**Batch T4**

**Practical No. 3**

**Title of Assignment :**

**MIS system**

**Student Name: Parshwa Herwade**

**Student PRN: 22510064**

Design and implement a web-enabled (portal) student MIS (Management

Information System) for University schema attached in separate file.

Technology stack : Angular 19, NodeJS , MySQL

Requirement Specifications :

1. Role based User management at different levels

2. Data Entry form for each entity with CRUD

3. Generic report generator : Should be able to display / list any data.

Note : Follow the submission guidelines.

Objective / Aim

The primary objective of this project is to design and implement a web‑based Student Management Information System (MIS) that automates and streamlines various administrative processes in a university setting. The system will:

Provide role‑based user management (e.g., admin, instructor, student) to ensure secure access to different modules.

Allow data entry for different entities (such as students, instructors, courses, etc.) using forms that support full CRUD (Create, Read, Update, Delete) operations.

Include a generic report generator that can display and export any table’s data from the MySQL database.

Introduction

Modern universities require efficient, accessible, and secure systems to manage large amounts of data related to students, faculty, courses, and other administrative functions. Traditional manual or paper‑based systems are error‑prone and inefficient. This project leverages modern web technologies—Angular 19 for the front‑end, NodeJS for the backend, and MySQL for the database—to create an integrated MIS portal. This portal not only supports role‑based access control, ensuring that different users see only what they need to, but also provides dynamic data entry and reporting capabilities, thereby streamlining operations and improving data accuracy and accessibility.

Theory / Algorithms

Architecture & Technology Stack:

Angular (Front‑end):

Angular’s component‑based architecture is used to create dynamic, single‑page applications. It provides robust features such as routing, dependency injection, and two‑way data binding, which are essential for building a responsive MIS portal.

NodeJS (Backend):

NodeJS, with its non‑blocking, event‑driven architecture, is used to create RESTful API endpoints that handle business logic, user authentication, and data processing.

MySQL (Database):

MySQL is used to store structured data related to the university (e.g., students, instructors, courses). SQL queries are used to perform CRUD operations.

Role‑Based User Management:

The system distinguishes between different user roles (admin, instructor, student) by verifying user credentials and generating JSON Web Tokens (JWT) upon successful login.

The JWT is stored in localStorage and used by an AuthGuard to protect routes, ensuring that only authorized users can access sensitive data.

CRUD Operations:

Create: Data entry forms are provided for each entity. When submitted, the front‑end sends POST requests to NodeJS, which then inserts data into MySQL.

Read: Data is fetched via GET requests from the backend and rendered in Angular components (tables, dashboards).

Update: Update forms (often through modals or inline editing) allow users to modify records. The updated data is sent via PUT requests to the backend.

Delete: Deletion of records is handled via DELETE requests after user confirmation.

Generic Report Generator:

The report generator module provides a dropdown list of allowed table names. When a user selects a table and clicks “Generate Report,” Angular sends a GET request to a backend endpoint (e.g., /api/report?table=student).

The backend verifies the requested table against a whitelist, executes a SELECT query, and returns the results as JSON.

On the front‑end, Angular dynamically generates table headers and rows using a custom KeysPipe to iterate over object keys.

Users can then download the report as a CSV file.

Documentation: Functional Block Diagram / DFD

(A high‑level description is provided here. Actual diagrams should be drawn using a tool such as Visio or Draw.io and attached as part of the submission.)

Client (Angular):

– Contains the Login, Dashboard (Admin, Instructor, Student), and Report Generator components.

– Communicates with the backend using HTTP requests (GET, POST, PUT, DELETE).

Server (NodeJS):

– Exposes RESTful API endpoints for user authentication, CRUD operations, and report generation.

– Implements middleware for token verification (AuthGuard).

Database (MySQL):

– Stores university-related data such as students, instructors, departments, courses, etc.

The Data Flow Diagram (DFD) shows the flow of data from the user (login, data entry, report requests) through the Angular front‑end to the NodeJS API and finally into the MySQL database, and vice‑versa for data retrieval.

Procedure

Requirement Analysis:

– Reviewed the University schema and requirement specifications.

– Defined user roles, CRUD operations, and report generator functionality.

System Design:

– Designed the overall architecture (Angular front‑end, NodeJS backend, MySQL database).

– Created diagrams (DFD, block diagrams) outlining data flow and module interactions.

Implementation:

– Developed Angular components (login, dashboards, report generator).

– Implemented RESTful API endpoints in NodeJS for authentication, CRUD, and reporting.

– Designed the MySQL database schema and inserted sample data.

Testing and Debugging:

– Tested API endpoints using Postman.

– Performed integration testing of Angular with NodeJS.

– Debugged issues related to routing, token storage, and data rendering.

Deployment:

– Deployed the application on a local server for demonstration and further testing.

Actual Experiments/Simulation, Results/Observations

(Attach screenshots as part of your submission; below is a summary of observations.)

Login Functionality:

– Users can log in as admin, instructor, or student.

– JWT tokens are stored and used for route protection.

CRUD Operations:

– Admins can add, update, delete, and view instructors.

– Similar functionality was implemented for students and instructors.

Report Generation:

– The Report Generator displays a dropdown of allowed tables.

– Upon selection and clicking “Generate Report,” the system fetches and displays data in a table.

– The CSV download functionality allows saving reports locally.

Performance:

– The system responded quickly to CRUD operations and report generation, and all tests were successful.

Conclusion

This project successfully implements a web‑enabled Student Management Information System (MIS) for a university using Angular 19, NodeJS, and MySQL. The system supports role‑based user management, provides data entry forms with full CRUD functionality for each entity, and includes a generic report generator that allows users to view and download data from any database table. The project demonstrates the effective use of modern web technologies to improve administrative efficiency and data accuracy in a university setting.

***Verdict:  
[This project runs successfully:  
There is login page for admin ,student and instructor option, with each having their own dashboards and also there is report generator which gives u report about the info in the database and which can be even downloaded too in the form of .csv file .  
Plus CURD operations can be done tooo***

***Testing has been also done with help of postman….]***





