Ex. No: 7	Routing Implementation using ExpressJS
28.09.2023	

Aim: To Implement the routing feature(s) using the ExpressJS.

Algorithm:

1. Include the required header files thread creation and sleep() function.
2. Write a function that executes as a thread when it is called. (sleep – print - return)
3. thread_id is declared to identify the thread in the system, we call pthread_create() function to create a thread.
4. The pthread_join() function for threads is the equivalent of wait() for processes. A call to pthread_join blocks the calling thread until the thread with identifier equal to the first argument terminates.

Program:

```
const express = require('express');
const app = express();
const port = 3000;
app.use(express.json());
let notes = [];
app.use(express.static('public'));
app.use((req, res, next) => {
console.log(`Received ${req.method} request at
${req.url}`);
next();
});
app.get('/api/notes', (req, res) => {
res.json(notes);
});
app.post('/api/notes', (req, res) => {
const { title, content } = req.body;
const newNote = { id: notes.length + 1, title, content };
notes.push(newNote);
console.log(`Added a new note: "${title}"`);
res.status(201).json(newNote);
});
app.delete('/api/notes/:id', (req, res) => {
const idToDelete = parseInt(req.params.id);
notes = notes.filter(note => note.id !== idToDelete);
console.log(`Deleted note with ID: ${idToDelete}`);
res.sendStatus(204);
});
app.listen(port, () => {
console.log(`Server is running on port ${port}`);
});
<!DOCTYPE html>
```

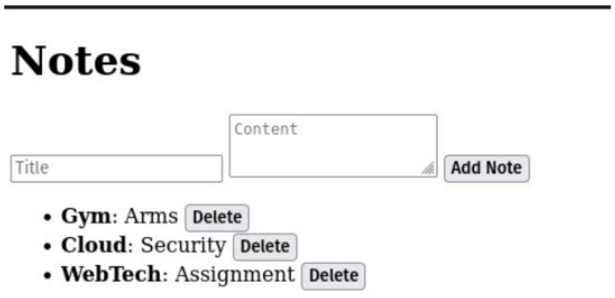
```
<html>
<head>
<title>Note Taking App</title>
</head>
<body>
<h1>Notes</h1>
<form id="noteForm">
<input type="text" id="noteTitle"
placeholder="Title" required>
<textarea id="noteContent" placeholder="Content"
required></textarea>
<button type="submit">Add Note</button>
</form>
<ul id="noteList"></ul>
<script>
const noteForm =
document.getElementById('noteForm');
const noteTitle =
document.getElementById('noteTitle');
const noteContent =
document.getElementById('noteContent');
const noteList =
document.getElementById('noteList');
noteForm.addEventListener('submit', async (e) => {
e.preventDefault();
const title = noteTitle.value;
const content = noteContent.value;
if (!title || !content) return;
try {
const response = await fetch('/api/notes',
{
method: 'POST',
headers: {
'Content-Type':
'application/json',
},
body: JSON.stringify({ title, content
}),
});
const newNote = await response.json();
noteTitle.value = '';
noteContent.value = '';
displayNote(newNote);
} catch (error) {
console.error('Error adding note:',
error);
}
});
async function fetchNotes() {
try {
```

```
const response = await
fetch('/api/notes');
const notes = await response.json();
notes.forEach(displayNote);
} catch (error) {
console.error('Error fetching notes:',
error);
}
}
function displayNote(note) {
const listItem = document.createElement('li');
listItem.innerHTML = `<strong>${note.title}
</strong>: ${note.content} <button>Delete</button>`;
const deleteButton =
listItem.querySelector('button');
deleteButton.addEventListener('click', async
() => {
try {
await fetch(`/api/notes/${note.id}`, {
method: 'DELETE' });
listItem.remove();
} catch (error) {
console.error('Error deleting note:',
error);
}
});
noteList.appendChild(listItem);
}
// Initial fetch
fetchNotes();
</script>
</body>
</html>
```

Output:

Github Link: https://github.com/AsHtrich/Web_tech2023

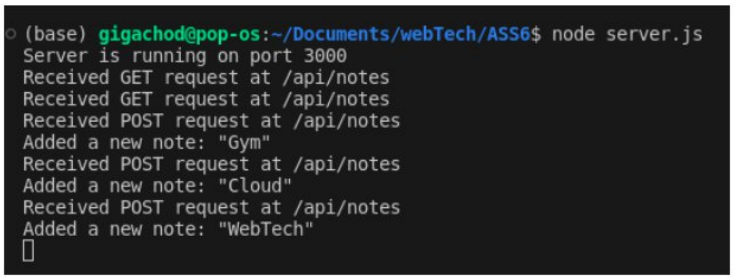
1. Initial webpage when the server is started.



Notes

- **Gym**: Arms
- **Cloud**: Security
- **WebTech**: Assignment

2. All the content for that session can be assesed in the /api/notes route in json formate



```
(base) gigachod@pop-os:~/Documents/webTech/ASS6$ node server.js
Server is running on port 3000
Received GET request at /api/notes
Received GET request at /api/notes
Received POST request at /api/notes
Added a new note: "Gym"
Received POST request at /api/notes
Added a new note: "Cloud"
Received POST request at /api/notes
Added a new note: "WebTech"
```

Result:
Therefore, we've successfully implemented a basic routing implementation using Express JS