**CE – 431 (DBMS)**

**Final Project Report**

**By:**

**Azib Zahid (2017092)**

**……….**

**……….**

**……….**

Table of Contents

[1. Introduction 2](#_Toc61038978)

[2. Project Description 2](#_Toc61038979)

[3. Detailed Project Requirements 3](#_Toc61038980)

[4. Technologies Used with Description 3](#_Toc61038981)

[5. EER Diagram 4](#_Toc61038982)

[6. Database Schema 5](#_Toc61038983)

[**7.** **API Endpoints** 6](#_Toc61038984)

[Faculty 6](#_Toc61038985)

[Faculty Member 7](#_Toc61038986)

[Course 7](#_Toc61038987)

[Student 8](#_Toc61038988)

[Section 8](#_Toc61038989)

[Enrollment 9](#_Toc61038990)

[Attendance 9](#_Toc61038991)

[Result 9](#_Toc61038992)

[Admin 9](#_Toc61038993)

[8. References 10](#_Toc61038994)

[Recorded Presentation Link 11](#_Toc61038995)

# Introduction

This report explains our project report for the Database Management Systems (DBMS) course. The proposal is about developing a Campus management system commonly referred to as CMS to be used by a university or college to handle details about students, faculty members, academics, and exams. The developed CMS will be a web application having backend developed in Node JS and frontend in React JS. The application will use MySQL relational database management system. All technologies to be used is explained in detail.

# Project Description

Campus Management System (CMS) deals with all kind of student details, academic related reports, institute details, course details, curriculum, batch details and other resource related details too. It tracks all the details of a student from the day one to the end of his course which can be used for all reporting purpose, tracking of attendance, progress in the course, completed semesters years, coming semester year curriculum details, exam details, project or any other assignment details, final exam result; and all these will be available for future references too. Our application will have the databases of Courses offered by the institute under all levels of graduation or main streams, teacher or faculty's details, batch execution details, students' details in all aspects. This application can facilitate us explore all the activities happening in the institute, even we can get to know which teacher / faculty is assigned to which batch, the status of a batch, attendance percentage of a batch and upcoming requirements of a batch. Different reports and Queries can be generated based of vast options related to students, batch, course, teacher / faculty, exams, semesters, certification and even for the entire institute.

# Detailed Project Requirements

We have a very versatile range of technologies that can be used to address the problem. It can be concluded from the formal problem statement; we need technologies capable of performing the tasks below:

1. We need a platform that is easily accessible
2. We need a database server which can hold the records of Students, Teachers and Courses
3. An interface that can communicate with the server and present the information in a user-friendly way.

# Technologies Used with Description

We are developing a web-based application and for that we have selected the following technologies:

1. Node.js + Express.js for Backend

Node.js will be our application's back-end platform that will run our application on the Cloud Server. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. (www.guru99.com, n.d.)

1. React.js for frontend

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used for building fast and interactive user interfaces for web and mobile applications. (Reactjs.org, 2019)

1. MySQL

MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the database.

1. A Database Server
2. Sequelize ORM

Sequelize is a powerful library in JavaScript that makes it easy to manage a SQL database. Sequelize can layer over different protocols, but here we will use MySQL. At its core, Sequelize is an Object-Relational Mapper – meaning that it maps an object syntax onto our database schemas. Sequelize uses Node.JS and JavaScript’s object syntax to accomplish its mapping.

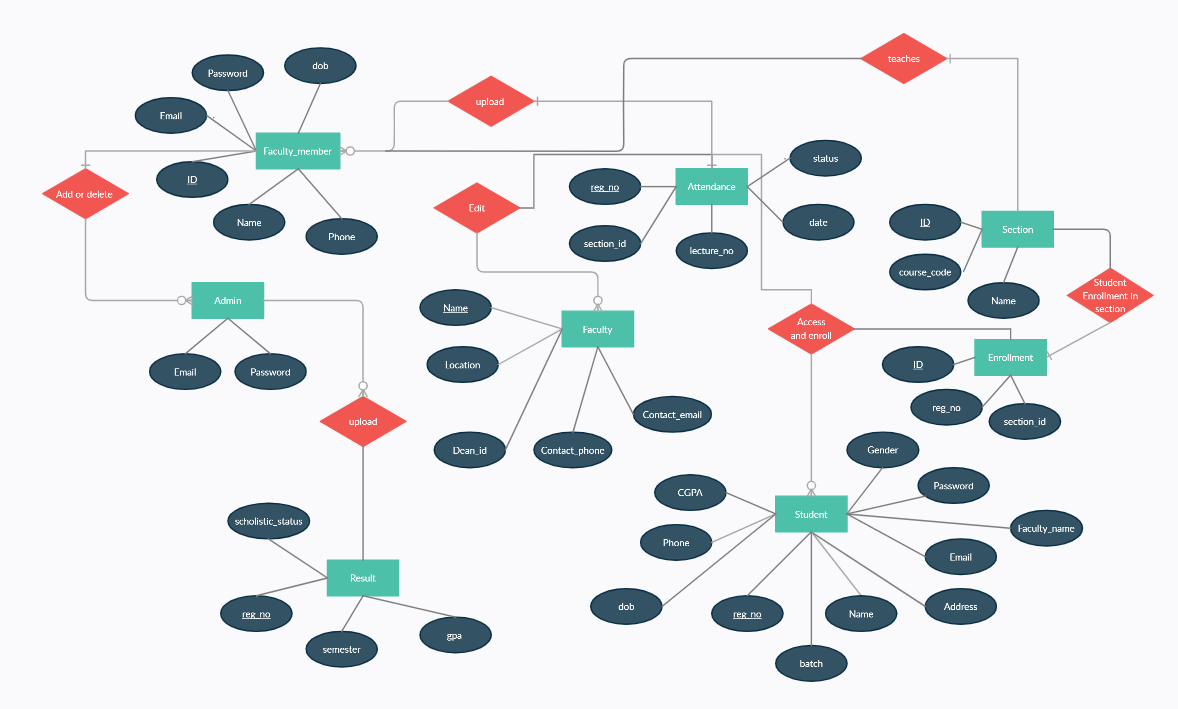
1. GitHub

GitHub can be divided into the Git, and the Hub. The service includes access controls as well as several collaboration features like tools for basic task management and for all projects you handle. GitHub hosts your source code projects in a variety of different programming languages and keeps track of the various changes made to every iteration. So, the “Git” implies the version control system; a tool which allows developers to keep track of the constant revisions to their code. The “Hub” is the community of like-minded individuals who participate. (Apiumhub, 2017)

# EER Diagram

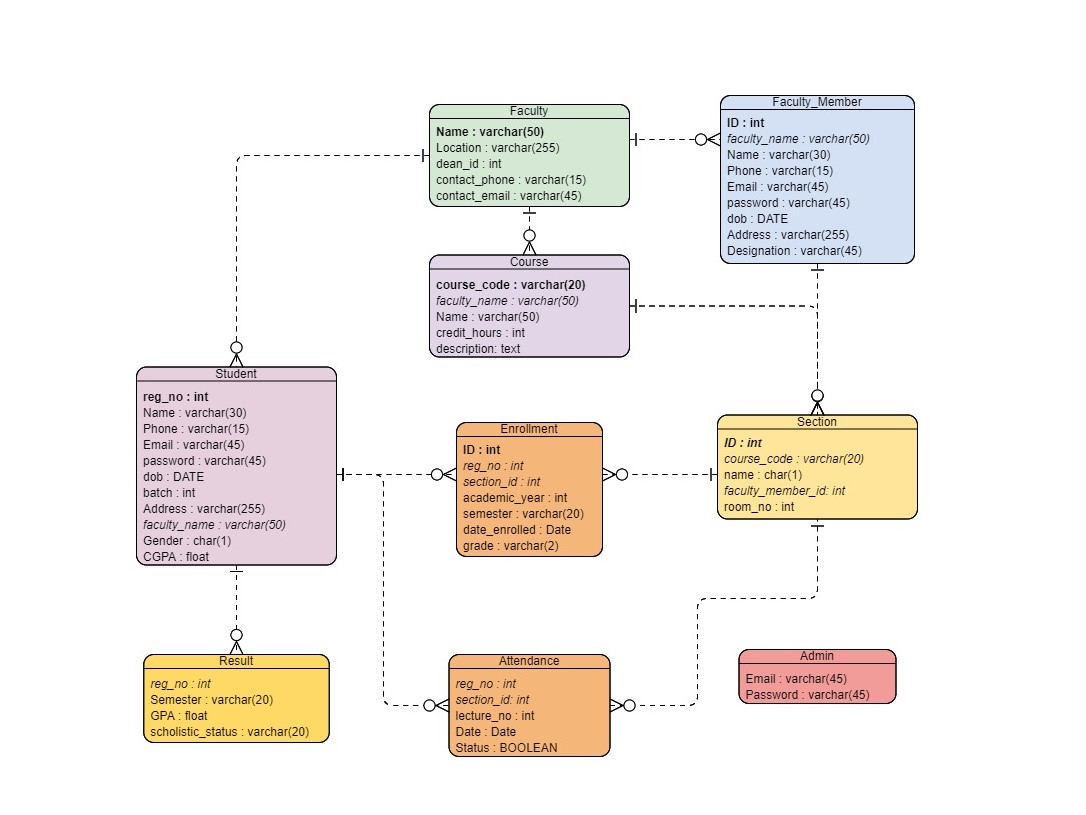
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known **as Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. (Cacoo, 2018)



# Database Schema

A database schema represents the logical configuration of all or part of a relational database. It can exist both as a visual representation and as a set of formulas known as integrity constraints that govern a database. These formulas are expressed in a data definition language, such as SQL. As part of a data dictionary, a database schema indicates how the entities that make up the database relate to one another, including tables, views, stored procedures, and more.



1. **API Endpoints**

Faculty

* Get all faculties: /api/faculty
* Get a faculty /api/faculty/:id
* Create a faculty /api/faculty
* Update a faculty /api/faculty/:id
* Delete a faculty /api/faculty/:id
* Get all faculty members of Faculty /api/faculty/:facultyId/members
* Get a faculty member of Faculty /api/faculty/:facultyId/members/:id
* Get Dean of Faculty /api/faculty/:facultyId/dean
* Create a faculty member of Faculty /api/faculty/:facultyId/members/
* Update a faculty member of Faculty /api/faculty/:facultyId/members/:id
* Delete a faculty member of Faculty /api/faculty/:facultyId/members/:id
* Get all courses of Faculty /api/faculty/:facultyId/courses
* Get a course of Faculty /api/faculty/:facultyId/courses/:id
* Create a course of Faculty /api/faculty/:facultyId/courses/
* Update a course of Faculty /api/faculty/:facultyId/courses/:id
* Delete a course of Faculty /api/faculty/:facultyId/courses/:id
* Get all students of Faculty /api/faculty/:facultyId/students
* Get a student of Faculty /api/faculty/:facultyId/students/:id
* Create a student of Faculty /api/faculty/:facultyId/students/
* Update a student of Faculty /api/faculty/:facultyId/students/:id
* Delete a student of Faculty /api/faculty/:facultyId/students/:id

Faculty Member

* Get all faculty members: /api/faculty-member
* Get a faculty member /api/faculty-member/:id
* Create a faculty member /api/faculty-member
* Update a faculty member /api/faculty-member/:id
* Delete a faculty member /api/faculty-member/:id
* Faculty Member Login /api/faculty-member/login
* Get all courses of Faculty Member /api/faculty-member/:facultyMemberId/courses

Course

* Get all courses: /api/course
* Get a course /api/course/:id
* Create a course /api/course
* Update a course /api/course/:id
* Delete a course /api/course/:id
* Get all sections of a course: /api/course /:course\_code/sections
* Get a section of a course /api/course/:course\_code/sections/:id
* Create a section of a course /api/course/:course\_code/sections
* Update a section of a course /api/course/:course\_code/sections/:id
* Delete a section of a course /api/course/:course\_code/sections/:id
* Get all students of a course: /api/course /:course\_code/students

Student

* Get all students: /api/student
* Get a student /api/student/:id
* Create a student /api/student
* Update a student /api/student/:id
* Delete a student /api/student/:id
* Student Login /api/student/login
* Get all results of a student: /api/student/:reg\_no/results
* Get a result of a student /api/student/:reg\_no/results/:id
* Create a result of a student /api/student/:reg\_no/results
* Update a result of a student /api/student/:reg\_no/results/:id
* Delete a result of a student /api/student/:reg\_no/results/:id
* Get all attendances of a student: /api/student/:reg\_no/attendances
* Get an attendance of a student /api/student/:reg\_no/attendances/:id
* Create an attendance of a student /api/student/:reg\_no/attendances
* Update an attendance of a student /api/student/:reg\_no/attendances/:id
* Delete an attendance of a student /api/student/:reg\_no/attendances/:id
* Get all enrollments of a student: /api/student/:reg\_no/enrollments
* Get all enrollments of semester: /api/student/:reg\_no/enrollments/:semester
* Get an enrollment of a student /api/student/:reg\_no/enrollments/:id
* Create an enrollment of a student /api/student/:reg\_no/enrollments
* Update an enrollment of a student /api/student/:reg\_no/enrollments/:id
* Delete an enrollment of a student /api/student/:reg\_no/enrollments/:id
* Get all courses of a student: /api/student/:reg\_no/courses

Section

* Get all sections: /api/section
* Get a section /api/section/:id
* Create a section /api/section
* Update a section /api/section/:id
* Delete a section /api/section/:id
* Get all enrollments of a section: /api/section/:sectionId/enrollments
* Get an enrollment of a section: /api/section/:sectionId/enrollments/:id
* Create an enrollment of a section: /api/section/:sectionId/enrollments
* Update an enrollment of a section: /api/section/:sectionId/enrollments/:id
* Delete an enrollment of a section: /api/section/:sectionId/enrollments/:id
* Get all attendances of a section: /api/section/:sectionId/attendances
* Get an attendance of a section: /api/section/:sectionId/attendances/:id
* Create an attendance of a section: /api/section/:sectionId/attendances
* Update an attendance of a section: /api/section/:sectionId/attendances/:id
* Delete an attendance of a section: /api/section/:sectionId/attendances/:id

Enrollment

* Get all enrollments: /api/enrollment
* Get an enrollment /api/enrollment/:id
* Create an enrollment /api/enrollment
* Update an enrollment /api/enrollment/:id
* Delete an enrollment /api/enrollment/:id
* Get a course from enrollment /api/enrollment/:eId/course

Attendance

* Get all attendances: /api/attendance
* Get an attendance /api/attendance/:id
* Create an attendance /api/attendance
* Update an attendance /api/attendance/:id
* Delete an attendance /api/attendance/:id

Result

* Get all results: /api/result
* Get a result /api/result/:id
* Create a result /api/result
* Update a result /api/result/:id
* Delete a result /api/result/:id

Admin

* Get all admins: /api/admin
* Get an admin /api/admin/:id
* Create an admin /api/admin
* Update an admin /api/admin/:id
* Delete an admin /api/admin/:id
* Admin Login /api/admin/login

# References

* *Apiumhub. (2017). Top 7 benefits you get by using Github | Apiumhub. [online] Available at:* [*https://apiumhub.com/tech-blog-barcelona/using-github/*](https://apiumhub.com/tech-blog-barcelona/using-github/)*.*
* *‌Cacoo. (2018). ER diagrams vs. EER diagrams: what’s the difference? [online] Available at:* [*https://cacoo.com/blog/er-diagrams-vs-eer-diagrams-whats-the-difference/*](https://cacoo.com/blog/er-diagrams-vs-eer-diagrams-whats-the-difference/)*.*
* *www.guru99.com. (n.d.). Node.js Express FrameWork Tutorial - Learn in 10 Minutes. [online] Available at: https://www.guru99.com/node-js-express.html#:~:text=js%3F- [Accessed 4 Jan. 2021].*
* *Reactjs.org. (2019). React – A JavaScript library for building user interfaces. [online] Available at: https://reactjs.org/.*

‌

‌

‌

# Recorded Presentation Link