

STUDENT DETAILS

College Name: AALIM
MUHAMMAD SALEGH
COLLEGE OF
ENGINEERING

College Code: 1101
Name: MOHAMED IBRAHIM
S.A

Department: Electronics &
Communication Engineering
Naan Mudhalvan (NM) ID:

4c5491433317e7dbfe14cd54
55f3d94c

Phone Number: 9361505023

Email ID:

ibucool6@gmail.com

1. INTRODUCTION

The Learning Management System is a database-driven application used to manage course information efficiently

This project demonstrates CRUD operations using MongoDB

MongoDB Shell is used for database operations while Node.js Express.js and React.js are used for backend and frontend development.

2. OBJECTIVE OF THE TASK

The objective of this task is to implement CRUD operations using MongoDB and to understand database connectivity with backend and frontend applications

3. TOOLS AND TECHNOLOGIES USED

VS Code

Node.js

Express.js

MongoDB

MongoDB Shell (mongosh)

React.js

Postman

Git & GitHub

4. DATABASE DETAILS

Database Name: lmsDB

Collection Name: courses

Schema Design

Field Name	Data Type
1.title	String
2.instructor	String
3.category	String
4.publishedYear	Number
5.availableSeats	Number

STEP-BY-STEP IMPLEMENTATION PROCESS

Step 1: Environment Setup

VS Code was used as the development environment

Node.js and MongoDB were installed and verified using command-line tools

Step 2: MongoDB Connection Using mongosh

MongoDB Shell (mongosh) was opened using Command Prompt

A new database named lmsDB was created and selected

Step 3: Collection Creation

A collection named courses was created automatically during data insertion

Step 4: Insert Data

Course records were inserted into the courses collection using insertOne and insertMany commands

Step 5: Read Data

Course data was retrieved using find queries to view all courses courses by category and courses published after a specific year

Step 6: Update Data

Existing course records were updated using updateOne to modify available seats and course category details

Step 7: Delete Data

Course records were deleted using deleteOne based on conditions such as zero available seats

Step 8: Backend Integration

The backend was developed using Node.js and Express.js

MongoDB was connected using Mongoose and REST APIs were created to handle CRUD operations

Step 9: Frontend Integration

The frontend was developed using React.js.

The frontend interacted with backend APIs to perform CRUD operations and display course details

Step 10: Testing and Verification

CRUD operations were tested using MongoDB Shell MongoDB Compass Postman and browser output verification

6. TESTING

CRUD operations tested using MongoDB Shell

Database verified using MongoDB Compass

Backend APIs tested using Postman

Frontend tested using browser

7. ERROR HANDLING

Course not found handling

Negative seat count prevention

Deletion restricted if available seats are not zero

Invalid update prevention.

8. GIT AND GITHUB IMPLEMENTATION

Git repository was initialized in the project folder.

Project files were added and committed with meaningful messages.

The project was pushed to GitHub repository and maintained as public.

9. CONCLUSION

The Learning Management System project provided practical exposure to implementing CRUD operations using MongoDB

The task helped in understanding database management backend API development frontend integration and version control using GitHub

IMAGE OF CRUD OPERATION :

1.Create Database:

Creation of lmsDB database using MongoDB Shell

```
use
lmsDB

switched to db
lmsDB
```

2.Insert Course Record:

Inserting course details into the courses collection using insertOne

```
db.courses.insertMany([
  {
    title: "MongoDB Basics",
    instructor: "John Smith",
    category: "Database",
    publishedYear: 2022,
    availableSeats: 40
  },
  {
    title: "React Fundamentals",
    instructor: "Alice Brown",
    category: "Frontend",
    publishedYear: 2021,
    availableSeats: 25
  },
  {
    title: "Node.js & Express",
    instructor: "David Lee",
    category: "Backend",
    publishedYear: 2023,
    availableSeats: 0
  }
])

db.courses.insertOne({
  title: "MongoDB
Basics",
  instructor: "Ravi
Kumar",
  category:
"Database",
  publishedYear: 2022,
  availableSeats: 30
})

{
  acknowledged: true,
  insertedId:
ObjectId("65ab1234abcd1234abcd1234"
)
```

3.Read All Courses

Retrieving all course records from the database.

```
db.courses.find()

{
  _id:
ObjectId("65ab1234abcd1234abcd1234"),
  title: "MongoDB Basics",
  instructor: "Ravi Kumar",
  category: "Database",
  publishedYear: 2022,
  availableSeats: 30
}
```

4.Read All Courses (Pretty Format)

Displaying course records in readable format

```
db.courses.find().pretty()

{
  "_id" :
  ObjectId("65ab1234abcd1234abcd1234")
, "title" : "MongoDB Basics",
  "instructor" : "Ravi Kumar",
  "category" : "Database",
  "publishedYear" : 2022,
  "availableSeats" : 30
}
```

5.Read Courses by Category

Fetching course details based on category.

```
db.courses.find({
  category: "Database"
})

{
  title: "MongoDB
Basics",
  category:
  "Database"
}
```

6.Read Courses Published After 2021

Retrieving courses published after a specific year

```
db.courses.find({
  publishedYear: {
    $gt: 2021 } })

{
  title: "MongoDB
Basics",

  publishedYear: 2022
}
```

7.Read Course by Instructor

Displaying course details based on instructor

```
db.courses.find({
  instructor:
  "Ravi Kumar" })
```



```
{
  title: "MongoDB
Basics",
  instructor: "Ravi
Kumar"
}
```

8.Update Available Seats

Updating available seats of a course

```
db.courses.updateOne(
  { title: "MongoDB
Basics" },
  { $set: {
    availableSeats: 25 }
  }
)
```

```
{
  acknowledged:
true,
  matchedCount: 1,
  modifiedCount: 1
}
```

9.Delete Course

Deleting course record when available seats become zero

```
db.courses.
deleteOne({
  availableSeats: 0 })
```

```
{
  acknowledged:
true,
  deletedCount: 1
}
```