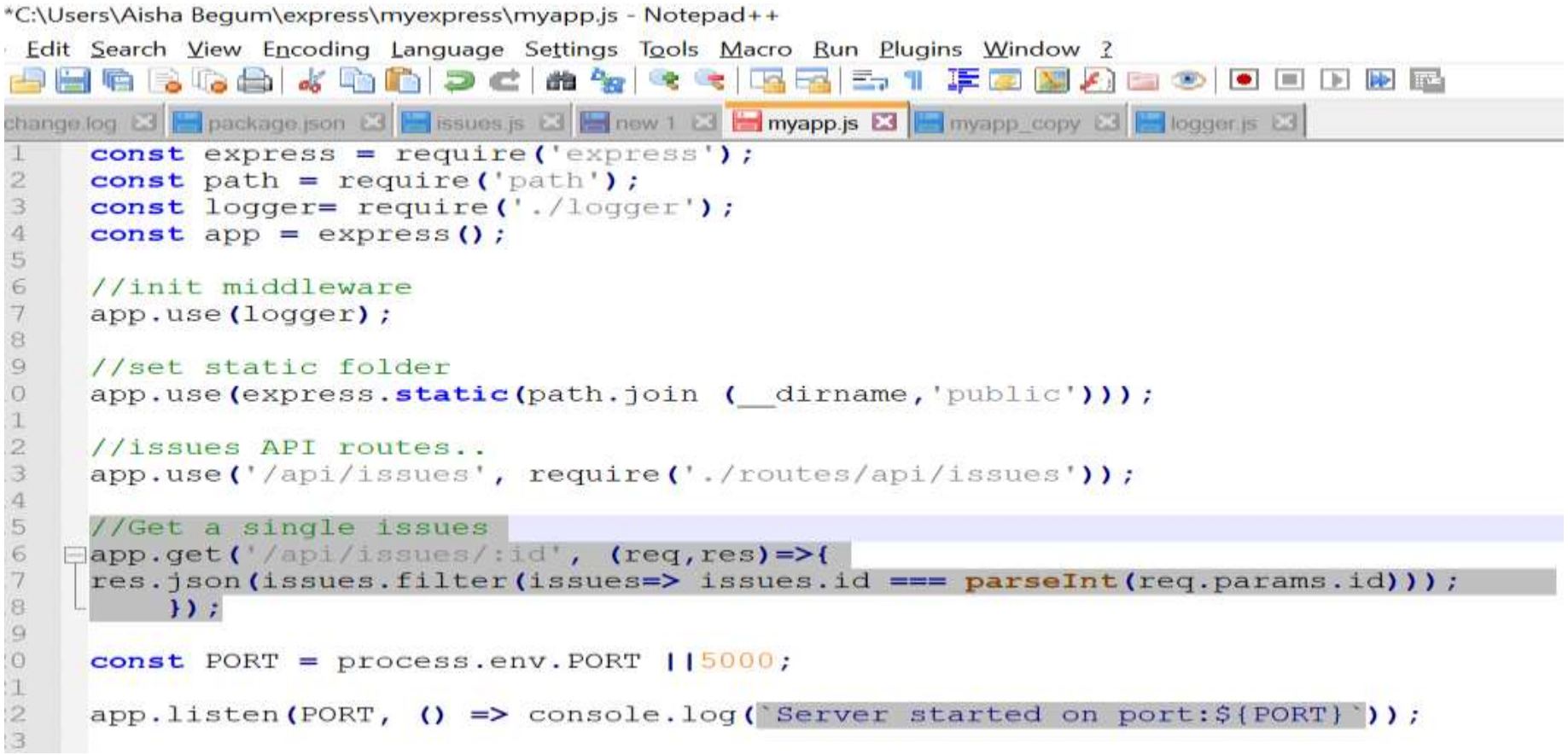
- We will return an object with a single property called message that holds a readable as the description.

- At the server, sending an error is simple; just set the status using res.status() and send the error message as the response.

WEB TECHNOLOGIES
ERROR HANDLING

Express JS

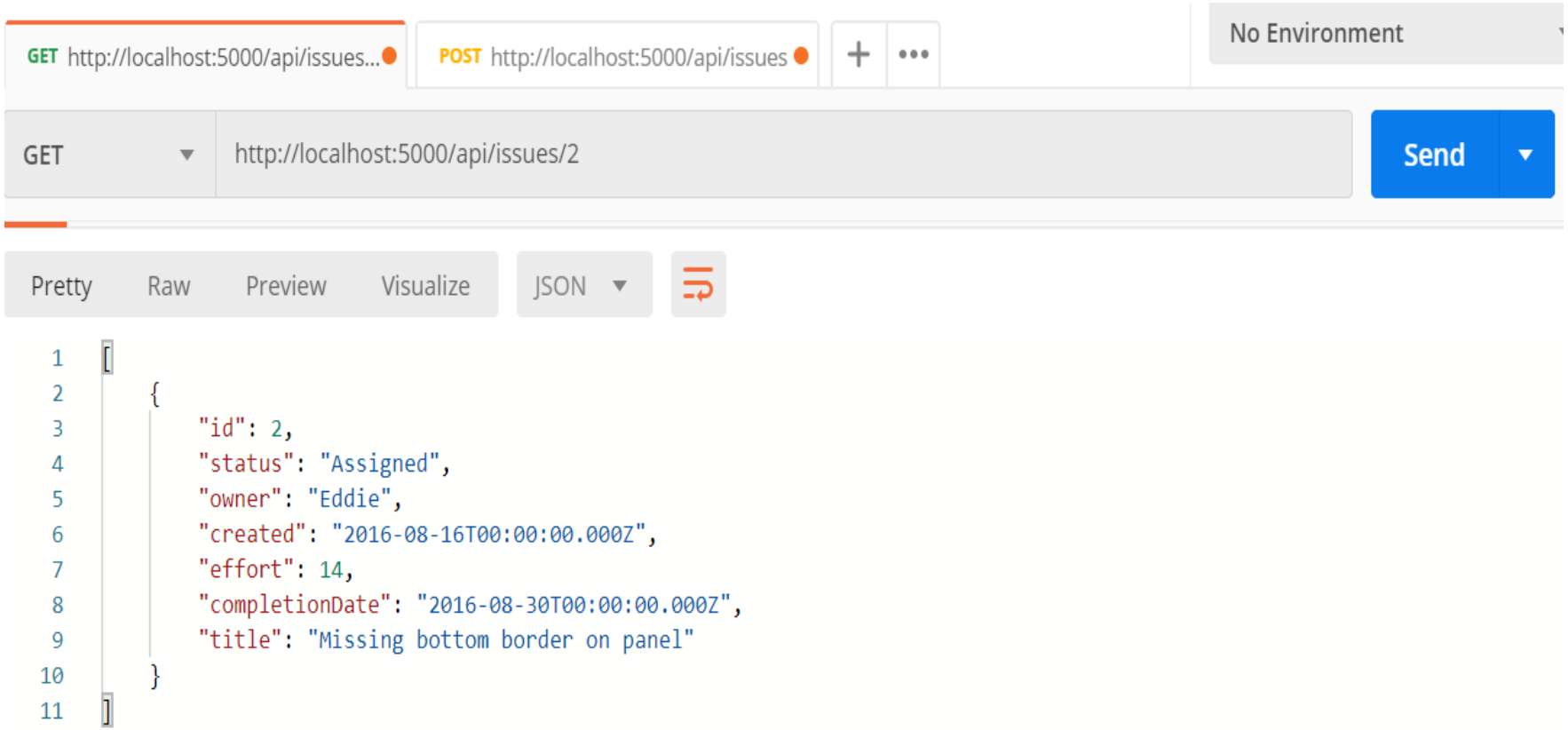
Here we are trying to get the single issues from our API List



WEB TECHNOLOGIES
ERROR HANDLING

Express JS

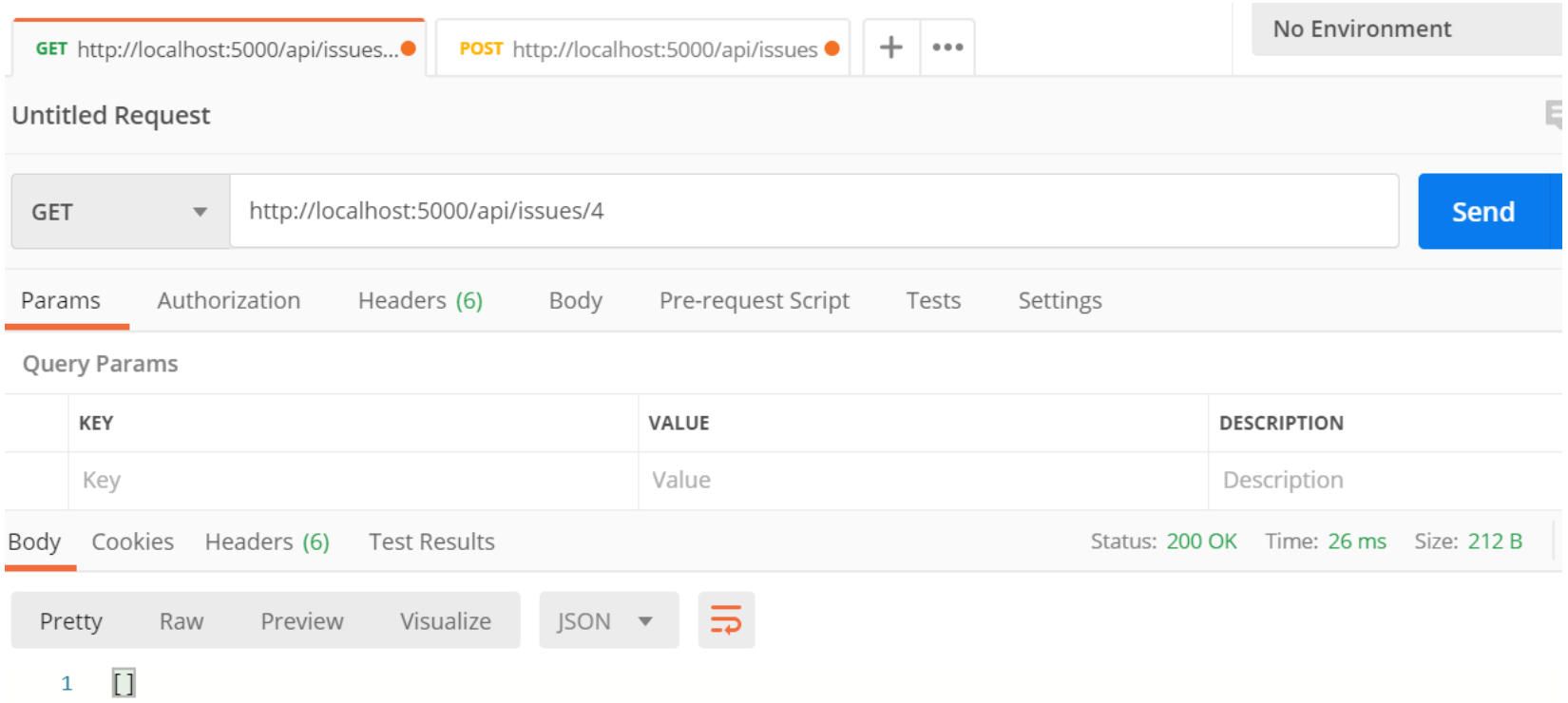
Here we are trying to get the single issues from our API List



WEB TECHNOLOGIES
ERROR HANDLING

Express JS

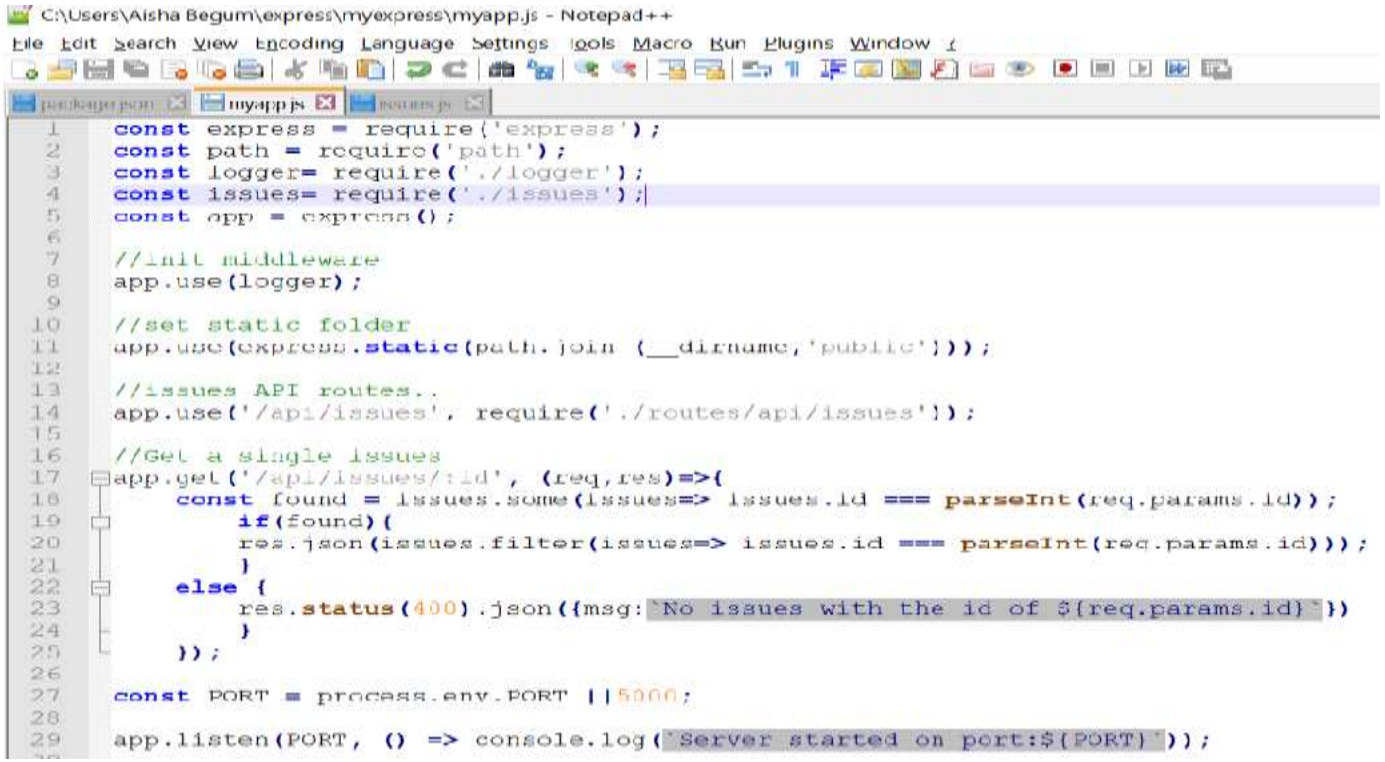
Here we are trying to get the single issues from our API List
Notice that we have requested for id=4 which is not there in the API List. Hence, does not show any result



WEB TECHNOLOGIES
ERROR HANDLING

Express JS

This code is trying to handle the issue with res.status HTTP response



WEB TECHNOLOGIES
ERROR HANDLING

Express JS

- Here we are trying to get the single issues from our API List
- Notice that we have requested for id=4 which is not there in the API List. Hence, error message is shown.

THANK YOU

Aisha Begam
Department of Computer Science Engineering

aisha.b@pes.edu

+91 9741907626

WEB TECHNOLOGIES

Module 5:
Express JS

Aisha Begam
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS
MIDDLEWARE

Aisha Begam
Department of Computer Science and Engineering

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Express is a framework that does minimal work and gets most of the job done by functions called *middleware.*

- *A middleware is a function that takes in an* HTTP request and response object, plus the next middleware function in the chain.

- The function can look at and
  - modify the request and response objects,
  - respond to requests, or
  - decide to continue with middleware chain by calling the next function.

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- The ***express.static*** generator function generates one such middleware function.

- This middleware responds to a request by trying to match the request URL with a file under a directory specified by the parameter to the generator function.

- If a file exists,
  - It returns the contents of the file as the response; else
  - It chains to the next middleware function.

- The middleware is *mounted on the application using the* ***application's use()*** method.

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- The middleware generator takes the parameter static to indicate that this is the directory where all the static files reside.

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

const app = express();
app.use(express.static('static'));

app.listen(3000, function () {
  console.log('App started on port 3000');
});
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- The ***express.static*** generator function generates one such middleware function.
- Helps to serve static files

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

const app = express();

//set static folder
app.use(express.static(path.join (__dirname,'public')));

const PORT = process.env.PORT ||5000;

app.listen(PORT, () => console.log(`Server started on port:${PORT}`));
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Created a public folder and added index.html and about.html.

- Parameter static to indicate that this is the directory where all the static files reside

log | package.json | issues.js | new 1 | myapp.js | index.html | about.html

```
<!doctype html>
<html lang="en">
  <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, intial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="css/style.css"/>
    <title>My website</title>
  </head>
  <body>
    <h1>My Website!!!</h1>
    </body>
</html>
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Created a public folder and added index.html and about.html.

- Parameter static to indicate that this is the directory where all the static files reside

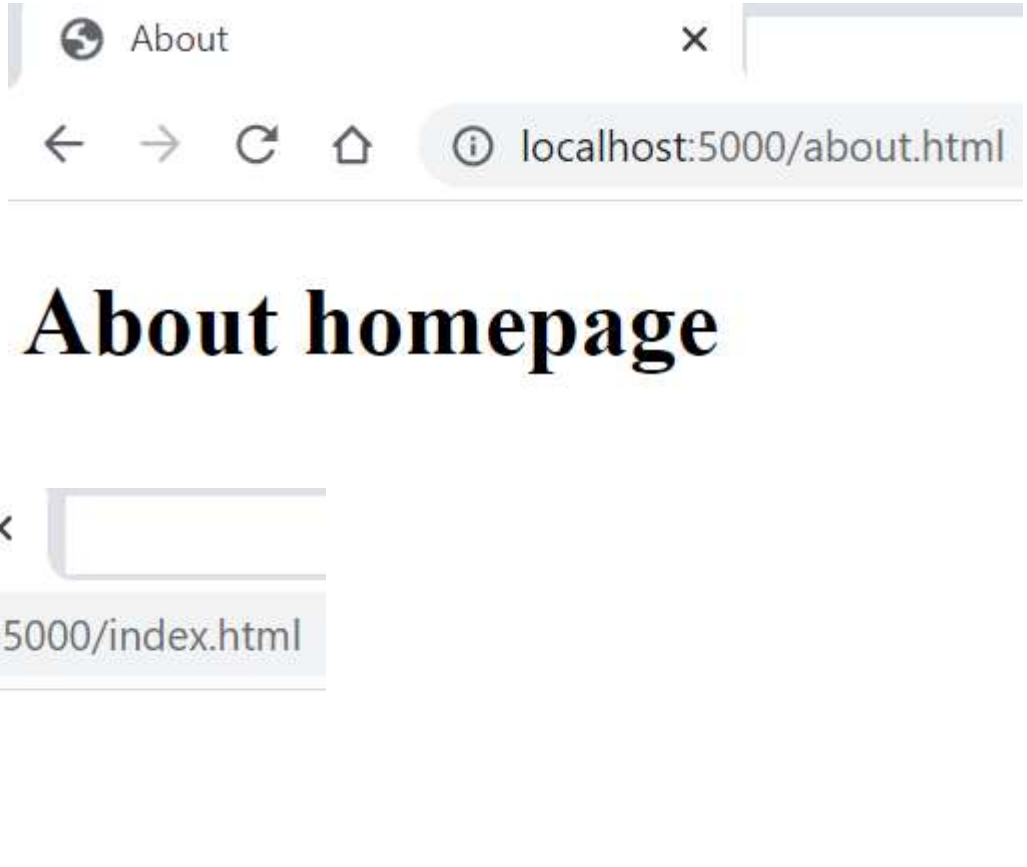
og | package.json | issues.js | new 1 | myapp.js | index.html | about.html

```
<!doctype html>
<html lang="en">
  <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, intial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="css/style.css"/>
    <title>About</title>
  </head>
  <body>
    <h1>About homepage</h1>
    </body>
</html>
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Output in the browser



WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- In express the Router itself is a middleware function.

- Other very useful middleware supported by the Express team, is the body-parser.

- Middleware can be at
  - the application level (i.e., applies to all requests)
  - or
  - The router level (applies to specific request path patterns).

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- To use a middleware at the application level supply the function to the application,
  - using: ***app.use(middlewareFunction);***

- To use the same middleware in a route-specific way, you could have done the following:
  - ***app.use('/public', express.static('static'));***

- This would have mounted the static files on the path /public and all static files would have to be accessed with the prefix /public, for example, /public/index.html.

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- The JSON parser middleware is created using ***bodyParser.json(),*** and is mounted at the application level using ***app.use().***

- The middleware ***bodyParse***r looks at the Content-Type header and determines if and how the body can be parsed.

- For JSON, the default content type is application/json.

- In the absence of this header, ***bodyParser*** does not parse the request body, and the variable ***req.body*** ends up as an empty
- object.

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Initializing a middleware called logger which will print Hello in the console when a request is made.

nge.log | package.json | issues.js | new 1 | myapp.js | myapp_copy

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

const app = express();

const logger=(req,res,next)=>{
    console.log('Hello');
    next();
};
//init middleware
app.use(logger);

//set static folder
app.use(express.static(path.join (__dirname,'public')));

//issues API routes..
app.use('/api/issues', require('./routes/api/issues'));

const PORT = process.env.PORT ||5000;

app.listen(PORT, () => console.log(`Server started on port:${PORT}`));
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

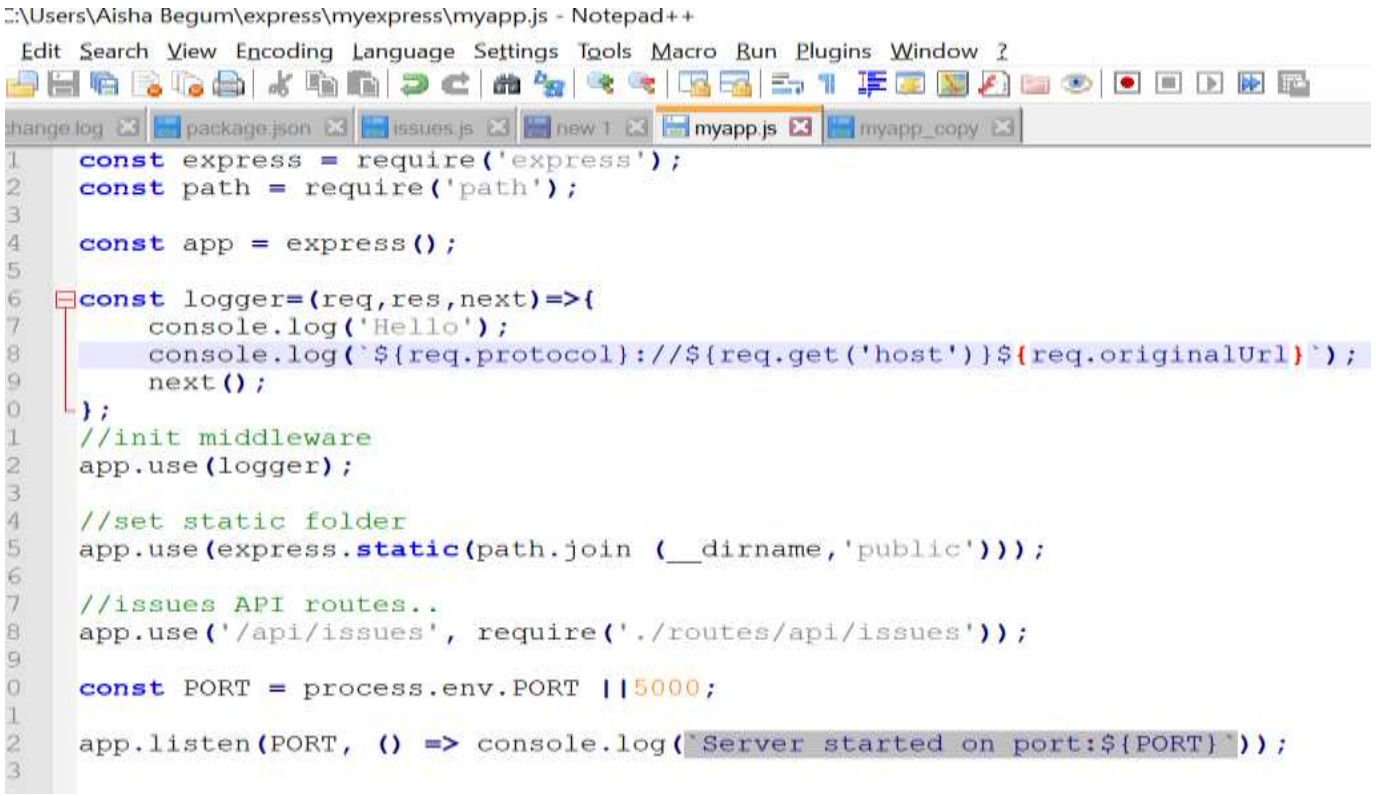
- Initializing a middleware called logger which will print Hello in the console when a request is made.

```
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting `node myapp.js`
Server started on port:5000
[nodemon] restarting due to changes...
[nodemon] starting `node myapp.js`
Server started on port:5000
Hello
Hello
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Initializing a middleware called logger which will print Hello, and the whole URL in the console when a request is made.



WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Output: after sending a request on the postman we get the original URL

```
    at C:\Users\Aisha Begum\express\myexpress\
[nodemon] restarting due to changes...
[nodemon] starting `node myapp.js`
Server started on port:5000
Hello
http://localhost:5000/api/issues/
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

Command Prompt

```
C:\Users\Aisha Begum\express\myexpress>npm i moment
npm WARN myexpress@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.3 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ moment@2.26.0
added 1 package from 6 contributors and audited 179 packages in 8.36s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


C:\Users\Aisha Begum\express\myexpress>
```

- Lets print the date and time
- Need to install pacage moment
- Use: ***npm i moment***

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- Included the package with require

C:\Users\Aisha Begum\express\myexpress\myapp.js - Notepad++

Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

change.log | package.json | issues.js | new 1 | myapp.js | myapp_copy

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

const app = express();
const moment = require('moment');
const logger=(req,res,next)=>{
    console.log('Hello');
    console.log(`${req.protocol}://${req.get('host')}
    ${req.originalUrl}:
    ${moment().format()}`);
    next();
};
//init middleware
app.use(logger);

//set static folder
app.use(express.static(path.join (__dirname,'public')));

//issues API routes..
app.use('/api/issues', require('./routes/api/issues'));

const PORT = process.env.PORT ||5000;

app.listen(PORT, () => console.log(`Server started on port:${PORT}`));
```

WEB TECHNOLOGIES
MIDDLEWARE

Express JS

- After sending a request on postman to display the issues
- ***Notice that the console has the date and time displayed on it***

```
C:\Users\Aisha Begum\express\myexpress>npm run dev

> myexpress@1.0.0 dev C:\Users\Aisha Begum\express\myexpress
> nodemon myapp

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node myapp.js`
Server started on port:5000
Hello
http://localhost:5000
        /api/issues/:
        2020-06-16T11:57:09+05:30
```

THANK YOU

Aisha Begam
Department of Computer Science Engineering

aisha.b@pes.edu

+91 9741907626

WEB TECHNOLOGIES

Module 5:
Express JS

Shruthi L
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS
FORM DATA

Shruthi L
Department of Computer Science and Engineering

WEB TECHNOLOGIES
FORM DATA

Express JS-Form data

- Forms are an integral part of the web.
- Install
  - *body-parser*(for parsing JSON and url-encoded data)
  - multer(for parsing multipart/form data) middleware.

npm install body-parser multer

Create index.js file and add the following modules –

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

var bodyParser = require('body-parser');

var multer = require('multer');
```

WEB TECHNOLOGIES
FORM DATA

- To read form (form.pug)
  app.get()
- To create a view
  app.set()
- For parsing application/json
  app.use()
- To display
  app.post()
- Specify port
  app.listen(3000);

```
6
7   app.get('/', function(req, res){
8      res.render('form');
9   });
10
11  app.set('view engine', 'pug');
12  app.set('views', './views');
13
14  app.use(bodyParser.json());
15  app.use(bodyParser.urlencoded({ extended: true }));
16  //form-urlencoded
17  // for parsing multipart/form-data
18  app.use(upload.array());
19  app.use(express.static('public'));
20
21  app.post('/', function(req, res){
22     console.log(req.body);
23     res.send("recieved your request!");
24  });
```

WEB TECHNOLOGIES
FORM DATA

- Create a template html form using form.pug

HTTPserver.js | form.pug | eventemitter.js

: > Users > shrug > express > form.pug

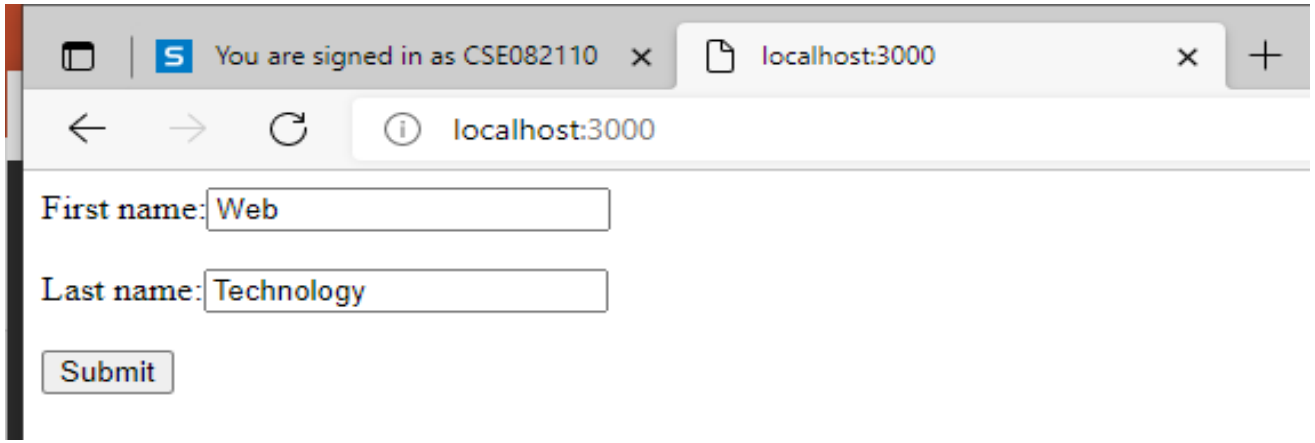
```
1   html
2   html
3       head
4
5       body
6           form(action = "/", method = "POST")
7               div
8                   label(for = "first") First name:
9                   input(name = "first" value = "Web")
10              br
11              div
12                  label(for = "last") Last name:
13                  input(name = "last" value = "Technology")
14              br
15              button(type = "submit") Submit
```

WEB TECHNOLOGIES
FORM DATA

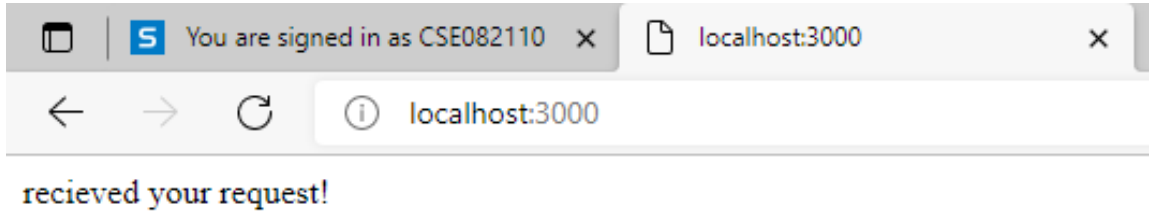
- Run your server using the following.

nodemon index.js

Now go to localhost:3000/ following is displayed in browser---



- Response is generated after submit



WEB TECHNOLOGIES

FORM DATA

- In console; it will show you the body of your request as a JavaScript object as in the following screenshot –

```
PS C:\Users\shrug> cd .\express\
PS C:\Users\shrug\express> cd .\express_eg\
PS C:\Users\shrug\express\express_eg> node index.js
PS C:\Users\shrug\express\express_eg> node index.js
{ first: 'Web', last: 'Technology' }
```

THANK YOU

Shruthi L
Department of Computer Science Engineering

shruthil@pes.edu

WEB TECHNOLOGIES

Module 5:
Express JS

Aisha Begam
Department of Computer Science and Engineering.

WEB TECHNOLOGIES

Express JS
FILE UPLOAD

Aisha Begam
Department of Computer Science and Engineering

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

- In Express.js, file upload is slightly difficult because of its asynchronous nature and networking approach.
- It can be done by using middleware to handle multipart/form data.
- There are many middleware that can be used like multer, connect, body-parser etc.
- There are two popular and robust options for multipart form processing: Busboy and Formidable.
- Formidable to be slightly easier, as it has a callback that provides objects containing the fields and the files,
- In Busboy, you must listen for each field and file event.

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

- Simple express middleware for uploading files.
- ***npm i express-fileupload***

```
C:\Users\Aisha Begum\express\myexpress>npm i express -fileupload
npm WARN myexpress@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.3 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ express@4.17.1
updated 1 package and audited 180 packages in 7.902s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


C:\Users\Aisha Begum\express\myexpress>
```

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

- When you upload a file, the file will be accessible from req.files

- It is similar to req.body and is turned on either by express.bodyParser() or express.multipart() middlewares.

- Express.js (and other modules behind the scene) process the request data (which is usually a form) and give us extensive information in the req.files.FIELD_NAME object.

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

To illustrate how req.files works, let's add this route to the req project:

```
app.post('/upload', function(req, res){
console.log(req.files.archive);
//read req.files.archive.path
//process the data
//save the data
res.end();
})
```

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

Example:

- You're uploading a file called **car.jpg**
- Your input's name field is **foo**: <input name="foo" type="file" />
- In your express server request, you can access your uploaded file from req.files.foo:

```
app.post('/upload', function(req, res) {
  console.log(req.files.foo); // the uploaded file object
});
```

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

The **req.files.foo** object will contain the following:
req.files.foo.name: "car.jpg“

req.files.foo.mv: A function to move the file elsewhere on your server. Can take a callback or return a promise.

req.files.foo.mimetype: The mimetype of your file

req.files.foo.data: A buffer representation of your file, returns empty buffer in case useTempFiles option was set to true.

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

The **req.files.foo** object will contain the following:

req.files.foo.tempFilePath: A path to the temporary file in case useTempFiles option was set to true.

req.files.foo.truncated: A boolean that represents if the file is over the size limit

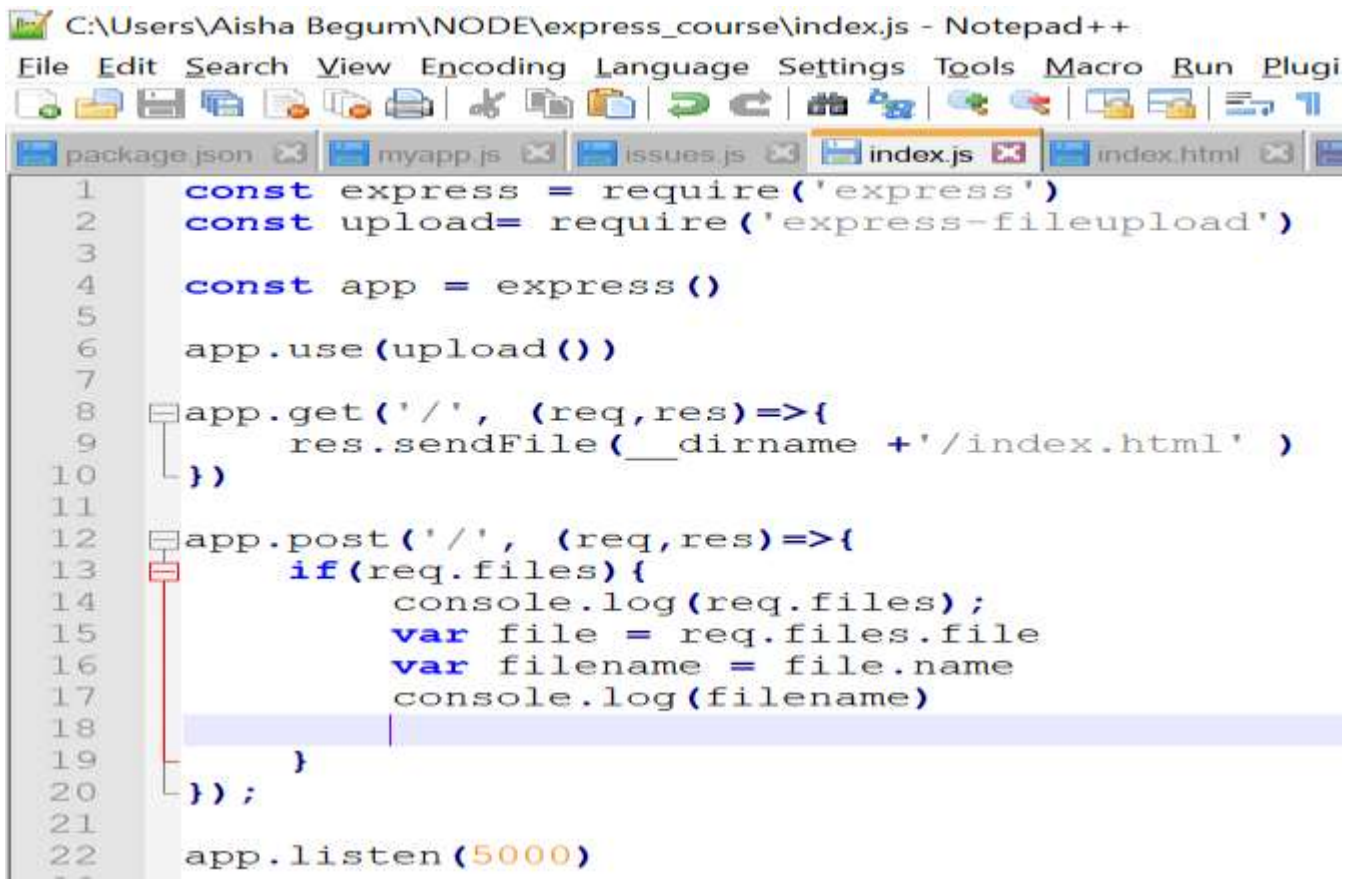
req.files.foo.size: Uploaded size in bytes

req.files.foo.md5: MD5 checksum of the uploaded file

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

Index.js for printing out the file name and details of uploaded file

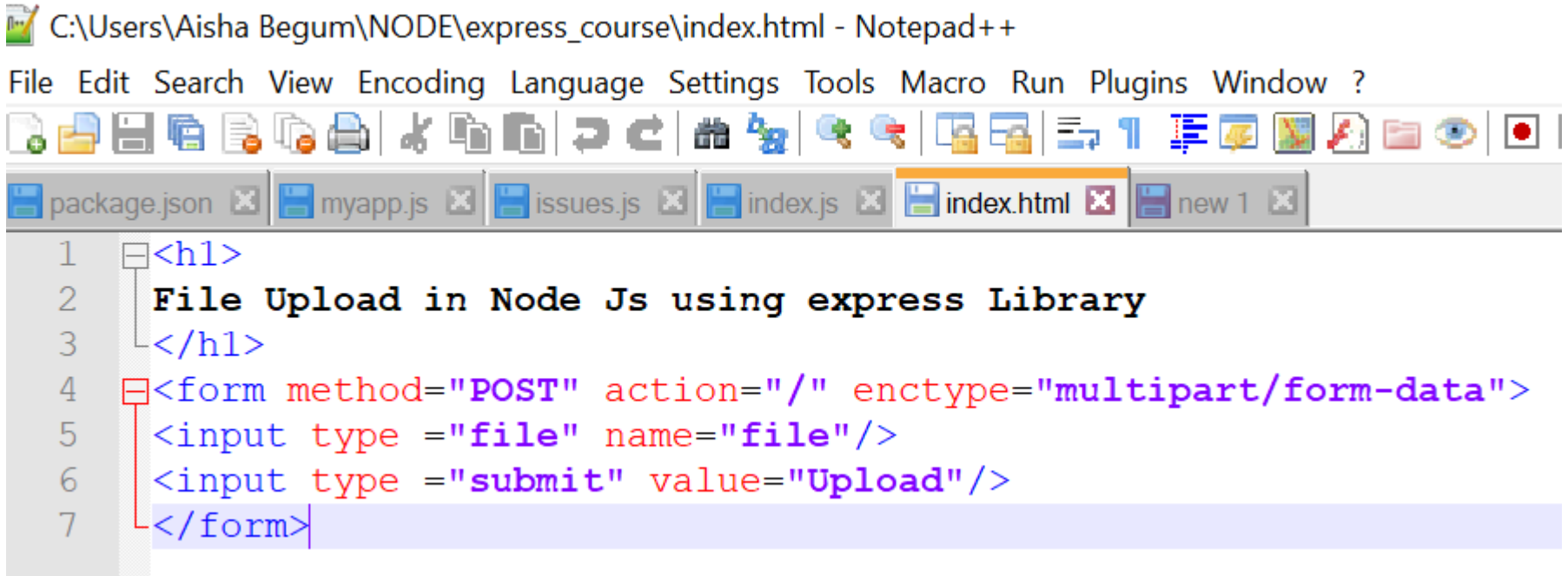


WEB TECHNOLOGIES
FILE UPLOAD

PES UNIVERSITY ONLINE

Express JS

Index.html: as simple file upload script shown as below:



WEB TECHNOLOGIES
FILE UPLOAD

Express JS

In browser when we type the url: http://localhost:5000

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

In browser when we type the url: http://localhost:5000 and choose file.
Here I have chosen images.jpg

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

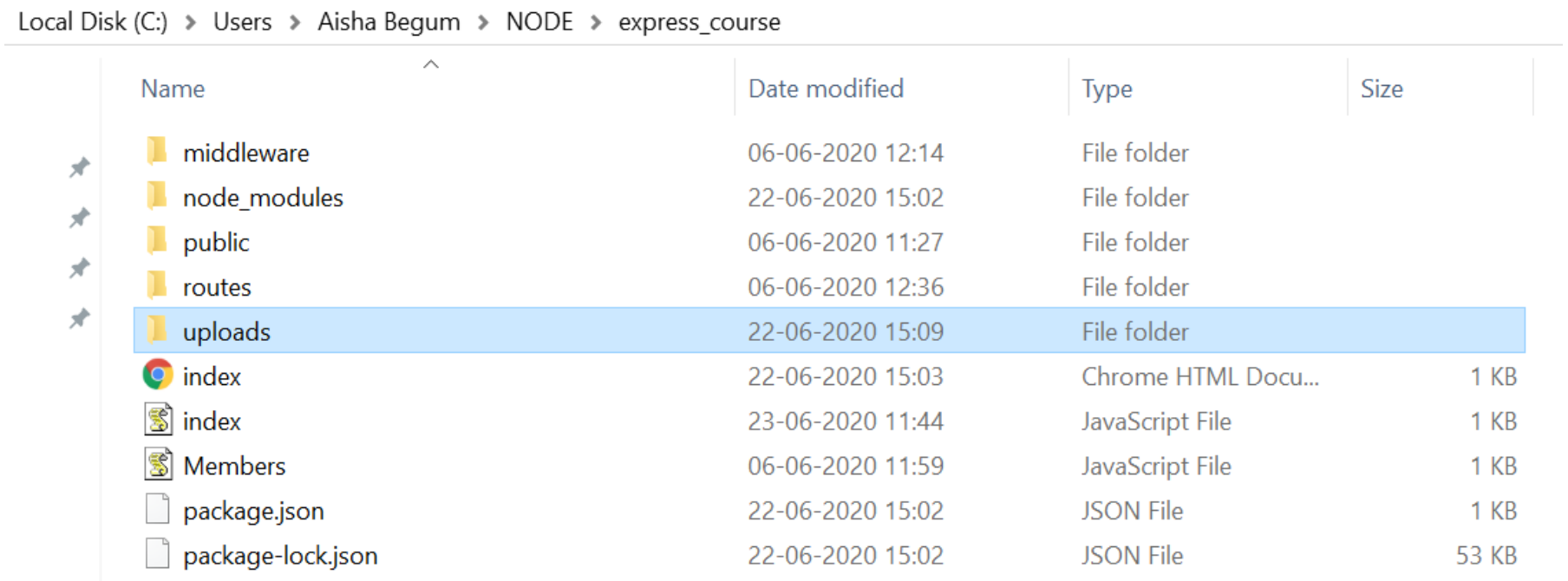
Upload a file then you can see the details in the command prompt about the file

```
C:\Users\Aisha Begum\NODE\express_course>node index
{
  file: {
    name: 'images.jpg',
    data: <Buffer ff d8 ff e0 00 10 4a 46 49 46 00 01 01 00 00 01 00 01 00 00 ff db 00 84 00 09 06 07 13 13 12 15 13 13 13 15 16 15 17 17 1d 1a 18 18 18 18 1b 1d 1d 18 ...
10523 more bytes>,
    size: 10573,
    encoding: '7bit',
    tempFilePath: '',
    truncated: false,
    mimetype: 'image/jpeg',
    md5: '0627265b42b7f89b5db7a4cb9925bd08',
    mv: [Function: mv]
  }
}
images.jpg
```

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

Create a file for upload in the root folder.
The uploaded files will be saved in the created folder



WEB TECHNOLOGIES
FILE UPLOAD

Express JS

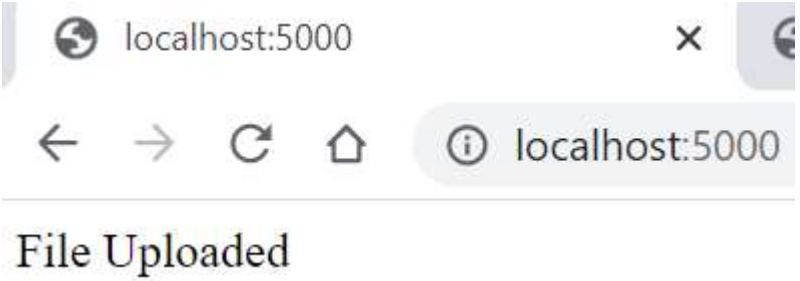
Using file.mv method we will be storing the file in the created folder

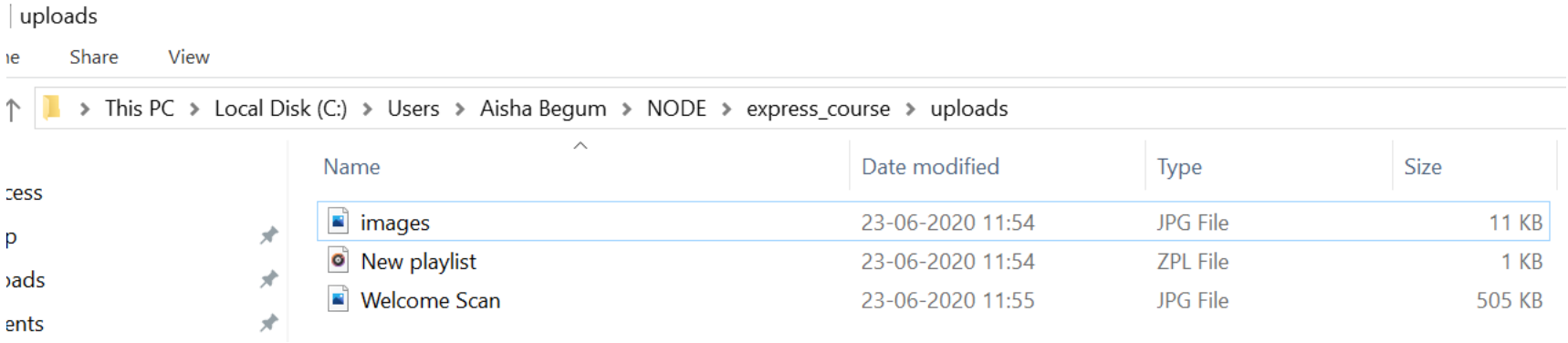
```
const express = require('express')
const upload= require('express-fileupload')

const app = express()

app.use(upload())

app.get('/', (req,res)=>{
    res.sendFile(__dirname +'/index.html' )
})

app.post('/', (req,res)=>{
    if(req.files){
        console.log(req.files);
        var file = req.files.file
        var filename = file.name
        console.log(filename)
        file.mv('./uploads/' + filename,function(err){
            if(err){
                res.send(err)
            }else {
                res.send("File Uploaded")
            }
        })
    }
});

app.listen(5000)
```

WEB TECHNOLOGIES
FILE UPLOAD

Express JS

In the browser you can see that file has been uploaded and when u check the uploads folder you can see that file has been uploaded. The file upload can be on any type





THANK YOU

Aisha Begam
Department of Computer Science Engineering

aisha.b@pes.edu

+91 9741907626