

Batch: A1

Roll No.: 16010123012

Experiment / assignment / tutorial No.: 4.2

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Express.js

AIM: Demonstrate the use of Express.js functionalities.

Problem Definition:

Consider the basic concepts of Express.js, which are useful in the creation of an application.

Considering the following points, demonstrate the functionality of each with a simple script

1) Scaffolding:

- Demonstrate express scaffolding to fulfill the following requirements. Example: Consider Grocery Delivery Application and demonstrate the Scaffolding, Scaffold the application to create different routes such as Sign up Page: (Root/ Homepage)

2) Serving static files using Express.js: With the help of Built in middleware, express.Static () to demonstrate the usage of serving static files in express. To demonstrate the above make

- Use of images where it should accept any type of image
- Use of CSS and HTML files.
- Make a Use json file of employee information, add file to the static folder, and show the response on the browser.

Note:

- Assume your own data whenever required to perform the operation.**

Resources used:

<https://expressjs.com/>

Expected OUTCOME of Experiment:

CO 4: Test the concepts and components of various front-end, back-end web app

Books/ Journals/ Websites referred:

- Shelly Powers Learning Node O' Reilly 2 nd Edition, 2016.

Pre Lab/ Prior Concepts:

Write details about the following content

- **Express.js** is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.
Key Features:
 1. Middleware support: Allows adding functions that process requests before sending responses.
 2. Routing system: Handles different HTTP requests based on URL paths.
 3. Template engines: Integrates with engines like EJS or Handlebars for rendering dynamic HTML pages.
 4. Static file serving: Easily serves static files such as images, CSS, and JavaScript.
 5. RESTful APIs: Simplifies the creation of APIs to interact with databases and clients.**Advantages:**
 1. Fast and lightweight.
 2. Easy to set up and use with Node.js.
 3. Highly customizable using middleware.
 4. Well-documented and supported by a large community.
- **Scaffolding** in Express.js refers to the process of automatically generating the basic structure or skeleton of an application. It helps developers quickly set up a project with a predefined folder structure and essential files, saving time and ensuring consistency.

Purpose:

1. To speed up project initialization.
 2. To maintain a standard structure across multiple projects.
 3. To help beginners start coding without manually creating files.
- **Routing** in Express.js refers to how an application's endpoints (URIs) respond to client requests. It defines how the server handles various HTTP methods (GET, POST, PUT, DELETE) and URL paths.

Types of Routing:

1. Basic Routing: Uses app-level routes as shown above.
2. Router-level Routing: Uses the express.Router() object to organize routes into separate files.
3. Dynamic Routing: Uses parameters or query strings in URLs (e.g., /user/:id).

Advantages:

1. Keeps code modular and clean.
2. Separates concerns between different parts of the application.
3. Makes it easy to handle multiple request types efficiently.

Methodology:

Implemented GET and POST endpoints following REST principles, Configured multiple static directories for different asset types

1. **Route Organization** - Separate route files for each feature. It improves code maintainability and scalability. Files are `signup.js`, `products.js`, `orders.js`, `employee.js`, `images.js`
2. **Static File Structure** - Multiple static directories (`/images`, `/css`, `/js`, `/data`). Helps to organize asset management and clear file categorization. Easy to locate and update specific asset types
3. **Image Handling** - Universal image type support. Demonstrates Express's automatic MIME type detection. `express.static()` middleware handles all formats automatically
4. **Data Serving** - JSON file in static folder rather than API endpoint. Simpler implementation demonstrating static file serving. Direct browser access to `/data/employees.json`

Implementation Details:

1. Signup.js

```
const express = require('express');
const router = express.Router();
const path = require('path');

router.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, '../public/signup.html'));
});

router.post('/signup', (req, res) => {
    const { name, email, password } = req.body;
    console.log('New User Signup:', { name, email });
    res.send(`

        <html>
            <head>
                <title>Signup Successful</title>
                <link rel="stylesheet" href="/css/style.css">
            </head>
            <body>
                <div class="container">
                    <div class="success-message">
                        <h1>Signup Successful!</h1>
                        <p>Welcome, ${name}!</p>
                        <p>Your account has been created successfully.</p>
                        <p>Email: ${email}</p>
                        <div style="margin-top: 20px;">
                            <a href="/" class="btn">Back to Home</a>
                            <a href="/products" class="btn btn-primary">Browse
Products</a>
```

```

        </div>
    </div>
</div>
</body>
</html>
`);
});

module.exports = router;

```

Fresh Grocery Delivery

Home Products Orders Employees Images Demo

Welcome to Fresh Grocery Delivery

Get fresh groceries delivered to your doorstep!

Sign Up for Free Delivery

Full Name:

Email Address:

Password:

Sign Up Now

© 2025 Fresh Grocery Delivery. All rights reserved.

Signup Successful!

Welcome, Aaryan !
Your account has been created successfully.
Email: test@gmail.com

[Back to Home](#) [Browse Products](#)

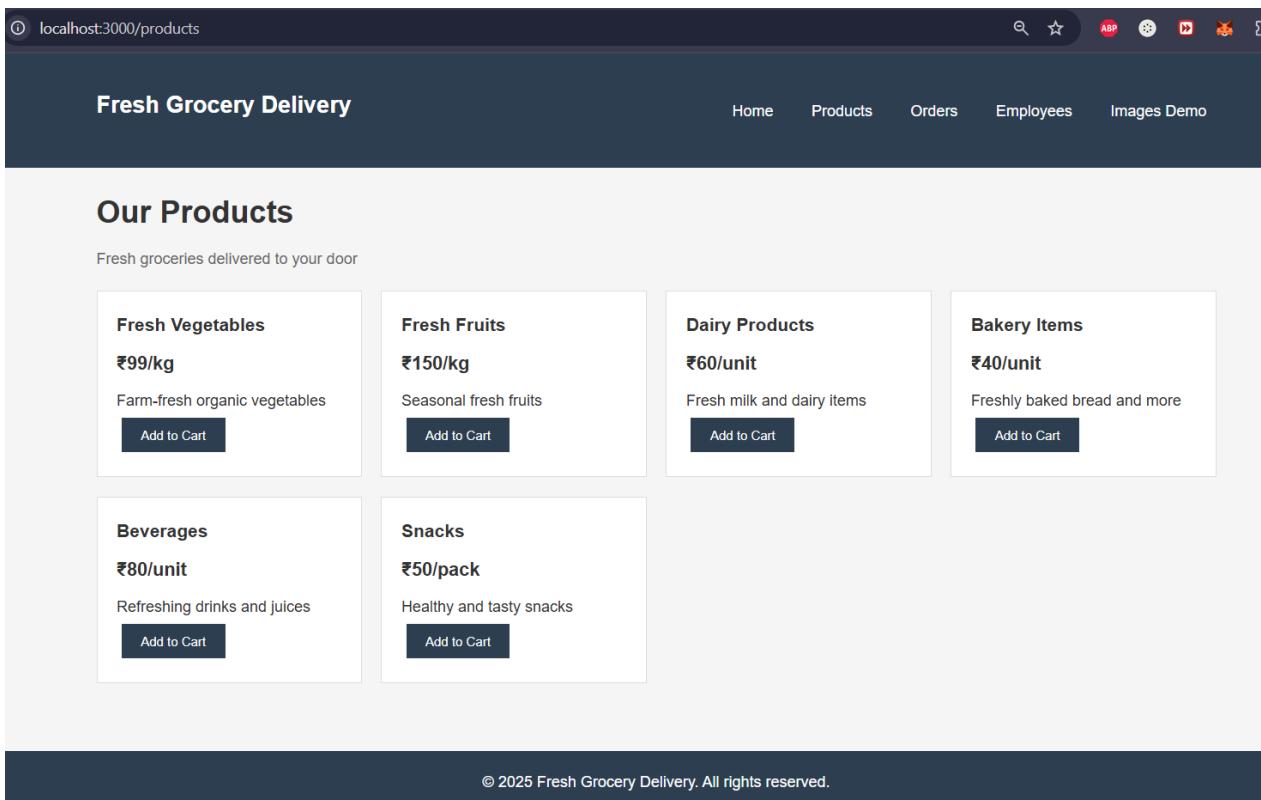
2. Products.js

```
const express = require('express');
const router = express.Router();
const path = require('path');

router.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, '../public/products.html'));
});

router.get('/:id', (req, res) => {
    const productId = req.params.id;
    res.send(`
        <html>
            <head>
                <title>Product Details</title>
                <link rel="stylesheet" href="/css/style.css">
            </head>
            <body>
                <div class="container">
                    <h1>Product Details - ID: ${productId}</h1>
                    <p>This is a detailed view of product ${productId}</p>
                    <a href="/products">Back to Products</a>
                </div>
            </body>
        </html>
    `);
});

module.exports = router;
```



The screenshot shows a web application for "Fresh Grocery Delivery". At the top, there's a navigation bar with links for Home, Products, Orders, Employees, and Images Demo. Below the navigation is a section titled "Our Products" with a subtitle "Fresh groceries delivered to your door". There are six product categories displayed in boxes:

- Fresh Vegetables**: ₹99/kg. Farm-fresh organic vegetables. **Add to Cart**.
- Fresh Fruits**: ₹150/kg. Seasonal fresh fruits. **Add to Cart**.
- Dairy Products**: ₹60/unit. Fresh milk and dairy items. **Add to Cart**.
- Bakery Items**: ₹40/unit. Freshly baked bread and more. **Add to Cart**.
- Beverages**: ₹80/unit. Refreshing drinks and juices. **Add to Cart**.
- Snacks**: ₹50/pack. Healthy and tasty snacks. **Add to Cart**.

At the bottom of the page, a dark footer bar contains the text "© 2025 Fresh Grocery Delivery. All rights reserved."

3. Orders.js

```
const express = require('express');
const router = express.Router();
const path = require('path');

router.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/orders.html'));
});

router.post('/create', (req, res) => {
  const { items, totalAmount } = req.body;
  console.log('New Order Created:', { items, totalAmount });

  res.json({
    success: true,
    message: 'Order placed successfully!',
    orderId: Math.floor(Math.random() * 10000)
  });
});

module.exports = router;
```

localhost:3000/orders

Fresh Grocery Delivery

Home Products Orders Employees Images Demo

My Orders

Track your grocery orders

Order #1234	Delivered
Date: October 10, 2025	
Items: Fresh Vegetables, Fruits, Dairy	
Total: ₹450	
Order #1235	Processing
Date: October 12, 2025	
Items: Bakery, Beverages, Snacks	
Total: ₹320	
Order #1236	Pending
Date: October 13, 2025	
Items: Fresh Vegetables, Dairy Products	
Total: ₹280	

Order More Groceries

4. Employee.js

```
const express = require('express');
const router = express.Router();
const path = require('path');

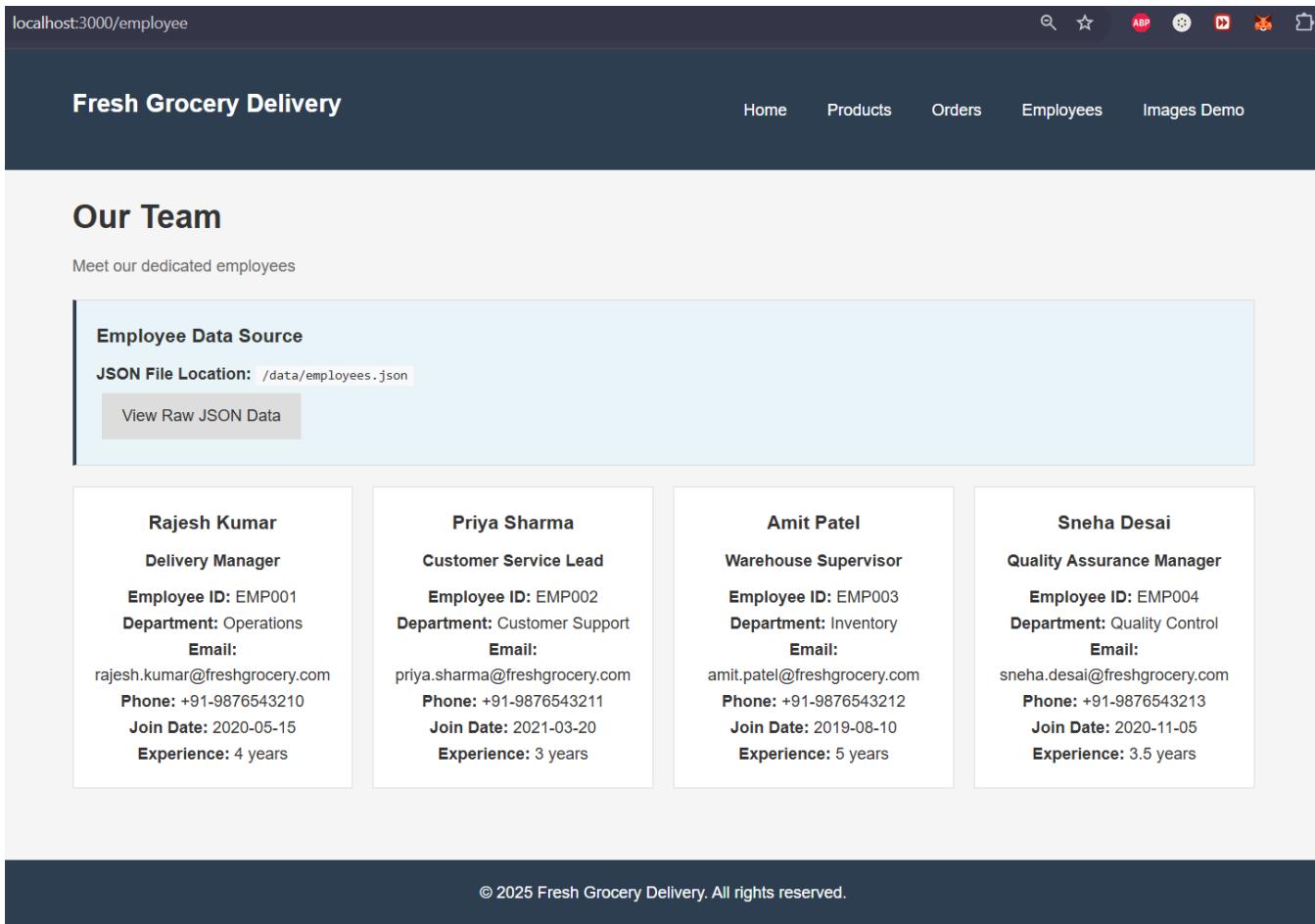
router.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, '../public/employee.html'));
});

module.exports = router;
```

localhost:3000/data/employees.json

```
{
  "employees": [
    {"id": "EMP001", "name": "Rajesh Kumar", "role": "Delivery Manager", "department": "Operations", "email": "rajesh.kumar@freshgrocery.com", "phone": "+91-9876543210", "joinDate": "2020-05-15", "experience": "4 years"}, 
    {"id": "EMP002", "name": "Priya Sharma", "role": "Customer Service Lead", "department": "Customer Support", "email": "priya.sharma@freshgrocery.com", "phone": "+91-9876543211", "joinDate": "2021-03-20", "experience": "3 years"}, 
    {"id": "EMP003", "name": "Amit Patel", "role": "Warehouse Supervisor", "department": "Inventory", "email": "amit.patel@freshgrocery.com", "phone": "+91-9876543212", "joinDate": "2019-08-10", "experience": "5 years"}, 
    {"id": "EMP004", "name": "Desai", "role": "Quality Assurance Manager", "department": "Quality Control", "email": "sneha.desai@freshgrocery.com", "phone": "+91-9876543213", "joinDate": "2020-11-05", "experience": "3.5 years"}], 
    "totalEmployees": 4, 
    "departments": [
      {"name": "Operations", "Customer Support", "Inventory", "Quality Control"}, 
      "companyInfo": {"name": "Fresh Grocery Delivery", "founded": "2019", "headquarters": "Mumbai, India", "totalStaff": 150}
    ]
}
```

localhost:3000/employee



Fresh Grocery Delivery

Home Products Orders Employees Images Demo

Our Team

Meet our dedicated employees

Employee Data Source

JSON File Location: /data/employees.json

[View Raw JSON Data](#)

Rajesh Kumar
Delivery Manager
Employee ID: EMP001
Department: Operations
Email: rajesh.kumar@freshgrocery.com
Phone: +91-9876543210
Join Date: 2020-05-15
Experience: 4 years

Priya Sharma
Customer Service Lead
Employee ID: EMP002
Department: Customer Support
Email: priya.sharma@freshgrocery.com
Phone: +91-9876543211
Join Date: 2021-03-20
Experience: 3 years

Amit Patel
Warehouse Supervisor
Employee ID: EMP003
Department: Inventory
Email: amit.patel@freshgrocery.com
Phone: +91-9876543212
Join Date: 2019-08-10
Experience: 5 years

Sneha Desai
Quality Assurance Manager
Employee ID: EMP004
Department: Quality Control
Email: sneha.desai@freshgrocery.com
Phone: +91-9876543213
Join Date: 2020-11-05
Experience: 3.5 years

© 2025 Fresh Grocery Delivery. All rights reserved.

5. Images.js

```
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  res.send(` 
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Serving Demo - All Types</title>
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <div class="navbar">
    <div class="container">
```

```

<h2>Fresh Grocery Delivery</h2>
<nav>
    <a href="/">Home</a>
    <a href="/products">Products</a>
    <a href="/orders">Orders</a>
    <a href="/employee">Employees</a>
    <a href="/images-demo">Images Demo</a>
</nav>
</div>
</div>

<div class="container">
    <h1>Image Serving Demonstration</h1>
    <p class="subtitle">Express.js can serve ANY type of image format</p>

    <h2>Example Image Formats</h2>
    <div class="image-demo-grid">
        <div class="demo-card">
            <h3>JPG / JPEG</h3>
            <p><strong>Extension:</strong> .jpg, .jpeg</p>
            <p><strong>Access URL:</strong></p>
            <code>/images/sample.jpg</code>
        </div>

        <div class="demo-card">
            <h3>PNG</h3>
            <p><strong>Extension:</strong> .png</p>
            <p><strong>Access URL:</strong></p>
            <code>/images/sample.png</code>
        </div>
    </div>
</div>

<footer>
    <p>&copy; 2025 Fresh Grocery Delivery. All rights reserved.</p>
</footer>
</body>
</html>
`);
});

module.exports = router;

```

localhost:3000/images-demo

Fresh Grocery Delivery

Home Products Orders Employees Images Demo

Image Serving Demonstration

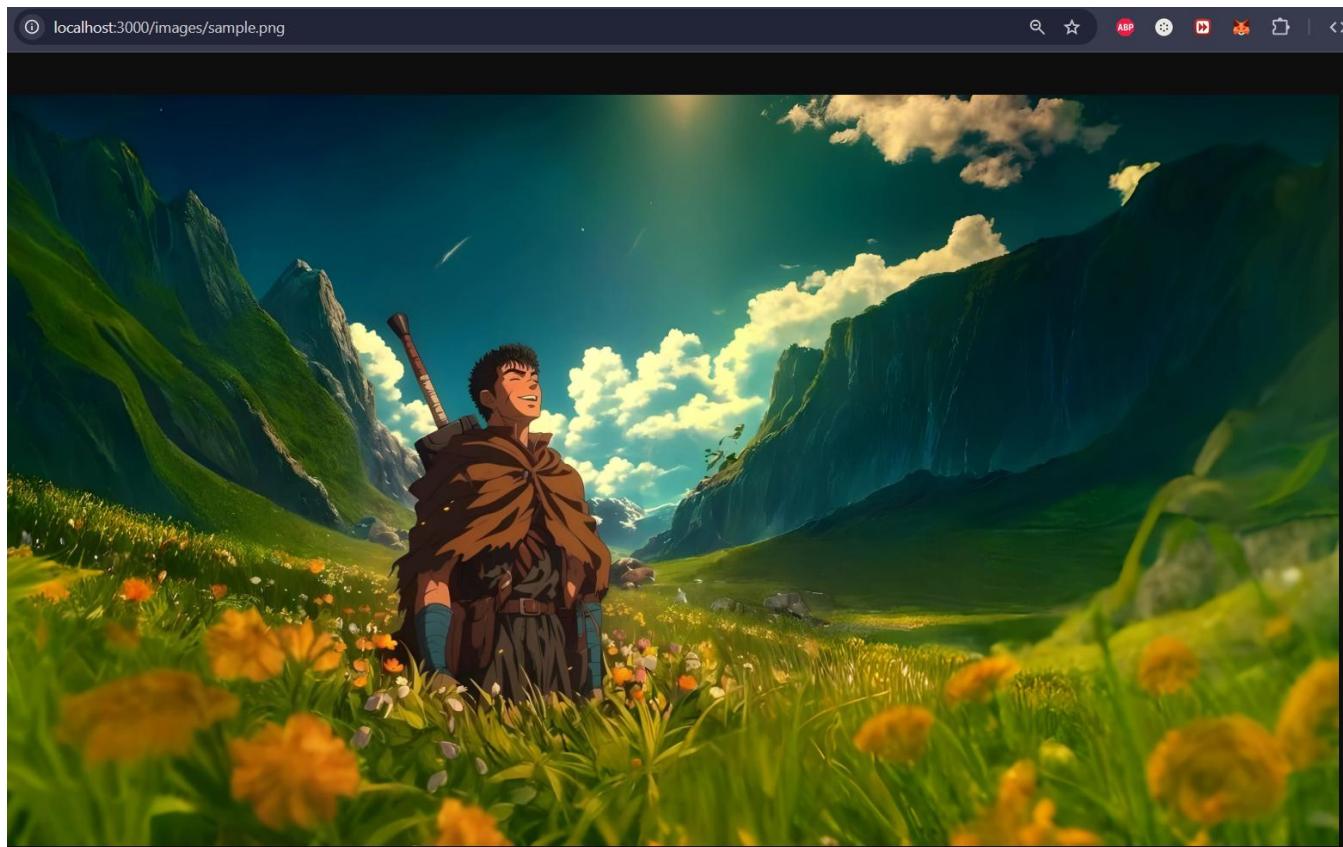
Express.js can serve ANY type of image format

Example Image Formats

JPG / JPEG
Extension: .jpg, .jpeg
Access URL:
`/images/sample.jpg`

PNG
Extension: .png
Access URL:
`/images/sample.png`

© 2025 Fresh Grocery Delivery. All rights reserved.



```
PS D:\KJSCE\BTech\TY\Sem V\MERN\Code\express> npm run dev

> grocery-delivery-app@1.0.0 dev
> nodemon app.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Grocery Delivery App Server Started
Server running at: http://localhost:3000
Homepage: http://localhost:3000/
Products: http://localhost:3000/products
Orders: http://localhost:3000/orders
Employees: http://localhost:3000/employee
Images Demo: http://localhost:3000/images-demo
JSON Data: http://localhost:3000/data/employees.json
New User Signup: { name: 'Aaryan ', email: 'test@gmail.com', phone: undefined }
```

Steps for execution:

1. Open the terminal and navigate to the project folder.
2. Run npm install to install dependencies.
3. Start the server using:
4. node app.js
5. Open a browser and visit:
 - o http://localhost:3000/ - Homepage
 - o http://localhost:3000/signup - Signup Page
 - o http://localhost:3000/products - Products
 - o http://localhost:3000/orders - Orders
 - o http://localhost:3000/images-demo - Images Demo
 - o http://localhost:3000/employees - JSON data

Conclusion:

I have successfully completed the experiment on Express.js implementation. Through this experiment, I learned how to set up an Express.js application, scaffold routes, and serve static files such as images, CSS, HTML, and JSON. I also understood the role of middleware and routing in building modular web applications. This experiment helped me gain hands-on experience with Node.js and Express.js, which are essential for backend web development.