Experiment 2

Question: Creating a Software Requirements Specification (SRS) Document:

- **Objective: Develop a comprehensive SRS document for a hypothetical software project.**
- **Procedure: Conduct requirement analysis, distinguish between functional and nonfunctional requirements and document them according to IEEE standards.**
- **Expected Outcome: A well-documented SRS that includes all the requirements and specifications for a software project.**

Result:

Software Requirements Specification (SRS)

For: *E-Learning platform*
Version: 1.0
Date: 17 August 2025
Prepared by: Sameer Upadhyay, Saurabh Rana, Shiwani Bhandari, Shresth Dwivedi

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the E-learning Platform, a webbased application designed to facilitate online education. This SRS serves as the primary reference for developers, testers, and stakeholders to ensure the final product meets all specified functional and nonfunctional requirements.

1.2 Scope

The E-learning Platform will be a web application that offers a complete solution for online education. The system will manage user accounts, provide tools for instructors to create and deliver courses, and enable students to enroll in courses and track their progress. The platform's features will include secure authentication, course content delivery, and interactive learning elements.

1.3 Definitions, Acronyms, and Abbreviations

- **MERN**: MongoDB, Express.js, React.js, and Node.js
- **SRS**: Software Requirements Specification
- **UI/UX**: User Interface / User Experience
- **API**: Application Programming Interface
- **CRUD**: Create, Read, Update, Delete

1.4 References

- IEEE Std 830-1998 (IEEE Recommended Practice for Software Requirements Specifications)
- UML 2.5 Standard Documentation
- MySQL & PHP Development Guides

1.5 Overview

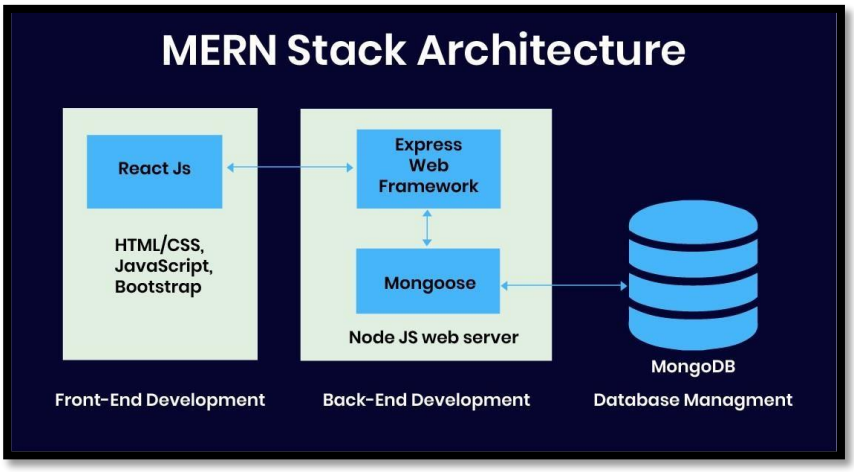
This document details the functional and non-functional requirements of the E-learning Platform, along with user needs, system features, and architectural details

2. Overall Description

2.1 Product Perspective

The E-learning Platform is a standalone MERN stack application. It consists of a React.js frontend, Node.js + Express.js backend, and a MongoDB database. The system communicates via REST APIs and runs on modern browsers.

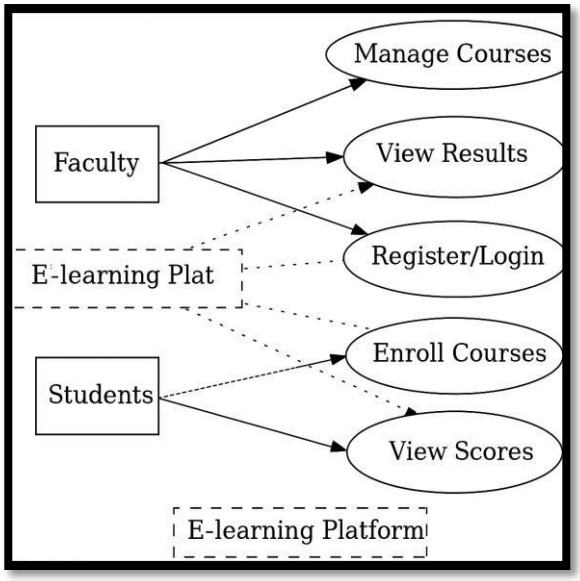
High-Level Architecture Diagram:



2.2 Product Functions

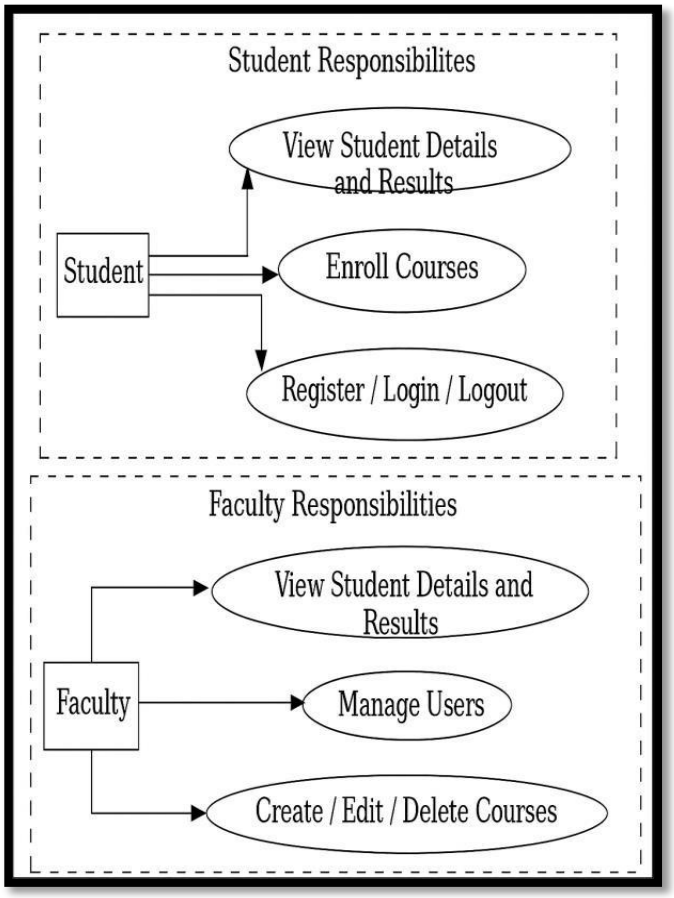
Major functions:

- **Student Registration & Records**: Enabling registration of new students and maintaining personal and academic details.
- **Faculty Management**: Allowing administrators to add, update, and manage faculty profiles.
- **Attendance Tracking**: Recording and monitoring student attendance by faculty members.
- **Exam Scheduling & Result Processing**: Supporting automated exam timetable generation and publishing of results.
- **Fee Management**: Tracking student fee status, generating receipts, and managing payments.



2.3 User Classes and Characteristics

- **Administrator**: Holds the highest privileges, responsible for managing users, system settings, and institutional configurations.
- **Faculty**: Medium-level privileges, responsible for managing courses, recording attendance, and evaluating students.
- **Students**: Limited access, primarily for viewing personal academic records, attendance, fee status, and accessing learning resources.

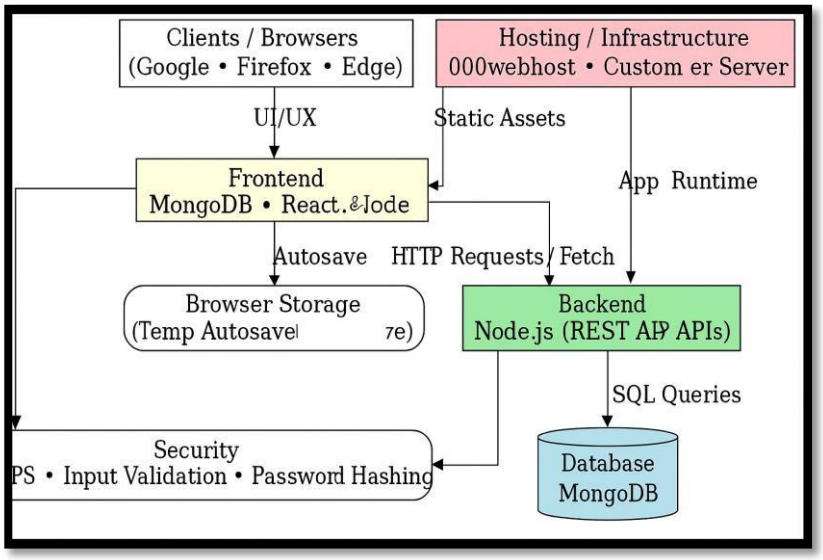


2.4 Operating Environment

- **Platform**: Web-based (React.js, Node.js, Express.js, MongoDB)
- **Server**: Node.js runtime with Express.js framework
- **Database**: MongoDB (v5.0 or above)
- **Supported Browsers**: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari

2.5 Design and Implementation Constraints

- Must comply with data protection and privacy regulations.
- Requires stable internet or intranet connectivity for proper functioning.
- All communication must be secured with SSL/TLS encryption.



2.6 Assumptions and Dependencies

- Each user will have unique login credentials to access the system.
- Deployment is dependent on the college's IT infrastructure for hosting and database management. Faculty and students are expected to have basic digital literacy to interact with the platform.

3. Specific Requirements

3.1 Functional Requirements

FR1: User Authentication & Authorization

- The system shall allow new users (students and instructors) to register with a unique email and password.
- The system shall provide role-based access control (student, instructor, admin)

FR2: User Profile Management □ Users shall be able to view and update their personal details (name, email, password).

- Users shall be able to reset their password via a secure mechanism.

FR3: Course Management (Instructor)

- Instructors shall be able to create, update, and delete courses.
- Instructors shall be able to upload course materials (videos, PDFs, assignments).
- Instructors shall be able to create quizzes/assessments.

FR4: Course Enrollment (Student)

- Students shall be able to browse/search available courses.
- Students shall be able to enroll in a course.
- Students shall be able to view enrolled courses in their dashboard.

FR5: Learning Progress Tracking

□ The system shall track and display each student's progress for every enrolled course. □ Students shall be able to see completed lessons, pending tasks, and quiz scores.

FR6: Communication & Interaction

- The platform shall provide a discussion forum or Q&A section for students and instructors.
- Students shall be able to submit assignments, and instructors shall be able to grade them.

FR7: Notifications

- The system shall send notifications for new assignments, deadlines, and announcements.
- Students shall receive alerts for course updates.

3.2 Non-Functional Requirements NFR1:

Performance

- The system shall load course content within 3 seconds under normal network conditions.
- The system shall support at least 1000 concurrent users without performance degradation.

NFR2: Security

- All data transfers shall be encrypted using HTTPS/TLS.
- Only authorized users shall access their own data.

NFR3: Interactive

- The system shall provide a responsive and intuitive UI accessible on desktop, tablet, and mobile devices.
- The system shall support accessibility standards (WCAG 2.1) for differently-abled users.

NFR4: Reliability and Scalability

- The system shall be developed using a modular architecture to allow easy updates.
- The system shall support scaling up (adding more servers/databases) as the number of users grows.

4. System Models

4.1 Use Case Diagram

Actors:

- Admin
- Instructor
- Student

Use Cases:

Admin:

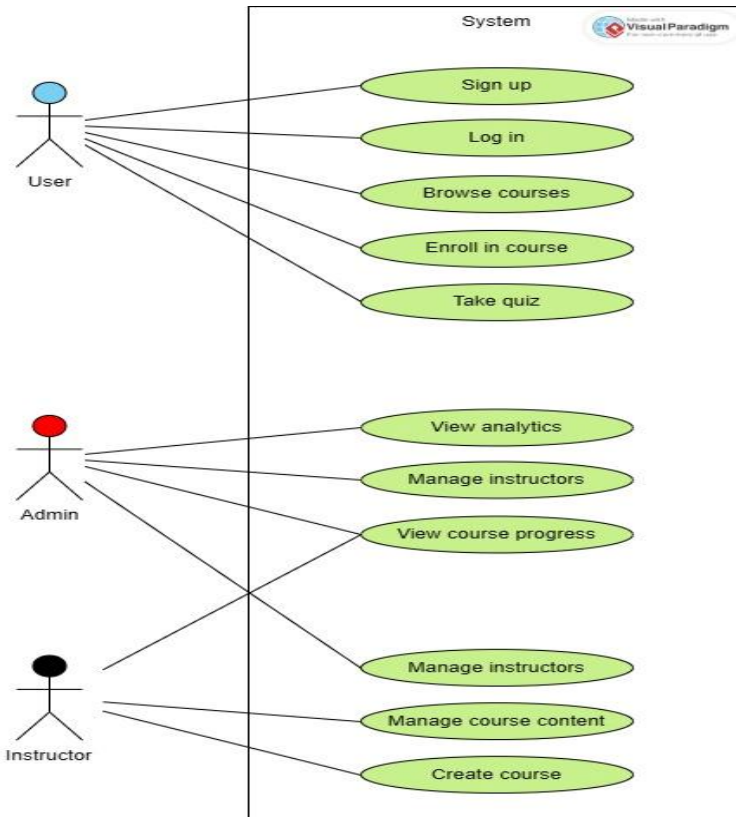
- Manage Users (Students/Faculty)
- Manage Courses
- Assign Faculty to Courses
- Monitor Reports/Logs

Instructor:

- Upload Learning Materials (PDF, Video, Assignments)
- Create & Manage Quizzes/Exams
- Evaluate Submissions
- View Student Performance

Student:

- Register/Login to Platform
- Enroll in Courses
- Access Course Content (Notes, Videos, Assignments)
- Attempt Quizzes/Exams
- View Results & Feedback

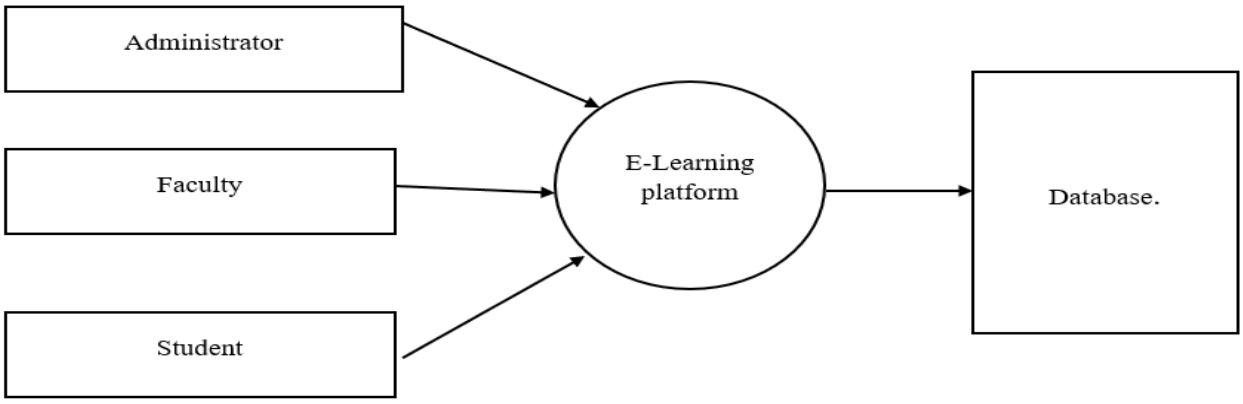


USE CASE DIAGRAM.

4.2 Data Flow Diagram (DFD)

Level 0:

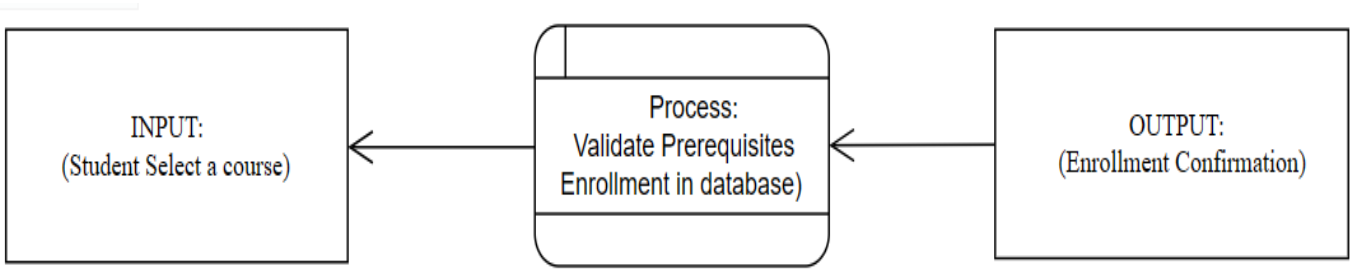
 Users (Admin, Faculty, Student) → E-learning Platform → Database.



Level 1 (Example – Course Enrollment)

- Input: Student selects a course to enroll
- Process: System validates prerequisites & stores enrollment details in DB

 Output: Confirmation of successful enrollment



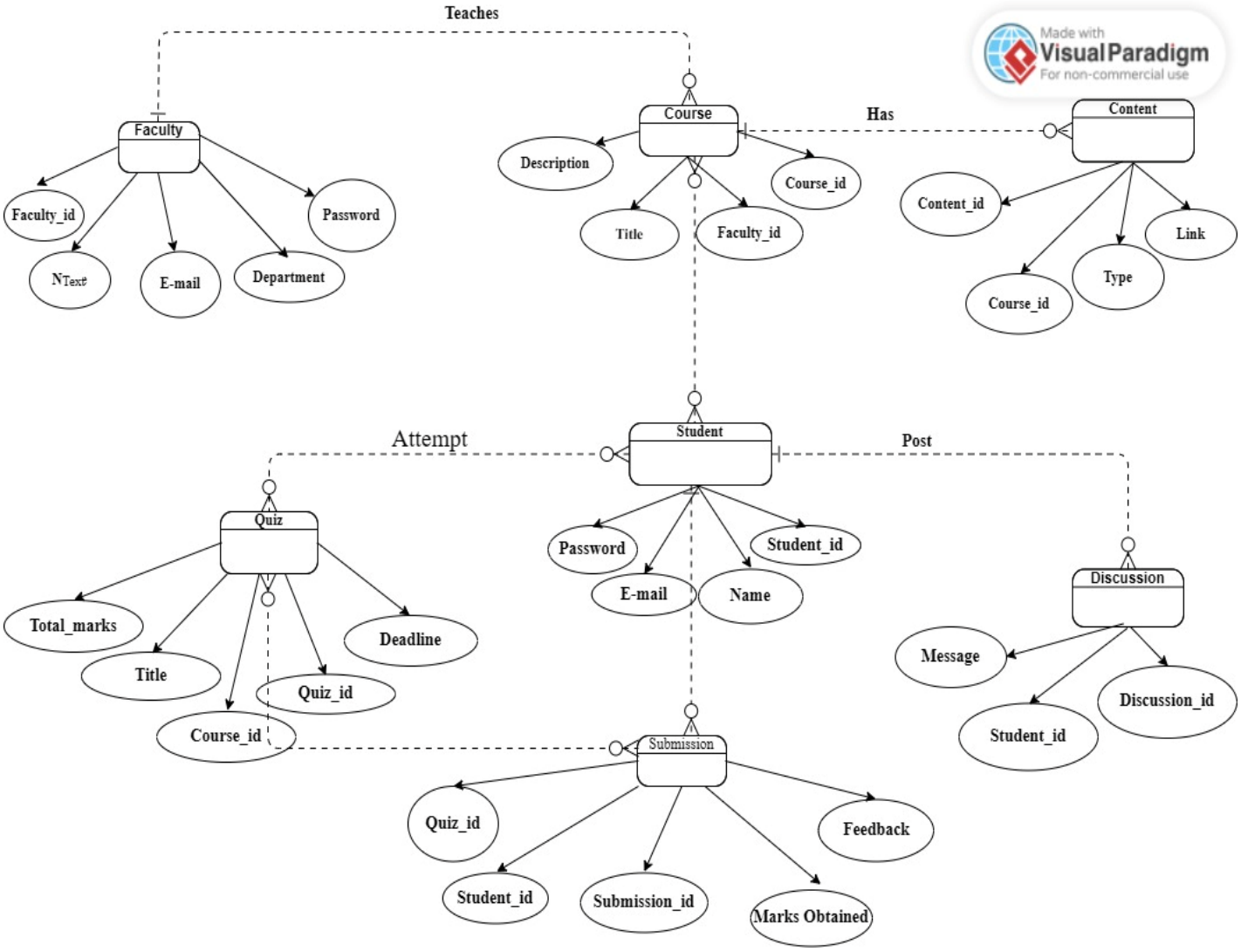
4.3 ER Diagram Entities & Relationship :

Entity

- Student (Student_ID, Name, Email, Password, Enrolled_Courses)
- Faculty (Faculty_ID, Name, Email, Password, Department)
- Course (Course_ID, Title, Description, Faculty_ID)
- Content (Content_ID, Course_ID, Type [PDF/Video/Assignment], File_Link)
- Quiz (Quiz_ID, Course_ID, Title, Total_Marks, Deadline)
- Submission (Submission_ID, Student_ID, Quiz_ID, Marks_Obtained, Feedback)

Relationships:

- Student ↔ Course (Many-to-Many)
- Faculty ↔ Course (One-to-Many)
- Course ↔ Content (One-to-Many)
- Student ↔ Quiz (Many-to-Many through Submissions)
- Student ↔ Discussion (One-to-Many)



5. Data Dictionary

5.1 User Table

Attribute	Type	Description
User_ID	INT (PK)	Unique identifier for each user (auto-generated).
Username	VARCHAR	Unique login name of the user.
Email	VARCHAR	User's registered email for communication.
Password	VARCHAR	Hashed password for secure authentication.
Role	ENUM(Admin/User)	Defines whether the user is an **Admin** or a **Student**.

5.2 Course Table

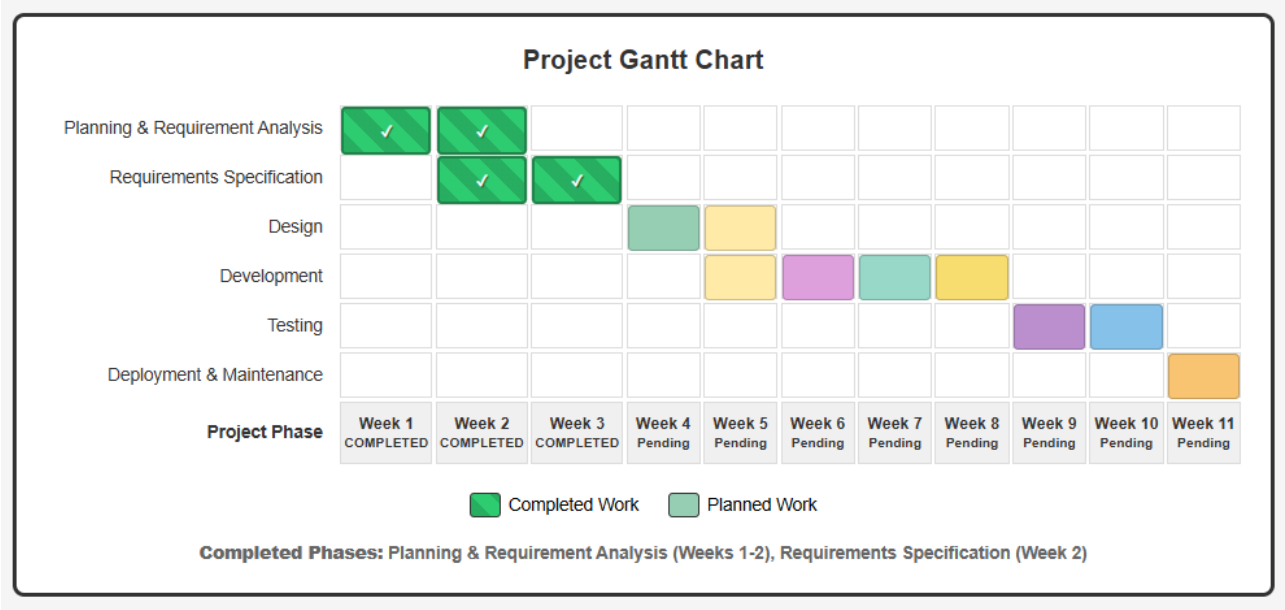
Attribute	Type	Description
Course_ID	INT (PK)	Unique identifier for each course.
Title	Varchar	Course Name.
Faculty_ID	INT	ID of the teacher that provided the course.

5.3 Enrollment

Attribute	Type	Description
Enroll_ID	INT (PK)	Unique identifier.
Student_ID	FK→USER	Student that enrolled that course .
Course_ID	FK→Course	Course that was enrolled.

6. Appendices

Gantt Chart: Development Schedule



Hardware Requirements

- Server: Quad-core processor, 16 GB RAM, 500 GB SSD, Linux/Windows OS
- Client PCs: Dual-core processor, 4 GB RAM, 100 GB HDD, Windows/Linux/Mac
- Networking: Stable LAN/Wi-Fi, Internet (10+ Mbps), Router/Switch

Test Cases

- Verify user registration with valid/invalid inputs.
- Check unique login for students, teachers, and admin.
- Validate course creation by teachers/admin.
- Verify student enrollment in courses.
- Test uploading assignments (valid file types, large file sizes).

Expected Outcome:

The expected outcome of the Moodle-based E-Learning System is to provide a reliable, secure, and user friendly platform for online education. Students will be able to seamlessly register, access courses, submit assignments, participate in quizzes, and communicate through forums. Faculty members will effectively manage courses, upload study materials, evaluate assignments, and track student performance. The system ensures smooth administration by enabling role-based access control, data security, and course backups. Overall, Moodle will enhance teaching and learning efficiency while offering flexibility, accessibility, and scalability for both students and educators