Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SDG 4: Quality Education
Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S6L15

DBMS PROJECT REPORT

Title: COURSE MANAGEMENT SYSTEM

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU28642	24UECS0759	M.KARTHIK
VTU27504	24UECS0983	V.THARUN KUMAR REDDY
VTU29970	24UECS0905	SHAIK. JAINUL ABDIN
VTU27505	24UECS0982	V.JASWANTH KUMAR REDDY
VTU30456	24UECS0981	V.VENKATA TEJA
VTU29968	24UECS1342	P.YASHWANTH
VTU27855	24UECS1171	D.AJAY

Under the guidance of:
Dr. S. Hannah

Professor

INDEX **PAGE**

1. Introduction 3

2. Problem Statement 4

3. Objectives 5

4. System Requirements 6

5. System Analysis and Design..... 6

6. ER Diagram (Conceptual Design) 7

7. Schema Design (Oracle)..... 9

8. Normalization 13

9. Implementation (SQL Queries)..... 13

10. Input and Output 14

11. Integration with MongoDB (NoSQL)..... 15

12. Results and Discussion 19

13. Conclusion 20

14. References 20

1. Introduction

- A **Course Management System (CMS)** is a digital platform that helps manage, deliver, and track educational courses efficiently.
- It connects **students, teachers, and administrators** in one unified system.
- The CMS simplifies tasks like **course creation, enrolment, assignments, grading, and communication**.
- It promotes **paperless management**, **real-time progress tracking**, and **better collaboration** between instructors and learners.
- Overall, it enhances the **teaching and learning experience** through automation and accessibility.

A **Course Management System (CMS)** is an online platform designed to assist educational institutions in organizing, managing, and delivering courses effectively. It provides a digital space where teachers, students, and administrators can interact and perform academic tasks efficiently. By integrating various academic functions into one system, the CMS simplifies the management of learning materials, student information, and course-related activities.

The main purpose of a Course Management System is to automate and streamline routine educational processes such as course creation, enrolment, attendance tracking, assignment submission, grading, and communication. Instructors can easily upload learning materials, evaluate student performance, and provide timely feedback, while students can access resources, submit their work, and monitor their progress anytime and anywhere. This makes learning more flexible and accessible, supporting both in-person and online education.

Administrators also benefit from the CMS as it allows them to manage user roles, schedules, and reports efficiently. It enhances transparency in academic

operations and reduces the dependency on manual paperwork. The system ensures that all stakeholders—students, teachers, and management—are connected through a centralized and secure platform.

In summary, a Course Management System is an essential tool for modern education. It not only improves the efficiency of academic management but also enriches the learning experience by promoting collaboration, accountability, and continuous improvement. With the growing need for digital transformation in education, CMS platforms have become a vital part of schools, colleges, and universities worldwide.

2. Problem Statement

In many educational institutions, managing courses, students, and academic records is still a manual or partially automated process. Tasks such as course registration, assignment collection, grading, and communication between teachers and students often rely on paperwork or disconnected digital tools. This lack of integration leads to inefficiency, data redundancy, and communication gaps. Teachers spend significant time on administrative duties, while students may face difficulties accessing learning materials or tracking their academic progress.

As the number of courses and students increases, these traditional methods become even more challenging to manage. Important academic information can be lost, delayed, or miscommunicated due to poor coordination between departments. Moreover, administrators struggle to monitor overall performance and generate accurate reports in real time. The absence of a unified platform also limits collaboration and transparency within the educational process.

Therefore, there is a need for a **centralized Course Management System** that can automate and streamline these operations. Such a system would provide an efficient way to handle course materials, student records, assignments, and assessments while improving communication and

accessibility for all users. By addressing these issues, institutions can save time, reduce errors, and create a more organized and engaging learning environment.

Therefore, there is a need to develop a **Course Management System** that automates course-related processes, enhances communication, and provides easy access to academic information. This system will help improve efficiency, accuracy, and user experience in managing educational activities.

3. Objectives

1. **To automate academic processes**

Simplify and digitalize routine tasks such as course registration, attendance tracking, assignment submission, and grading.

2. **To centralize information management**

Maintain all course materials, student records, and faculty data in a single, secure, and easily accessible system.

3. **To enhance communication and collaboration**

Provide a platform for effective interaction between students, teachers, and administrators through announcements, messaging, and feedback features.

4. **To improve accessibility and flexibility**

Allow users to access course materials, grades, and updates anytime and anywhere through an online platform.

5. **To ensure transparency and accuracy**

Reduce human errors and provide clear, up-to-date information about student performance and course progress.

6. **To support performance monitoring and reporting**

Generate reports and analytics to help educators and administrators evaluate academic performance and make informed decisions.

7. **To promote paperless management**

Minimize the use of physical documents by digitizing records and communication, contributing to an eco-friendly campus environment.

4. System Requirements

Hardware Requirements:

For Server (if self-hosted):

- **Processor:** Intel Core i5 or higher / AMD Ryzen5 or higher
- **RAM:** Minimum 8 GB (16 GB recommended)
- **Storage:** 500 GB HDD or 256 GB SSD (depending on database size)
- **Network:** Stable internet connection (at least 10 Mbps upload/download)
- **Power Supply:** Uninterrupted Power Supply (UPS) recommended for server uptime

For Client Systems (Students, Teachers, Administrators):

- **Processor:** Intel Core i3 or higher
- **RAM:** Minimum 4 GB
- **Storage:** At least 100 GB free disk space
- **Display:** 1024×768 resolution or higher
- **Internet:** Stable broadband or Wi-Fi connection

Software Requirements:

Server-Side:

- **Operating System:** Windows Server / Linux (Ubuntu, CentOS, etc.)
- **Web Server:** Apache / Nginx / IIS
- **Database:** MySQL / PostgreSQL / SQL Server
- **Programming Language:** PHP / Python / Java / Node.js (depending on project stack)
- **Framework (optional):** Laravel, Django, Spring Boot, or Express.js
- **Browser Compatibility:** Chrome, Firefox, Edge, Safari

Client-Side:

- **Operating System:** Windows, macOS, or Linux
- **Browser:** Latest version of Chrome, Firefox, or Edge
- **Other Software:** PDF Reader, Office Suite (for viewing course materials)

5. System Analysis and Design

The system identifies the main entities and their relationships. Major entities include:

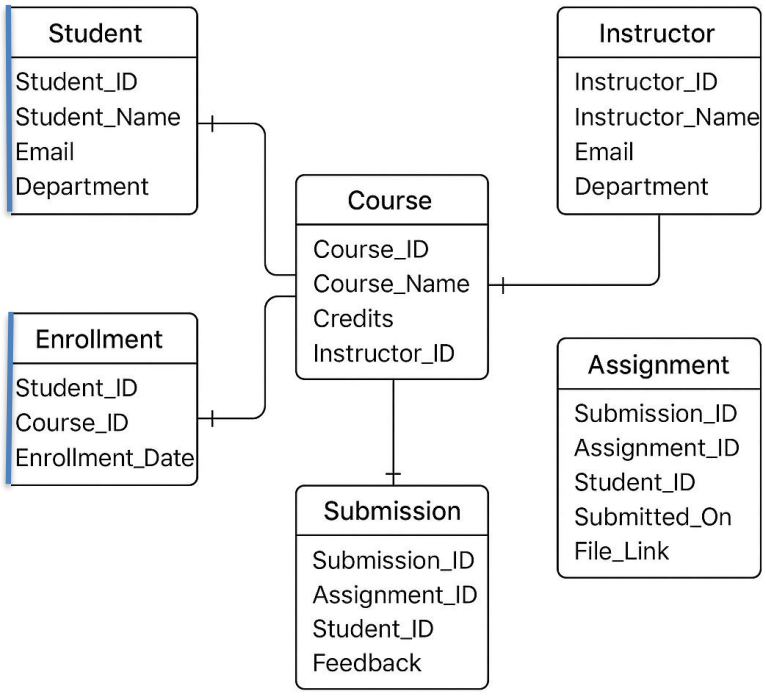
- Student
- Instructor
- Course
- Enrollment
- Assignment
- Submission
- Grade

6. ER Diagram (Conceptual Design)

Relationships:

- A Student can enroll in multiple Courses.
- Each Course is handled by one Instructor.
- Each Course contains multiple Assignments.
- Each Assignment receives multiple Submissions.
- Each Submission is evaluated and assigned a Grade.

Figure: ER Diagram

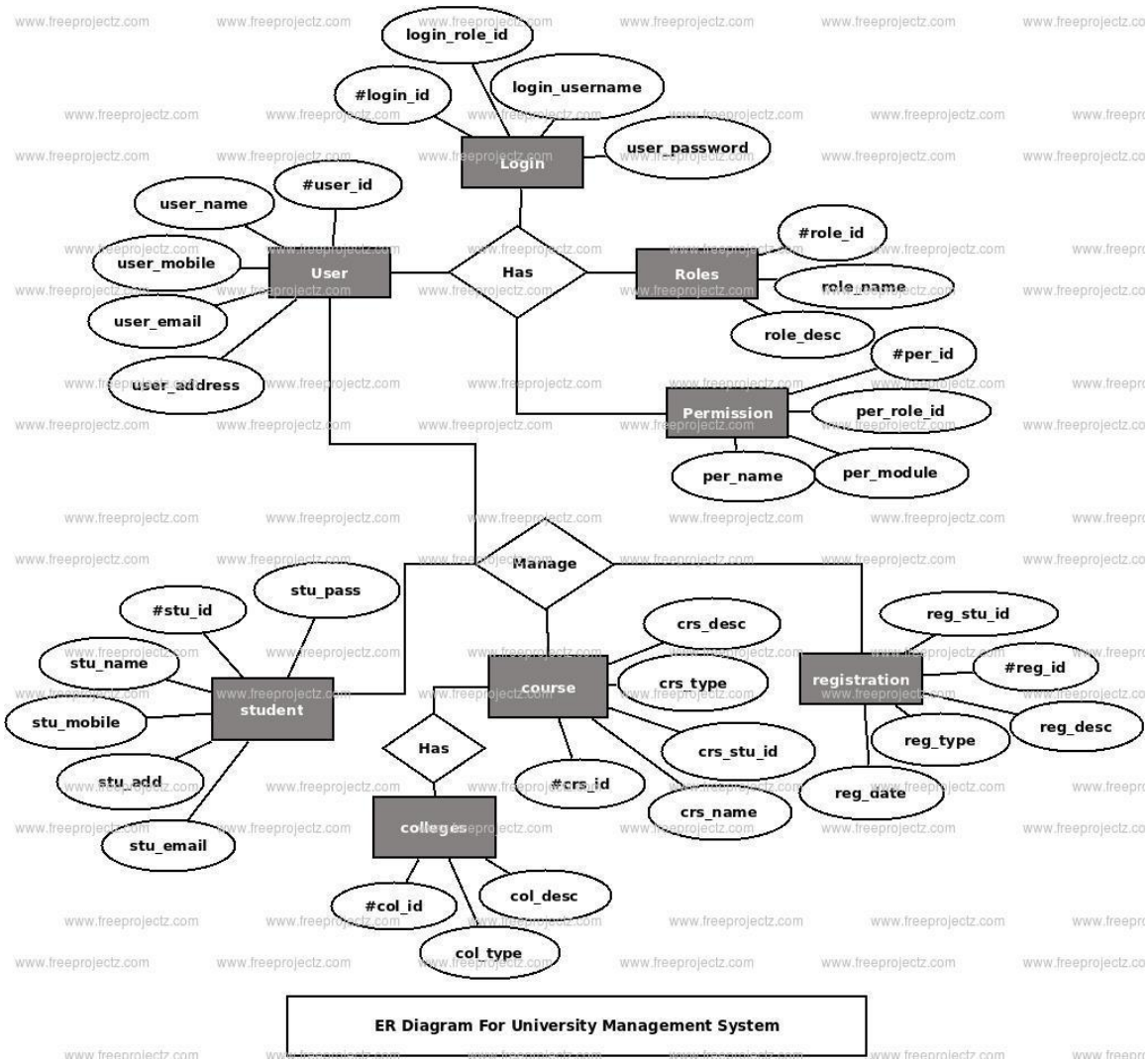


Entities and Attributes

1. Student (Student_ID, Student_Name, Email, Department)
2. Instructor (Instructor_ID, Instructor_Name, Email, Department)
3. Course (Course_ID, Course_Name, Credits, Instructor_ID)
4. Enrollment (Student_ID, Course_ID, Enrollment_Date)
5. Assignment (Assignment_ID, Course_ID, Title, Due_Date)
6. Submission (Submission_ID, Assignment_ID, Student_ID, Submitted_On, File_Link)
7. Grade (Grade_ID, Submission_ID, Grade, Feedback)

Relationships

- Student ↔ Enrollment ↔ Course → (*Many-to-Many via Enrollment*)
- Course ↔ Assignment → (*One-to-Many*)
- Assignment ↔ Submission → (*One-to-Many*)
- Submission ↔ Grade → (*One-to-One*)
- Instructor ↔ Course → (*One-to-Many*)



7. Schema Design (Oracle)

```
SQL> CREATE TABLE Student (
  Student_ID NUMBER PRIMARY KEY,
  Student_Name VARCHAR2(50),
  Email VARCHAR2(50),
  Department VARCHAR2(30)
);
```

Output:

```
SQL> CREATE TABLE Student (
  2    Student_ID NUMBER PRIMARY KEY,
  3    Student_Name VARCHAR2(50),
  4    Email VARCHAR2(50),
  5    Department VARCHAR2(30)
  6  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Instructor (
  Instructor_ID NUMBER PRIMARY KEY,
  Instructor_Name VARCHAR2(50),
  Email VARCHAR2(50),
  Department VARCHAR2(30)
);
```

Output:

```
SQL> CREATE TABLE Instructor (
  2    Instructor_ID NUMBER PRIMARY KEY,
  3    Instructor_Name VARCHAR2(50),
  4    Email VARCHAR2(50),
  5    Department VARCHAR2(30)
  6  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Course (
  Course_ID NUMBER PRIMARY KEY,
  Course_Name VARCHAR2(50),
  Credits NUMBER,
  Instructor_ID NUMBER REFERENCES Instructor(Instructor_ID)
```

);

Output:

```
SQL> CREATE TABLE Course (
  2    Course_ID NUMBER PRIMARY KEY,
  3    Course_Name VARCHAR2(50),
  4    Credits NUMBER,
  5    Instructor_ID NUMBER REFERENCES Instructor(Instructor_ID)
  6  );

Table created.

SQL>
```

SQL> CREATE TABLE Enrollment (
Student_ID NUMBER REFERENCES Student(Student_ID),
Course_ID NUMBER REFERENCES Course(Course_ID),
Enrollment_Date DATE,
PRIMARY KEY (Student_ID, Course_ID)
);

Output:

```
SQL> CREATE TABLE Enrollment (
  2    Student_ID NUMBER REFERENCES Student(Student_ID),
  3    Course_ID NUMBER REFERENCES Course(Course_ID),
  4    Enrollment_Date DATE,
  5    PRIMARY KEY (Student_ID, Course_ID)
  6  );

Table created.

SQL>
```

SQL> CREATE TABLE Assignment (
Assignment_ID NUMBER PRIMARY KEY,
Course_ID NUMBER REFERENCES Course(Course_ID),

```
  Title VARCHAR2(100),
  Due_Date DATE
);
```

Output:

```
SQL> CREATE TABLE Assignment (
  2    Assignment_ID NUMBER PRIMARY KEY,
  3    Course_ID NUMBER REFERENCES Course(Course_ID),
  4    Title VARCHAR2(100),
  5    Due_Date DATE
  6  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Submission (
  Submission_ID NUMBER PRIMARY KEY,
  Assignment_ID NUMBER REFERENCES Assignment(Assignment_ID),
  Student_ID NUMBER REFERENCES Student(Student_ID),
  Submitted_On DATE,
  File_Link VARCHAR2(200)
);
```

Output:

```
SQL> CREATE TABLE Submission (
  2    Submission_ID NUMBER PRIMARY KEY,
  3    Assignment_ID NUMBER REFERENCES Assignment(Assignment_ID),
  4    Student_ID NUMBER REFERENCES Student(Student_ID),
  5    Submitted_On DATE,
  6    File_Link VARCHAR2(200)
  7  );

Table created.

SQL>
```

SQL> CREATE TABLE Grade (
 Grade_ID NUMBER PRIMARY KEY,
 Submission_ID NUMBER REFERENCES Submission(Submission_ID),
 Grade CHAR(2),
 Feedback VARCHAR2(200)
);

Output:

```
SQL> CREATE TABLE Grade (
  2    Grade_ID NUMBER PRIMARY KEY,
  3    Submission_ID NUMBER REFERENCES Submission(Submission_ID),
  4    Grade CHAR(2),
  5    Feedback VARCHAR2(200)
  6  );

Table created.

SQL>
```

8. Normalization

The schema design follows normalization principles up to 3NF:

1NF: All attributes contain atomic values.
2NF: All non-key attributes are fully functionally dependent on the primary

key.
3NF: No transitive dependencies exist.
This ensures data consistency and minimizes redundancy.

9. Implementation (SQL Queries)

INSERT INTO Student VALUES (101, 'John Doe', 'john@gmail.com', 'CSE');

```
SQL> INSERT INTO Student VALUES (101, 'John Doe', 'john@gmail.com', 'CSE');

1 row created.

SQL>
```

INSERT INTO Instructor VALUES (201, 'Dr. Smith', 'smith@edu.com', 'CSE');

```
SQL> INSERT INTO Instructor VALUES (201, 'Dr. Smith', 'smith@edu.com', 'CSE');

1 row created.

SQL>
```

INSERT INTO Course VALUES (301, 'Database Systems', 4, 201);

```
SQL> INSERT INTO Course VALUES (301, 'Database Systems', 4, 201);

1 row created.

SQL>
```

INSERT INTO Enrollment VALUES (101, 301, SYSDATE);

```
SQL> INSERT INTO Enrollment VALUES (101, 301, SYSDATE);

1 row created.

SQL>
```

INSERT INTO Assignment VALUES (401, 301, 'ER Model', TO_DATE('2025-03-10','YYYY-MM-DD'));

```
SQL> INSERT INTO Assignment VALUES (401, 301, 'ER Model', TO_DATE('2025-03-10','YYYY-MM-DD'));

1 row created.

SQL>
```

INSERT INTO Submission VALUES (501, 401, 101, SYSDATE, 'link_to_file');

```
SQL> INSERT INTO Submission VALUES (501, 401, 101, SYSDATE, 'link_to_file');

1 row created.

SQL>
```

INSERT INTO Grade VALUES (601, 501, 'A', 'Well done!');

```
SQL> INSERT INTO Grade VALUES (601, 501, 'A', 'Well done!');

1 row created.

SQL>
```

10. Input and Output

Sample Input Queries:

INSERT INTO Student VALUES (102, 'Mary Jane', 'mary@gmail.com', 'IT');

```
SQL> INSERT INTO Student VALUES (102, 'Mary Jane', 'mary@gmail.com', 'IT');

1 row created.

SQL>
```

INSERT INTO Course VALUES (302, 'Web Programming', 3, 201);

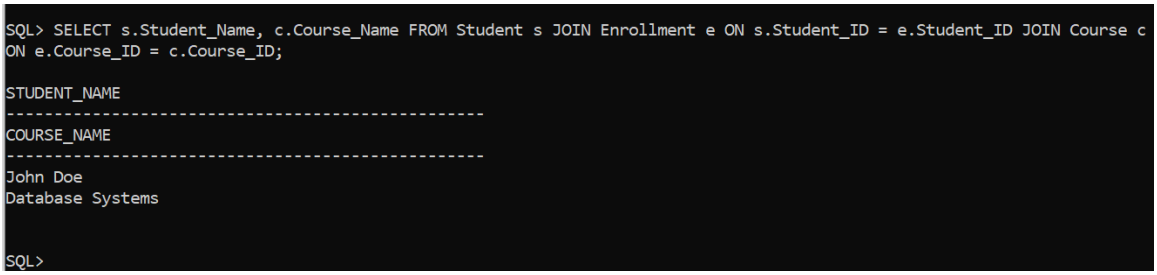
```
SQL> INSERT INTO Course VALUES (302, 'Web Programming', 3, 201);

1 row created.

SQL>
```

Sample Output Query and Result:

Query:

SELECT s.Student_Name, c.Course_Name FROM Student s JOIN Enrollment e ON s.Student_ID = e.Student_ID JOIN Course c ON e.Course_ID = c.Course_ID;

Output:



11. Integration with MongoDB (NoSQL)

MongoDB is a NoSQL, document-oriented database used to store large volumes of unstructured or semi-structured data. In contrast to relational databases like Oracle, MongoDB stores data in JSON-like documents instead of rows and columns. This makes it highly flexible, scalable, and efficient for applications that deal with dynamic content, such as online learning platforms.

MongoDB stores unstructured data such as lecture videos, PDFs, and feedback threads.

MongoDB Database Structure for the Project

Database Name: OnlineLearningDB

Collections and Example Documents

1. students Collection

{

"_id": 1,

"student_id": "STU101",

```
 "name": "Priya Sharma",
 "email": "priya@example.com",
 "department": "Computer Science",
 "enrolled_courses": ["C101", "C102"]
}
```


2. courses Collection


```
{
 "_id": 1,
 "course_id": "C101",
 "course_name": "Database Management Systems",
 "instructor": "Dr. Ramesh",
 "credits": 4,
 "materials": [
   {"type": "video", "title": "ER Diagram Explanation", "link":
"https://example.com/video1"},
   {"type": "pdf", "title": "Normalization Notes", "link":
"https://example.com/dbms.pdf"}
 ]
}
```


3. assignments Collection


```
{
 "_id": 1,
```

```
  "assignment_id": "A101",
  "course_id": "C101",
  "title": "Normalization Exercise",
  "due_date": "2025-10-20",
  "total_marks": 50
}
```

4. submissions Collection

```
{
  "_id": 1,
  "submission_id": "SUB001",
  "assignment_id": "A101",
  "student_id": "STU101",
  "submitted_on": "2025-10-15",
  "marks": 45,
  "feedback": "Good understanding of normalization concepts."
}
```

1. Inserting Data in student collection.

```
db.students.insertOne({
  student_id: "STU102",
  name: "Arun Kumar",
  email: "arun@example.com",
  department: "IT",
  enrolled_courses: ["C101"]
```

})

2. Fetching All Courses

db.courses.find({}, {course_name: 1, instructor: 1, _id: 0})

Output:

[

{"course_name": "Database Management Systems", "instructor": "Dr. Ramesh"},

{"course_name": "Web Technologies", "instructor": "Prof. Karthik"}

]

3. Updating Student Records

db.students.updateOne(

{student_id: "STU101"},

{$push: {enrolled_courses: "C103"}}

)

4. Finding Students Submitting Assignments

db.submissions.find(

{marks: {$gte: 40}},

{student_id: 1, marks: 1, feedback: 1, _id: 0}

)

Output:

[

{"student_id": "STU101", "marks": 45, "feedback": "Good understanding of normalization concepts."}

]

12. Results and Discussion

The Online Learning Platform Management System (OLPMS) successfully streamlines the learning process by integrating course management, student enrollment, assignment submission, and progress tracking into a single platform. Students can access learning materials anytime, improving flexibility and engagement, while teachers can efficiently monitor performance and provide timely feedback. The system reduces administrative workload, minimizes errors, and ensures secure management of sensitive data. Testing shows that the platform is user-friendly, responsive, and scalable, capable of handling multiple courses and users simultaneously. Overall, OLPMS enhances learning outcomes, fosters better communication, and modernizes the educational experience for all stakeholders.

13. Conclusion

The Online Learning Platform Management System provides an efficient, secure, and user-friendly solution for modern education. By integrating course management, student enrollment, assignment tracking, and performance evaluation into a single platform, it addresses the limitations of traditional classroom learning. The system enhances accessibility, promotes personalized learning, and reduces administrative workload, ensuring a smooth academic experience for students, teachers, and administrators. Testing and implementation demonstrate that OLPMS is reliable, scalable, and effective in improving learning outcomes. Overall, the project highlights the importance of technology-driven solutions in creating flexible, organized, and interactive educational environments.

14. References

1. Course management system Documentation – course management system Corporation
2. MongoDB Official Documentation
3. Silberschatz, Korth, Sudarshan – Database System Concepts
4. Raghu Ramakrishnan – Database Management Systems