Batch: D-2	**Roll No.: 16010123325**
Experiment / assignment / tutorial No._4.2___	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Demonstrate the Use of node.js and Express.js

AIM: To implement the node js Concepts based on the following topics.

Problem Definition:

Problem statement:

1) Scaffolding:

1. 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 a use of

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.

Resources used:
https://www.geeksforgeeks.org/web-tech/scaffolding-expressjs-app-scratch/
https://expressjs.com/en/starter/static-files.html

Expected OUTCOME of Experiment:

CO 2:. Illustrate the concepts of various front-end, back-end web application development technologies & frameworks using different web development tools.

Books/ Journals/ Websites referred:

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

Pre Lab/ Prior Concepts:

Write details about the following content

1) Scaffolding

- **Definition:** Scaffolding in web development refers to the automatic generation of a basic project structure or boilerplate code to help you start quickly.
- **Purpose:** It saves time, enforces best practices, and sets up folders, configuration files, and sometimes default routes.
- **In Express.js:**
  - express-generator is commonly used.
  - Command:
  - npx express-generator myApp

    This creates a folder structure like:

    myApp/
    ├── bin/
    ├── public/
    ├── routes/
    ├── views/
    ├── app.js
    └── package.json

  - It gives you a ready-to-run Express app with middleware, routing, and view engine configured.

2) Serving Static Files in Express.js

- **Definition:** Static files are files that are served "as-is" to the client, without any server-side processing. Examples: images, CSS, JavaScript, HTML files.
- **Purpose:** Allows the browser to load stylesheets, scripts, images, etc., efficiently.
- **Implementation in Express:**

```
const express = require('express');
const app = express();
app.use(express.static('public')); // 'public' folder contains static files
app.listen(3000, () => console.log('Server running on port 3000'));
```

- **How it works:**
  - If a user requests /style.css, Express looks in public/style.css and serves it directly.
  - Static files are **served**, not executed, meaning their content is sent as-is to the browser.

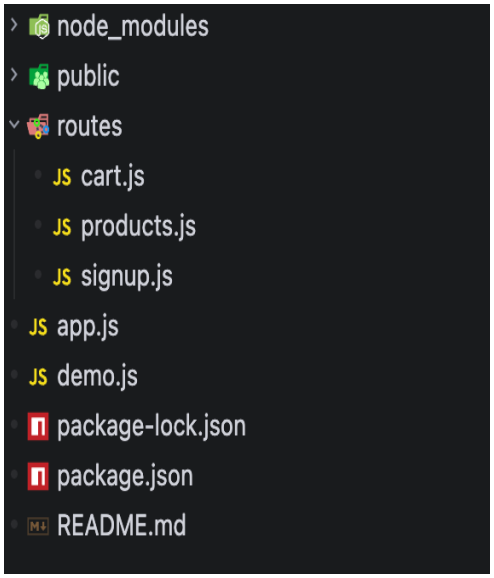
1) Scaffolding

Methodology:

- Initialize the project with npm init -y.
- Install required dependencies:
  - npm install express
- Create folders:
  - public/ → for static assets (HTML, CSS, images, JSON)
  - routes/ → for different route handlers
- Create route files:
  - signup.js → for homepage and signup requests
  - products.js → for product-related API endpoints
  - cart.js → for cart-related operations
- Main entry file: app.js integrates all routes using Express middleware.
- Add demo.js for demonstrating/test-running all API endpoints.

Implementation Details:

- The **Signup Route (routes/signup.js)** handles homepage and user registration requests.
- The **Products Route (routes/products.js)** manages product listing and filtering by category.
- The **Cart Route (routes/cart.js)** allows adding items and retrieving a user's cart.
- The **app.js** serves as the core server, mounting all routes and enabling middleware.



```
// Import routes
const signupRoutes = require('./routes/signup');
const productRoutes = require('./routes/products');
const cartRoutes = require('./routes/cart');

// Use routes
app.use('/', signupRoutes);
app.use('/api/products', productRoutes);
app.use('/api/cart', cartRoutes);

// Error handling middleware
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send('Something went wrong!');
});

// 404 handler
app.use((req, res) => {
    res.status(404).send('Page not found!');
});

// Start the server
app.listen(PORT, () => {
    console.log(`🚀 Grocery Delivery App is running on http://localhost:${PORT}`);
    console.log(`📝 Sign up page available at: http://localhost:${PORT}`);
    console.log(`📊 Employee data available at: http://localhost:${PORT}/employees.json`);
    console.log(`🛒 Products API available at: http://localhost:${PORT}/api/products`);
    console.log(`👜 Cart API available at: http://localhost:${PORT}/api/cart`);
    console.log(`🖼️ Sample images available at: http://localhost:${PORT}/images/`);
});
```

Steps for execution:

- Open terminal and navigate to the project directory.
- Run the server:
  - node app.js
- Visit:
  - Home/Signup page → http://localhost:3000/
  - Products API → http://localhost:3000/api/products
  - Cart API → http://localhost:3000/api/cart

2) Serving static files using Express.js

Methodology:

- Create a public/ folder to hold all static files.
  - Include:
    - index.html → homepage layout
    - style.css → linked stylesheet
    - /images/ → folder with various images
    - employees.json → static JSON file containing employee details
- Use Express static middleware in app.js:
  - app.use(express.static(path.join(__dirname, 'public')));

This allows Express to automatically serve any file from the public directory.

Implementation Details:

- Any file in the public directory can now be accessed directly via browser:
  - index.html → http://localhost:3000/index.html
  - style.css → http://localhost:3000/style.css
  - Image files (any type) → http://localhost:3000/images/<filename>.jpg/png/svg
  - Employee data → http://localhost:3000/employees.json
- employees.json contains employee data that is displayed directly as a browser JSON response.

```
// Serve static files from the 'public' directory
app.use(express.static(path.join(__dirname, 'public')));
```

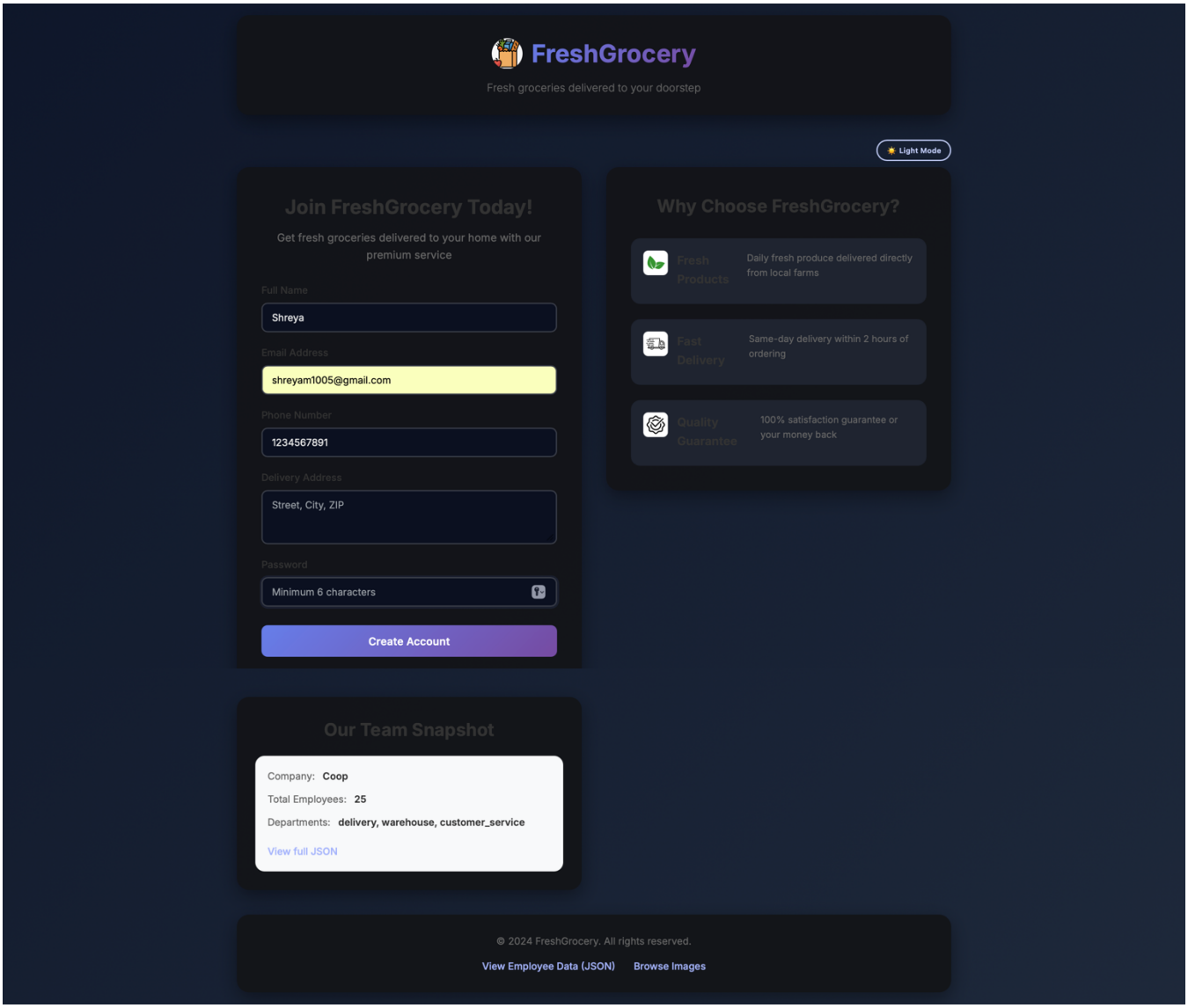
```
app.listen(PORT, () => {
    console.log(`🚀 Grocery Delivery App is running on http://localhost:${PORT}`);
    console.log(`📱 Sign up page available at: http://localhost:${PORT}`);
    console.log(`📊 Employee data available at: http://localhost:${PORT}/employees.json`);
    console.log(`🛒 Products API available at: http://localhost:${PORT}/api/products`);
    console.log(`🛍️  Cart API available at: http://localhost:${PORT}/api/cart`);
    console.log(`🖼️  Sample images available at: http://localhost:${PORT}/images/`);
});
```

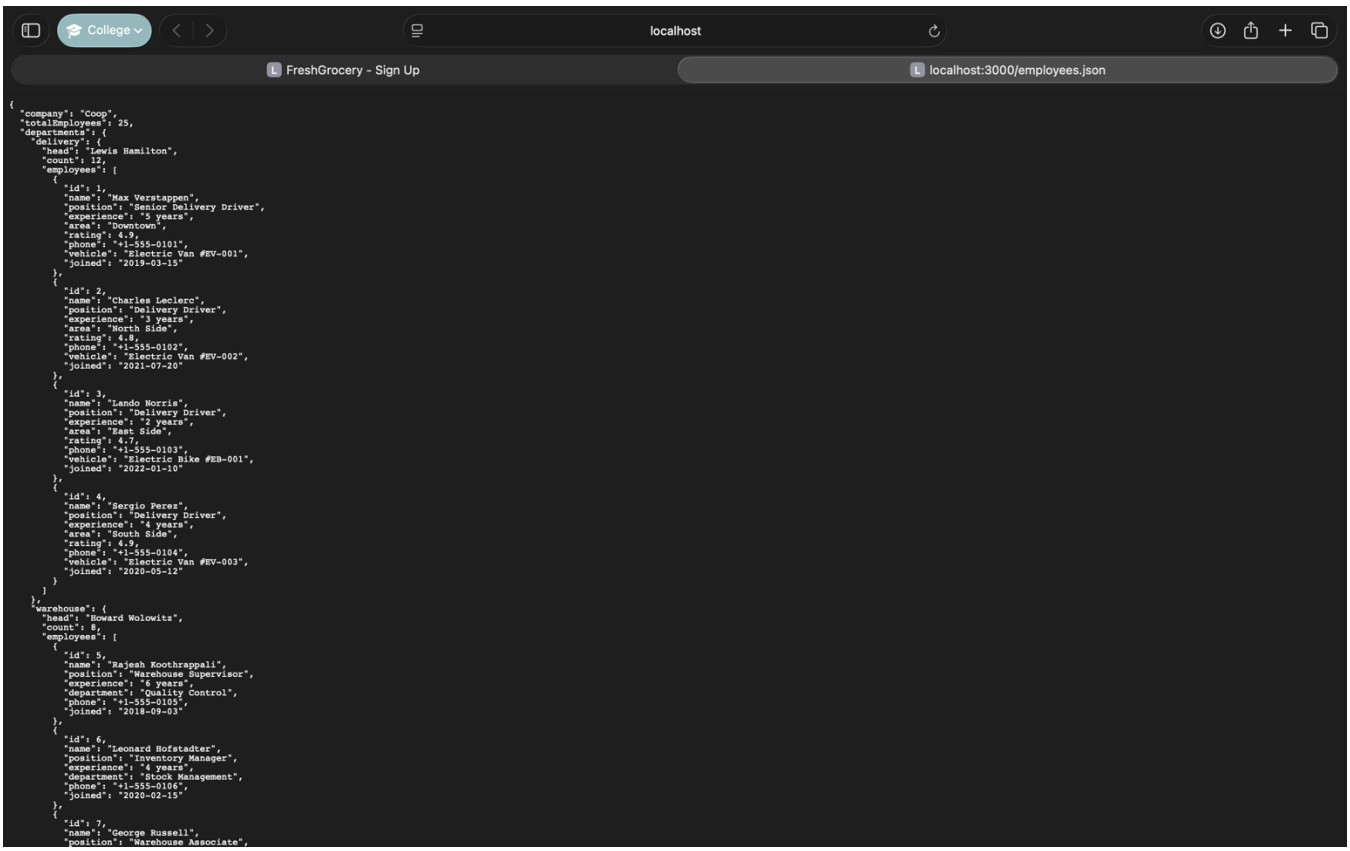
Steps for execution:

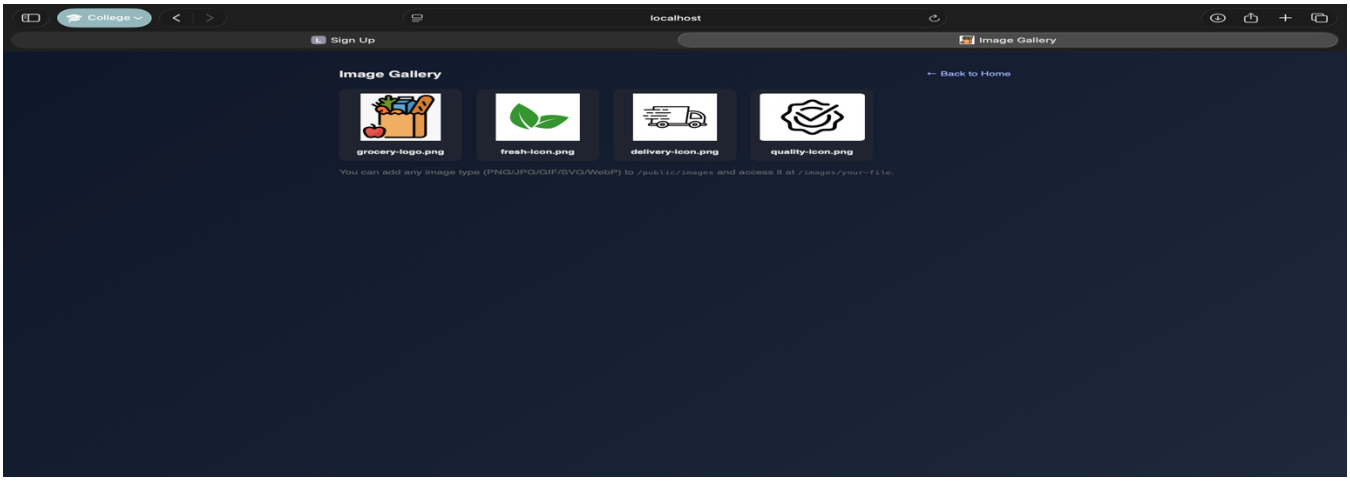
- Start the server using node app.js.
- Open browser and check:
  - http://localhost:3000/index.html → loads HTML page with CSS styling.

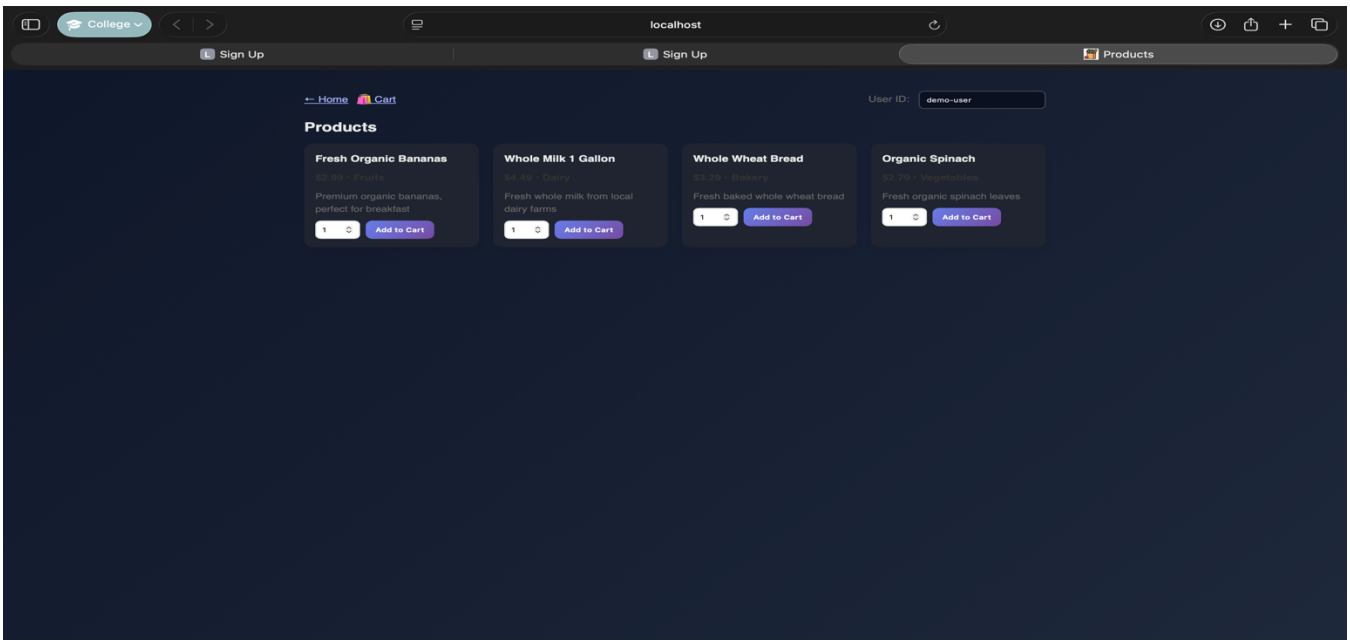
- http://localhost:3000/images/sample.png → loads static image.
- http://localhost:3000/employees.json → displays JSON employee information.

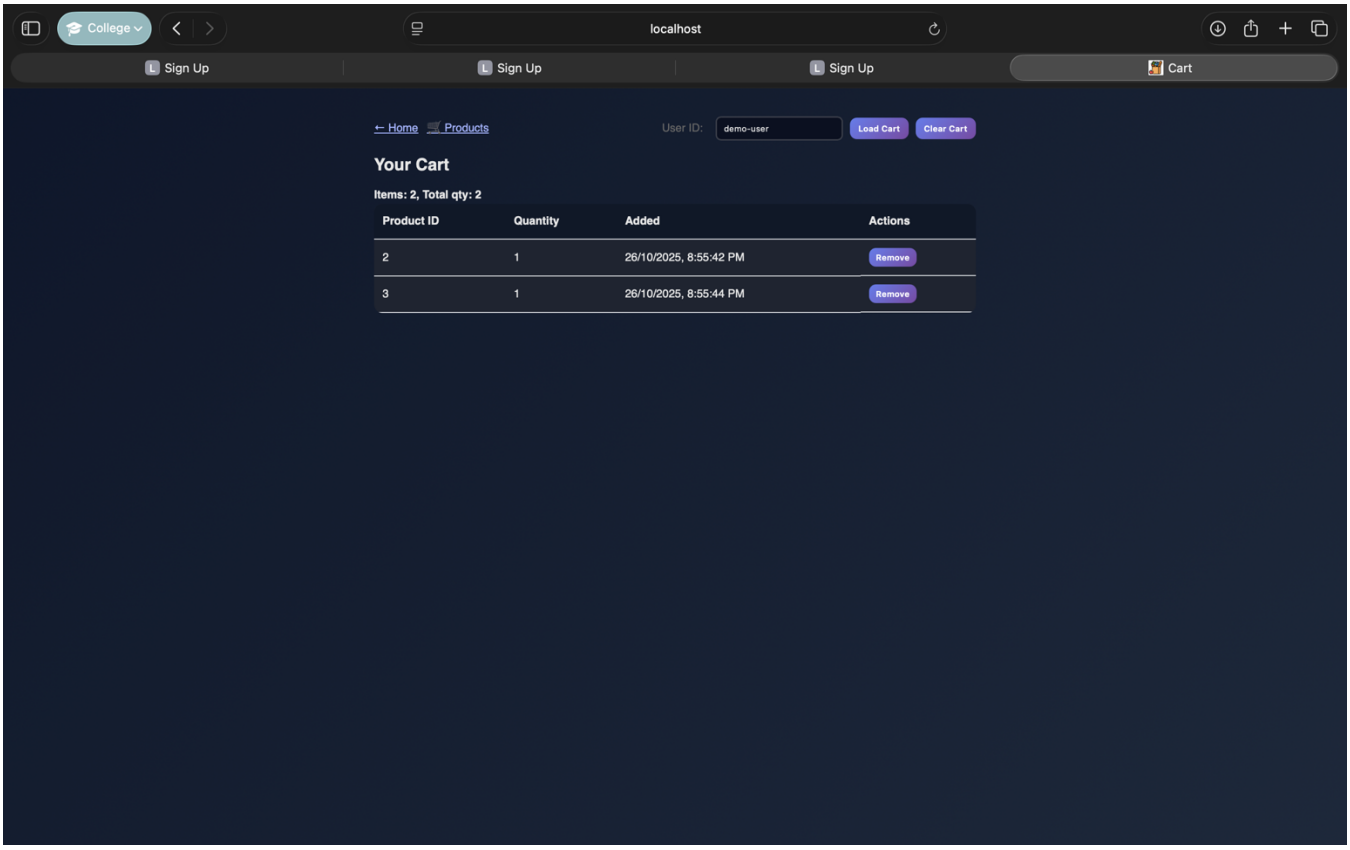
Outputs-




College
localhost
FreshGrocery - Sign Up
localhost:3000/employees.json
{
"company": "Coop",
"totalEmployees": 25,
"departments": {
"delivery": {
"head": "Lewis Hamilton",
"count": 12,
"employees": [
{
"id": 1,
"name": "Max Verstappen",
"position": "Senior Delivery Driver",
"experience": "3 years",
"area": "Downtown",
"rating": 4.9,
"phone": "+1-555-0101",
"vehicle": "Electric Van #EV-001",
"joined": "2019-03-15"
},
{
"id": 2,
"name": "Charles Leclerc",
"position": "Delivery Driver",
"experience": "3 years",
"area": "North Side",
"rating": 4.8,
"phone": "+1-555-0102",
"vehicle": "Electric Van #EV-002",
"joined": "2021-07-20"
},
{
"id": 3,
"name": "Lando Norris",
"position": "Delivery Driver",
"experience": "2 years",
"area": "East Side",
"rating": 4.7,
"phone": "+1-555-0103",
"vehicle": "Electric Bike #EB-001",
"joined": "2022-01-10"
},
{
"id": 4,
"name": "Sergio Perez",
"position": "Delivery Driver",
"experience": "4 years",
"area": "South Side",
"rating": 4.9,
"phone": "+1-555-0104",
"vehicle": "Electric Van #EV-003",
"joined": "2020-05-12"
}
]
},
"warehouse": {
"head": "Howard Wolowitz",
"count": 8,
"employees": [
{
"id": 5,
"name": "Rajesh Koothrappali",
"position": "Warehouse Supervisor",
"experience": "6 years",
"department": "Quality Control",
"phone": "+1-555-0105",
"joined": "2018-09-03"
},
{
"id": 6,
"name": "Leonard Hofstadter",
"position": "Inventory Manager",
"experience": "4 years",
"department": "Stock Management",
"phone": "+1-555-0106",
"joined": "2020-02-15"
},
{
"id": 7,
"name": "George Russell",
"position": "Warehouse Associate",


College
localhost
Sign Up
Image Gallery
Image Gallery
← Back to Home
grocery-logo.png
fresh-icon.png
delivery-icon.png
quality-icon.png
You can add any image type (PNG/JPG/GIF/SVG/WebP) to /public/images and access it at /images/your-file.


College
localhost
Sign Up
Sign Up
Products
← Home
Cart
User ID:
demo-user
Products
Fresh Organic Bananas
Premium organic bananas, perfect for breakfast
Add to Cart
Whole Milk 1 Gallon
Fresh whole milk from local dairy farms
Add to Cart
Whole Wheat Bread
Fresh baked whole wheat bread
Add to Cart
Organic Spinach
Fresh organic spinach leaves
Add to Cart


College
localhost
Sign Up
Sign Up
Sign Up
Cart
← Home
Products
User ID:
demo-user
Load Cart
Clear Cart
Your Cart
Items: 2, Total qty: 2
Product ID
Quantity
Added
Actions
2
1
26/10/2025, 8:55:42 PM
Remove
3
1
26/10/2025, 8:55:44 PM
Remove

Conclusion:

This project successfully demonstrates:

- Express **Scaffolding** for a modular application.
- Use of **express.static()** middleware to serve HTML, CSS, image, and JSON files.
- **Route segregation** for clean and maintainable code.