Note:- when including CSS or JS files in 'EJS' file, always use '/' before the name of file (eg: /style.css), otherwise it will throw error while you try to load same page with different. (http://localhost:8080/posts) ✓ (http://localhost:8080/posts/) ✗ Won't include CSS here.

RESTful APIs

What is REST?

REST or Representational State Transfer is an architectural style that defines a set of constraints to be used for creating web services.
REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web services. All communication done via REST API uses only HTTP request.

Two main rules to follow while creating RESTful API

*1. must Use HTTP request Verbs for CRUD operations i.e,

(Http verbs)	(operations)
GET	to get data
POST	to post data
PUT	to update existing data
PATCH	to update existing data partially
DELETE	removes data

*2. for routing of RESTful APIs, use specific nouns, not verbs.

E.g:- If there is any website that shows posts uploaded their then URL :-

↳ http://localhost:8080/showingposts ✗

↳ http://localhost:8080/posts ✓

How to make 'PUT' or 'PATCH' or 'DELETE' req HTTP request through HTML?

As we know, there are only two methods available for submitting a form in HTML i.e GET or POST, but we are supposed to use other HTTP verbs as well as. To do so, we need a middleware, 'method-override'.

Steps to add 'method-override' middleware

step-1: install this middleware
```
npm install method-override
```

step-2: include in your index.js
```
const methodOverride = require('method-override');
```

step-3: add middleware,
```
app.use(methodOverride('_method'));
```

step-4: You can use 'PUT', 'PATCH' or 'DELETE' in your index.js, (app.put, app.patch or app.delete).

step-5: Changes you have to make in your HTML form

```
<form method="POST" action="/posts?_method=PUT">
    <button type="submit">Submit</button>
</form>
```

Getting all Posts using GET Request

```
// route for getting all posts
app.get('/posts',(req,res)=>{
    res.render('index.ejs',{posts});
});
```

Getting specific Post based on Id using GET Request

```
// route to handle display specifie post based on id
app.get('/posts/:id',(req,res)=>{
    let { id } = req.params;

    let post = posts.find((p)=>{
        return (id === p.id);
    });
    res.render('show.ejs',{post});
});
```

Requesting to create a new Post using POST Request

Step-1 :- requesting for a page containing form to enter new post detail

```
// route to render form to add new post
app.get('/posts/new',(req,res)=>{
    res.render('new.ejs');
});
```

Step-2 :- route handling form data and submit new post detail and redirect to homepage (/posts)

```
// route to handle new post data
app.post('/posts',(req,res)=>{
    // console.log(req.body);
    let id = uuidv4();
    let { username , content} = req.body;
    posts.push({id, username,content});

    /* redirecting to /posts
       after updating or pushing data
    --to database or array in this case */
    res.redirect('/posts');
});
```

Requesting to Update a Post based on ID using PATCH Request

Step-1 :- Include method-override package to receive PUT, PATCH or DELETE Request

```
// including module to receive http requests like put, patch, or delete
const methodOverride = require('method-override');
```

Step-2 :- Include method-override middleware in **index.js**

```
// midleware TO OVERRIDE METHODS REQUEST
app.use(methodOverride('_method'));
```

Step-3 :- Changing form action method in **edit.ejs**

```
<form method="post" action="/posts/<%= post.id %>?_method=PATCH">
    <textarea name="content" rows="10" cols="35">
        <%= post.content %>
    </textarea>
```

Step-4 :- route handling form data and submit edit post detail and redirect to homepage (/posts)

```
// route to handle edit request
app.get('/posts/:id/edit',(req,res)=>{
    let { id } = req.params;
    let post = posts.find((p)=>{
        return (id === p.id);
    });

    res.render('edit.ejs', { post });
});
```

```
// route to update our specific post
app.patch('/posts/:id',(req,res)=>{
    let { id } = req.params;
    // console.log(id);
    let post = posts.find((p)=>{
        return (id === p.id);
    });
    // update partial data of post
    post.content = req.body.content;
    // redirect to /posts after updating data
    res.redirect('/posts');
});
```

Requesting to Update a Post based on ID using DELETE Request

Step-1 :- adding delete button and changing form action method in **index.ejs**

```
<!-- creating form to make a delete request for related post -->
<form action="/posts/<%= post.id %>?_method=DELETE" method="post">
    <button type="submit">Delete</button>
</form>
```

Step-2 :- route handling form data and submit delete post detail and redirect to homepage (/posts)

```
// ROUTE TO HANDLE DELETE REQUEST
app.delete('/posts/:id',(req,res)=>{
    let { id } = req.params;
    /* filter all elements and remove specific
       related post */
    posts = posts.filter((p)=>{
        return (id !== p.id);
    });
    // redirecting to /posts
    res.redirect('/posts');
});
```


SCAN this QR code to get full notes and source code