MASTERCLASS MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

DIVYA SURESH 220701069

JAYAJOTHI KUMAR 220701100

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023-24

BONAFIDE CERTIFICATE

Certified that this project report "**MASTERCLASS MANAGEMENT SYSTEM**" is the bonafide work of "**DIVYA SURESH (220701069), JAYAJOTHI KUMAR (220701100)**" who carried out the project work under my supervision.

Submitted for Practical Examination held on ____________________

SIGNATURE	SIGNATURE
Dr. R. Sabitha	**Dr. G. Dharani Devi**
Professor and II Year Academic Head	**Associate Professor**
Computer Science and Engineering	**Computer Science and Engineering**
Rajalakshmi Engineering College	**Rajalakshmi Engineering College**
(Autonomous)	**(Autonomous)**
Thandalam	**Thandalam**
Chennai - 602105	**Chennai - 602105**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project aims to develop a dynamic web application for managing student enrollments in various courses. The application provides an intuitive interface for users to search for their enrollment records based on their name and date of birth. It also allows administrators to manage these records efficiently.

The key features of the application include a search functionality that displays the enrollment records in a tabular format and an "Unenroll" button beside each record. When the "Unenroll" button is clicked, the corresponding record is removed from the enrollments table, and the table is dynamically updated to reflect this change.

The front-end of the application is designed using HTML, CSS, and JavaScript, ensuring a user-friendly and visually appealing interface. The back-end is implemented using Node.js and Express.js, with MySQL serving as the database to store and manage the enrollment records.

This project demonstrates effective use of modern web development technologies to create a robust and interactive enrollment management system, providing users with a seamless experience while maintaining data integrity and consistency. The application is designed to handle real-time updates and ensure that users can efficiently manage their enrollments.

ACKNOWLEGEMENT

We express our sincere thanks to our beloved and honorable chairman **Mr.S.MEGANATHAN** and the chairperson **Dr.M.THANGAM MEGANATHAN** for their timely support and encouragement. We are greatly indebted to our respected and honorable principal **Dr.S.N.MURUGESAN** for his able support and guidance. No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr.P.KUMAR**, for being ever supporting force during our project work We also extend our sincere and hearty thanks to our internal guide **Dr.G.DHARANI DEVI,** for her valuable guidance and motivation during the completion of this project.

1. DIVYA SURESH
2. JAYAJOTHI KUMAR

TABLE OF CONTENTS

1. INTRODUCTION
    1.1. INTRODUCTION
    1.2. OBJECTIVE
2. TECHNOLOGIES
    2.1. SOFTWARE
    2.2. LANGUAGES
3. REQUIREMENTS AND ANALYSIS
    3.1. REQUIREMENTS SPECIFICATIONS
    3.2. HARDWARE AND SOFTWARE REQUIREMENTS
    3.3. ARCHITECTURE DIAGRAM
    3.4. ER DIAGRAM
    3.5. NORMALIZATION
4. PROGRAM CODE
5. RESULT
6. CONCLUSION
7. REFERENCES

INTRODUCTION

1.1. INTRODUCTION

In the modern educational environment, managing student enrollments effectively is crucial for both administrators and students. Traditional methods of handling enrollments, often involving manual processes, can be time-consuming and prone to errors. To address these challenges, this project introduces a web-based application designed to streamline the process of managing course enrollments and unenrollments.

The primary objective of this project is to create a user-friendly and efficient system for managing student enrollments in various courses. The application allows users to search for their enrollment records using specific criteria such as name and date of birth. It provides a clear and concise display of the search results in a tabular format, enhancing the ease of record management.

One of the notable features of this application is the inclusion of an "Unenroll" button beside each record in the search results. This feature enables users to easily remove their enrollment from a course, with the system dynamically updating the records to reflect this change. This functionality is particularly useful for administrators who need to maintain up-to-date and accurate enrollment records.

The front-end of the application is developed using HTML, CSS, and JavaScript, ensuring a visually appealing and responsive user interface. The back-end is powered by Node.js and Express.js, providing a robust and scalable server-side framework. The MySQL database is utilized to store and manage the enrollment data, ensuring data integrity and security.

By integrating these technologies, the application offers a comprehensive solution for managing student enrollments. It simplifies the enrollment process, reduces the administrative burden, and provides a seamless experience for users. This project exemplifies the effective use of web development technologies to address real-world challenges in educational administration.

1.2. OBJECTIVE

The objective of this project is to develop a web-based application that streamlines the management of student enrollments and unenrollments in various courses. The application aims to provide a user-friendly interface for searching, displaying, and managing enrollment records, with features such as dynamic record updates and seamless integration of front-end and back-end technologies.

TECHNOLOGIES

2.1. SOFTWARE

Visual Studio Code:

Visual Studio Code (VS Code) is a powerful and versatile code editor used for writing and editing the codebase. It offers features such as syntax highlighting, code completion, and integrated debugging, making it an ideal tool for both front-end and back-end development.

2.2. LANGUAGES

i. **HTML (HyperText Markup Language):**
HTML is the standard markup language used to create the structure of web pages. In this project, HTML is used to define the layout and elements of the web interface, including forms, tables, and buttons, ensuring a clear and organized presentation of enrollment data.

ii. **CSS (Cascading Style Sheets):**
CSS is used to style the HTML elements and enhance the visual appeal of the web application. It allows for the customization of colors, fonts, spacing, and layout, ensuring a responsive and visually appealing user interface.

iii. **JavaScript:**
JavaScript is a scripting language used to create dynamic and interactive web content. In this project, JavaScript is employed to handle user interactions, such as form submissions and button clicks, as well as to dynamically update the content displayed on the web pages.

iv. **Express.js:**
Express.js is a web application framework for Node.js that simplifies the process of building web applications and APIs. It is used to define the routes and middleware for handling various HTTP requests, making the back-end development more organized and manageable.

v. **MySQL:**
MySQL is a relational database management system used to store and manage the enrollment data. It ensures data integrity and provides efficient querying capabilities. MySQL is used to perform CRUD (Create, Read, Update, Delete) operations on the enrollment records.

REQUIREMENTS AND ANALYSIS

3.1. REQUIREMENTS SPECIFICATION

User Requirements

i. Search Enrollment Records:
Users must be able to search for their enrollment records using their name and date of birth.

ii. Display Enrollment Details:
The system should display the searched enrollment records in a tabular format, showing details such as name, age, date of birth, sex, phone number, and course.

iii. Unenroll Functionality:
Users should have the ability to unenroll from courses by clicking an "Unenroll" button next to their record. The record should be removed from the database and the table updated dynamically.

iv. Form Submission and Feedback:
After submitting an enrollment form, users should be redirected to a page where they can search and view their enrollment details. A success or error message should be displayed based on the form submission outcome.

System Requirements

i. Front-end Technologies:
- HTML: To structure the web pages.
- CSS: To style the web pages and enhance their visual appeal.
- JavaScript: To handle user interactions and dynamically update the web content.

ii. Back-end Technologies:
- Node.js: To run the server-side code and handle client requests.
- Express.js: To manage routing and middleware for handling HTTP requests.

iii. Database:
MySQL: To store and manage the enrollment data, providing efficient querying and data integrity.

iv. Development Environment:
Visual Studio Code: As the code editor for writing and debugging the application's code.

3.2. HARDWARE AND SOFTWARE REQUIREMENTS

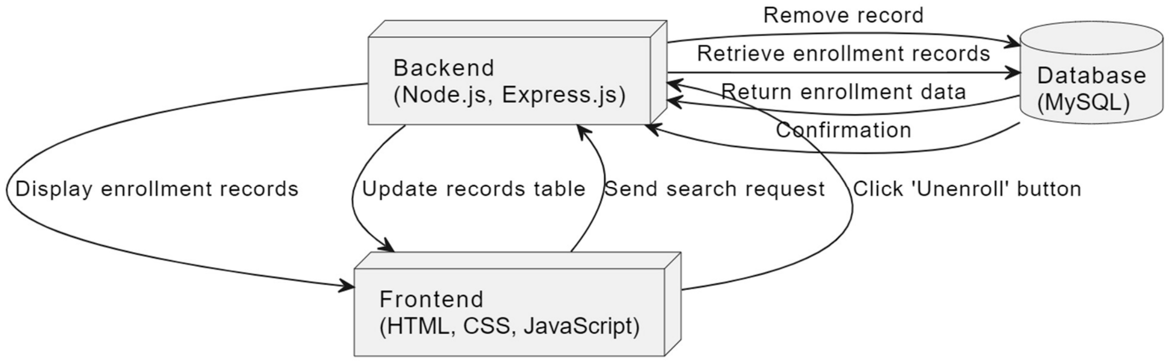
SOFTWARE REQUIREMENTS

- Operating system: Windows 11
- Front end: HTML, CSS, Javascript
- Back end: Node.js, Express.js, MySQL

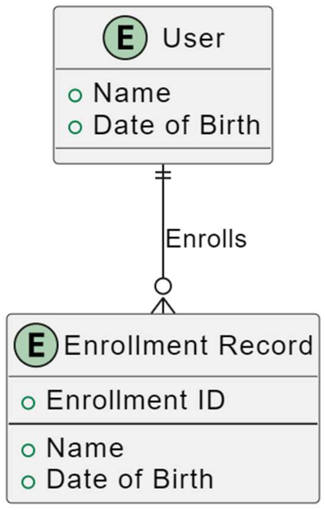
HARDWARE REQUIREMENTS

- Laptop
- Operating System: Windows 11
- 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz   2.80 GHz
- 64-bit operating system, x64-based processor
- Keyboard and mouse

3.3. ARCHITECTURE DIAGRAM



3.4. ER DIAGRAM



3.5. NORMALIZATION

To design a robust and efficient database for managing student enrolments, we need to ensure the database is normalized. Normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. The goal is to divide the data into multiple related tables to reduce data redundancy and improve data integrity.

Here's how we can normalize the database for the student enrolment management system:

1. First Normal Form (1NF)

1NF requires that the data in each column of a table be atomic (indivisible). Each table should have a primary key, and there should be no repeating groups.

Tables:

- **Students**: Stores student information.
  - student_id (Primary Key)
  - first_name
  - last_name
  - date_of_birth
  - email
- **Courses**: Stores course information.
  - course_id (Primary Key)
  - course_name
  - course_description
- **Enrollments**: Stores enrollment information.
  - enrollment_id (Primary Key)
  - student_id (Foreign Key referencing Students)
  - course_id (Foreign Key referencing Courses)
  - enrollment_date

2. Second Normal Form (2NF)

2NF requires that the table be in 1NF and all non-key attributes are fully functionally dependent on the primary key.

In our design, each table from 1NF is already in 2NF because:

- In the **Students** table, all columns depend on the student_id.
- In the **Courses** table, all columns depend on the course_id.
- In the **Enrollments** table, all columns depend on the enrollment_id.

3. Third Normal Form (3NF)

3NF requires that the table be in 2NF and all the attributes are functionally dependent only on the primary key.

In our design, each table from 2NF is already in 3NF because there are no transitive dependencies.

Final Database Schema

Students Table

Column Name	Data Type	Constraints
student_id	INT	PRIMARY KEY, AUTO_INCREMENT
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
date_of_birth	DATE	NOT NULL
email	VARCHAR	UNIQUE, NOT NULL

Courses Table

Column Name	Data Type	Constraints
course_id	INT	PRIMARY KEY, AUTO_INCREMENT
course_name	VARCHAR	NOT NULL
course_description	TEXT	

Enrollments Table

Column Name	Data Type	Constraints
enrollment_id	INT	PRIMARY KEY, AUTO_INCREMENT
student_id	INT	FOREIGN KEY REFERENCES Students(student_id)
course_id	INT	FOREIGN KEY REFERENCES Courses(course_id)
enrollment_date	DATE	NOT NULL

Additional Considerations

- **Indexes**: Indexes can be created on student_id, course_id, and enrollment_date in the Enrollments table to improve search performance.
- **Triggers and Constraints**: Triggers can be used to handle cascading deletes or updates. For instance, if a student is deleted, their corresponding enrollments can be automatically removed using a cascading delete constraint.
- **Transactions**: Ensure that operations such as enrollment and unenrollment are handled within transactions to maintain data consistency.

This normalization ensures that the database design is efficient, maintains data integrity, and eliminates redundancy, providing a solid foundation for the enrollment management system.

PROGRAM CODE

Server.js

```
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');
const path = require('path');
const bcrypt = require('bcrypt');

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

// Serve static files from the "public" directory
app.use(express.static(path.join(__dirname, 'public')));

const connection = mysql.createConnection({
  host: process.env.DB_HOST || 'localhost',
  user: process.env.DB_USER || 'root',
  password: process.env.DB_PASSWORD || '10012005',
  database: process.env.DB_NAME || 'mastery'
});

connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL database:', err);
    return;
  }
  console.log('Connected to MySQL database');
});
```

```
// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
// Signup route
app.post('/signup', async (req, res) => {
  const { username, password } = req.body;

  try {
    // Hash the password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Insert user into database
    const insertQuery = 'INSERT INTO users (username, password) VALUES (?, ?)';
    await connection.promise().query(insertQuery, [username, hashedPassword]);

    return res.status(201).json({ message: 'User registered successfully!', redirectUrl:
'/contents' });
  } catch (error) {
    console.error('Error signing up:', error);
    return res.status(500).json({ error: 'Failed to sign up.' });
  }
});

// Login route
app.post('/login', async (req, res) => {
  const { username, password } = req.body;

  try {
    // Check if the user exists
```

```
        const [rows] = await connection.promise().query('SELECT * FROM users WHERE
username = ?', [username]);


        if (rows.length === 0) {
            return res.status(404).json({ error: 'User not found' });
        }


        const user = rows[0];


        // Compare the hashed password
        const isPasswordMatch = await bcrypt.compare(password, user.password);


        if (!isPasswordMatch) {
            return res.status(401).json({ error: 'Invalid credentials' });
        }


        return res.status(200).json({ message: 'Logged in successfully!', redirectUrl: '/contents'
});
    } catch (error) {
        console.error('Error logging in:', error);
        return res.status(500).json({ error: 'Failed to log in.' });
    }
});



// Serve the enroll form page
//app.get('/enroll', (req, res) => {
//    res.sendFile(path.join(__dirname, 'public', 'enroll.html'));
//});
app.get('/enroll', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'enroll.html'));
});
```

```
// Enrollment route - GET and POST
app.post('/enroll', (req, res) => {
   //if (req.method === 'GET') {
      //res.sendFile(path.join(__dirname, 'public', 'enroll.html'));
   if (req.method === 'POST') {

   const { name, age, dob, sex, phone, course } = req.body;

      if (!name || !age || !dob || !sex || !phone || !course) {
         res.status(400).json({ status: 'error', message: 'All fields are required' });
         return;
      }

      const insertQuery = "INSERT INTO enrollments (name, age, dob, sex, phone, course)
VALUES (?, ?, ?, ?,?,?)";
      connection.query(insertQuery, [name, age, dob, sex, phone, course], (err, result) => {
         if (err) {
            console.error('Failed to enroll:', err);
            res.status(500).json({ status: 'error', message: 'Failed to enroll' });
            return;
         }
         console.log('Enrollment successful:', result);
         //res.json({ status: 'success', message: 'Enrolled successfully' });
         res.status(200).json({ message: 'Successfully enrolled', redirectUrl: '/display' });
      });
   }else {
      res.status(405).json({ status: 'error', message: 'Method Not Allowed' });
   }
});

// Unenroll route
```

```
/*app.delete('/unenroll/:course', (req, res) => {
   const course = req.params.course;
   const { name } = req.body;

   // Check if the user is enrolled in the course
   const checkQuery = 'SELECT * FROM enrollments WHERE name = ? AND course = ?';
   connection.query(checkQuery, [name, course], (err, results) => {
      if (err) {
         return res.status(500).json({ error: err });
      }

      if (results.length === 0) {
         return res.status(404).json({ message: 'User not enrolled in this course' });
      }

      // If user is enrolled, proceed with unenrollment
      const deleteQuery = 'DELETE FROM enrollments WHERE name = ? AND course = ?';
      connection.query(deleteQuery, [name, course], (err, result) => {
         if (err) {
            return res.status(500).json({ error: err });
         }
         res.json({ message: 'Unenrollment successful!' });
      });
   });
});*/



app.get('/display', (req, res) => {
   res.sendFile(path.join(__dirname, 'public', 'display.html'));
});

app.post('/search', (req, res) => {
    const { name, dob } = req.body;

    const sql = 'SELECT * FROM enrollments WHERE name = ? AND dob = ?';
    connection.query(sql, [name, dob], (err, results) => {
        if (err) {
            console.error('Error during search:', err);
            return res.status(500).json({ message: 'Failed to search. Please try again later.' });
        }
        res.status(200).json(results);
    });
});

app.delete('/unenroll/:id', (req, res) => {
    const id = req.params.id;

    const sql = 'DELETE FROM enrollments WHERE id = ?';
    connection.query(sql, id, (err, result) => {
        if (err) {
            console.error('Error during unenrollment:', err);
            return res.status(500).json({ success: false });
        }
        res.status(200).json({ success: true });
    });
});



// Gracefully handle MySQL connection closure
process.on('SIGINT', () => {
    connection.end((err) => {
        if (err) {
            console.error('Error closing MySQL connection:', err);
        } else {
            console.log('MySQL connection closed');
        }
        process.exit();
    });
});
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome Page</title>
  <style>
    body {
      background-color: rgb(65, 38, 47);
      color:white ;
      text-align: center;
      font-family: Arial, sans-serif;
      display: flex;
      flex-direction: column;
      justify-content: center;
      height: 100vh;
      margin: 0;
    }
    .welcome {
      font-weight: bold;
      font-size: 100px;
      padding-top: 30px;

    }
    .subtext {
      font-size: 18px;
    }
    .login-button {
      background-color: #ffffff;
      color: purple;
      border: none;
      padding: 10px 20px;
```

```
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin-top: 20px;
            cursor: pointer;
            border-radius: 5px;
            width: 100px;
        }
        .login-button-container {
            margin-top: 20px;
            text-align: center; /* Center the button */
        }
        .login-button:hover {
            background-color: #f1f1f1;


        }
    </style>
</head>
<body>
    <div class="welcome">Welcome</div>
    <div class="subtext">to Master Class</div></br>
    <div class="subtext">"Studying made easier"</div>
    <div class="login-button-container">
    <a href="login-details.html" class="login-button">Login</a>
</body>
</html>
```

Login-details.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Form</title>
   <style>
      body {
         background-color: lavenderblush;
         color: white;
         font-family: Arial, sans-serif;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
         margin: 0;
      }
      .login-container {
         background-color: rgb(65, 38, 47);
         padding: 20px;
         border-radius: 5px;
         text-align: center;
      }
      .login-container label {
         display: block;
         margin-bottom: 10px;
      }
      .login-container input[type="text"],
      .login-container input[type="password"] {
         width: 200px;
         padding: 5px;
         border: none;
```

```
            background-color: white;
            border-radius: 5px;
        }
        .login-container input[type="submit"] {
            padding: 5px 20px;
            background-color: #fff;
            color: purple;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            margin-right: 10px;
        }
        .login-container input[type="submit"]:hover {
            background-color: #f1f1f1;
        }
        .login-container button {
            padding: 5px 20px;
            background-color: #fff;
            color: purple;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        .login-container button:hover {
            background-color: #f1f1f1;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login Form</h2>
        <form id="loginForm" onsubmit="return login(event)">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Login">
        <button type="button" onclick="signup()">Sign Up</button>
        <button type="button" onclick="forgotPassword()">Forgot Password</button>
    </form>
</div>

<script>
    function login(event) {
        event.preventDefault();
        const username = document.getElementById('username').value;
        const password = document.getElementById('password').value;

        // Validate input
        if (!username || !password) {
            alert("Username and password are required.");
            return;
        }

        // Create a new request to validate login
        const xhr = new XMLHttpRequest();
        xhr.open('POST', '/login', true);
        xhr.setRequestHeader('Content-Type', 'application/json');

        xhr.onload = function() {
            if (xhr.status === 200) {
                // Assuming the server returns a success response
                window.location.href = "contents page.html";
            } else {
```

```
            alert("Invalid username or password.");
        }
    };

    // Send the request with the username and password
    xhr.send(JSON.stringify({ username, password }));
}

function signup() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    // Validate input
    if (!username || !password) {
        alert("Username and password are required.");
        return;
    }

    // Create a new request to update the database
    const xhr = new XMLHttpRequest();
    xhr.open('POST', '/signup', true);
    xhr.setRequestHeader('Content-Type', 'application/json');

    xhr.onload = function() {
        if (xhr.status === 200) {
            // Assuming the server returns a success response
            window.location.href = "contents page.html";
        } else {
            alert("Signup failed. Try again.");
        }
    };
    // Send the request with the username and password
    xhr.send(JSON.stringify({ username, password }));
}

function forgotPassword() {
    const username = document.getElementById('username').value;

    // Validate input
    if (!username) {
        alert("Username is required.");
        return;
    }

    const newPassword = prompt("Please enter your new password:");

    if (!newPassword) {
        alert("New password is required.");
        return;
    }

    // Create a new request to handle forgot password logic
    const xhr = new XMLHttpRequest();
    xhr.open('POST', '/forget-password', true);
    xhr.setRequestHeader('Content-Type', 'application/json');

    xhr.onload = function() {
        if (xhr.status === 200) {
            alert("Password has been reset successfully.");
        } else {
            alert("Password reset failed. Try again.");
        }
    };
```

```
            // Send the request with the username and new password
            xhr.send(JSON.stringify({ username, newPassword }));

            // Clear the form fields
            document.getElementById('username').value = '';
            document.getElementById('password').value = '';
        }
    </script>
</body>
</html>
```

Contents-page.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Courses</title>
    <style>
        .course {
            border: 1px solid #ccc;
            border-radius: 5px;
            padding: 10px;
            margin-bottom: 10px;
            background-color: #f9f9f9;
        }
        .course-details {
            display: none;
            margin-top: 10px;
            padding: 10px;
            background-color: #f0f0f0;
        }
        .enroll-btn {
            background-color: #4CAF50;
            color: white;
            border: none;
            padding: 5px 10px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            border-radius: 3px;
            cursor: pointer;
        }
        .unenroll-btn {
            background-color: #f44336;
            color: white;
            border: none;
            padding: 5px 10px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            border-radius: 3px;
            cursor: pointer;
        }
        .enroll-form-btn {
            background-color: #2196F3;
            color: white;
            border: none;
            padding: 5px 10px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            border-radius: 3px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="course" id="java">
        <h2>Java</h2>
        <p>A course on Java programming language.</p>
        <button class="enroll-btn" onclick="showDetails('java')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="window.location.href='/enroll'">Enroll
Form</button>
        <div class="course-details" id="java-details">
            <h3>Contents</h3>
            <ul>
                <li>Introduction to Java</li>
                <li>Java Syntax</li>
                <li>Object-Oriented Programming in Java</li>
                <li>Java Collections Framework</li>
                <li>Java Database Connectivity (JDBC)</li>
            </ul>
        </div>
    </div>

    <div class="course" id="dbms">
        <h2>DBMS (Database Management System)</h2>
        <p>A course on database management systems.</p>
        <button class="enroll-btn" onclick="showDetails('dbms')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="dbms-details">
            <h3>Contents</h3>
            <ul>
                <li>Introduction to DBMS</li>
                <li>Relational Database Concepts</li>
                <li>SQL (Structured Query Language)</li>
                <li>Database Design</li>
                <li>Normalization and Optimization</li>
            </ul>
        </div>
    </div>
    <div class="course" id="c">
        <h2>C Programming</h2>
        <p>A course on C programming language.</p>
        <button class="enroll-btn" onclick="showDetails('c')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="c-details">
            <h3>Contents</h3>
            <ul>
                <li>Introduction to C</li>
                <li>C Syntax and Data Types</li>
                <li>Control Flow in C</li>
                <li>Functions and Pointers</li>
                <li>File Handling in C</li>
            </ul>
        </div>
    </div>
    <div class="course" id="python">
        <h2>python</h2>
        <p>A course on python  programming language.</p>
        <button class="enroll-btn" onclick="showDetails('python')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="python-details">
            <h3>Contents</h3>
            <ul>
                <li>Introduction to python</li>
                <li>python Syntax and Data Types</li>
                <li>Advanced Python Libraries </li>
                <li>web Development with Python</li>
                <li>Data Science and Machine Learning with Python</li>
            </ul>
        </div>
    </div>
    <div class="course" id="cooking">
        <h2>Cooking Basics</h2>
        <p>A course on culinary arts .</p>
        <button class="enroll-btn" onclick="showDetails('cooking')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="cooking-details">
            <h3>Contents</h3>
            <ul>
                <li>Knife Skills</li>
                <li>Basic Cooking Techniques </li>
                <li>Understanding Ingredients</li>
                <li>Food Safety and Hygiene</li>
                <li>Simple Recipes for Beginners</li>
            </ul>
        </div>
    </div>

    <div class="course" id="baking">
        <h2>Baking</h2>
        <p>A course on baking and types.</p>
        <button class="enroll-btn" onclick="showDetails('baking')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div    class="course-details" id="baking-details">
```

```
            <h3>Contents</h3>
            <ul>
                <li>Introduction to Baking</li>
                <li>Baking Equipment and Tools</li>
                <li>Types of Flour and Sugar</li>
                <li>Basic Baking Techniques</li>
                <li>Recipes for Cakes, Cookies, and Breads</li>
            </ul>
        </div>
    </div>
    <div class="course" id="drawing">
        <h2>Drawing</h2>
        <p>A course on drawing.</p>
        <button class="enroll-btn" onclick="showDetails('drawing')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="drawing-details">
            <h3>Contents</h3>
            <ul>
                <li>Basic Drawing Techniques</li>
                <li>Understanding Light and Shadow</li>
                <li>Composition and Perspective</li>
                <li>Sketching Exercises</li>
                <li>Drawing from Observation</li>
            </ul>
        </div>
    </div>
    <div class="course" id="dancing">
        <h2>Dancing</h2>
        <p>A course on dancing and types.</p>
        <button class="enroll-btn" onclick="showDetails('dancing')">Enroll</button>
        <button class="unenroll-btn">Unenroll</button>
        <button class="enroll-form-btn" onclick="redirectToEnrollForm()">Enroll
Form</button>
        <div class="course-details" id="dancing-details">
            <h3>Contents</h3>
            <ul>
                <li>Introduction to Dance</li>
                <li>Popular Dance Styles </li>
                <li>Dance Techniques</li>
                <li>Choreography Basics</li>
                <li>Dance Routines</li>
            </ul>
        </div>
    </div>
    <div id="enroll" style="display: none;">
        <h2>Enrollment Form</h2>
        <form action="/enroll" method="GET">
            <label for="name">Name:</label><br>
            <input type="text" id="name" name="name" required><br>

            <label for="age">Age:</label><br>
            <input type="number" id="age" name="age" required><br>

            <label for="dob">Date of Birth:</label><br>
            <input type="date" id="dob" name="dob" required><br>

            <label for="sex">Sex:</label><br>
            <select id="sex" name="sex" required>
                <option value="male">Male</option>
                <option value="female">Female</option>
                <option value="other">Other</option>
            </select><br>
            <label for="phone">Phone:</label><br>
            <input type="tel" id="phone" name="phone" required><br>


            <input type="hidden" id="course" name="course" value=""><!-- This will be filled
dynamically via JavaScript -->


            <button type="submit">Enroll</button>
        </form>
    </div>
    <script>
         function showDetails(courseId) {
            var details = document.getElementById(courseId + "-details");
            details.style.display = "block";
            var enrollBtn = document.getElementById(courseId + "-enroll-btn");
            enrollBtn.style.display = "none";
        }

        //function redirectToEnrollForm() {
            //window.location.href = '/enroll';
        //}
    </script>
</body>
</html>
```

Enroll.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Enroll</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f9f9f9;
      padding: 20px;
      margin: 0;
    }

    h1 {
      text-align: center;
      margin-bottom: 20px;
    }

    form {
      max-width: 400px;
      margin: 0 auto;
      background-color: #fff;
      padding: 20px;
      border-radius: 5px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    label {
```

```
    display: block;
    margin-bottom: 5px;
}

input[type="text"],
input[type="number"],
input[type="date"],
input[type="tel"],
select {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 3px;
    box-sizing: border-box;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #2196F3;
    color: #fff;
    border: none;
    border-radius: 3px;
    cursor: pointer;
}

button:hover {
    background-color: #0b7dda;
}

.button-container {
```

```
            display: flex;
            justify-content: space-between;
        }

        .button-container button {
            width: 48%;
        }

    </style>
</head>
<body>
    <h1>Enroll Form</h1>
    <form id="enrollmentForm">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required><br><br>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" required><br><br>

        <label for="sex">Sex:</label>
        <select id="sex" name="sex" required>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="other">Other</option>
        </select><br><br>

        <label for="phone">Phone Number:</label>
        <input type="tel" id="phone" name="phone" required><br><br>
        <label for="courseSelect">Select Course:</label>
        <select id="courseSelect" name="course" required>
            <option value="Java">Java</option>
            <option value="DBMS">DBMS</option>
            <option value="C">C</option>
            <option value="Python">Python</option>
            <option value="Cooking Basics">Cooking Basics</option>
            <option value="Baking">Baking</option>
            <option value="Drawing">Drawing</option>
            <option value="Dancing">Dancing</option>
        </select><br><br>

        <button type="submit">Enroll</button>
    </form>



    <script>
        document.getElementById('courseSelect').addEventListener('change', function() {
            let selectedCourse = document.getElementById('courseSelect').value;
            console.log(selectedCourse);
            //document.getElementById('course').value = selectedCourse;
        });

        document.getElementById('enrollmentForm').addEventListener('submit',
function(event) {
            event.preventDefault();

            const enrollmentData = {
                name: document.getElementById('name').value,
                age: parseInt(document.getElementById('age').value, 10),
                dob: document.getElementById('dob').value,
                sex: document.getElementById('sex').value,
                phone: document.getElementById('phone').value,
                course: document.getElementById('courseSelect').value
            };

            fetch('/enroll', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(enrollmentData)
            })
            .then(response => {
                if (!response.ok) {
                    return response.json().then(error => { throw new Error(error.message); });
                }
                return response.json();
            })
            .then(data => {
                    alert(data.message);
                    if (data.redirectUrl) {
                        window.location.href = data.redirectUrl;
                    }
                })
            .catch(error => {
                console.error('Error1:', error.message);
                alert('Failed to enroll. Please try again later.');
            });
        });

        // Initialize the course field with the default selected course
        document.getElementById('course').value =
document.getElementById('courseSelect').value;
```

```
        function unenroll() {
            const form = document.getElementById('enrollForm');
            const formData = new FormData(form);
            const course = formData.get('course');


            fetch(`/unenroll/${course}`, { // Make sure `course` is defined and contains the
correct value
    method: 'DELETE',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({ name: formData.get('name') })
})


            .then(response => response.json())
            .then(data => {
                alert(data.message);
                // Redirect to another page or update UI as needed
            })
            .catch(error => {
                console.error('Error:', error);
                alert('Failed to unenroll.');
            });
        }
        // Initialize the course field with the default selected course
        //document.getElementById('course').value =
document.getElementById('courseSelect').value;


    </script>
</body>
</html>
```

Unenroll.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <!-- Meta tags, title, and styles -->
   <style>
      #unenroll {
         display: none;
      }
   </style>
</head>
<body>
   <!-- Course sections with enroll and unenroll buttons -->

   <!-- Unenrollment form for each course -->
   <div class="course" id="unenroll">
      <h2>Unenroll Form</h2>
      <form id="unenrollForm">
         <label for="course">Course:</label><br>
         <input type="text" id="course" name="course" readonly><br>
         <input type="hidden" id="userId" name="userId" value="USER_ID"><!-- Assuming
USER_ID is set dynamically -->
         <button type="button" onclick="unenroll()">Unenroll</button>
      </form>
   </div>

   <!-- JavaScript to handle showing the unenrollment form -->
   <script>
      function showUnenrollForm(courseId) {
         var unenrollForm = document.getElementById('unenroll');
         var courseInput = unenrollForm.querySelector('#course');
```

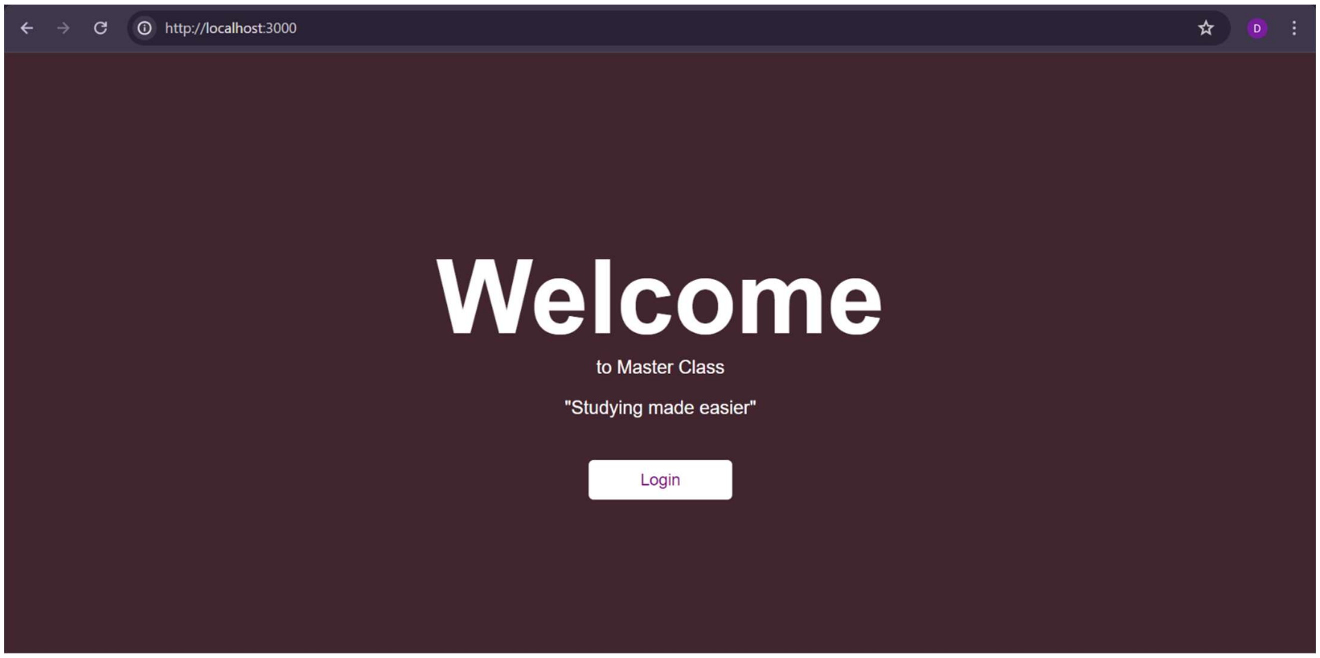
```
            courseInput.value = courseId;
            unenrollForm.style.display = 'block';
        }

        function unenroll() {
            var course = document.getElementById('course').value;
            var userId = document.getElementById('userId').value;

            fetch(`/unenroll/${course}`, {
                method: 'DELETE',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ userId: userId })
            })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Failed to unenroll');
                }
                alert('Unenrollment successful!');
                document.getElementById('unenrollForm').reset();
                document.getElementById('unenroll').style.display = 'none';
            })
            .catch(error => {
                console.error('Error:', error);
                alert('Failed to unenroll');
            });
        }
    </script>
</body>
</html>
```

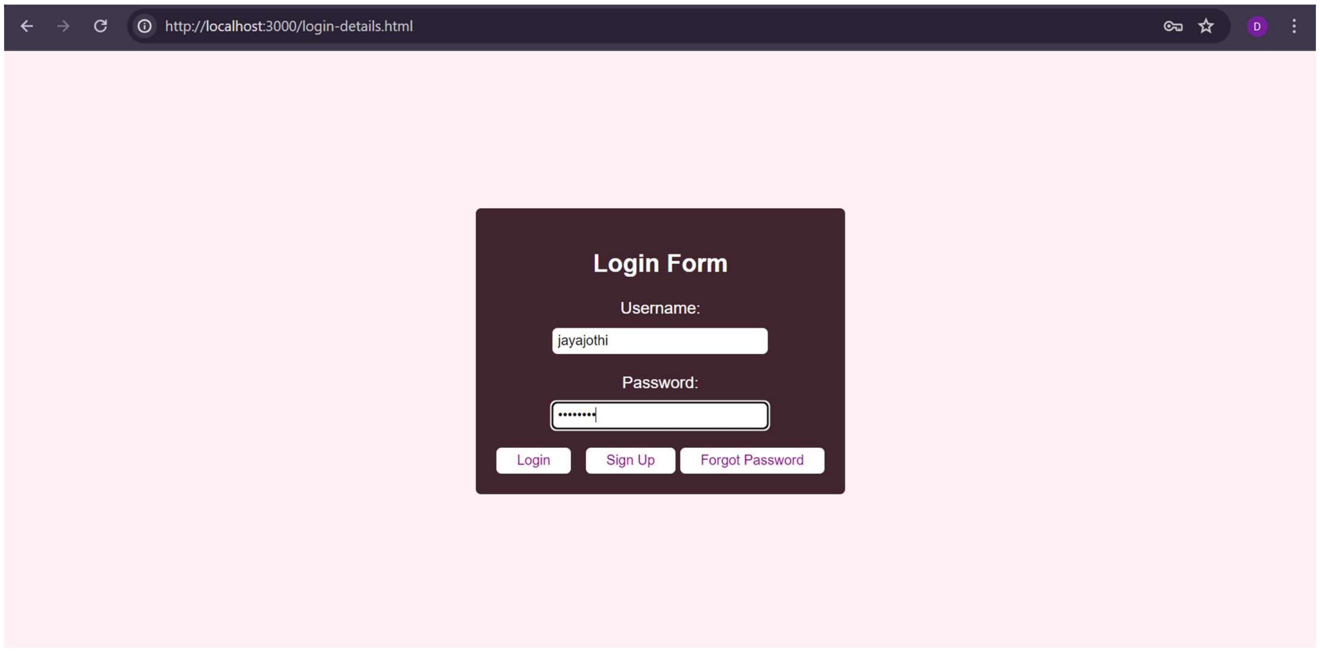
Display.html

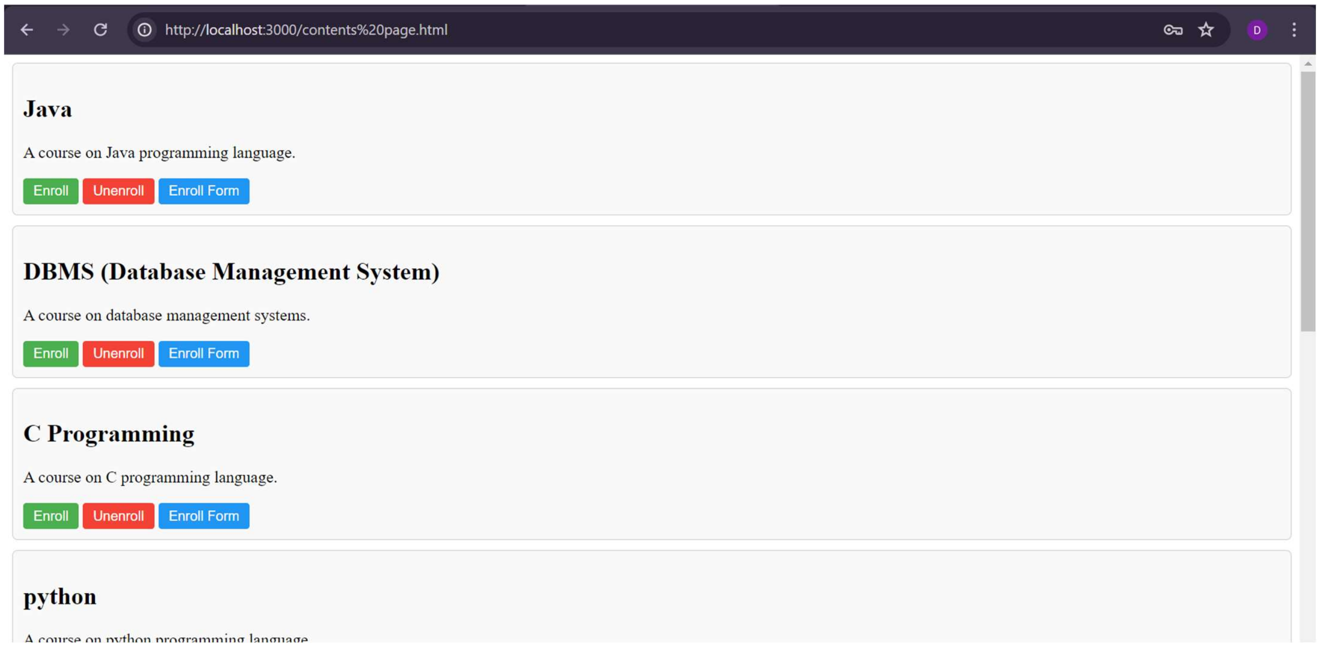
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Search Enrollments</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f0f0f0;
        }
        h1, h2 {
            text-align: center;
            color: #333;
        }
        form {
            margin-bottom: 20px;
            text-align: center;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input[type="text"],
        input[type="date"],
        button {
            padding: 5px 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 3px;
        background-color: #fff;
    }
    button {
        background-color: #007bff;
        color: #fff;
        cursor: pointer;
    }
    table {
        width: 100%;
        border-collapse: collapse;
        background-color: #fff;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }
    th, td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
    tr:nth-child(even) {
        background-color: #f9f9f9;
    }
    tr:hover {
        background-color: #f1f1f1;
    }
</style>
<script>
    function unenroll(id) {
    fetch(`/unenroll/${id}`, {
        method: 'DELETE'
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert('Record successfully unenrolled');
            // Optionally, remove the row from the table
            document.getElementById(`row-${id}`).remove();
        } else {
            alert('Failed to unenroll. Please try again later.');
        }
    })
    .catch(error => {
        console.error('Error:', error);
        alert('Failed to unenroll. Please try again later.');
    });
}

    function searchEnrollments(event) {
        event.preventDefault();

        const name = document.getElementById('name').value;
        const dob = document.getElementById('dob').value;

        fetch('/search', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ name, dob })
        })
        .then(response => response.json())
        .then(data => {
                const tableBody = document.getElementById('resultsTableBody');
                tableBody.innerHTML = '';

                if (data.length > 0) {
                    data.forEach(row => {
    const tr = document.createElement('tr');
    tr.id = `row-${row.id}`;
    tr.innerHTML = `
        <td>${row.name}</td>
        <td>${row.age}</td>
        <td>${new Date(row.dob).toISOString().split('T')[0]}</td>
        <td>${row.sex}</td>
        <td>${row.phone}</td>
        <td>${row.course}</td>
        <td><button onclick="unenroll(${row.id})">Unenroll</button></td>
    `;
    tableBody.appendChild(tr);
});
                } else {
                    const tr = document.createElement('tr');
                    tr.innerHTML = '<td colspan="6">No results found</td>';
                    tableBody.appendChild(tr);
                }
            })
            .catch(error => {
                console.error('Error:', error);
                alert('Failed to search. Please try again later.');
            });
        }
    </script>
</head>
<body>
```

```
    <h1>Search Enrollments</h1>
    <form onsubmit="searchEnrollments(event)">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob"><br>

        <button type="submit">Search</button>
    </form>

    <h2>Results</h2>
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Date of Birth</th>
                <th>Sex</th>
                <th>Phone</th>
                <th>Course</th>
            </tr>
        </thead>
        <tbody id="resultsTableBody">
            <!-- Results will be inserted here -->
        </tbody>
    </table>
</body>
</html>
```

RESULT


http://localhost:3000
Welcome
to Master Class
"Studying made easier"
Login


http://localhost:3000/login-details.html
Login Form
Username:
jayajothi
Password:
••••••••
Login
Sign Up
Forgot Password


http://localhost:3000/contents%20page.html

Java

A course on Java programming language.

Enroll Unenroll Enroll Form

DBMS (Database Management System)

A course on database management systems.

Enroll Unenroll Enroll Form

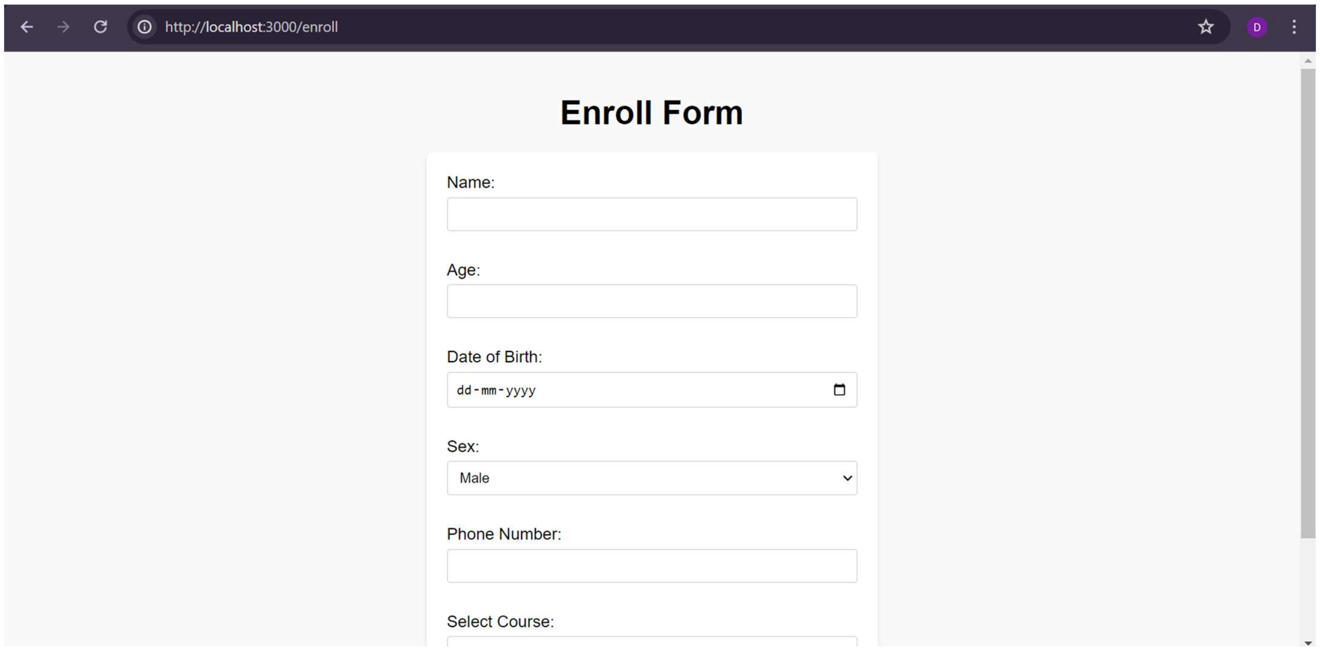
C Programming

A course on C programming language.

Enroll Unenroll Enroll Form

python

A course on python programming language.


http://localhost:3000/enroll

Enroll Form

Name:

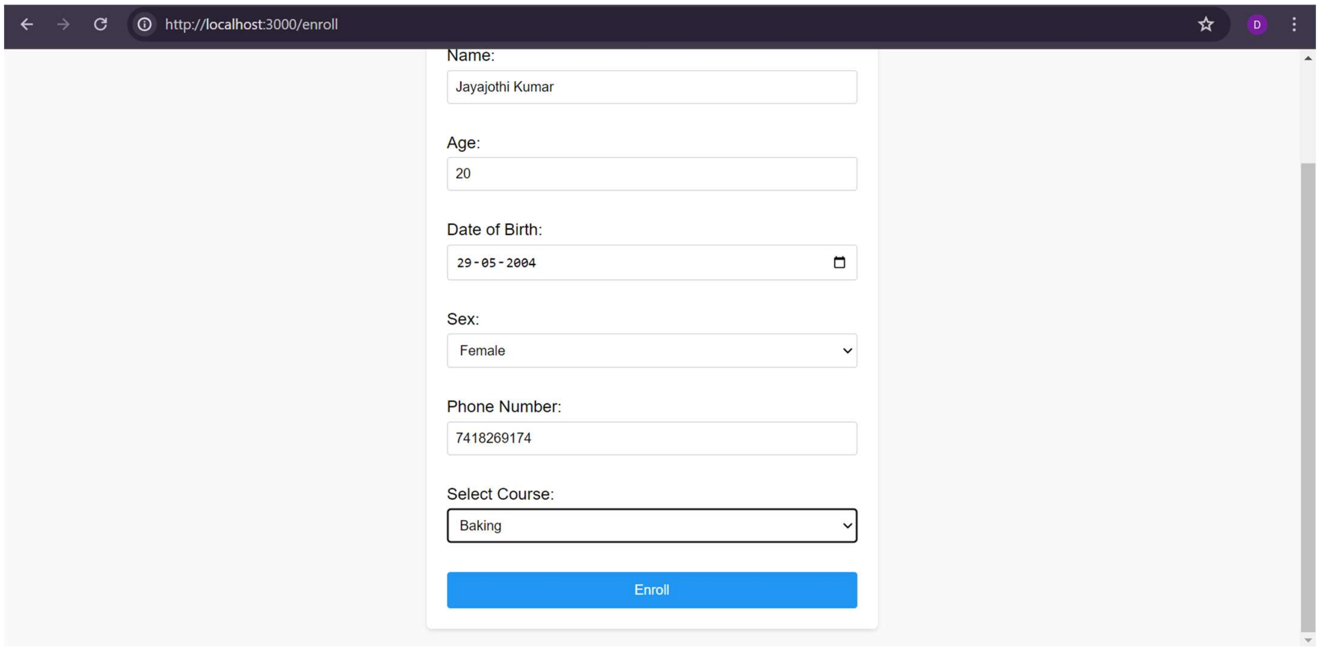
Age:

Date of Birth: dd-mm-yyyy

Sex: Male

Phone Number:

Select Course:


http://localhost:3000/enroll

Name:
Jayajothi Kumar

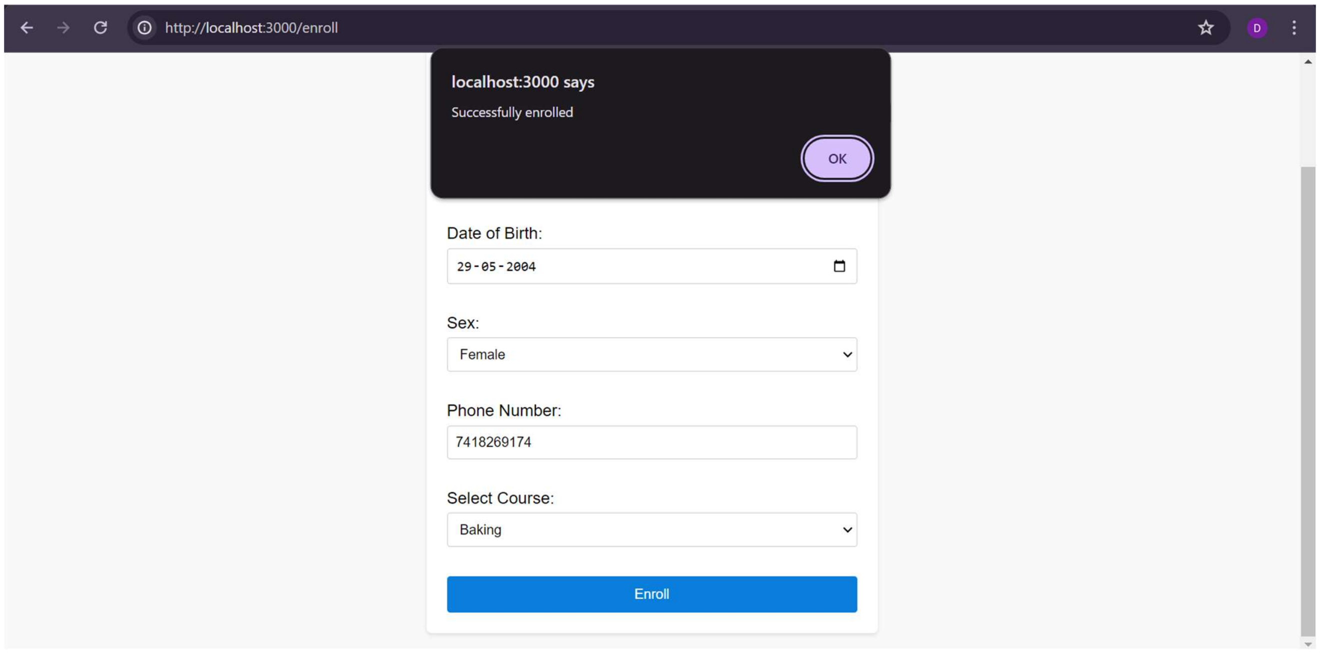
Age:
20

Date of Birth:
29-05-2004

Sex:
Female

Phone Number:
7418269174

Select Course:
Baking


http://localhost:3000/enroll

localhost:3000 says

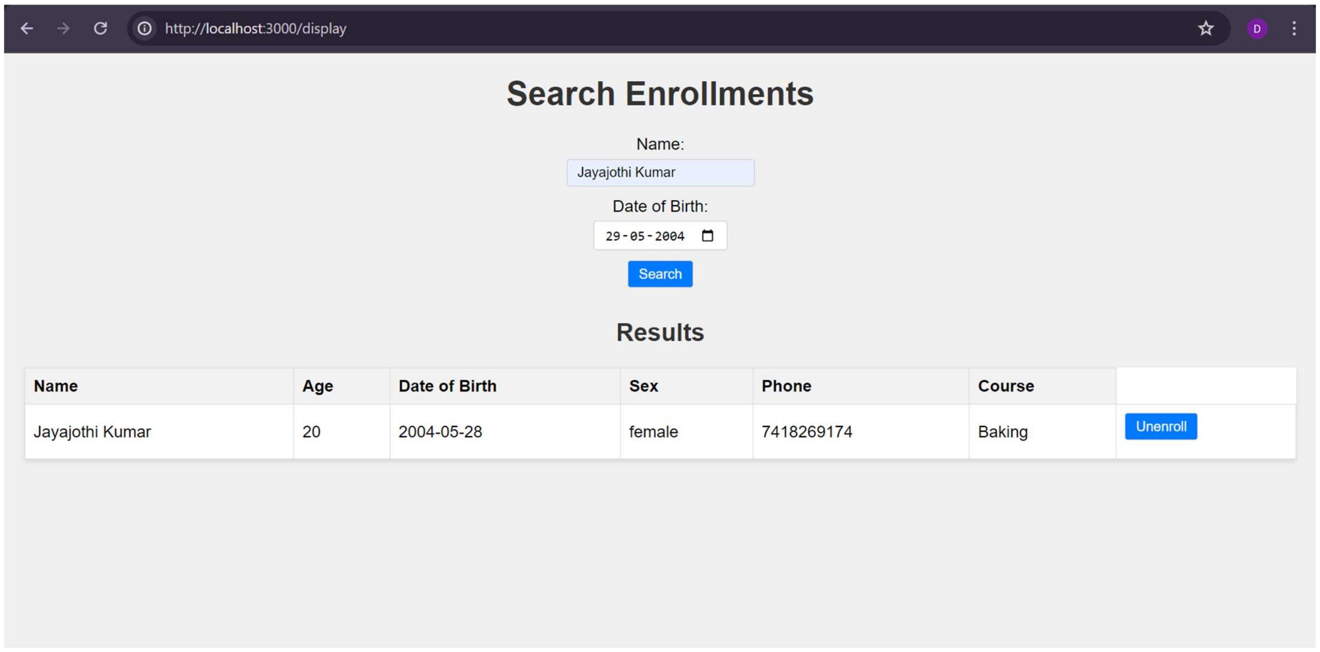
Successfully enrolled

Date of Birth:
29-05-2004

Sex:
Female

Phone Number:
7418269174

Select Course:
Baking



http://localhost:3000/display

Search Enrollments

Name: Jayajothi Kumar

Date of Birth: 29-05-2004

Results

Name	Age	Date of Birth	Sex	Phone	Course	
Jayajothi Kumar	20	2004-05-28	female	7418269174	Baking	Unenroll

CONCLUSION

In conclusion, the development of a web-based application for managing student enrollments and unenrollments has been successfully achieved. The application provides users with a user-friendly interface for searching, displaying, and managing enrollment records, while also offering administrators the ability to efficiently manage these records.

The use of modern web development technologies, including HTML, CSS, JavaScript, Node.js, and MySQL, has enabled the creation of a robust and efficient system. The integration of front-end and back-end components ensures a seamless user experience, while the use of MySQL ensures data integrity and efficient data management.

Overall, this project demonstrates the effectiveness of using web technologies to address real-world challenges in educational administration. The application provides a valuable tool for managing student enrollments and unenrollments, streamlining the process and improving overall efficiency.

REFERENCES

1. https://www.geeksforgeeks.org/
2. https://www.w3schools.com/nodejs/
3. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
4. https://expressjs.com/
5. https://www.mysqltutorial.org/