

| 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:**

          1. Demonstrate express scaffolding to fulfil 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.

**Note:**

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

**Resources used:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

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

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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**

* **Express js**

Express.js is a fast, minimalistic, and flexible web application framework for Node.js, designed for building web applications and APIs. It simplifies tasks like routing, handling HTTP requests and responses, and serving static files. Express provides a robust set of features for building single-page, multi-page, and hybrid web applications. Its lightweight nature makes it an essential part of the MERN stack.

* **Scaffolding**

Scaffolding in Express refers to quickly generating the basic structure of an application. Express provides a command-line tool called express-generator that sets up the directories and essential files required for the application, such as routes, views, and public assets. Scaffolding accelerates development by giving developers a standard folder structure, helping them focus on the business logic rather than the setup.

* **Express Routing**

Routing is the process of determining how an application responds to client requests made to specific paths (or endpoints). Express allows developers to define routes for different HTTP methods (GET, POST, etc.) and associate them with URL patterns. The routes can respond with HTML, JSON, files, or any other content. Routing helps in creating different pages and functionalities for an Express application.

**Methodology:**

1. **Install Express and Set Up Project:**

* Use npm init to initialize the project and npm install express to install Express.
* Optionally, use express-generator to scaffold the project.

1. **Create Basic Folder Structure:**

* Use express-generator to generate a default folder structure including routes/, public/, and views/.
* Alternatively, create folders manually for static files and routes.

1. **Set Up Static Files:**

* Use express.static() middleware to serve static assets like HTML, CSS, images, and JavaScript.

1. **Define Routes:**

* Create routes in the routes/ folder for the home page, sign-up page, and other endpoints.
* Use HTTP methods like GET, POST, PUT, etc., in the routes to define functionality.

1. **Link Static Content to Routes:**

* Serve HTML files through routes using res.sendFile() or render dynamic content using view engines like Pug or EJS.

1. **Test the Application:**

* Run the app using npm start or a custom script.
* Verify that routes, static content, and functionality work as expected.

**Implementation Details:**

**Code Snippets:**

**app.js**

var express = require('express');

var path = require('path');

var indexRouter = require('./routes/index');

var app = express();

// Serve static files from 'public' directory

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', indexRouter);

module.exports = app;

**index.js**

var express = require('express');

var router = express.Router();

var path = require('path');

// Homepage route

router.get('/', function(req, res, next) {

// Serve static HTML file for the homepage

res.sendFile(path.join(\_\_dirname, '../public/html/index.html'));

});

// Sign up route

router.get('/signup', function(req, res, next) {

// Serve static HTML file for the sign-up page

res.sendFile(path.join(\_\_dirname, '../public/html/signup.html'));

});

module.exports = router;

**users.js**

var express = require('express');

var router = express.Router();

/\* GET users listing. \*/

router.get('/', function(req, res, next) {

res.send('respond with a resource');

});

module.exports = router;

**Stepwise Details:**

1. **Install Express.js**: Initialize your project and install Express using the following commands:

npm init -y

npm install express

1. **Scaffold the Project**:
   * If using express-generator, scaffold the project using:

npx express-generator

* + This will create a project structure like:

routes/

public/

views/

app.js

1. **Define Routes**:
   * In routes/index.js, define basic routes:

var express = require('express');

var router = express.Router();

router.get('/', function(req, res) {

res.send('Homepage');

});

router.get('/signup', function(req, res) {

res.send('Signup Page');

});

module.exports = router;

1. **Serve Static Files**:
   * Create static assets like logo.png, style.css inside the public/ directory.
   * Serve these files using app.use(express.static('public'));.
2. **Connect Everything**: Ensure the app.js file is properly configured to use routes and serve static files:

var express = require('express');

var path = require('path');

var indexRouter = require('./routes/index');

var app = express();

app.use(express.static(path.join(\_\_dirname, 'public')));

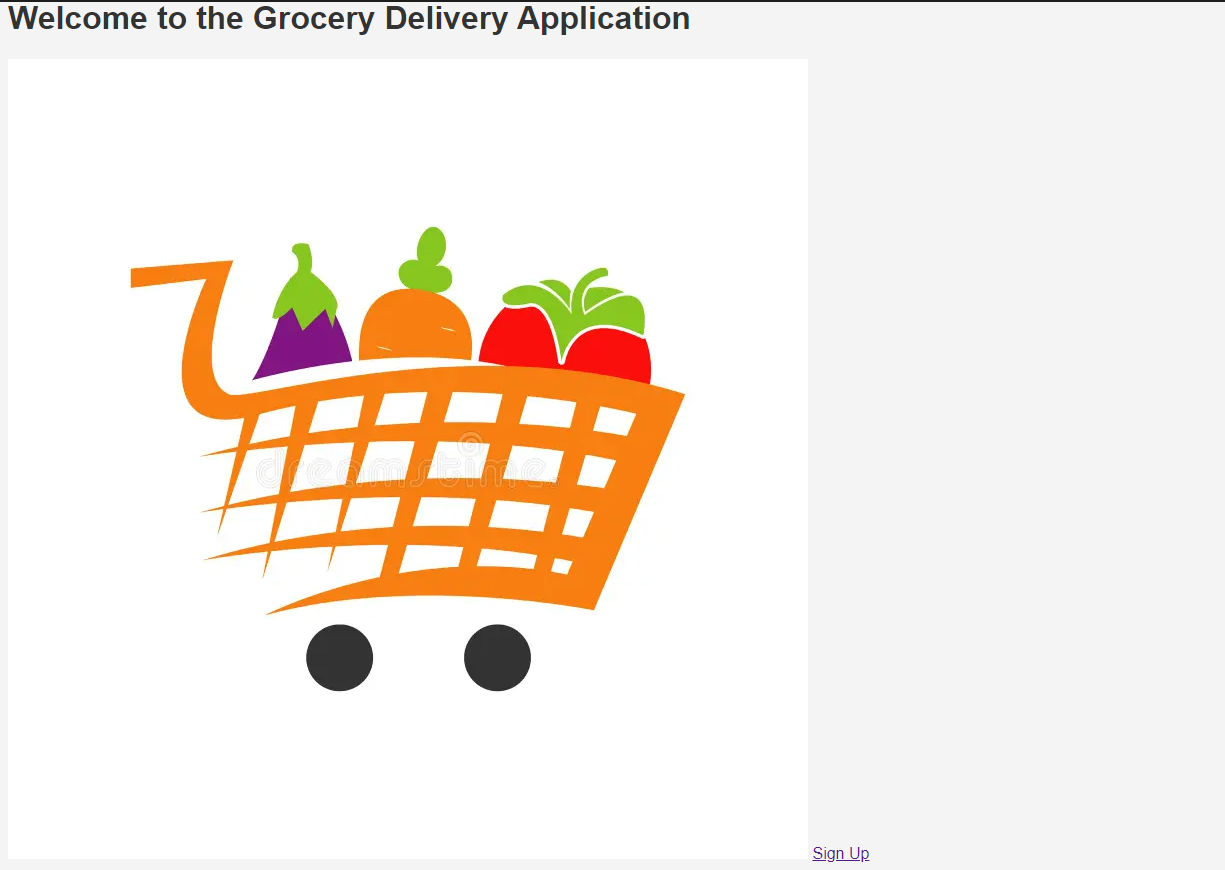
app.use('/', indexRouter);

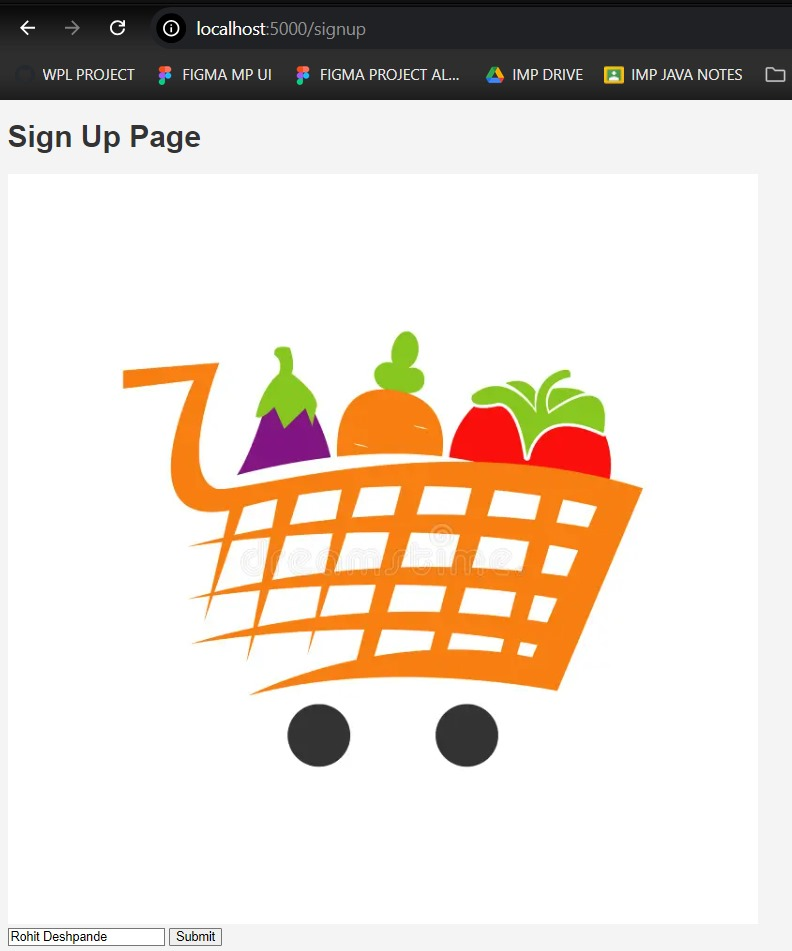
module.exports = app;

1. **Run the Application**:
   * Start the server:

npm start

**SCREENSHOTS:**

****

****

**Steps for execution:**

1. Install Node.js and Express dependencies.
2. Scaffold or manually set up the project structure.
3. Define routes for the homepage and sign-up page in routes/index.js.
4. Add static files like HTML, CSS, and images to the public/ folder.
5. Start the application using npm start.
6. Access the routes via http://localhost:5000/ and test functionality.
7. Check for correct display of the HTML, CSS, and images.

**Conclusion:** Learnt and implemented Express.js