BASAVARAJESWARI GROUP OF INSTITUTIONS

**Ballari Institute of Technology& Management**

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA, BELAGAVI 590018

**INTERNSHIP**

**Report On**

# FITNESS CLASS BOOKING SYSTEM

Submitted in partial fulfilment of the requirements for the award of degree of

## Bachelor of Engineering

**In**

## COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

**Submitted by**

**PRAVEEN**

**3BR23CD067**

**Internship Carried Out By**

**EZ TRAININGS & TECHNOLOGIES PVT.LTD HYDERABAD**

**Internal Guide**

**V P ANUSHYA**

**Asst. prof, CSE-DS**

**KAMALAPADU VARSHA**

**Teaching Asst. CSE-DS**

**External Guide**

**VISHAL KUMAR**

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

**BELAGAVI 590018**

NACC Accredited Institution*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)**

"JnanaGangotri"Campus,No.873/2,Ballari-HospetRoad,Allipur,

**Ballar1-583 104 (Karnataka)(India)**

**Ph: 08392 – 237100 / 237190, Fax: 08392 –23719**

BASAVARAJESWARI GROUP OF INSTITUTIONS

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**

Autonomous institute under VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA,

BELAGAVI 590018

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri"Campus,No.873/2,Ballari-HospetRoad,Allipur,

Ballar1-583 104 (Karnataka)(India)

Ph: 08392 – 237100 / 237190, Fax: 08392 –237197

## DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING – DATA SCIENCE

# CERTIFICATE

This is to certify that the Internship entitled **"FITNESS CLASS BOOKING SYSTEM"** has been successfully completed by **PRAVEEN** bearing USN **3BR23CD067** a bonafide student of Ballari Institute of Technology and Management, Ballari. For the partial fulfilment of the requirements for the **Bachelor's Degree in Computer Science and Engineering – Data Science** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2024-2025.

**Signature of Internship Co-ordinator**

**V P ANUSHYA**

**Asst. prof, CSE-DS**

**KAMALAPADU VARSHA**

**Teaching Asst. CSE-DS**

**Signature of HOD**

**DR. ARADHANA**

**Prof. and HOD(CSE-DS)**

# DECLARATION

I, PRAVEEN**,** second year student of Computer Science and Engineering, Ballari Institute of Technology, Ballari, declare that Internship entitled **FITNESS CLASS BOOKING SYSTEM** is a part of Internship Training successfully carried out by **EZ TECHNOLOGIES & TRAININGS PVT.LTD,**

**Hyderabad** at "**BITM, BALLARI".** This report is submitted in partial fulfilment of the requirements for the award of the degree, Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi.

**Date : 28/09/2024** **Signature of the Student**
**Place : BALLARI**

# ACKNOWLEDGEMENT

The satisfactions that a company the successful completion of my internship on "FITNESS CLASS BOOKING SYSTEM" would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is my privilege to express my gratitude and respect to all those who inspired me in the completion of my internship.

I am grateful to our respective coordinator **"V P ANUSHYA (Asst. Prof, CSE-DS), Kamalapadu Varsha (Teaching. Asst, CSE_DS)"** for his noble gesture, support co-ordination and valuable suggestions given to me in the completion of Internship.

I also thank **DR ARADHANA,** H.O.D. Department of **Computer science and engineering-Data Science** for extending all his valuable support and encouragement.

| Day | Date | Content Covered | Signature of the faculty in-charge |
|---|---|---|---|
| | 09.09.24 | Introduction to Python, Setup & Installation, First Python Program, Variables, Data Types, and Basic I/O | |
| 2 | 10.09.24 | Control Structures: If-else, Loops, Functions and Modules | |
| 3 | 11.09.24 | Lists, Tuples, and Dictionaries, File Handling | |
| 4 | 12.09.24 | Exception Handling, Practice exercises on Python basics | |
| 5 | 13.09.24 | Introduction to OOP, Classes, and Objects | |
| 6 | 14.09.24 | Inheritance, Polymorphism, and Encapsulation | |
| 7 | 15.09.24 | Abstract Classes and Interfaces | |
| 8 | 17.09.24 | Practice exercises on OOP concepts | |
| 9 | 18.09.24 | Introduction to DSA, Arrays, and Linked Lists | |
| 10 | 19.09.24 | Introduction to DSA, Arrays, and Linked Lists | |
| 11 | 20.09.24 | Introduction to stacks and Queue | |
| 12 | 21.09.24 | Practice Exercise on basic concept (Reduce, Lambda, Function, List Comprehension) | |
| 13 | 23.09.24 | Introduction to Tree Data structures | |
| 14 | 24.09.24 | Introduction to Graph Data Structures | |
| 15 | 25.09.24 | Searching and Sorting Algorithms Project Building & Presentations | |
| 16 | 26.09.24 | Project Building & Presentations | |
| 17 | 27.09.24 | Project Building & Presentations | |
| 18 | 28.09.24 | Project Building & Presentations | |

**Table of Contents**


| Chapter No. | Chapter Name | Page No. |
|---|---|---|
| 1 | Day to day activity (student diary extract) | 01-02 |
| 2 | Company Profile | 03 |
| 3 | Abstract | 04 |
| 4 | Introduction of the project | 05 |
| 5 | Description | 06-07 |
| 6 | Algorithm | 08-09 |
| 7 | Flowchart | 10 |
| 8 | Source Code | 11-15 |
| 9 | Output | 16-20 |
| 10 | Conclusion | 21 |
| 11 | References | 22 |

# COMPANY PROFILE

## Company Name: EZ Trainings and Technologies Pvt. Ltd.

### Introduction:

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

### Mission:

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

### Services:

### College Trainings:

- Tailored training programs designed to enhance the employability of students.
- Industry-aligned curriculum covering technical and soft skills.
- Placement assistance and career guidance.

### Development Projects:

- End-to-end development services, from ideation to execution.
- Expertise in diverse technologies and frameworks.
- Custom solutions to meet specific business needs.

**Locations:** Hyderabad | Delhi NCR

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

# ABSTRACT


- The Fitness Class Booking System is a comprehensive software solution designed to streamline the management of fitness class offerings in gyms and fitness centers. This system employs object-oriented programming principles in Python to create a modular and efficient framework for handling various aspects of class management, including creating, reading, updating, and deleting fitness class listings (CRUD operations).
- Key functionalities include the ability to manage bookings, where members can reserve spots in fitness classes while the system tracks available capacity. Additionally, the system allows for attendance tracking, recording the participants of each class to assess popularity and optimize scheduling.
- The implementation is encapsulated within three primary classes: FitnessClass, representing individual fitness classes; FitnessCenter, which manages the overall class listings and operations; and FitnessClassBookingSystem, focusing on booking management and attendance tracking.
- A user-friendly menu interface facilitates interaction, enabling gym staff and members to easily navigate through class options and perform operations efficiently. Unit tests using Python's unittest framework ensure the reliability and correctness of the system, making it a practical solution for modern fitness facilities.

# INTRODUCTION OF THE PROJECT

In today's fast-paced world, the fitness industry has seen significant growth, with more individuals seeking structured workout programs to achieve their health and wellness goals. Gyms and fitness centers, such as Planet Fitness and Gold's Gym, are responding to this demand by offering a variety of fitness classes, from yoga and spin classes to strength training and aerobics. As a result, the management of class schedules, bookings, and attendance has become increasingly complex and challenging.

To address these challenges, the Fitness Class Booking System is designed to provide a robust solution for managing fitness class offerings efficiently. This system leverages object-oriented programming (OOP) principles in Python to create an intuitive, modular, and scalable application that can easily adapt to the needs of different fitness facilities.

The core functionalities of this system include:

- **Class Listings Management (CRUD Operations):** The system enables gym staff to create, read, update, and delete fitness class listings. Each class is represented by the FitnessClass class, which encapsulates essential attributes such as class ID, name, capacity, booked spots, and attendance records. This structure allows for easy management and retrieval of class information.
- **Booking Management:** Members can book spots in fitness classes through a straightforward booking interface. The system tracks the number of booked spots against the class capacity, ensuring that overbooking does not occur. This functionality enhances the member experience by allowing individuals to secure their spots in desired classes while minimizing confusion.
- **Attendance Tracking:** Post-class attendance tracking is vital for evaluating class popularity and effectiveness. The system records the actual attendees of each class, providing valuable data that can be analyzed to improve class offerings, adjust schedules, and enhance overall member satisfaction.
- **User-Friendly Interface:** The application includes a simple menu-driven interface that facilitates seamless interaction between users and the system. Gym staff can easily navigate through various operations, such as adding new classes, managing bookings, and tracking attendance, without requiring extensive technical knowledge.
- **Unit Testing:** To ensure the reliability and functionality of the system, comprehensive unit tests are incorporated using Python's unittest framework. This testing phase validates the various operations and interactions within the system, ensuring that each component functions as intended and that any potential bugs are identified and resolved early.

By implementing this Fitness Class Booking System, gyms and fitness centers can enhance operational efficiency, improve member engagement, and ultimately contribute to a more organized and enjoyable fitness experience. As the demand for fitness services continues to rise, this system offers a scalable and effective solution to meet the evolving needs of the industry.

# MODULE DESCRIPTION

The Fitness Class Booking System is structured into several key modules, each designed to fulfill specific functions within the overall application. These modules work together to create a cohesive and efficient system for managing fitness classes, bookings, and attendance tracking. Below is a detailed description of each module:

1. FitnessClass Module

- Purpose: This module represents individual fitness classes offered by the gym or fitness center. It encapsulates class-specific data and behavior, allowing for the management of class attributes and operations related to the class itself.
- Key Components:
    - Attributes:
        - class_id: A unique identifier for each fitness class.
        - name: The name of the fitness class (e.g., "Yoga", "Spin Class").
        - capacity: The maximum number of participants allowed in the class.
        - booked_spots: The number of spots that have been reserved for the class.
        - attendance_list: A list that tracks the attendees for each session.
    - Methods:
        - __init__: Initializes a new fitness class with the specified ID, name, and capacity.
        - update_class_details: Allows updates to the class name and capacity.
        - available_spots: Returns the number of available spots for the class.
        - __repr__: Provides a string representation of the fitness class for easy display.

2. FitnessCenter Module

- Purpose: This module serves as the primary management interface for all fitness classes. It provides CRUD operations and manages bookings and attendance, facilitating the interaction between users and the class listings.
- Key Components:
    - Attributes:
        - classes: A dictionary that stores all the fitness classes, indexed by their unique class_id.
    - Methods:
        - create_class: Adds a new fitness class to the system, ensuring the class ID is unique.
        - read_class: Retrieves and displays the details of a specific fitness class by its ID.
        - update_class: Updates the details (name and capacity) of an existing class.
        - delete_class: Removes a class from the system using its ID.
        - manage_bookings: Handles the booking process for a class, updating the number of booked spots and checking availability.

        - track_class_attendance: Records the attendees of a class and updates the attendance list.

3. FitnessClassBookingSystem Module

- Purpose: This module coordinates the overall functionality of the booking system, integrating the FitnessClass and FitnessCenter modules. It provides a menu-driven interface that allows users to interact with the system easily.
- Key Components:
    - Attributes: None (this module primarily serves as a facilitator for interaction).
    - Methods:
        - A series of user interaction methods that present options for class management, booking, and attendance tracking through a console-based menu system.
        - Input validation to ensure correct and meaningful user inputs.

4. Unit Testing Module

- Purpose: This module employs Python's unittest framework to validate the functionality of the Fitness Class Booking System. It ensures that all components work correctly and meet specified requirements.
- Key Components:
    - Test Cases: A series of unit tests that cover:
        - Creation, retrieval, updating, and deletion of fitness classes.
        - Booking management functionality to confirm that bookings are processed correctly and that availability is respected.
        - Attendance tracking to verify that attendee data is recorded and reported accurately.
    - Assertions: Use of assertions to validate expected outcomes against actual results, ensuring that any bugs or issues are identified during the development process.

# ALGORITHM

- **Initialization**

- Create an instance of the FitnessCenter class.

- **Main Menu Loop**

- Repeat the following steps until the user chooses to exit:

    1. Display the menu options:

        1. Create a class
        2. View a class
        3. Update a class
        4. Delete a class
        5. Book a spot in a class
        6. Track class attendance
        7. Exit

    2. Get user input for menu selection.
    3. Based on user input, perform the corresponding operation:

        1. **Option 1: Create a Class**
            - Get class_id, name, and capacity from user input.
            - Call create_class(class_id, name, capacity) method of the FitnessCenter class.
            - Display the result message.
        2. **Option 2: View a Class**
            - Get class_id from user input.
            - Call read_class(class_id) method of the FitnessCenter class.
            - Display the class details or a "not found" message.
        3. **Option 3: Update a Class**
            - Get class_id from user input.
            - Get new name (or leave blank to keep unchanged).
            - Get new capacity (or leave blank to keep unchanged).
            - Call update_class(class_id, name, capacity) method of the FitnessCenter class.
            - Display the result message.

4. **Option 4: Delete a Class**
    - Get class_id from user input.
    - Call delete_class(class_id) method of the FitnessCenter class.
    - Display the result message.
5. **Option 5: Book a Spot in a Class**
    - Get class_id from user input.
    - Call manage_bookings(class_id) method of the FitnessCenter class.
    - Display the result message regarding booking status.

**Option 6: Track Class Attendance**

- Get class_id from user input.
- Get a list of attendees (comma-separated) from user input.
- Convert the input into a list of names.
- Call track_class_attendance(class_id, attendees) method of the FitnessCenter class.
- Display the result message regarding attendance recording.

6. **Option 7: Exit**
    - Print an exit message and break the loop.

4. If the user inputs an invalid option, display an error message and prompt to try again.

# SOURCE CODE

```
# FitnessClass class for class listings
class FitnessClass:
    def __init__(self, class_id, name, capacity):
        self.class_id = class_id
        self.name = name
        self.capacity = capacity
        self.booked_spots = 0
        self.attendance_list = []

    def update_class_details(self, name=None, capacity=None):
        if name:
            self.name = name
        if capacity:
            self.capacity = capacity

    def available_spots(self):
        return self.capacity - self.booked_spots

    def __repr__(self):
        return f"Class: {self.name}, Capacity: {self.capacity}, Booked: {self.booked_spots}"

# FitnessCenter class for managing bookings and attendance
class FitnessCenter:
    def __init__(self):
        self.classes = {}

    # CRUD operations for class listings
    def create_class(self, class_id, name, capacity):
        if class_id in self.classes:
            return f"Class ID {class_id} already exists."
        self.classes[class_id] = FitnessClass(class_id, name, capacity)
        return f"Class {name} created with ID {class_id}."
```

```
def read_class(self, class_id):
    if class_id in self.classes:
        return self.classes[class_id]
    return "Class not found."

def update_class(self, class_id, name=None, capacity=None):
    if class_id in self.classes:
        self.classes[class_id].update_class_details(name, capacity)
        return f"Class {class_id} updated."
    return "Class not found."

def delete_class(self, class_id):
    if class_id in self.classes:
        del self.classes[class_id]
        return f"Class {class_id} deleted."
    return "Class not found."

# Manage bookings for classes
def manage_bookings(self, class_id):
    if class_id not in self.classes:
        return "Class not found."
    fitness_class = self.classes[class_id]
    if fitness_class.available_spots() > 0:
        fitness_class.booked_spots += 1
        return f"Booked spot in {fitness_class.name}. Remaining spots: {fitness_class.available_spots()}."
    return f"Class {fitness_class.name} is fully booked."

# Track class attendance
def track_class_attendance(self, class_id, attendees):
    if class_id not in self.classes:
        return "Class not found."
    fitness_class = self.classes[class_id]
    fitness_class.attendance_list = attendees
    return f"Attendance for class {fitness_class.name} recorded: {len(attendees)} attendees."
```

```
# Menu loop and interaction
fc = FitnessCenter()  # Instantiate the FitnessCenter object

while True:
    print("\n--- Fitness Center Menu ---")
    print("1. Create a class")
    print("2. View a class")
    print("3. Update a class")
    print("4. Delete a class")
    print("5. Book a spot in a class")
    print("6. Track class attendance")
    print("7. Exit")

    choice = input("Choose an option (1-7): ")

    if choice == "1":
        class_id = int(input("Enter class ID: "))
        name = input("Enter class name: ")
        capacity = int(input("Enter class capacity: "))
        result = fc.create_class(class_id, name, capacity)
        print(result)

    elif choice == "2":
        class_id = int(input("Enter class ID to view: "))
        result = fc.read_class(class_id)
        print(result)

    elif choice == "3":
        class_id = int(input("Enter class ID to update: "))
        name = input("Enter new class name (leave blank to keep unchanged): ")
        capacity = input("Enter new class capacity (leave blank to keep unchanged): ")
        capacity = int(capacity) if capacity else None
        result = fc.update_class(class_id, name, capacity)
        print(result)
```

```
elif choice == "4":
    class_id = int(input("Enter class ID to delete: "))
    result = fc.delete_class(class_id)
    print(result)

elif choice == "5":
    class_id = int(input("Enter class ID to book a spot: "))
    result = fc.manage_bookings(class_id)
    print(result)

elif choice == "6":
    class_id = int(input("Enter class ID to track attendance: "))
    attendees = input("Enter attendees (comma-separated): ").split(",")
    attendees = [attendee.strip() for attendee in attendees]
    result = fc.track_class_attendance(class_id, attendees)
    print(result)

elif choice == "7":
    print("Exiting the Fitness Center app.")
    break

else:
    print("Invalid option. Please try again.")
```

# OUTPUT

--- Fitness Center Menu ---

1. Create a class
2. View a class
3. Update a class
4. Delete a class
5. Book a spot in a class
6. Track class attendance
7. Exit

Choose an option (1-7): 1

Enter class ID: 101

Enter class name: Yoga

Enter class capacity: 20

Class Yoga created with ID 101.

--- Fitness Center Menu ---

1. Create a class
2. View a class
3. Update a class
4. Delete a class
5. Book a spot in a class
6. Track class attendance
7. Exit

Choose an option (1-7): 2

Enter class ID to view: 101

Class: Yoga, Capacity: 20, Booked: 0

--- Fitness Center Menu ---

1. Create a class
2. View a class
3. Update a class
4. Delete a class
5. Book a spot in a class
6. Track class attendance

7. Exit

Choose an option (1-7): 5

Enter class ID to book a spot: 101

Booked spot in Yoga. Remaining spots: 19.

--- Fitness Center Menu ---

1. Create a class

2. View a class

3. Update a class

4. Delete a class

5. Book a spot in a class

6. Track class attendance

7. Exit

Choose an option (1-7): 6

Enter class ID to track attendance: 101

Enter attendees (comma-separated): Alice, Bob

Attendance for class Yoga recorded: 2 attendees.

--- Fitness Center Menu ---

1. Create a class

2. View a class

3. Update a class

4. Delete a class

5. Book a spot in a class

6. Track class attendance

7. Exit

Choose an option (1-7): 4

Enter class ID to delete: 101

Class 101 deleted.

--- Fitness Center Menu ---

1. Create a class

2. View a class

3. Update a class

4. Delete a class

5. Book a spot in a class

6. Track class attendance

7. Exit

Choose an option (1-7): 7

Exiting the Fitness Center app.

# CONCLUSION

The Fitness Class Booking System designed in this implementation effectively demonstrates the principles of object-oriented programming (OOP) through its structured class definitions and modular design. By utilizing classes such as FitnessClass, FitnessCenter, and a clear interaction loop, this system allows for robust management of fitness classes, including creating, reading, updating, and deleting class listings (CRUD operations).

Key Features:

1. Class Management:
   - The FitnessClass class encapsulates all necessary attributes and methods related to a single fitness class, including class ID, name, capacity, booked spots, and an attendance list. This modular approach allows for easy expansion and maintenance of class-related functionality.
2. Booking Management:
   - The manage_bookings method in the FitnessCenter class handles bookings efficiently by checking the availability of spots before allowing new bookings. This ensures that users cannot overbook classes, thereby enhancing user experience and operational efficiency.
3. Attendance Tracking:
   - The ability to track attendance through the track_class_attendance method provides valuable insights into class popularity. This data can inform future scheduling and marketing strategies, enabling gyms to optimize their offerings based on member preferences.
4. User Interaction:
   - The menu-driven interaction allows users to easily navigate through the various functionalities of the system, making it user-friendly. This approach simplifies the user experience and encourages engagement with the system.
5. Data Integrity:
   - The system includes validation checks, such as confirming the existence of a class before performing operations, which helps maintain data integrity and prevents errors.

Potential Enhancements:

While the current implementation is functional, there are several areas for potential enhancement:

- Unit Testing: Introducing unit tests using Python's unittest framework would ensure that each component of the system works as intended, providing confidence in the system's reliability.
- User Authentication: Implementing user accounts with authentication could allow personalized experiences, where members can view their booking history and manage their profiles.
- Notifications: Adding a notification system for reminders about upcoming classes or confirmations of bookings could enhance user engagement.
- Graphical User Interface (GUI): Transitioning from a command-line interface to a GUI would make the system more accessible to users who prefer visual interactions.

In summary, the Fitness Class Booking System serves as a foundational example of how OOP concepts can be applied to solve real-world problems in fitness management. By managing class listings, bookings, and attendance effectively, the system not only streamlines operations for fitness centers but also enhances the overall member experience. Future developments and enhancements can build upon this solid foundation to create a more comprehensive and user-friendly system.

# REFERENCES

→Google

→ChatGPT