Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	TE-AIDS	Semester:	V
Course Code:	CSC502	Course Name:	WC

Name of Student:	Divya P. Davane
Roll No. :	14
Assignment No.:	5
Title of Assignment:	NodeJS applications using express
Date of Submission:	
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	5
Demonstrated Knowledge	3	3
Legibility	2	2
Total	10	10

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge Legibility	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty :

ature : Bharat

: 6/10/25

WC Assignment 5

DATE:

Q.1 Create a simple Express application that integrates with React to display a list of items fetch from a server. Describe the steps involved and provide code snippets

Ans steps involved :- I] Setup Express Backend

1. Create a new folder express-react-app and inside it create a folder server

2. Initialize Node.js and install Express :

```
cd server
npm init -y
npm install express cors.
```

3. Create server.js inside server/:

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(cors());
const items = [
{id:1, name: "Apple"};
{id:2, name : "Banana"};
{id; 3, name: "Orange"},
];
app.get("/api/items",(req,res) => {
  res.json(items);
});
const PORT=5000.
app.listen(PORT, () => console.log ('Server running
on http://localhost:${PORT}'));
```

4. Run server

```
node server.js
```

FOR EDUCATIONAL USE

DATE:

4] Setup React Frontend

1. In the root folder, create react app :

```
npx create-react-app client
cd client
npm start
```

2. Inside client/src/App.js fetch from express Backend

```
import React, {useEffect, useState} from "react";
function App() {
const [items, setItems] = useState([]);
useEffect(() => {
fetch("http://localhost:5000/api/items")
.then((res => res.json())
.then((data) => setItems(data));
}; []);
return (
<div style={{ padding: "20px" }}>
<h1> Items List </h1>
<ul>
{items.map((item) => (
<li key = {item.id}> {item.name}</li>
))}
</ul>
</div>
);
}
export default App;
```

3. Start the React frontend with
npm start.

DATE:

Q.2. Apply Express Router to organize route handling in a Node js application by creating a modular structure. Implement a simple example to demonstrate how express Router improves code organization.

Ans I] Project setup.

```
mkdir express-router example
cd express-router example
npm init -y
npm install express
```

II] create Modular Routes

• server.js (Main entry file)

```
const express = require("express");
const app = express();
const itemRoutes = require("./routes/items");
app.use ("/api/items, itemRoutes);
const PORT = 5000;
app.listen (PORT, () ⇒ console.log ('Server running on
http://localhost:${PORT}'));
```

• routes/item.js

```
const express = require("express");
const router = express.Router();
let items =[
{id; 1, name : "Apple"},
{id: 2, name : "Banana"},
];
router.get("/", (req, res) ⇒ (
 res.json (items);
});
router.~~post~~ get ("/:id", {req, res) ⇒ {
 const items = items.find (i⇒i.id === parseInt (req.pare
.id));
```

FOR EDUCATIONAL USE

DATE

```
if (!item) return res.status(404).json({messag
"Item not found" });
res.json(item);
});
router.post ("/", express.json(), (req, res) => {
const newItem ={
id: items.length +1,
name:req.body.name,
};
items.push(newItem);
res.status(201).json(newItem);
});
module.exports = router;
```

III] Run server

```
node server.js
```

IV] Test Routes

- GET http://localhost:5000/api/items → list of items
- GET http://localhost:5000/api/items/1 → {id:1, name : "Apple"}
- POST http://localhost:5000/api/items with JSON body {"name": "Orange"} → adds new item

3 You are developing a login system in an Express Application. After a user successfully logs in, you need to store their username in a cookie that should only be accessible via HTTP (not javascript) and should expire after 1 minute. Write the express route that sets this cookie and another route that reads and displays the username from the cookie.

Ans

```
const express = require("express");
const cookieParser = require("cookie-parser");
const app = express();
app.use(express.json());
app.use(cookieParser());
app.post("/login", (req, res) => {
const {username} = req.body;
res.cookie("username", username, {
httpOnly: true,
maxAge: 60 * 1000,
});
res.send("Login successful, cookie set!");
});
app.get ("/profile", (req, res) => {
const username = req.cookies.username;
if (username) {
res.send('Welcome back, ${username}!'); } else {
res.send("No user logged in or cookie expired.");
}
});
app.listen (5000, () => console.log ("Server running on
http://localhost: 5000"));
```

DATE:

Q.7. You are building a web application where users can view a list of products. Apply your understanding of REST APIs by creating an Express route that handles an HTTP GET request at /products and returns a sample JSON array of products show the code snippet for the route

Ans

```
const express = require("express");
const app = express();
app.get ("/products", (req, res)=>{
const products = [
{id:1, name:"Laptop", price: 50000},
{id:2, name:"Phone", price:20000},
{id: 3, name:"Headphones", price: 3000},
];
res.json(products);
});
app.listen(5000, ()=> console.log("Server running on
http://localhost:5000"));
```