Advanced JavaScript Framework Project Activities

Employee Management Dashboard using Angular and TypeScript

Team Members:

1. Jacques Paul (2462527) – s.jacques@btech.christuniversity.in
2. Vinay Viswanath (2462535) – vinay.viswanath@btech.christuniversity.in
3. Prarthana Puthan Purayil (2462526) – Prarthana.puthan@btech.christuniversity.in
4. V. O. Anirudh (2462536) – vo.anirudh@btech.christuniversity.in
5. Bharath URS P (2462508) – Bharath.urs@btech.christuniversity.in

Course: L&T Advanced JavaScript Framework

Instructor Name: Mrs. Pallavi

Institution: Christ (Deemed to be university)

Date of Submission: 23.01.2026

Abstract

This project is a web-based Employee Management Dashboard developed using an advanced JavaScript framework (Angular) to manage and visualize employee information in an organized and interactive manner. The primary objective of the application is to demonstrate the use of modern front-end development concepts such as component-based architecture, routing, services, dependency injection, and reactive programming.
The system allows users to view employee details, manage departmental information, and perform basic employee operations through a clean and responsive user interface. Angular Material components are used to enhance the visual appeal and usability of the dashboard. The final outcome is a professional, scalable, and maintainable front-end application that simulates real-world enterprise-level employee management systems.

Objectives:

The objectives of this project are:

- To design a structured and user-friendly Employee Management Dashboard
- To implement a component-based architecture using an advanced JavaScript framework (Angular)
- To utilize TypeScript for strong typing, maintainability, and scalability
- To implement routing and navigation for seamless user experience
- To integrate Angular Material components for a professional and responsive UI
- To demonstrate the use of services, dependency injection, and observables for data handling

Scope of the Project:

- Focused on front-end application development using Angular
- Simulates employee data management without real backend integration
- Uses mock data / in-memory data services for demonstration
- Designed to be responsive across desktop, tablet, and mobile devices
- Covers core modules such as Employee List, Employee Details, and Add/Edit Employee

Tools & Technologies Used:

Tool / Technology	Purpose
Angular (v17+)	Front-end framework
TypeScript	Strongly typed scripting language
Angular Material	UI components and theming
HTML5	Application structure
CSS3 / SCSS	Styling and layout
VS Code	Code editor
Google Chrome	Testing and debugging

Application Architecture Overview:

- Component-based architecture for modular development
- Centralized services for employee data management
- Angular Router for page navigation
- Use of observables (RxJS) for asynchronous data flow
- Separation of concerns using models, services, components, pipes, and directives

Component Structure:

- Navbar Component – Provides application-wide navigation
- Employee List Component – Displays employee records in a tabular format
- Employee Detail Component – Displays detailed employee information
- Employee Form Component – Allows adding and editing employee data

UI Design & Styling Strategy:

- Angular Material used for consistent and responsive UI design
- Material tables, forms, buttons, and toolbars implemented
- Responsive layout achieved using Flexbox and Material grid system
- Consistent color themes and typography for professional appearance
- Clean spacing and alignment for improved readability

Key Features:

Feature	Description
Responsive Dashboard	Adapts to different screen sizes
Modular Components	Easy maintenance and scalability
Routing & Navigation	Smooth page transitions
Material UI	Professional and accessible interface
Data Binding	Real-time UI updates

Challenges Faced & Solutions:

Challenge	Solution
Managing component communication	Used shared services and dependency injection
Complex form validation	Implemented reactive forms with validators
UI consistency across components	Adopted Angular Material theming

Outcome:

- Successfully developed a structured and interactive Employee Management Dashboard
- Demonstrated effective use of Angular framework concepts
- Achieved a clean, responsive, and professional UI
- Improved understanding of enterprise-level front-end application development

Future Enhancements:

- Integration with a real backend REST API
- Role-based authentication and authorization
- Advanced search and filtering options
- Data visualization using charts and graphs
- Export employee data to PDF or Excel

Sample Code:

Employee Model:

```
export interface Employee {
  id: number;
  name: string;
  email: string;
  role: string;
  department: string;
  salary: number;
  joiningDate: string;
}
```

Employee Service:

```
@Injectable({ providedIn: 'root' })
export class EmployeeService {
  private apiUrl = 'http://localhost:3000/employees';

  constructor(private http: HttpClient) {}

  getEmployees(): Observable<Employee[]> {
    return this.http.get<Employee[]>(this.apiUrl);
  }

  getEmployee(id: number): Observable<Employee> {
    return this.http.get<Employee>(`${this.apiUrl}/${id}`);
  }

  addEmployee(emp: Employee): Observable<Employee> {
    return this.http.post<Employee>(this.apiUrl, emp);
  }

  updateEmployee(emp: Employee): Observable<Employee> {
    return this.http.put<Employee>(`${this.apiUrl}/${emp.id}`, emp);
  }

  deleteEmployee(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Routing Configuration:

```
const routes: Routes = [
  { path: '', redirectTo: 'employees', pathMatch: 'full' },
  { path: 'employees', component: EmployeeListComponent },
  { path: 'employee/add', component: EmployeeFormComponent },
  { path: 'employee/:id', component: EmployeeDetailComponent, canActivate: [AuthGua
rd] }
];
```

Auth Guard:

```
@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  canActivate(): boolean {
    return true; // simulate authentication
  }
}
```

Mock Backend (Json server):

```
{
  "employees": [
    {
      "id": 1,
      "name": "John Doe",
      "email": "john@company.com",
      "role": "Developer",
      "department": "IT",
      "salary": 60000,
      "joiningDate": "2022-05-01"
    }
  ]
}
```

Run server:

```
npx json-server --watch db.json --port 3000
```

Conclusion:

This project demonstrates the practical application of an advanced JavaScript framework in building a real-world Employee Management Dashboard. By leveraging Angular, TypeScript, and Angular Material, the application achieves modularity, scalability, and a professional user interface. The project significantly enhanced understanding of modern front-end architectures, reactive programming, and UI component design.

References:

- L&T LMS: https://learn.lntedutech.com/Landing/MyCourse
- Angular Documentation: https://angular.io/docs
- W3Schools: https://www.w3schools.com/