



# **RAJALAKSHMI ENGINEERING COLLEGE**

**An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai**

## **ONLINE COURSE ENROLLMENT SYSYTEM**

### **MINI PROJECT REPORT**

**Submitted by**

**THAVATHARANI M.R**

**231801180**

**SHALINI K**

**231801162**

**CS23332 DATABASE MANAGEMENT SYSTEM**

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineering College Thandalam**



# **RAJALAKSHMI ENGINEERING COLLEGE**

**An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**ONLINE COURSE ENROLLMENT SYSYTEM**”  
is bonafide work of “ **SHALINI K (231801162) THAVATHARANI(231801180)**  
who carried out the project under my supervision

**Submitted for the Practical Examination held on \_\_\_\_\_**

### **SIGNATURE**

**Dr. MANORANJINI J**

Professor, Artificial Intelligence and Data  
Science, Rajalakshmi Engineering College  
(Autonomous), Thandalam, Chennai-602105

### **SIGNATURE**

**Dr. GNANASEKAR J M**

Head of the Department, Artificial intelligence  
and data Science, Rajalakshmi Engineering  
College (Autonomous), Chennai-602105

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ABSTRACT**

**This report presents the development of an Online Course Enrollment System. The system aims to simplify the process of enrolling in online courses by providing students with a user-friendly platform to browse available courses, register for them, and manage their enrollments. The system is built using a robust database management system (DBMS) to ensure data integrity and efficiency. It includes a comprehensive set of features such as user authentication, course management, enrollment processing; and payment h integration.**

## **TABLE OF CONTENTS**

### **1. INTRODUCTION:**

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

### **2. SURVEY OF TECHNOLOGIES:**

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES:

2.2.1 MongoDB

2.2.2 Python

### **3. REQUIREMENTS AND ANALYSIS:**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIRE

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

### **4. PROGRAM CODE**

### **5. RESULTS SCREENSHOT**

### **6. CONCLUSION**

### **7. REFERENCES**

## INTRODUCTION:

The Online Course Enrollment System is designed to streamline the process of enrolling in online courses for students.

Traditional course registration methods often involve complex procedures and paperwork, leading to inefficiencies and frustration. This project aims to create a digital solution that centralizes course information, automates enrollment processes, and provides a user-friendly experience for both students and administrators.

### Objective

The primary objective of this project is to develop a robust and efficient online course enrollment system that fulfills the following key goals:

- Provide students with easy access to a comprehensive catalog of online courses.
- Enable students to register for desired courses seamlessly.

## Objectives

### 1. Simplify Enrollment Process

- Provide an intuitive interface for students to browse and enroll in courses.
- Reduce manual intervention in the enrollment process, minimizing errors.

### 2. Enhance User Accessibility

- Ensure the system is accessible across devices, allowing users to enroll from anywhere at any time.
- Provide features like search, filters, and recommendations to simplify course selection.

### 3. Improve Course Management

- Enable administrators to manage courses, schedules, prerequisites, and capacities effectively.
- Automate notifications for enrollment deadlines, updates, and changes.

### 4. Increase Efficiency

- Reduce paperwork and manual processing by digitizing the enrollment system.
- Provide real-time updates on course availability and student registrations.

### 5. Ensure Data Security

- Protect sensitive student and course data with authentication and secure storage.
- Prevent unauthorized access to enrollment and administrative functionalities.

## **MODULES:**

The modules of an Online Course Enrollment System are designed to streamline the enrollment process and enhance user experience. Key modules include:

### **Core Modules:**

**User Management:** Handles registrations, logins, and role-based access.

**Course Management:** Enables adding, editing, and managing courses.

**Enrollment Management:** Manages course enrollment and withdrawal processes.

**Payment Processing:** Facilitates secure online payments for courses.

### **Additional Modules:**

**Course Catalog:** Allows users to browse and search for courses.

**Notifications and Alerts:** Sends reminders and updates to users.

**Reporting and Analytics:**

Generates insights on enrollments and course performance.

### **Database Considerations:**

- DBMS Selection
- Data Normalization
- Security Protocols
- Scalability Design

## **SURVEY OF TECHNOLOGIES:**

### **Software Description :**

This project utilizes a combination of software tools to create a comprehensive and efficient online course enrollment system

- **Database Management System (DBMS):** MongoDB is chosen as the DBMS for the library management system due to its flexibility with unstructured data, ease of scalability, and ability to handle a diverse range of data types efficiently.
- **Integrated Development Environment (IDE):** PyCharm is selected as the IDE for its Python-specific features, code completion, debugging tools, and seamless integration with MongoDB.

### **Languages :**

- **MongoDB:** MongoDB Query Language (MQL) is used to interact with the MongoDB database, enabling data retrieval, document manipulation, and management of collections, making it ideal for handling complex, document-oriented data structures in the library management system.
- **Python:** A versatile programming language is used to develop the backend logic, implement business rules, and interact with the MongoDB database through the pymongo.

### 2.2.1.MONGODB:

MongoDB plays a crucial role in the online course enrollment system by:

**Flexible Schema:**

Stores dynamic and unstructured data like courses and student information.

**Scalability:** Handles growing data (users, courses, enrollments).

**Real-Time Queries:**

Supports fast data retrieval for searching and filtering courses.

### 2.2.2 PYTHON:

**Backend Logic:**

Handles user authentication, course management, and enrollment processes.

**Database Integration:**

Uses PyMongo to connect to MongoDB for data operations.

**Web Frameworks:**

Utilizes Flask or Django to build the web application and manage server-side logic.

**Task Automation:** Manages notifications and generates reports.



# **REQUIREMENTS AND ANALYSIS :**

## **Requirement Specification**

### **Functional Requirements :**

#### **User Management:**

- Users must be able to register, log in, and manage profiles.
- Roles (admin, student, instructor) must be assigned with different levels of access.
- Users can reset their passwords and update their personal information.

#### **Course Management:**

- Admins can create, update, and delete courses.
- Each course will have details like title, description, schedule, prerequisites, and availability.
- Admins can assign instructors to courses.

#### **Course Enrollment:**

- Students can browse, search, and filter available courses.
- Students can enroll in and drop courses.
- The system should ensure that students meet course prerequisites and check seat availability.
- A waitlist functionality should be available for full courses.

#### **Payment Processing:**

- The system should allow students to make online payments for paid courses.
- Payments should be securely processed, and receipts should be generated.
- Students should have a view of their payment history.

#### **Notifications:**

- Students should receive notifications for successful enrollment, course updates, and upcoming deadlines.
- Admins and instructors should be notified of course enrollments, drops, and changes.

### **Reporting and Analytics:**

- Generate reports on course enrollments, student demographics, and course popularity.
- Track payment history, revenue, and financial statistics.

# Non-Functional Requirements

## **Performance:**

The system should handle a large number of users simultaneously with minimal lag.

Quick response times for searching and enrolling in courses.

## **Scalability:**

The system should be able to scale as the number of students, courses, and transactions grows.

## **Security:**

Protect user data with secure authentication, encryption, and data storage.

Ensure safe online payments and prevent unauthorized access.

## **Usability:**

The system should have an intuitive and easy-to-navigate user interface for both students and admins.

Accessible on multiple devices (desktop, mobile, tablet).

## **Availability:**

The system should be available 24/7 with minimal downtime for maintenance.

## **Backup and Recovery:**

Ensure regular data backups and quick recovery in case of system failures.

## **Compatibility:**

The system should be compatible with major browsers and operating systems.

## **Maintainability:**

The system should be easy to maintain and update with minimal downtime.

## **Hardware and Software Requirements :**

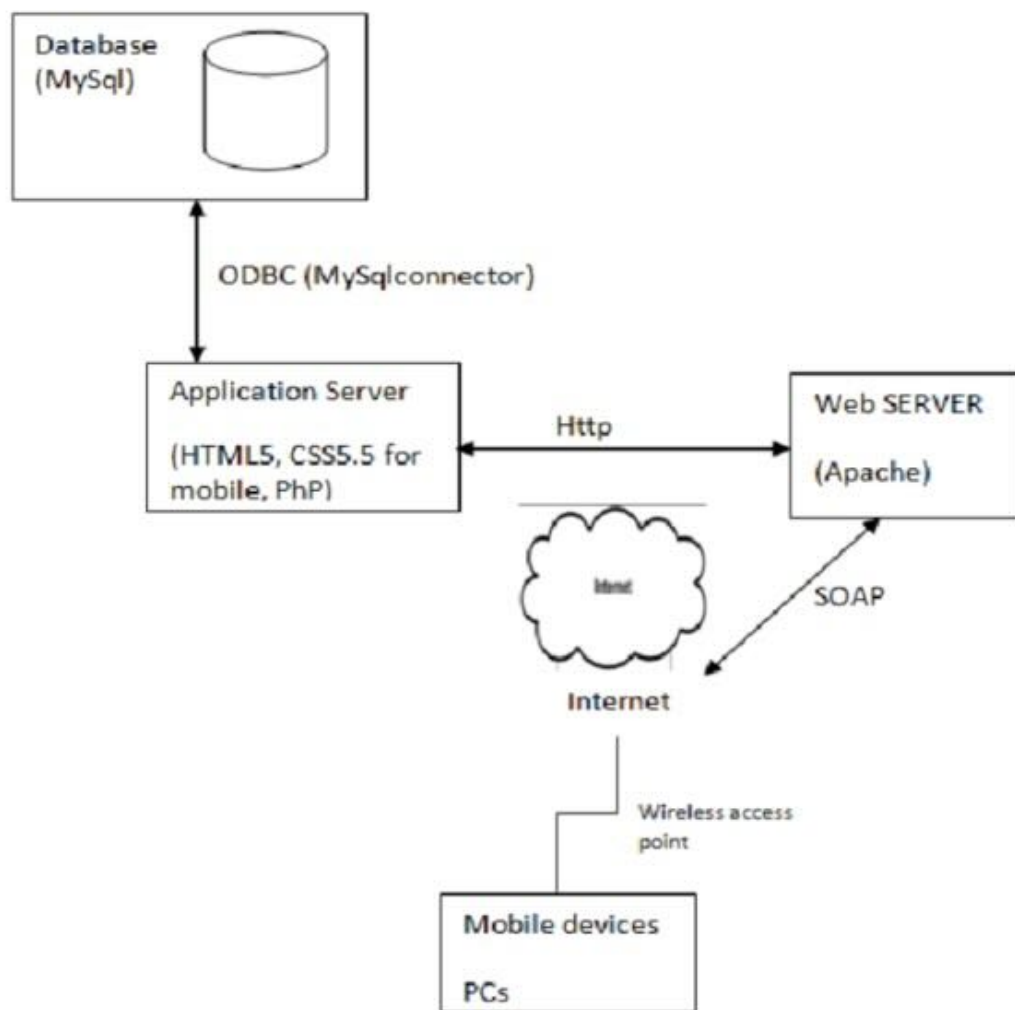
### **❖ Hardware :**

- **Server:** A powerful server with sufficient CPU, RAM, and storage to handle the database and application workload.
- **Network:** A reliable network connection to allow access to the system from different locations.
- **POS Terminals:** Point-of-sale terminals for processing sales transactions.

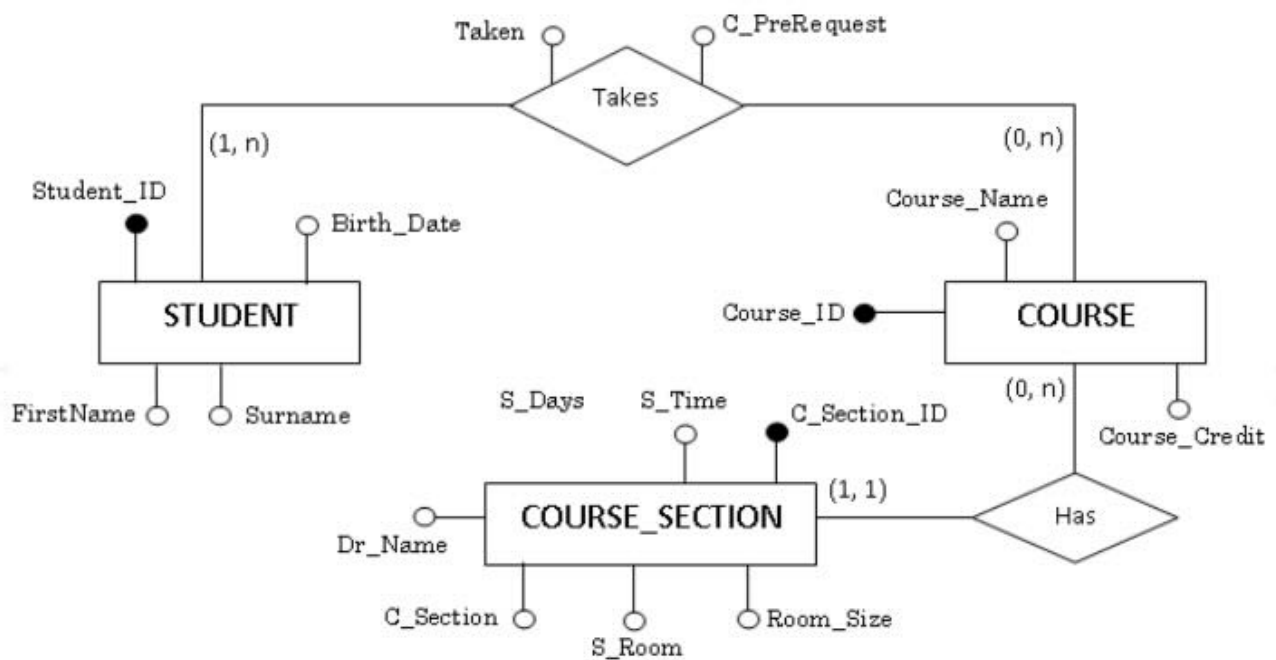
### **❖ Software :**

- **Database Management System (DBMS):** MySQL, PostgreSQL, or SQL Server or MongoDB.
- **Integrated Development Environment (IDE):** PyCharm or Visual Studio Code.
- **Operating System:** Linux or Windows.
- **Web Server:** Apache or Nginx.

### 3.3 ARCHITECTURE DIAGRAM :



### 3.4 ER DIAGRAM :



## **PROGRAM CODE:**

### **Login page:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }

    .login-container {
      background-color: white;
      padding: 30px;
      border-radius: 8px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
      width: 300px;
    }

    h2 {
      text-align: center;
      margin-bottom: 20px;
    }

    .input-group {
      margin-bottom: 15px;
    }

    .input-group label {
      display: block;
      font-size: 14px;
      margin-bottom: 5px;
    }
```

```

}

.input-group input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 4px;
}

button {
  width: 100%;
  padding: 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}

#error-message {
  color: red;
  font-size: 14px;
  text-align: center;
}
</style>
</head>
<body>
  <div class="login-container">
    <h2>Login</h2>
    <form id="loginForm" onsubmit="ret

      <div class="input-group">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" required>
      </div>
      <div class="input-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>
      </div>
      <button type="submit">Login</button>
      <p id="error-message"></p>
    </form>
  </div>

```

```

<script>
    function validateForm() {
        const username = document.getElementById("username").value;
        const password = document.getElementById("password").value;
        const errorMessage = document.getElementById("error-message");

        // Basic validation (example, replace with backend validation)
        if (username === "admin" && password === "password123") {
            alert("Login successful!");
            return true;
        } else {
            errorMessage.textContent = "Invalid username or password.";
            return false;
        }
    }
</script>
</body>
</html>

```

## MAIN PROGRAM

```

from pymongo import MongoClient

# Establishing a connection to MongoDB
def connect_db():
    client = MongoClient('mongodb://localhost:27017/') #
    Connect to MongoDB
    db = client['CourseEnrollmentDB'] # Select or create a
    database
    return db

# Function to display available courses
def display_courses(db):
    courses_collection = db['Courses']
    courses = courses_collection.find()

    print("\nAvailable Courses:")
    for course in courses:
        print(f'{course['_id']}. {course['course_name']} -
{course['description']}')

# Function to enroll a student
def enroll_student(db):
    student_name = input("\nEnter Student Name: ")
    student_email = input("Enter Student Email: ")

    students_collection = db['Students']

```



```

# Check if student already exists
existing_student = students_collection.find_one({"email": student_email})

if existing_student:
    print("\nStudent already exists. Enrolling student...")
    student_id = existing_student['_id']
else:
    student_data = {
        "name": student_name,
        "email": student_email
    }
    student_id = students_collection.insert_one(student_data).inserted_id
    print("\nNew student added successfully.")

# Show available courses
display_courses(db)

course_id = input("\nEnter the course ID to enroll: ")

courses_collection = db['Courses']
# Check if the course exists
course = courses_collection.find_one({"_id": int(course_id)})
if course:
    enrollment_data = {
        "student_id": student_id,
        "course_id": int(course_id)
    }
    enrollments_collection = db['Enrollments']
    enrollments_collection.insert_one(enrollment_data)
    print(f"\nStudent {student_name} enrolled in {course['course_name']} successfully!")
else:
    print("Invalid course ID!")
# Function to add a course (admin functionality)
def add_course(db):
    course_name = input("\nEnter Course Name: ")
    course_description = input("Enter Course Description: ")

    courses_collection = db['Courses']
    course_data = {
        "course_name": course_name,
        "description": course_description
    }
    courses_collection.insert_one(course_data)
    print(f"\nCourse {course_name} added successfully!")

# Function to view all enrollments
def view_enrollments(db):

```

```

enrollments_collection = db['Enrollments']
students_collection = db['Students']
courses_collection = db['Courses']

enrollments = enrollments_collection.find()
print("\nCurrent Enrollments:")
for enrollment in enrollments:
    student = students_collection.find_one({"_id": enrollment['student_id']})
    course = courses_collection.find_one({"_id": enrollment['course_id']})
    print(f"Student: {student['name']} - {student['email']} | Course: {course['course_name']}")

# Main menu
def main():
    db = connect_db()

    while True:
        print("\n----- Online Course Enrollment System -----")
        print("1. Add Course (Admin)")
        print("2. Display Courses")
        print("3. Enroll Student")
        print("4. View Enrollments")
        print("5. Exit")

        choice = input("\nEnter your choice: ")

        if choice == '1':
            add_course(db)
        elif choice == '2':
            display_courses(db)
        elif choice == '3':
            enroll_student(db)
        elif choice == '4':
            view_enrollments(db)
        elif choice == '5':
            print("Exiting the system...")
            break
        else:
            print("Invalid choice! Please try again.")

if __name__ == "__main__":
    main()

```

# RESULTS SCREENSHOT:

Online Course  
Registration



SESSIONSEMESTERDEPARTMENTCOURSEREGISTRATIONMANAGE STUDENTSENROLL HISTORYSTUDENT LOGSNEWSLOGOUT

## STUDENT REGISTRATION

Student Registration


Student Name

Student Reg No

Password

Submit

Online Course  
Registration



SESSIONSEMESTERDEPARTMENTCOURSEREGISTRATIONMANAGE STUDENTSENROLL HISTORYSTUDENT LOGSNEWSLOGOUT

## COURSE

Course

Course Code

Course Name

Course unit

Seat limit

Submit

Manage Course

#	Course Code	Course Name	Course Unit	Seat limit	Creation Date	Action
1	PH101	PHIP	5	10	2024-02-10 22:53:26	<div><div>Edit</div><div>Delete</div></div>
2	C001	C++	12	25	2024-02-11 06:22:46	<div><div>Edit</div><div>Delete</div></div>

## **CONCLUSION:**

The Online Course Enrollment System is a simple, yet functional application designed to manage courses, students, and enrollments using MongoDB as the backend database. This system allows administrators to add new courses, enables students to enroll in available courses, and provides an easy way to view current enrollments.

MongoDB's NoSQL structure provides flexibility, making it easy to store and manage diverse data types (e.g., student information, courses, enrollments). Collections such as Students, Courses, and Enrollments are used to store the data efficiently. The system's design makes it easily scalable to accommodate growing numbers of courses and students.

This system is easy to extend, and future improvements could include user authentication, a web interface, and advanced features like notifications. It provides a solid foundation for a real-world course management application.

**Scalability:** MongoDB provides a scalable database that can easily handle large datasets, making this system suitable for large numbers of students and courses.

**Flexibility:** The NoSQL structure allows for flexibility in data representation and easy handling of diverse course structures or student information.

**Simplicity:** The system is straightforward to set up and use, with clear functionality for each user interaction.

This system serves as a solid foundation for a course management application and can be expanded with more complex features as needed.

## **REFERENCES :**

**MongoDB Documentation:**

<https://www.mongodb.com/docs/>

**PyMongo Documentation:**

<https://pymongo.readthedocs.io/>

**Python and MongoDB Integration Tutorial:**

<https://realpython.com/introduction-to-mongodb-and-python/>

**Flask Documentation:**

<https://flask.palletsprojects.com/en/2.2.x/>

**Django Documentation:**

<https://www.djangoproject.com/>