MAHATMA EDUCATION SOCIETY’S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE

(Autonomous) NEW PANVEL

PROJECT REPORT ON

# “GOLF MEMBERSHIP MANAGEMENT”

IN PARTIAL FULFILLMENT OF

MASTER OF SCIENCE IN DATA ANALYTICS

SEMESTER I 2024-25

PROJECT GUIDE PROF. OMKAR SHERKHANE

SUBMITTED BY: ROHIT MALVIYA

ROLL NO: 10315

Mahatma Education Society's

**PILLAI COLLEGE OF ARTS,COMMERCE & SCIENCE**

**(Autonomous)**

## Re-accredited “A” Grade by NAAC (3rd Cycle)



Project Completion Certificate

# THIS IS TO CERTIFY THAT

**Rohit Malviya**

of **M.Sc. Data Analytics Part - 1** has completed the project titled **Swimming Classes Membership Management** of subject **Next Generation Databases** under our guidance and supervision during the academic year 2024-25 in the department of

Project Guide Course Coordinator Head of the Department

**INTRODUCTION**

The "Swimming Classes Membership Management" project is designed to streamline the administration of swimming class membership records using MongoDB. This system provides a user-friendly interface for managing essential database operations, such as adding new member records, updating existing information, deleting records, and retrieving membership details. By leveraging MongoDB’s flexible schema design, the project ensures efficient data management, accommodating various membership attributes, and allowing the database to adapt to changing requirements.

* **Inserting New Member Records:** Easily add new members to the swimming classes database with all necessary details, including preferred class times and skill levels.
* **Updating Existing Information:** Modify member information, such as contact details, class schedules, and membership types, ensuring that records remain up-to-date and accurate.
* **Deleting Records:** Remove members from the database when required, maintaining data integrity and compliance with privacy regulations.
* **Retrieving Membership Details:** Access and display comprehensive information about all members, facilitating efficient management and personalized communication.

**Tools used:** MongoDB, NetBeans, HTML/CSS, Java

**Techniques:**

* **CRUD Operations:** The fundamental operations of Create, Read, Update, and Delete, essential for effectively managing membership records for swimming classes.

**Index.html**

# CODE

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Swimming Classes Membership Management</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #e8f1f2;

}

.container {

width: 80%;

margin: auto;

overflow: hidden;

background: #ffffff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

h1 {

color: #333;

text-align: center;

}

h3 {

color: #333;

}

form {

margin-bottom: 20px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 4px;

background: #d3f9d8;

}

label {

display: block;

margin: 10px 0 5px;

font-weight: bold;

}

input[type="text"], input[type="email"] {

width: calc(100% - 22px);

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 4px;

}

input[type="submit"] {

padding: 10px 20px;

border: none;

border-radius: 4px;

background-color: #28a745;

color: white;

cursor: pointer;

font-size: 16px;

}

input[type="submit"]:hover {

background-color: #218838;

}

.result {

padding: 10px;

margin-top: 10px;

border: 1px solid #ddd;

border-radius: 4px;

background: #ffeeba;

color: #333;

}

</style>

</head>

<body>

<div class="container">

<h1>Swimming Classes Membership Management</h1>

<form method="get" action="swimmingclasses.jsp">

<h3>Register New Swimming Member:</h3>

<label for="name">Name:</label>

<input type="text" name="name" id="name" required>

<label for="email">Email:</label>

<input type="email" name="email" id="email" required>

<label for="registration\_id">Registration ID:</label>

<input type="text" name="registration\_id" id="registration\_id" required>

<label for="mem\_addr">Address:</label>

<input type="text" name="mem\_addr" id="mem\_addr" required>

<label for="mem\_type">Membership Type:</label>

<input type="text" name="mem\_type" id="mem\_type" required>

<input type="hidden" name="operation" value="insert">

<input type="submit" value="Register Member">

</form>

<form method="get" action="swimmingclasses.jsp">

<h3>Update Swimming Member Details:</h3>

<label for="registration\_id\_update">Registration ID:</label>

<input type="text" name="registration\_id" id="registration\_id\_update" required>

<label for="new\_name">New Name:</label>

<input type="text" name="new\_name" id="new\_name">

<label for="new\_email">New Email:</label>

<input type="email" name="new\_email" id="new\_email">

<label for="new\_mem\_addr">New Address:</label>

<input type="text" name="new\_mem\_addr" id="new\_mem\_addr">

<label for="new\_mem\_type">New Membership Type:</label>

<input type="text" name="new\_mem\_type" id="new\_mem\_type">

<input type="hidden" name="operation" value="update">

<input type="submit" value="Update Member">

</form>

<form method="get" action="swimmingclasses.jsp">

<h3>Delete Swimming Member by Registration ID:</h3>

<label for="registration\_id\_delete">Registration ID:</label>

<input type="text" name="registration\_id" id="registration\_id\_delete" required>

<input type="hidden" name="operation" value="delete">

<input type="submit" value="Delete Member">

</form>

<form method="get" action="swimmingclasses.jsp">

<h3>Retrieve All Swimming Members:</h3>

<input type="hidden" name="operation" value="retrieve">

<input type="submit" value="Retrieve Members">

</form>

</div>

</body>

</html>

**swimmingclasses.jsp**

<%@page import="com.mongodb.client.MongoCursor"%>

<%@page import="com.mongodb.MongoCredential"%>

<%@page import="java.util.Iterator"%>

<%@page import="com.mongodb.client.FindIterable"%>

<%@page import="java.util.ArrayList"%>

<%@page import="org.bson.Document"%>

<%@page import="com.mongodb.client.MongoCollection"%>

<%@page import="com.mongodb.client.MongoDatabase"%>

<%@page import="com.mongodb.MongoClient"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ page import="com.mongodb.client.\*, com.mongodb.MongoClient, org.bson.Document" %>

<%@ page import="com.mongodb.client.MongoCursor" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Swimming Classes Membership Management</title>

<style>

body {

font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0;

padding: 0;

}

.container {

max-width: 800px; margin: 50px auto; padding: 20px; background-color: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.result {

padding: 15px;

background-color: #e9ecef; border-left: 5px solid #17a2b8; margin-bottom: 20px;

}

hr {

margin: 20px 0; border: none;

border-top: 1px solid #ddd;

}

</style>

</head>

<body>

<div class="container">

<h1>MongoDB Operation Results - Swimming Classes Membership Management</h1>

<%

String dbName = "swimmingClassesDb"; // Updated database name

String collectionName = "swimmingMembers"; // Updated collection name

MongoClient mongoClient = null;

MongoCollection<Document> collection = null;

try {

mongoClient = new MongoClient("localhost", 27017);

MongoDatabase database = mongoClient.getDatabase(dbName);

collection = database.getCollection(collectionName);

String operation = request.getParameter("operation");

if ("insert".equals(operation)) {

String name = request.getParameter("name");

String email = request.getParameter("email");

String registrationId = request.getParameter("registration\_id");

String memAddr = request.getParameter("mem\_addr");

String memType = request.getParameter("mem\_type");

if (name != null && !name.trim().isEmpty() &&

email != null && !email.trim().isEmpty() &&

registrationId != null && !registrationId.trim().isEmpty() &&

memAddr != null && !memAddr.trim().isEmpty() &&

memType != null && !memType.trim().isEmpty()) {

Document member = new Document("name", name)

.append("email", email)

.append("registration\_id", registrationId)

.append("mem\_addr", memAddr)

.append("mem\_type", memType);

collection.insertOne(member);

out.println("<div class='result'>Swimming member data inserted successfully!</div>");

} else {

out.println("<div class='result'>All fields are required.</div>");

}

} else if ("update".equals(operation)) {

String registrationId = request.getParameter("registration\_id");

String newName = request.getParameter("new\_name");

String newEmail = request.getParameter("new\_email");

String newMemAddr = request.getParameter("new\_mem\_addr");

String newMemType = request.getParameter("new\_mem\_type");

if (registrationId != null && !registrationId.trim().isEmpty()) {

Document query = new Document("registration\_id", registrationId);

Document update = new Document("$set", new Document("name", newName)

.append("email", newEmail)

.append("mem\_addr", newMemAddr)

.append("mem\_type", newMemType));

long modifiedCount = collection.updateOne(query, update).getModifiedCount();

if (modifiedCount > 0) {

out.println("<div class='result'>Swimming member details updated successfully!</div>");

} else {

out.println("<div class='result'>No member found with registration ID: " + registrationId + "</div>");

}

} else {

out.println("<div class='result'>Registration ID is required.</div>");

}

} else if ("delete".equals(operation)) {

String registrationId = request.getParameter("registration\_id");

if (registrationId != null && !registrationId.trim().isEmpty()) {

Document query = new Document("registration\_id", registrationId);

collection.deleteOne(query);

out.println("<div class='result'>Swimming member data deleted successfully!</div>");

} else {

out.println("<div class='result'>Registration ID is required.</div>");

}

} else if ("retrieve".equals(operation)) {

MongoCursor<Document> cursor = collection.find().iterator();

if (!cursor.hasNext()) {

out.println("<div class='result'>No swimming member records found.</div>");

} else {

out.println("<div class='result'><h3>Swimming Member Data:</h3>");

while (cursor.hasNext()) {

Document doc = cursor.next();

out.println("<strong>Name:</strong> " + doc.getString("name") + "<br>");

out.println("<strong>Email:</strong> " + doc.getString("email") + "<br>");

out.println("<strong>Registration ID:</strong> " + doc.getString("registration\_id") + "<br>");

out.println("<strong>Address:</strong> " + doc.getString("mem\_addr") + "<br>");

out.println("<strong>Membership Type:</strong> " + doc.getString("mem\_type") + "<br><hr>");

}

out.println("</div>");

}

cursor.close();

}

} catch (Exception e) {

out.println("<div class='result'>Error: " + e.getMessage() + "</div>");

} finally {

if (mongoClient != null) {

mongoClient.close();

}

}

%>

<a href="index.html">Go Back to Forms</a>

</div>

</body>

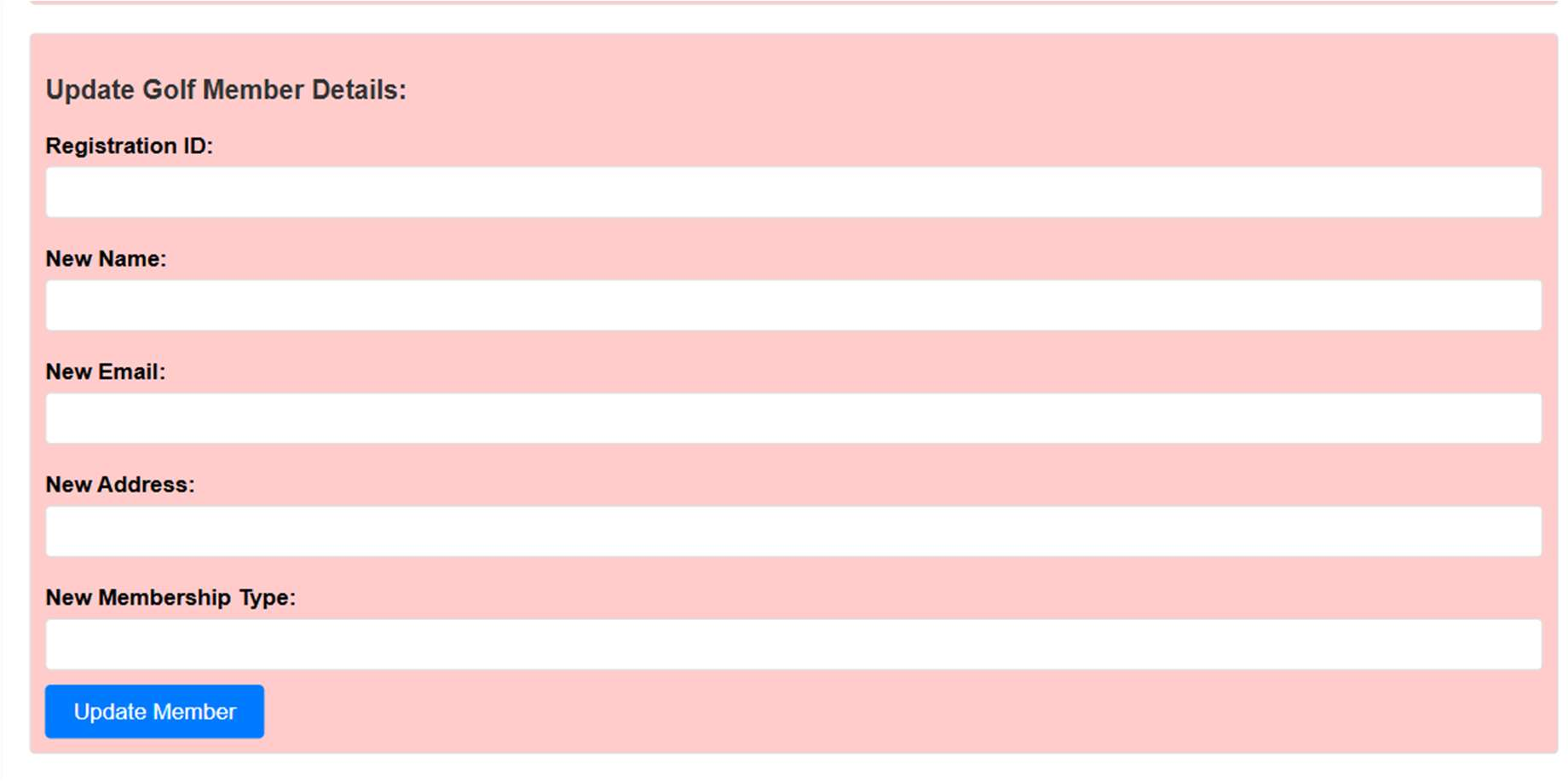
</html>

# OUTPUT SCREENSHOT

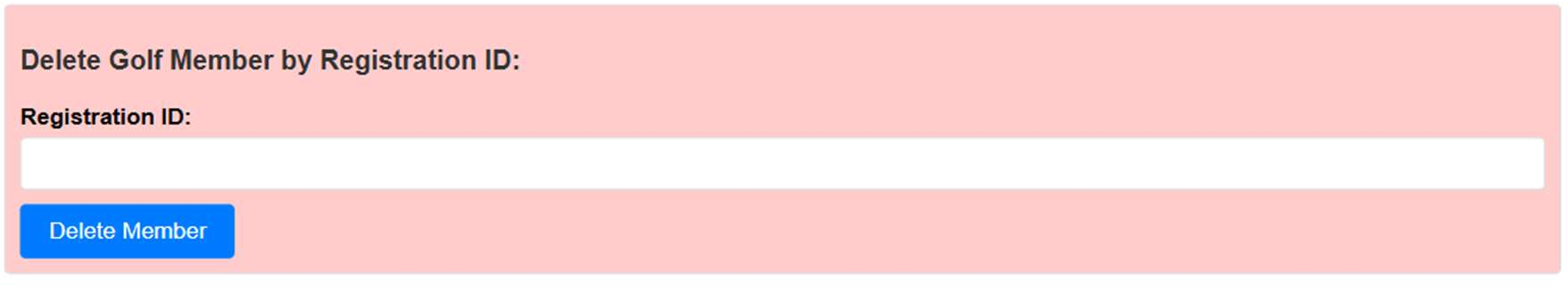
1. INSERT MEMBER DETAILS



1. UPDATE MEMBER DETAILS



1. DELETE MEMBER RECORD BY REGISTRATION ID



1. RETRIEVE MEMBER RECORDS FROM MONGODB DATABASE



**PERFORMING CRUD OPERATIONS:**

1. **Inserting a Member Record**



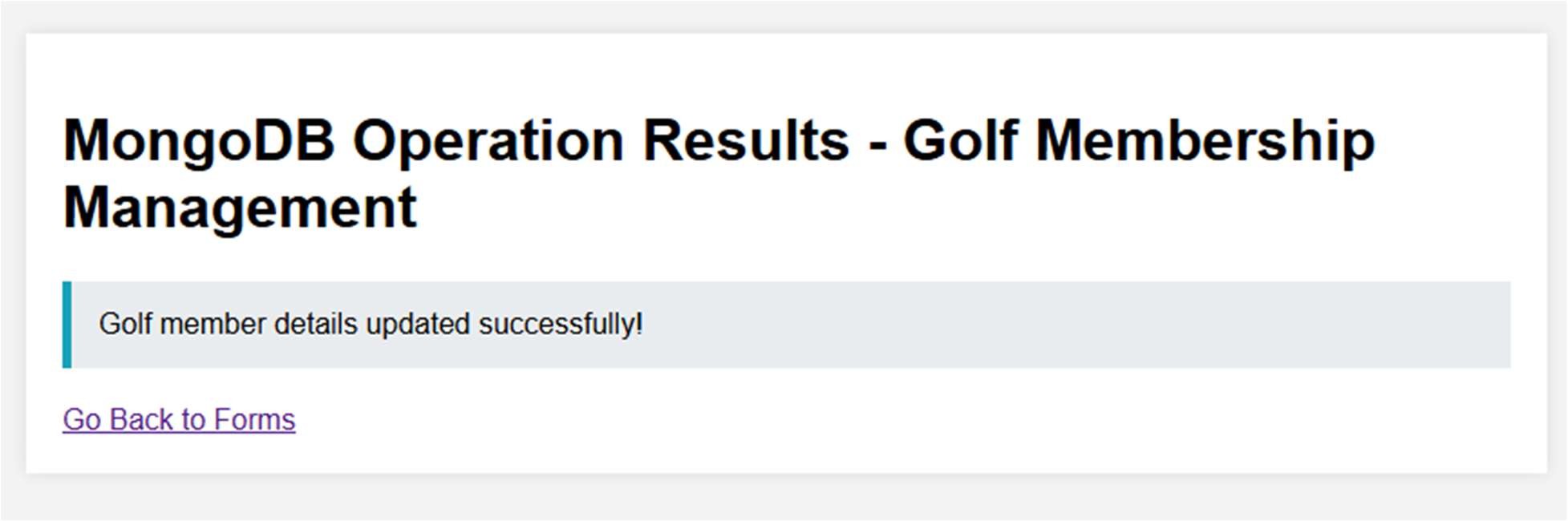
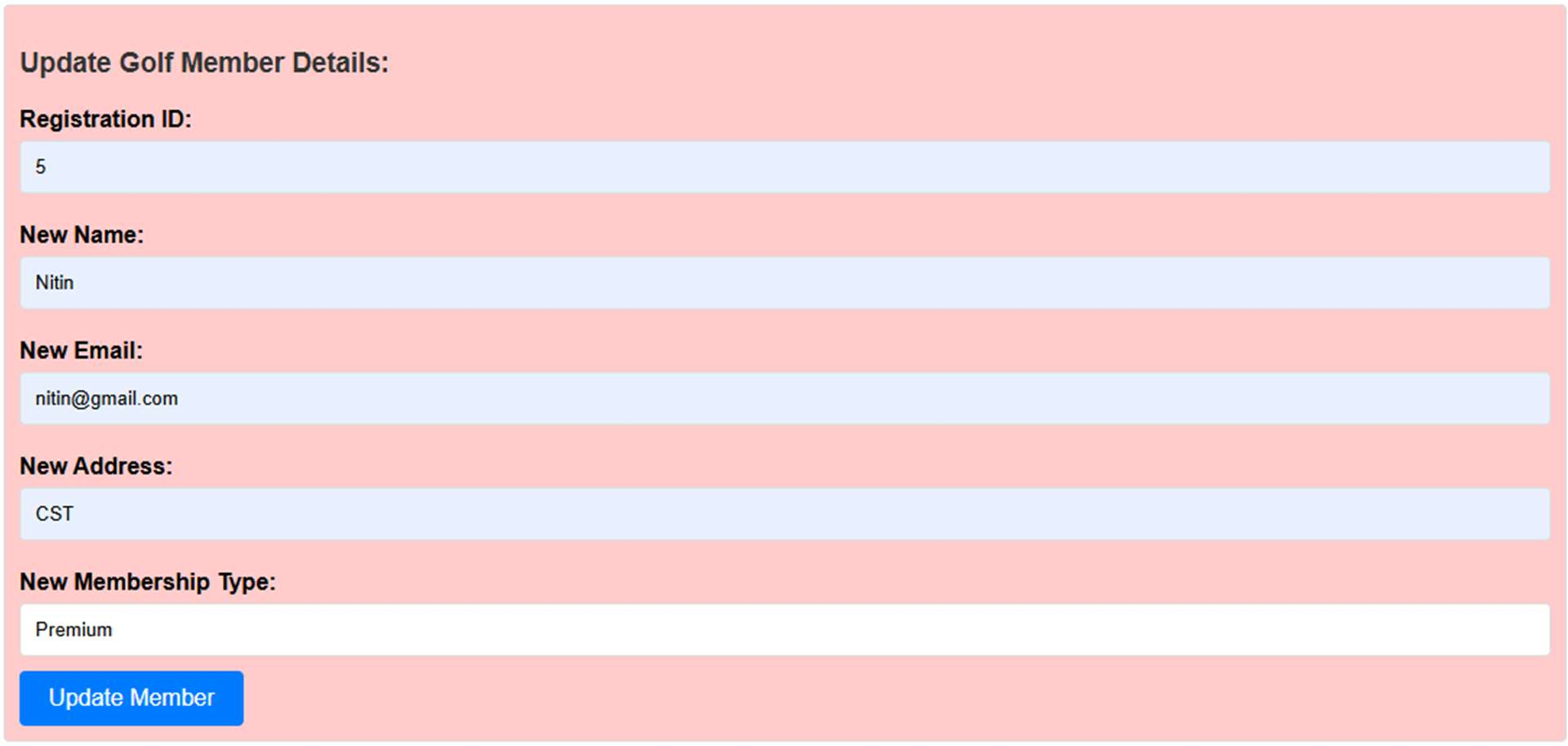
INSERTED A RECORD SUCCESSFULLY IN THE DATABASE

Now let’s check in client side the inserted records To view all inserted records the command is- db.golfmem.find()



## Updating a Member Record By Registration ID

We will update Nitin’s Record using their registration id and upgrade the membership from Gold to Premium



Now check the updated record in mongo using command db.golfmem.findOne({name:”Nitin”})



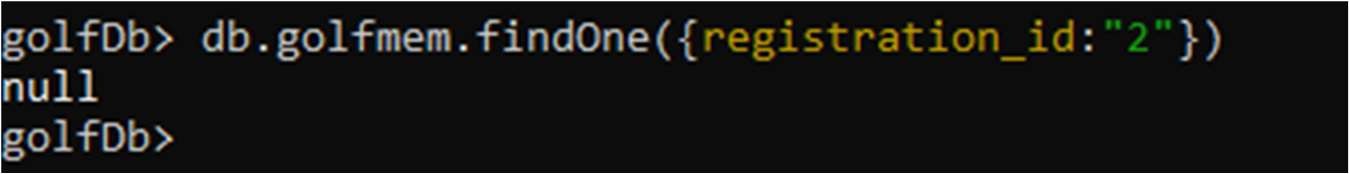
WE CAN SEE RECORD IS SUCCESSFULLY UPDATED

## Deleting a Member Record

Now let’s Delete the a record with registration id 2



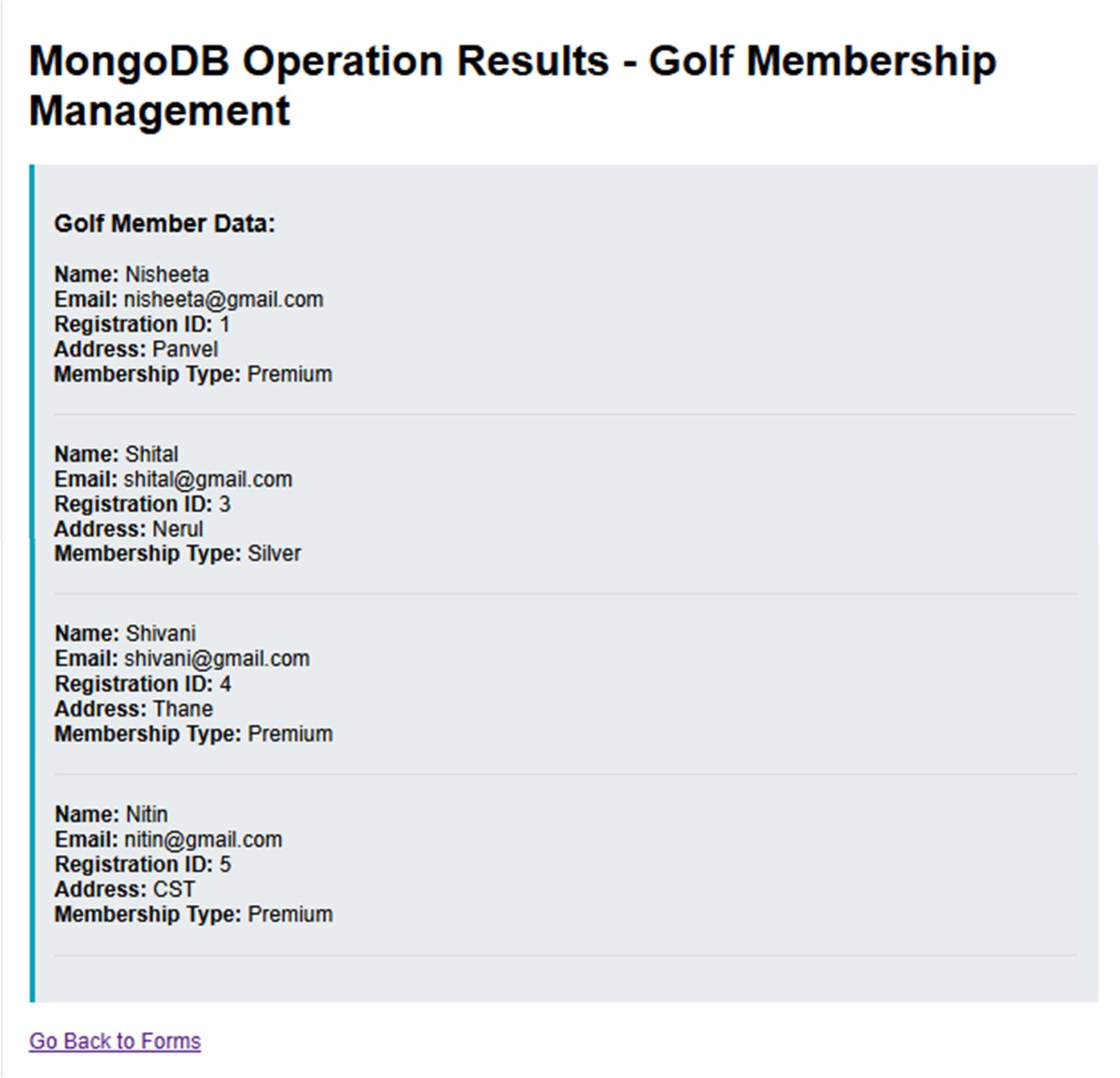
We can check on mongo to verify if record is deleted



WE HAVE SUCCESSFULLY DELETED THE RECORD OF REGISTRATION ID 2

## Retrieving All Member Records

Now let’s retrieve the All Member Records



Now we can also verify in Mongo



**CONCLUSION**

The "Swimming Classes Membership Management" system provides a comprehensive solution for managing membership records using MongoDB. By implementing essential database operations such as creating, updating, deleting, and retrieving member details, the system effectively streamlines administrative tasks while ensuring the accuracy and security of membership data. MongoDB's flexibility enables easy adaptation to new or evolving requirements, making it a versatile tool for managing the diverse needs of a swimming classes facility.

The integration of Java, HTML, and CSS ensures a user-friendly and intuitive interface, facilitating smooth interactions for both administrators and users. Overall, this project represents a robust and scalable solution for managing membership records, enhancing operational efficiency and improving the experience for both staff and members. With its strong foundation, the system is well-equipped to accommodate future enhancements and the growing demands of the facility.