Ex No: 5
Date:

Todo Application Using AngularJS with JSON File Storage via Node.js Server

AIM:

To create a Todo application using AngularJS, store the data in a JSON file using a simple Node.js server, and retrieve the data during page reloads.

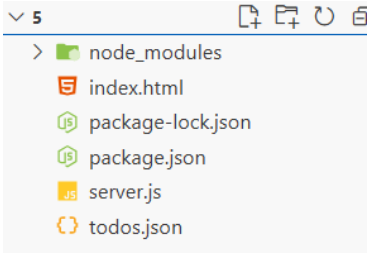
ALGORITHM:

Step 1: Start by importing the required modules: express, express-handlebars, fs, and path.
Step 2: Initialize the Express application using express() and assign a port number (e.g., 3000).
Step 3: Set up middleware using express.urlencoded({ extended: true }) to handle form data from req.body.
Step 4: Use express.json() middleware to support JSON data parsing.
Step 5: Configure the Handlebars view engine using app.engine() and app.set() to define the rendering engine and views folder.
Step 6: Create a views/ folder and add a Handlebars file (e.g., form.handlebars) with an HTML form to collect user data (e.g., name, email).
Step 7: Design the HTML form using standard input fields and set the method="POST" and action="/submit" attributes.
Step 8: In the server file, create a route for GET / to render the form view using res.render('form').
Step 9: Create a POST route /submit to receive the form data using req.body.
Step 10: Use the fs.readFile() method to read existing data from a data.json file. If the file is empty or missing, initialize an empty array.
Step 11: Append the new user data from req.body to the existing array.
Step 12: Use the fs.writeFile() method to save the updated array back into data.json in stringified format.
Step 13: After successful writing, redirect the user to a success page or send a response like "Data saved successfully".
Step 14: (Optional) Create a new route (e.g., GET /view) to read from data.json and render a

page that displays all submitted entries.

Step 15: Start the server using app.listen() and open http://localhost:3000 in the browser to interact with the form.

DESIGN :



Files:

index.html

- The frontend HTML file of the project.
- Provides the user interface (form input, buttons, and task list).
- May include form elements to enter tasks, and uses JavaScript to send data to the server**.**

package.json

- The project configuration file.
- Lists project metadata (name, version) and dependencies (like Express).
- Used by Node.js to manage packages.
- Created using npm init.

server.js

- The backend Node.js server file.
- Contains the Express server logic to:
- Serve static files.
- Handle GET and POST requests.
- Read from and write to the todos.json file.

todos.json

- A JSON file used for storage.
- Stores the list of todos entered by the user.
- Read and updated dynamically by the backend server (server.js) to persist data across reloads.

SOURCE CODE:

server.js (Node.js + Express)

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const todosPath = path.join(__dirname, 'data', 'todos.json');
const server = http.createServer((req, res) => {
    if (req.method === 'GET' && req.url === '/todos') {
        fs.readFile(todosPath, 'utf-8', (err, data) => {
            if (err) {
                res.writeHead(500);
                return res.end('Error reading file');
            }
            res.writeHead(200, { 'Content-Type': 'application/json' });
            res.end(data);
        });
    } else if (req.method === 'POST' && req.url === '/todos') {
        let body = '';
        req.on('data', chunk => body += chunk);
        req.on('end', () => {
            fs.writeFile(todosPath, body, err => {
                if (err) {
                    res.writeHead(500);
                    return res.end('Error writing file');
                }
                res.writeHead(200);
                res.end('Data saved');
            });
        });
    } else {
        let filePath = path.join(__dirname, 'public', req.url === '/' ? 'index.html' : req.url);
        let extname = path.extname(filePath);
        let contentType = {
            '.html': 'text/html',
            '.js': 'application/javascript',
            '.css': 'text/css'
        }[extname] || 'text/plain';
```

```
      fs.readFile(filePath, (err, content) => {
        if (err) {
          res.writeHead(404);
          res.end('Not Found');
        } else {
          res.writeHead(200, { 'Content-Type': contentType });
          res.end(content);
        }
      });
  }
});
server.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

index.html

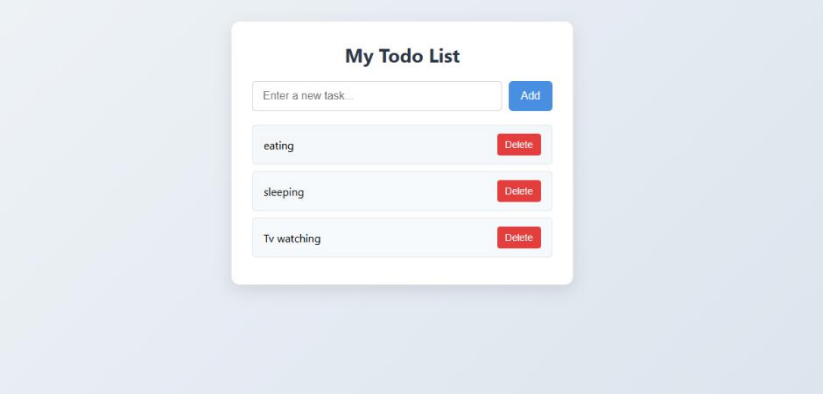
```
<!DOCTYPE html>
<html ng-app="todoApp">
<head>
  <title>Todo App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.min.js"></script>
  <script src="app.js"></script>
  <link rel="stylesheet" href="style.css">
</head>
<body ng-controller="TodoController">
  <div class="container">
    <h1>My Todo List</h1>
    <div class="input-group">
      <input type="text" ng-model="newTodo" placeholder="Enter a new task..." />
      <button ng-click="addTodo()">Add</button>
    </div>
    <ul>
      <li ng-repeat="todo in todos">
        {{ todo }}
        <button ng-click="removeTodo($index)">Delete</button>
      </li>
    </ul>
  </div>
</body>

</html>
```

tasks.json

```
[ {
  "text": "MERN Exp completed",
  "category": "Education",
  "datetime": "2025-07-21T04:35:00.000Z",
```

```
    "priority": "High"
  },
  {
    "text": "Upcoming CAT Exam",
    "category": "Exam",
    "datetime": "2025-07-23T03:36:00.000Z",
    "priority": "Medium"
  },
  {
    "text": "Start Preparing Full Stack Exam",
    "category": "Study",
    "datetime": "2025-07-22T01:36:00.000Z",
    "priority": "Low"
  }
]
```

OUTPUT:



COE (30):	
Observation(10):	
Record (10):	
Viva (10):	
Total (60):	

RESULT:

The AngularJS Todo application was successfully created. The tasks were stored and retrieved using a Node.js server with JSON file persistence, and tasks remained visible after page reloads was executed successfully.