Advanced JavaScript Framework Project Activities

Student Course Enrollment and Management System 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

1.Introduction

The **Student Course Enrollment and Management System** is a sophisticated web application designed to streamline the administrative and academic workflows of modern educational institutions. Developed using the **Angular** framework and **TypeScript**, this project serves as a comprehensive digital dashboard that allows for the efficient management of student data, course catalogs, and enrollment records. In an increasingly digital academic environment, such systems are vital for maintaining data integrity, improving accessibility, and providing a seamless experience for both administrators and students.

At its core, the application leverages a **modular, component-based architecture**. This design philosophy ensures that different functional areas—such as the student directory, course repository, and registration forms—are developed as independent units that communicate effectively through shared services. By using **TypeScript**, the project implements strong typing and object-oriented principles, which significantly reduces development errors and ensures that the data models for Students and Courses are consistent throughout the application's lifecycle.

The user interface is built using **Angular Material**, which provides a collection of professional, high-quality UI components. This ensures that the system is not only visually appealing but also fully responsive, allowing users to manage enrollments across various devices, from desktop workstations to mobile tablets. By integrating **Reactive Forms** and **RxJS Observables**, the system handles complex data entry and asynchronous backend communication with ease, simulating the performance and reliability of a production-grade enterprise application.

Through the implementation of this system, we demonstrate a mastery of modern front-end technologies. The project successfully showcases the transition from static web pages to a dynamic, single-page application (SPA) capable of handling real-time updates, secure routing, and robust data validation, providing a scalable foundation for future institutional growth.

2.Objective of the Project

- Design a structured and user-friendly interface for student management.
- Implement a component-based architecture using Angular.
- Utilize **TypeScript** for strong typing and scalability.
- Implement routing for seamless transitions between student and course modules.
- Integrate **Angular Material** components for a professional UI.
- Demonstrate data handling using services, dependency injection, and observables.

3.Tools and Technologies Used

Tool/Technology	Purpose
Angular (v17+)	Front-end framework
TypeScript	Strongly typed scripting language
Angular Material	UI components and theming
HTML5 / CSS3	Application structure and layout
VS Code	Code editor
JSON Server	Mock API for data simulation

4. System Architecture

The application is built using a **Component-Based Architecture**. It features a modular structure where concerns are separated into models, services, components, pipes, and directives.

5. Module Implementation

5.1 Dashboard Module

Adapts to different screen sizes and provides summary metrics.

5.2 Student List Module

Displays registered students in a tabular format using `MatTable`.

5.3 Student Registration Module

Features a form for adding new students with real-time UI updates.

5.4 Attendance Tracker Module

Designed to manage student presence logs dynamically.

5.5 Leave Request Module

Allows students to submit requests via specialized forms.

5.6 Leave Approval Module (HR/Admin)

Provides an interface for administrators to approve or reject pending requests

6. Routing and Navigation

The **Angular Router** manages smooth page transitions.

- **Default Path:** Redirects to `students` list.
- **Guarded Routes:** The enrollment page is protected by an `AuthGuard`.

7. Forms and Validation

The project implements **Reactive Forms** for complex data entry.

- **Validators:** Fields like "Email" and "Student Name" are strictly validated.
- **Feedback:** Provides dynamic error messages for better user accessibility.

8. Services and Dependency Injection

A centralized `StudentService` manages data flow.

- **Dependency Injection:** Injects `HttpClient` to communicate with the mock backend.
- **Observables:** Uses RxJS to handle asynchronous data streams.

9. Mock Backend (JSON Server)

A mock API simulates a real-world database.

- **Execution:** `npx json-server --watch db.json --port 3000`.
- **Data Structure:** Stores student details, course IDs, and enrollment dates .

10.Sample code

student.model.ts

```
export interface Student {
  id: number;
  name: string;
  email: string;
  department: string;
}
```

course.model.ts

```
export interface Course {
  id: number;
  title: string;
  department: string;
  credits: number;
}
```

enrollment.model.ts

```
export interface Enrollment {
  studentId: number;
  courseId: number;
}
```

Services & Dependency Injection

student.service.ts

```
@Injectable({ providedIn: 'root' })
export class StudentService {
  private students: Student[] = [
    { id: 1, name: 'Ananya', email: 'ananya@mail.com', department: 'CSE' },
    { id: 2, name: 'Rahul', email: 'rahul@mail.com', department: 'ECE' }
  ];

  getStudents() {
    return this.students;
  }
}
```

---

course.service.ts

```
@Injectable({ providedIn: 'root' })
export class CourseService {
  private courses: Course[] = [
    { id: 101, title: 'Angular', department: 'CSE', credits: 4 },
    { id: 102, title: 'DBMS', department: 'CSE', credits: 3 }
```

```
  ];

  getCourses() {
    return this.courses;
  }
}
```

Routing Configuration

app-routing.module.ts

```
const routes: Routes = [
  { path: '', component: DashboardComponent },
  { path: 'students', component: StudentListComponent },
  { path: 'courses', component: CourseListComponent },
  { path: 'enroll', component: EnrollStudentComponent, canActivate:
[AuthGuard] }
];
```

Route Guard

auth.guard.ts

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

Forms & Validation

enroll-student.component.ts

```
this.enrollForm = this.fb.group({
  studentId: ['', Validators.required],
  courseId: ['', Validators.required]
});
```

11. Result

The project successfully delivered a structured, professional-grade management system. It demonstrated mastery of Angular's modular design and achieved a clean, responsive UI.

12. Conclusion

Leveraging Angular, TypeScript, and Angular Material significantly improves the scalability and maintainability of enterprise-level applications. The project enhanced the team's understanding of reactive programming and modern front-end design.

13. Future Enhancements

- Integration with a live **REST API** backend.
- **Role-based** authentication for students and admins.
- **Data Visualization** using charts and graphs.
- **Export** features for PDF or Excel reports.