## A

**MINOR PROJECT**

(Based on BCA5001P)

ON

## Gym Management System

Submitted By

*YUVRAJ GAUTAM*

(BACHELOR OF COMPUTER APPLICATION)

5th SEMESTER 2024-25

*Under the Guidance of*

*MR. SAURABH SACHAN SIR*



SIGNA INSTITUTE OF PROFESSIONAL STUDIES

##### Department of Computer Science Dharmangadpur Kisan Nagar-209304

CANDIDATE’S DECLARATION

I hereby declare that the minor project work being presented in this report entitled ***“GYM MANAGEMENT”*** submitted in the department of computer science, ***SIGNA INSTITUTE OF PROFESSIONAL STUDIES, Kanpur*** is the authentic work carried out by me under the guidance of ***MR. SAURABH SACHAN SIR***, Department of BCA.

Date / / ***YUVRAJ GAUTAM***

BCA 5TH SEM

# ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher ***MR. SAURABH SIR*** who gave me the golden opportunity to do this wonderful project on the topic ***GYM MANAGEMENT***, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Secondly I would also like to thank my friend ***RAJVEER RAJPUT*** who helped me a lot in finalizing this project within the limited time frame.

Date / / ***YUVRAJ GAUTAM***

BCA 5TH SEM

# SYNOPSIS

### TITLE OF THE PROJECT:

Gym Management System

### OBJECTIVE OF THE PROJECT:

To design and implement a comprehensive **gym management system** that streamlines gym operations, improves member experiences, and integrates advanced technologies for efficient fitness tracking and management.

### PROJECT CATEGORY:

RDBMS

### KEY FEATURES:

#### Member Management:

Easy registration and profile creation. Membership status and renewal notifications.

* **Workout Plan Customization:** Personalized workout plans for members. Progress tracking and real-time updates.

#### Attendance Tracking:

RFID or QR-based check-ins to monitor member activity.

#### Payment Integration:

Secure online payment options for memberships and additional services.

#### Trainer Module:

Trainers can assign workout plans and monitor member performance.

#### Fitness Analytics:

Interactive dashboards displaying workout trends and fitness progress.

#### IoT Integration:

Smart equipment tracking for detailed performance analysis.

### LANGUAGES AND SOFTWARE TOOL USED:

**Frontend:** React.js for web and Flutter for mobile apps.

**Backend:** Node.js or Django for robust server-side logic.

**Database:** MySQL or MongoDB for secure and scalable data storage.

**Additional Tools:** AI for fitness recommendations, IoT for smart equipment integration, and cloud storage for data.

### STRUCTURE OF THE PROJECT:

#### Proposed System:

The proposed system is a **modern, technology-driven solution** designed to address inefficiencies in traditional gym operations and improve member experiences. It integrates key functions like registration, attendance tracking, workout scheduling, and performance monitoring into a seamless and user-friendly platform.

#### Features of the Proposed System:

* + - * **Member Registration and Management:** Streamlined member sign-up and profile creation. Automated membership renewals and notifications.

#### Workout Planning and Tracking:

Personalized workout plans generated using fitness goals and AI- based suggestions.

Progress tracking and feedback for members.

#### Attendance Monitoring:

QR code or RFID-based check-ins for tracking gym visits. Attendance logs available for both members and trainers.

#### Trainer Module:

Trainers can assign, edit, and monitor workout plans for individual members.

Performance insights to evaluate member progress.

#### Payment and Billing System:

Integrated payment gateways for memberships and services. Automated invoicing and payment reminders.

#### Real-Time Analytics and Dashboards:

Insights into member activity, equipment usage, and revenue trends. Dashboards for trainers and administrators to monitor performance.

#### IoT Integration:

Smart equipment sensors to monitor usage and track member performance.

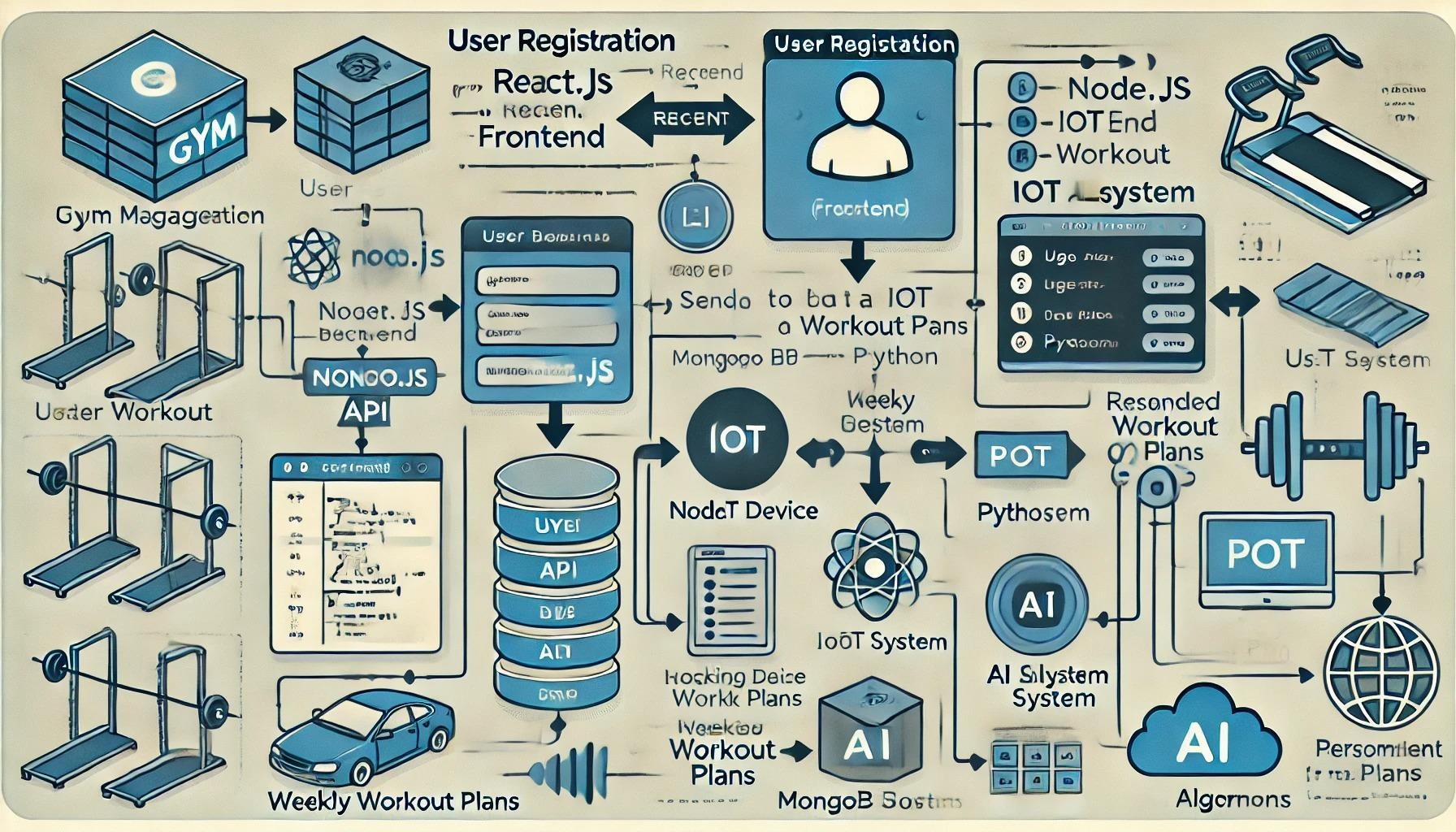
Data collected is synced with the system for analysis.

#### Advantages of the Proposed System:

* + - * **Efficiency:** Reduces manual effort in managing gym operations.
      * **Convenience:** Enables members to access workout plans and attendance records on mobile apps.
      * **Personalization:** Provides tailored workout plans for individual fitness goals.
      * **Transparency:** Keeps clear records of attendance, payments, and workout progress.
      * **Scalability:** Can support multiple gym branches with cloud- based data storage.

# FLOW CHART

##### Here is the flowchart representing the gym management system. It illustrates the interactions between the frontend, backend, database, IoT devices, AI algorithm, and cloud storage for analytics. Let me know if you'd like to make any adjustments!



***SOURCE CODE:***

## Frontend(React.js)

Feature: Member Registration Form

import React, { useState } from "react"; function RegisterUser() {

const [formData, setFormData] = useState({ name: "", email: "", password: "" });

const handleSubmit = async (e) => { e.preventDefault();

const response = await fetch("http://localhost:5000/api/register", { method: "POST",

headers: { "Content-Type": "application/json" }, body: JSON.stringify(formData),

});

const data = await response.json(); alert(data.message);

};

return (

<form onSubmit={handleSubmit}>

<h2>Register</h2>

<input type="text"

placeholder="Name"

onChange={(e) => setFormData({ ...formData, name: e.target.value })} required

/>

<input type="email"

placeholder="Email"

onChange={(e) => setFormData({ ...formData, email: e.target.value })} required

/>

<input type="password"

placeholder="Password"

onChange={(e) => setFormData({ ...formData, password: e.target.value })} required

/>

<button type="submit">Register</button>

</form>

);

}

export default RegisterUser;

## Output:

* A simple form where users input their name, email, and password.
* Upon submission, data is sent to the backend API, and a success message is shown.

## Backend(Node.js)

Feature: User Registration API MongoDB

const express = require("express"); const mongoose = require("mongoose");

const bodyParser = require("body-parser");

const app = express(); app.use(bodyParser.json());

// MongoDB connection

mongoose.connect("mongodb://localhost:27017/gym", { useNewUrlParser: true, useUnifiedTopology: true });

const userSchema = new mongoose.Schema({ name: String,

email: String, password: String,

});

const User = mongoose.model("User", userSchema);

// Registration API

app.post("/api/register", async (req, res) => { const { name, email, password } = req.body;

const newUser = new User({ name, email, password }); await newUser.save();

res.json({ message: "User registered successfully!" });

});

app.listen(5000, () => console.log("Server running on port 5000"));

## Output:

* The backend API saves user details to MongoDB when the React form sends data.
* A success response ("User registered successfully!") is sent back.

## Database(MySQL)

Feature: Storing Member Details

CREATE DATABASE gym\_management; USE gym\_management;

CREATE TABLE members (

id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100),

email VARCHAR(100) UNIQUE,

password VARCHAR(255),

joined\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

INSERT INTO members (name, email, password) VALUES ("John Doe", "[john@example.com](mailto:john@example.com)", "hashed\_password");

## Output:

* A database table members is created to store user information.
* Member details are saved when the backend receives a registration request.

## IoT Intregation(Python)

Feature: Tracking Equipement Usage

import time

from random import randint

def simulate\_equipment\_usage(): while True:

usage\_data = { "equipment\_id": 101,

"reps": randint(5, 15),

"time\_spent": randint(10, 60) # seconds

}

print("Sending data to server:", usage\_data) # Simulate sending data to backend time.sleep(5)

simulate\_equipment\_usage()

## Output:

* Simulates gym equipment usage (reps and time spent).
* Prints the data, which can be sent to a backend server for logging and analytics.

## AI Algorithm(Python)

Feature: Suggesting A Workout Plan

from sklearn.cluster import KMeans import numpy as np

# Simulated data: [age, weekly workout hours] data = np.array([

[25, 5], [30, 3], [35, 7], [40, 2], [22, 6],

[28, 4], [32, 8], [45, 1], [29, 5]

])

# Clustering similar users

kmeans = KMeans(n\_clusters=3, random\_state=0).fit(data)

def recommend\_workout(age, hours):

cluster = kmeans.predict([[age, hours]])[0]

plans = ["Beginner Plan", "Intermediate Plan", "Advanced Plan"] return plans[cluster]

# Example recommendation

print(recommend\_workout(27, 4)) # Output: e.g., "Intermediate Plan"

## Output:

Recommends a workout plan based on user input (age and workout hours).

## Cloud Storage

Feature: Uploading Workout Data

const AWS = require('aws-sdk'); const s3 = new AWS.S3();

const uploadToS3 = (filePath, bucketName, key) => {

const fileContent = fs.readFileSync(filePath); const params = {

Bucket: bucketName, Key: key,

Body: fileContent

};

s3.upload(params, (err, data) => { if (err) throw err;

console.log('File uploaded successfully:', data.Location);

});

};