SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
(An Autonomous Institution)
Adyar, Mangaluru-575007
2023-24

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

A PROJECT REPORT

ON

GYM MANAGEMENT SYSTEM

BY

Abhijith A
4SF21CS004

Swasthik P Gowda
4SF21CS052

In the partial fulfillment of the requirement for V Sem. B. E. (CSE)

DBMS LABORATORY WITH MINI PROJECT

Under the guidance of

Ms. Ashwini C S
Assistant Professor, Dept. of CS&E

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
(An Autonomous Institution)
Adyar, Mangaluru-575007

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**Gym Management System"** is submitted in partial fulfillment for the requirement of V sem. B. E. (Computer Science & Engineering), "DATABASE MANAGEMENT SYSTEMS LABORATORY" during the year 2023–24 is a result of bonafide work carried out by

Abhijith A	**4SF21CS004**
Swasthik P Gowda	**4SF21CS052**

……	**……..**
Ms. Ashwini C S	**Dr. Mustafa Basthikodi**
Asst. Prof. Dept. of CS&E	**HOD, Dept. of CS&E**
SCEM, Mangaluru	**SCEM, Mangaluru**

Signature of the Examiners

1. …….

2. …….

ABSTRACT

The Gym Management System (GMS) is a comprehensive software solution designed to streamline the operations of fitness centers and promote efficient management of gym facilities. Developed using Java NetBeans and MySQL, the system caters to the needs of gym owners, trainers, and members, offering a user-friendly interface and robust backend functionality.

Key features of the Gym Management System include member registration, subscription management, equipment tracking, and exercise scheduling. The system allows gym owners to efficiently manage resources, track member attendance, and optimize equipment utilization, thereby enhancing operational efficiency and customer satisfaction.

Moreover, the GMS provides data analytics capabilities, allowing gym owners to gain valuable insights into member engagement, equipment usage trends, and subscription patterns. This data-driven approach enables informed decision-making and facilitates continuous improvement initiatives to better meet the needs of gym members.

Overall, the Gym Management System represents a transformative tool for gym owners and members alike, offering a seamless platform for managing fitness facilities and promoting healthy lifestyles. With its emphasis on usability, efficiency, and data-driven insights, the system aims to revolutionize the fitness industry and improve the overall gym experience for users.

ACKNOWLEDGEMENT

It is with great satisfaction and euphoria that we are submitting the Mini Project Report On **"Gym Management System"** We have completed it as a part of the V semester DATABASE MANAGEMENT SYSTEMS LABORATORY (21CSL55) of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Ms. Ashwini C S, Assistant Professor, Department of Computer Science & Engineering** for her innumerable acts of timely advice, encouragement and we sincerely express our gratitude.

We express our sincere gratitude to **Dr. Mustafa Basthikodi, Professor & Head of the Department of Computer Science & Engineering** for his invaluable support and guidance.

We sincerely thankful to our beloved **Principal Dr. S S Injaganeri, Sahyadri College of Engineering & Management,** who have always been a great source of inspiration.

Finally, yet importantly, we express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

ABHIJITH A
4SF21CS004
V Sem, B.E., CSE
SCEM, Mangaluru

SWASTHIK P GOWDA
4SF21CS052
V Sem, B.E., CSE
SCEM, Mangaluru

PAGE INDEX

CHAPTER 1

INTRODUCTION

In today's fast-paced world, maintaining a healthy lifestyle is becoming increasingly important. Gyms play a pivotal role in helping individuals achieve their fitness goals by providing access to state-of-the-art equipment, expert guidance from trainers, and a supportive community environment. However, managing a gym efficiently and effectively can be a challenging task.

The Gym Management System (GMS) emerges as a comprehensive solution to address the complexities of gym administration and member management. Developed to streamline operations and enhance the overall gym experience, the GMS leverages technology to simplify tasks such as member registration, subscription management, equipment tracking, and workout scheduling.

By centralizing critical gym functions into a user-friendly platform, the GMS empowers gym owners and staff to focus on delivering exceptional service and personalized support to their members. From automating administrative processes to providing valuable insights into member preferences and performance, the GMS revolutionizes the way gyms are run and elevates the fitness journey for all stakeholders involved.

In this project, we will delve into the Gym Management System, exploring its features and benefits, analyzing its impact on gym operations and member satisfaction. Through this exploration, we aim to understand how GMS can revolutionize fitness management, promoting healthier lifestyles globally.

CHAPTER 2

DESIGN

2.1 E R Diagram

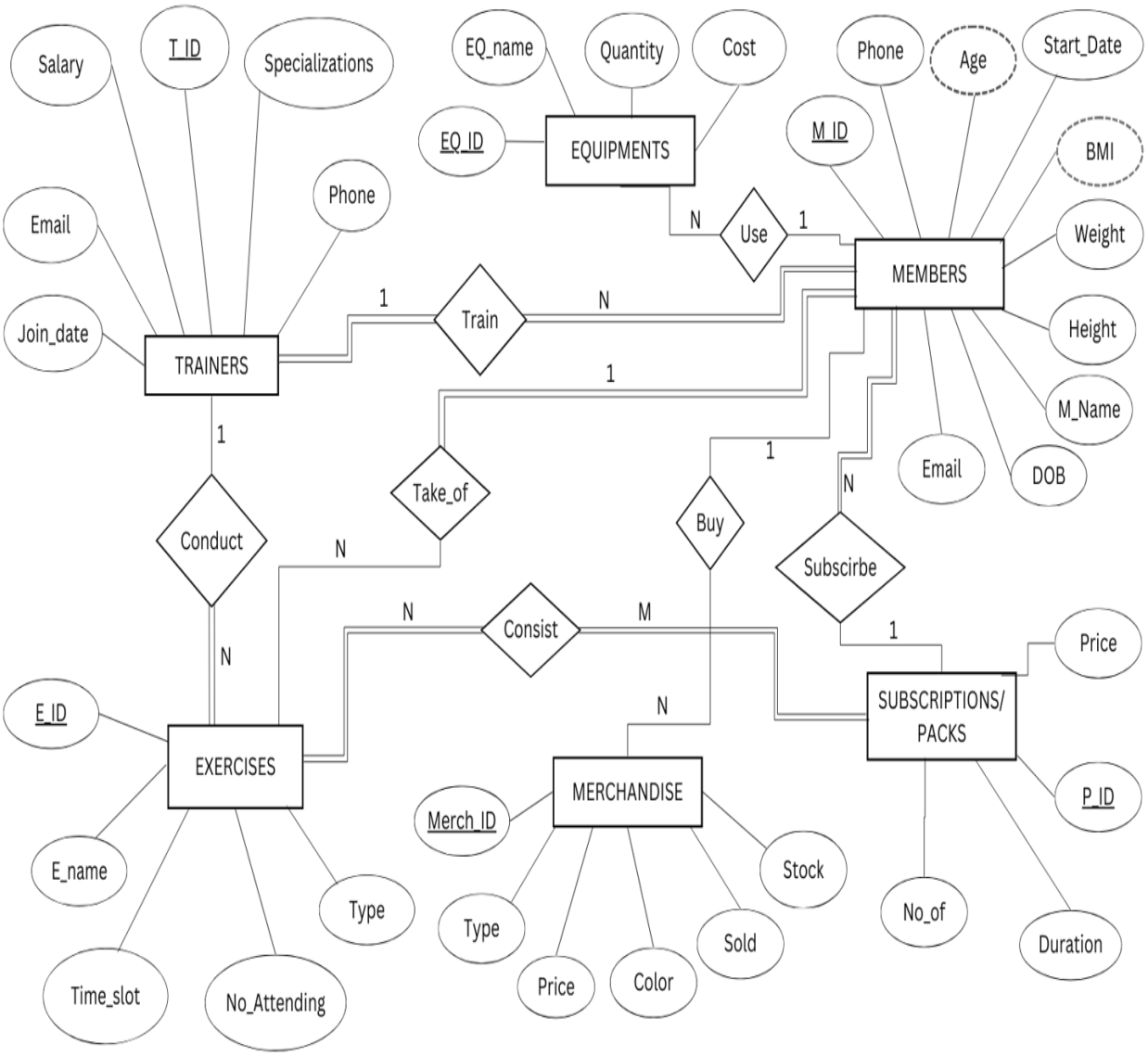


Fig 2.1 Gym management System ER Diagram

2.2 Relational Schema

2.2.1 Mapping From ER Diagram to Schema Diagram

STEP 1: Mapping of regular entity types. The regular entity types of our project are shown in figure.

TRAINERS

T_ID	E-MAIL	SALARY	JOIN_DATE	PHONE	SPECIALIZATION

EQUIPMENTS

EQ_ID	EQ_NAME	QUANTITY	COST

MEMBERS

M_ID	M_NAME	PHONE	AGE	START_DATE	BMI

HEIGHT	WEIGHT	DOB	EMAIL

SUBSCRIPTIONS

P_ID	PRICE	DURATION	NO_OF

MERCHANDISE

MERCH_ID	TYPE	PRICE	COLOR	SOLD	STOCK

EXERCISES

E_ID	E_NAME	TIME_SLOT	TYPE	NO_ATTENDING

Fig 2.2 Mapping of regular entity.

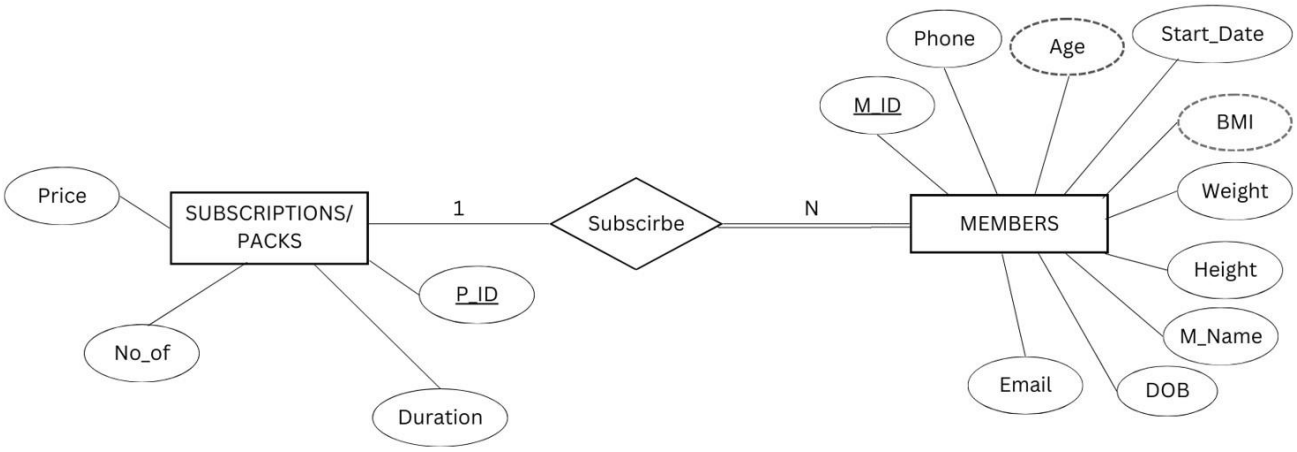
STEP 2: Mapping of Weak entity types

The ER Diagram contains no weak entities, this step is ignored in this project.

STEP 3: Mapping of 1:1 Relations

The ER diagram contains no 1:1 entity, this step is ignored in the project.

STEP 4: Mapping of 1:N Relations



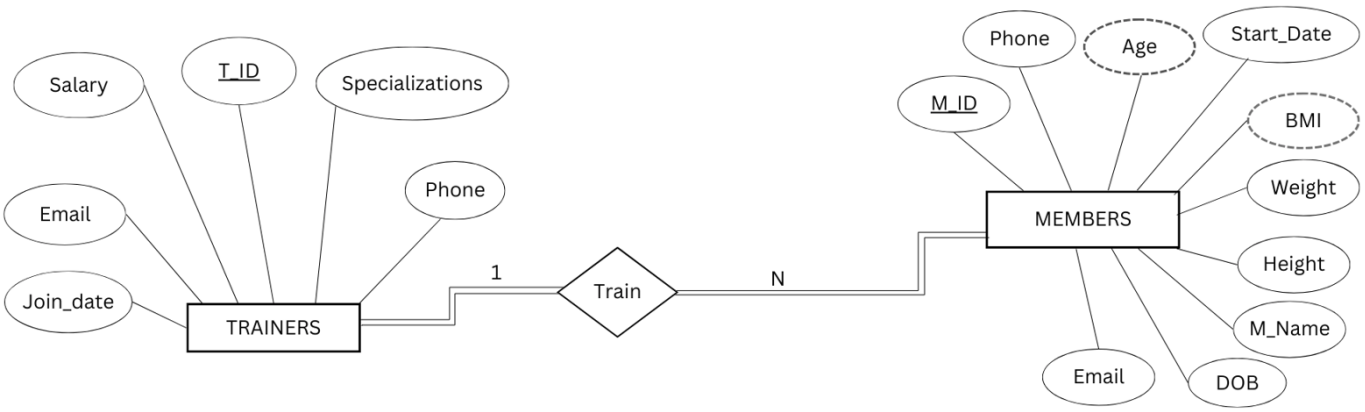
The 1: N relation can be mapped as:

SUBSCRIPTIONS

P_ID	PRICE	DURATION	NO_OF

MEMBERS

M_ID	M_NAME	E-MAIL	PHONE	DOB	AGE	START_DATE
HEIGHT	WEIGHT	BMI	TRAIN_ID	PACK_ID		



The 1: N relation can be mapped as:

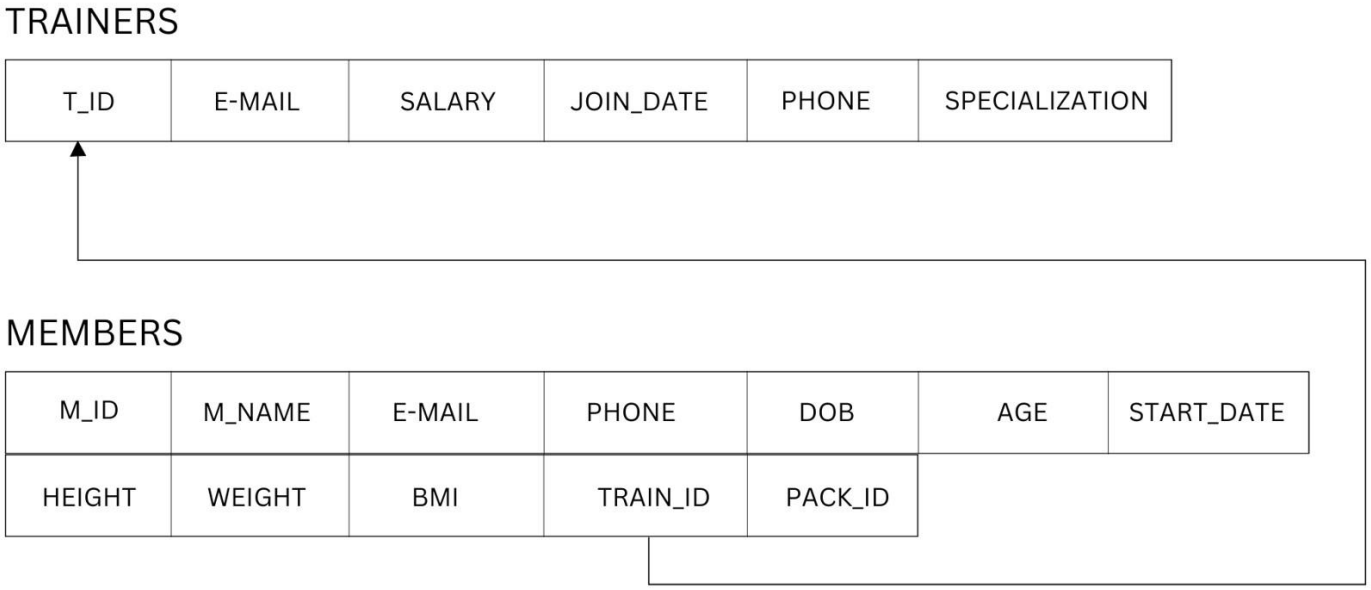
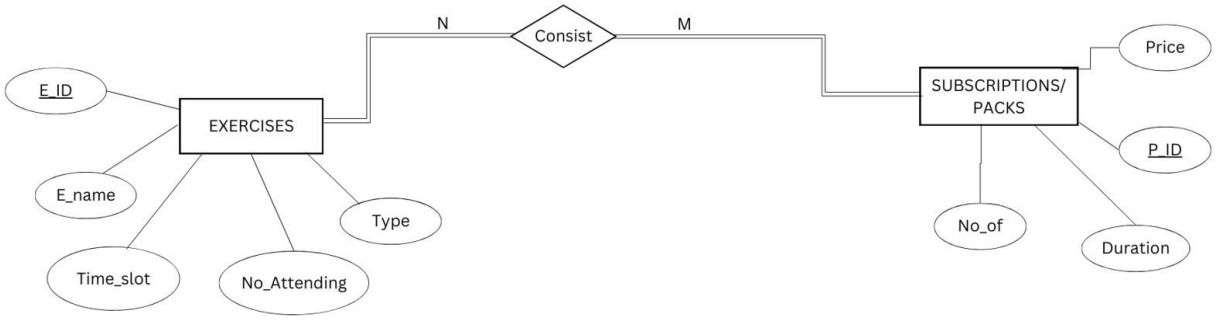


Fig.2.3 Mapping of 1: N relations

STEP 5: Mapping of M: N Relation.



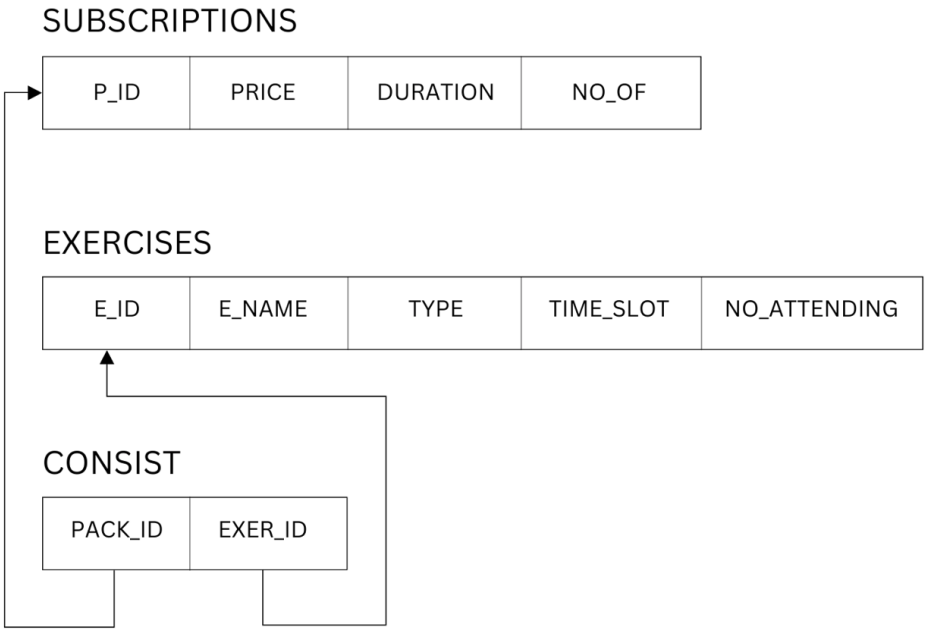


Fig.2.4 Mapping of M:N relations.

STEP 6: Mapping of N-Ary Relation.

The ER diagram contains no 1:1 entity, this step is ignored in the project.

2.3. Schema Diagram

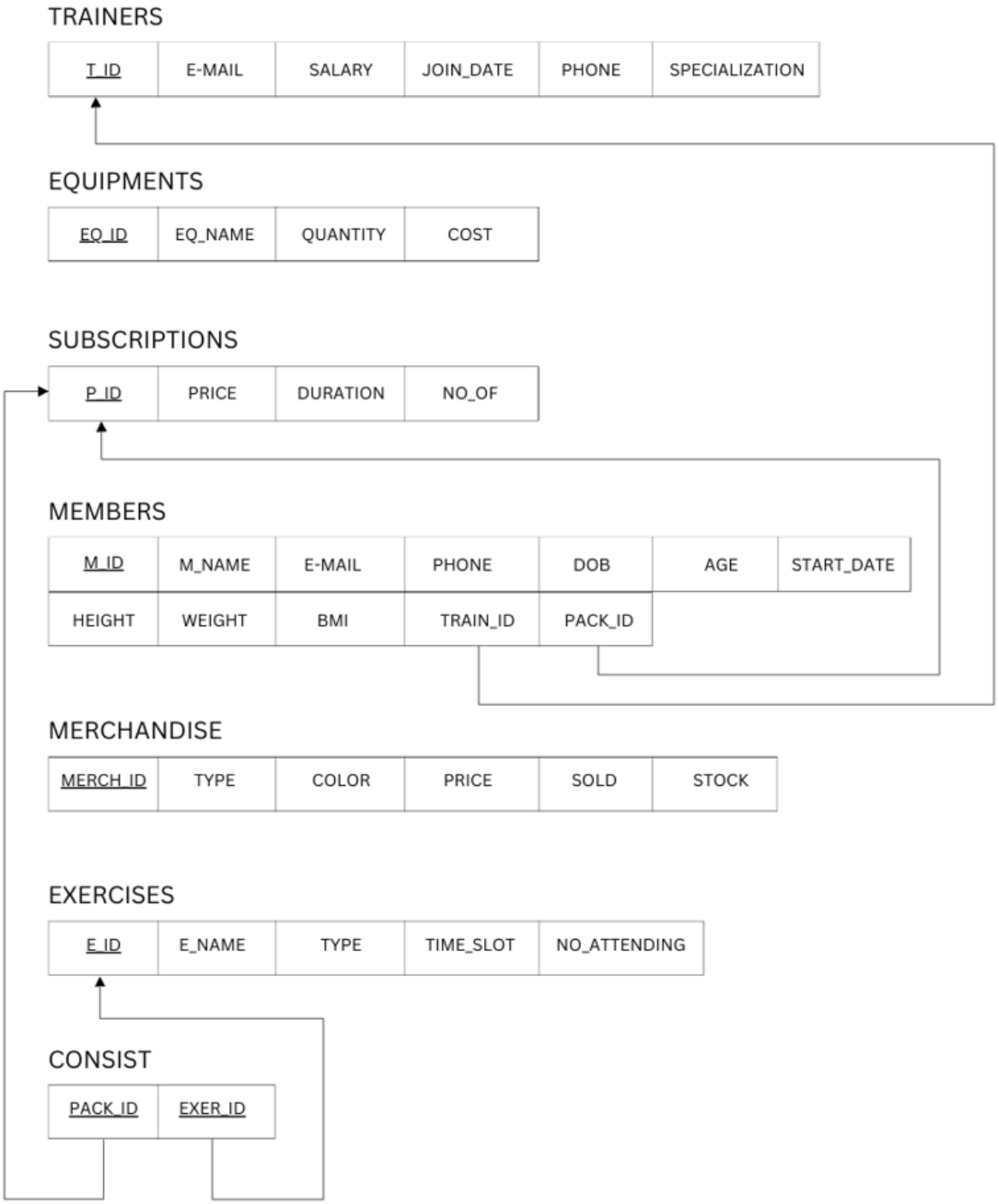


Fig.2.5. Schema Diagram

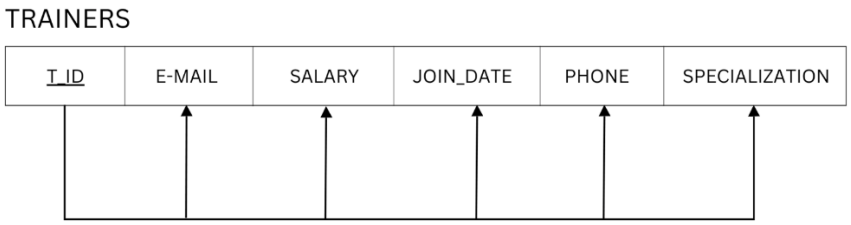
CHAPTER 3

NORMALIZATION

Normalization is the process of organizing data in a database efficiently. This involves creating tables and establishing relationships between those tables according to a set of rules designed to protect the data integrity and ensure that the database structure supports the intended operations without unnecessary redundancy.

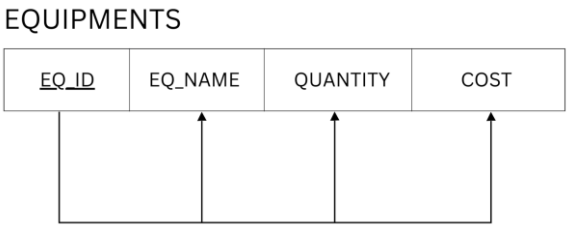
To normalize each table, we'll start by identifying the functional dependencies and then decompose the tables accordingly.

1. TRAINERS:



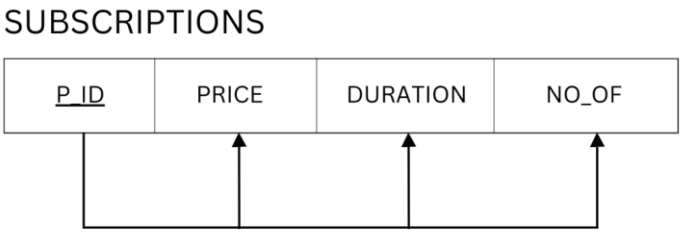
- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

2. EQUIPMENTS:



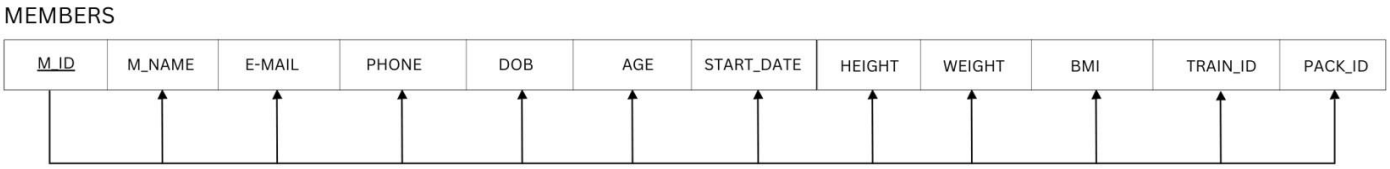
- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

3. SUBSCRIPTIONS:



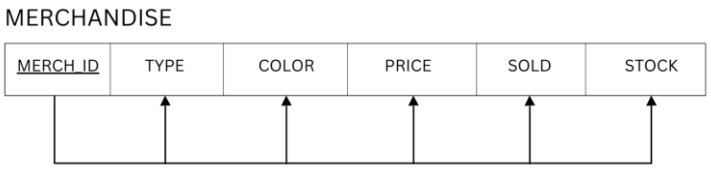
- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

4. MEMBERS:



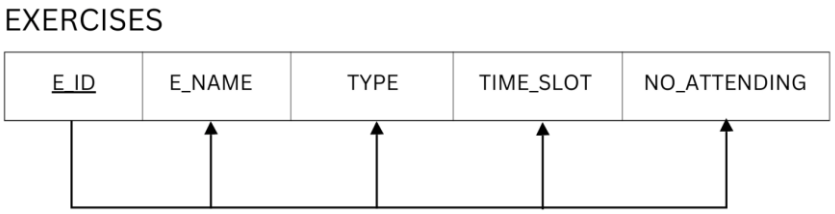
- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

5. MERCHANDISE:



- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

6. EXERCISES:



- It is in 1NF because there are no multivalued attributes in the relation schema.
- It is in 2NF because the relation is in 1NF and all the attributes in the relation schema are fully functionally dependent on the primary key.
- It is in 3NF because the relation is in 2NF and there are no transitive dependencies.

CHAPTER 4

IMPLEMENTATION

Implementing a Gym Management System using Java NetBeans and MySQL involves several steps. Below is a brief overview of how you might approach it:

1. Database Design:

   - Design the database schema considering the entities involved such as Trainers, Members, Equipments, Subscriptions, Merchandise, and Exercises.
   - Define tables with appropriate fields and establish relationships using foreign keys.
   - Ensure normalization to eliminate redundancy and maintain data integrity.

2. Backend Development with Java:

   - Develop Java classes for managing database operations including insertion, updating, and deletion of data.
   - Implement authentication and authorization mechanisms to control access to the system.
   - Design logic for overseeing gym operations such as member registration, equipment management, subscription management, and merchandise sales.
   - Use JDBC (Java Database Connectivity) to connect Java classes with the MySQL database and execute SQL queries for database interaction.
   - Handle exceptions and errors gracefully to ensure robustness and reliability.

3. Frontend Development with Java Swing:

   - Design the user interface using Java Swing to create a user-friendly experience.
   - Implement forms for data entry and validation to ensure data integrity.
   - Include features such as tables, buttons, text fields, and other GUI components to interact with the user.

4. Integration with MySQL Database:

- Connect Java classes with the MySQL database using JDBC.
- Execute SQL queries to perform CRUD (Create, Read, Update, Delete) operations on the database.
- Handle database transactions effectively to maintain data consistency and integrity.
- Utilize prepared statements to prevent SQL injection attacks and improve performance.

4.1 TABLE STRUCTURE

4.1.1 TRAINERS TABLE:

```
CREATE TABLE trainers (
   T_ID INT PRIMARY KEY AUTO_INCREMENT,
   Tname VARCHAR(255),
   email VARCHAR(255),
   phone VARCHAR(20),
   salary DECIMAL(10, 2),
   join_date DATE,
   Specialization VARCHAR(255)
);
```

4.1.2 EQUIPMENTS TABLE:

```
CREATE TABLE equipments(
      Eq_ID char(15) PRIMARY KEY,
      Quantity int,
      Cost int,
      Eq_Name varchar(50) NOT NULL
);
```

4.1.3 EXERCISES TABLE:

```
CREATE TABLE exercises(
      E_ID char(15) PRIMARY KEY,
      Ex_Name varchar(50) NOT NULL,
      Type varchar(50),
      Time_Slot time,
      No_Attending int
);
```

4.1.4 SUBSCRIPTIONS TABLE:

```
CREATE TABLE subscriptions(
    Sub_ID char(15) PRIMARY KEY,
    Price int NOT NULL,
    Duration_in_Months int NOT NULL,
    No_Subscribed int
);
```

4.1.5 MERCHANDISE TABLE:

```
CREATE TABLE merchandise(
    Merch_ID char(15) PRIMARY KEY,
    Color varchar(10) NOT NULL,
    Price int NOT NULL,
    Type varchar(10),
    No_Sold int,
    No_In_Stock int
);
```

4.1.6 MEMBERS TABLE:

```
CREATE TABLE members (
    M_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    M_name VARCHAR(255),
    Email VARCHAR(255),
    Phone VARCHAR(255),
    Join_date DATE,
    Height INT,
    Weight DECIMAL(10,2),
    BMI DECIMAL(10,2) AS (Weight / POWER(Height / 100.0, 2)),
    DOB DATE,
    Trainer_ID INT,
    Pack_ID char(15),
    Age INT AS (TIMESTAMPDIFF(YEAR, DOB, CURDATE())),
    FOREIGN KEY (Trainer_ID) REFERENCES trainers(T_ID) on delete set NULL,
    FOREIGN KEY (Pack_ID) REFERENCES Subscriptions (Sub_ID) on delete no action
);
```

4.1.7 CONSIST TABLE :

```
Pack_ID char(15),
    Exercise_ID char(15),
    PRIMARY KEY (Pack_ID, Exercise_ID),
    FOREIGN KEY (Pack_ID) REFERENCES Subscriptions(Sub_ID) on delete cascade,
    FOREIGN KEY (Exercise_ID) REFERENCES Exercises (E_ID) on delete cascade
);
```

4.2 FUNCTIONALITY

4.2.1 DATEBASE CONNECTION:

```
import java.sql.Connection;

    try {
        Class.forName("com.mysql.jdbc.Driver");

        Connection con =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/gym?useSSL
        =false&allowPublicKeyRetrieval=true","root","swasthik@52");

    }catch(ClassNotFoundException ex){
        return null;
    }
```

4.2.2 SELECTION:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
  Class.forName("com.mysql.jdbc.Driver");
  Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/gym?useSSL=false","root","swasthi
k@52");

  Statement st = con.createStatement();
  String sql = "SELECT * FROM trainers";
  ResultSet rs = st.executeQuery(sql);

  // Clear existing data outside the loop
  DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
  model.setRowCount(0);

  while (rs.next()) {
      int tID = rs.getInt("T_ID");
      String tName = rs.getString("Tname");
      String email = rs.getString("email");
      String phone = rs.getString("phone");
      int salary = rs.getInt("salary");
      java.sql.Date joinDate = rs.getDate("join_date");
      String specialization = rs.getString("specialization");

      // Add a new row to the JTable
      model.addRow(new Object[]{tID, tName, email, phone,salary,joinDate,specialization});
  }
```

```
            con.close();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

            }
```

4.2.3 INSERT:

```
private void InsertActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
        // Retrieve data from text fields
        String tName = textFieldName.getText();
        String email = textFieldEmail.getText();
        String phone = textFieldPhone.getText();

        // Check if the salary field is empty
        String salaryText = textFieldSalary.getText();
        if(b==1){
        if (salaryText.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Fill all the details", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if the salary field is empty
        }

        double salary;
        try {
            salary = Double.parseDouble(salaryText);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Invalid salary format. Please enter a valid number.",
"Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if there's an issue parsing the salary
        }

        String joinDate = textFieldJoinDate.getText();

        // Check if the phone field contains only digits
        if (!phone.matches("\\d+")) {
            JOptionPane.showMessageDialog(null, "Invalid phone number. Please enter only digits.",
"Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if phone number contains non-digits
        }

        // Check if the join date is in the proper format (yyyy-mm-dd)
        if (!isValidDateFormat(joinDate)) {
            JOptionPane.showMessageDialog(null, "Invalid date format. Please enter the date in YYYY-
MM-DD format.", "Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if the date format is invalid
        }
```

```
        String specialization = textFieldSpecialization.getText();

        // Check if any other field is empty
        if (tName.isEmpty() || email.isEmpty() || phone.isEmpty() || joinDate.isEmpty() ||
specialization.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Fill all the details", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if any other field is empty
        }

        // Establish database connection
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/gym?useSSL=false&allowPublicKeyRetri
eval=true", "root", "swasthik@52");

        // Create and execute the SQL INSERT statement
        String sql = "INSERT INTO trainers (Tname, email, phone, salary, join_date, specialization)
VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement pstmt = con.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {
            pstmt.setString(1, tName);
            pstmt.setString(2, email);
            pstmt.setString(3, phone);
            pstmt.setDouble(4, salary);
            pstmt.setString(5, joinDate);
            pstmt.setString(6, specialization);

            // Execute the INSERT statement
            int rowsInserted = pstmt.executeUpdate();

            if (rowsInserted > 0) {
                // Display a popup message
                JOptionPane.showMessageDialog(null, "Trainer inserted successfully!");
                refreshTrainersTable();
                textFieldName.setText("");
                textFieldEmail.setText("");
                textFieldPhone.setText("");
                textFieldSalary.setText("");
                textFieldJoinDate.setText("");
                textFieldSpecialization.setText("");
            } else {
                System.out.println("Failed to insert trainer details. No rows affected.");
            }
        }
        populateTrainerIDs();
        // Close the database connection
        con.close();
        }else{
            JOptionPane.showMessageDialog(null, "Members do not have permission to insert.",
"Permission Denied", JOptionPane.ERROR_MESSAGE);
            refreshTrainersTable();
        }
```

```
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

private boolean isValidDateFormat(String date) {
    try {
        LocalDate.parse(date, DateTimeFormatter.ISO_DATE);
        return true;
    } catch (DateTimeParseException e) {
        return false;
    }
    }
```

4.2.4 UPDATING:

```
private void trainerUpdateActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
     try {
        // Retrieve data from text fields
        String tName = updNametext.getText();
        String email = updEmailtext.getText();
        String phone = updPhonetext.getText();
        String salaryText = updSalarytext.getText();
        String joinDate = updJoindatetext.getText();
        String specialization = updSpectext.getText();

        if(b==1){

        // Check if any field is empty
        if (tName.isEmpty() || email.isEmpty() || phone.isEmpty() || salaryText.isEmpty() ||
joinDate.isEmpty() || specialization.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Enter all the details", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if any field is empty
        }

        // Check if the phone field contains only digits
        if (!phone.matches("\\d+")) {
            JOptionPane.showMessageDialog(null, "Invalid phone number. Please enter only
digits.", "Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if phone number contains non-digits
        }

        double salary;
        try {
            salary = Double.parseDouble(salaryText);
        } catch (NumberFormatException e) {
```

```
 JOptionPane.showMessageDialog(null, "Invalid salary format. Please enter a valid number.",
"Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if there's an issue parsing the salary
        }

        // Check if the join date is in the proper format (yyyy-mm-dd)
        if (!isValidDateFormat(joinDate)) {
            JOptionPane.showMessageDialog(null, "Incorrect date format. Enter in yyyy-mm-dd.",
"Error", JOptionPane.ERROR_MESSAGE);
            return; // Exit the method if the date format is invalid
        }

        // Establish database connection
        Class.forName("com.mysql.cj.jdbc.Driver");
        try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/gym?useSSL=false&allowPublicKe
yRetrieval=true", "root", "swasthik@52")) {
            // Execute SQL Update Statement
            String updateSql = "UPDATE trainers SET Tname = ?, email = ?, phone = ?, salary = ?,
join_date = ?, specialization = ? WHERE T_ID = ?";
            try (PreparedStatement pstmt = con.prepareStatement(updateSql)) {
                String selectedTrainerID = jComboBoxUpdateTID.getSelectedItem().toString();

                pstmt.setString(1, tName);
                pstmt.setString(2, email);
                pstmt.setString(3, phone);
                pstmt.setDouble(4, salary);
                pstmt.setString(5, joinDate);
                pstmt.setString(6, specialization);
                pstmt.setString(7, selectedTrainerID);

                int rowsUpdated = pstmt.executeUpdate();

                if (rowsUpdated > 0) {
                    System.out.println("Trainer details updated successfully!");
                    JOptionPane.showMessageDialog(null, "Trainer updated successfully!");
                    // Optionally, you can display a success message or perform additional actions
                    refreshTrainersTable();
                    updNametext.setText("");
                    updEmailtext.setText("");
                    updPhonetext.setText("");
                    updSalarytext.setText("");
                    updJoindatetext.setText("");
                    updSpectext.setText("");
                } else {
                    System.out.println("Failed to update trainer details. No rows affected.");
                    // Optionally, you can display an error message or perform additional actions
                }
            }
        }
```

```
        populateTrainerIDs();// Update the JComboBox after successful update
        }else{
            JOptionPane.showMessageDialog(null, "Members do not have permission to update.",
"Permission Denied", JOptionPane.ERROR_MESSAGE);
            refreshTrainersTable();
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }

}
```

4.2.5 DELETING:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String deleteTID = trainerDEL_TextField.getText();

    // Check if the T_ID is empty
    if(b==1){
    if (deleteTID.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Enter a Trainer ID to delete", "Error",
         JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Call the deleteTrainer method to perform the deletion
    deleteTrainer(deleteTID);
    populateTrainerIDs();
    }else{
            JOptionPane.showMessageDialog(null, "Members do not have permission to delete.",
         "Permission Denied", JOptionPane.ERROR_MESSAGE);
            refreshTrainersTable();
        }
}

// Method to delete a trainer based on T_ID
private void deleteTrainer(String tID) {
    try {
        // Establish a database connection
        Connection con =
         DriverManager.getConnection("jdbc:mysql://localhost:3306/gym?useSSL=false&allowPublic
         KeyRetrieval=true", "root", "swasthik@52");

        // Create and execute the SQL DELETE statement
        String sql = "DELETE FROM trainers WHERE T_ID = ?";
        try (PreparedStatement pstmt = con.prepareStatement(sql)) {
            pstmt.setString(1, tID);
```

```
    // Execute the DELETE statement
            int rowsDeleted = pstmt.executeUpdate();

            if (rowsDeleted > 0) {
                JOptionPane.showMessageDialog(null, "Trainer Deleted successfully!");
                System.out.println("Trainer with T_ID " + tID + " deleted successfully!");
                // Refresh the JTable to reflect the changes
                refreshTrainersTable();
                // Optionally, clear the text field after successful deletion
                trainerDEL_TextField.setText("");
            } else {
                System.out.println("No trainer found with T_ID " + tID);
                JOptionPane.showMessageDialog(null, "No trainer found with T_ID " + tID, "Error",
         JOptionPane.ERROR_MESSAGE);
            }

        }

        // Close the database connection
        con.close();
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error deleting trainer. Check the console for details.",
         "Error", JOptionPane.ERROR_MESSAGE);
    }

}
```

4.2.6 TRIGGERS

```
CREATE TRIGGER before_insert_consist
    BEFORE INSERT ON consist
    FOR EACH ROW
    BEGIN
        DECLARE exercise_count INT;
        DECLARE subscription_count INT;

        -- Check if the Exercise_ID exists in the exercises table
        SELECT COUNT(*) INTO exercise_count FROM exercises WHERE E_ID = NEW.Exercise_ID;

        -- Check if the Pack_ID exists in the subscriptions table
        SELECT COUNT(*) INTO subscription_count FROM subscriptions WHERE Sub_ID = NEW.Pack_ID;

        -- If either Exercise_ID or Pack_ID does not exist, prevent insertion
        IF exercise_count = 0 OR subscription_count = 0 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Cannot insert into consist table: Invalid Exercise_ID or Pack_ID';
        END IF;
    END;
```

CHAPTER 5

RESULTS

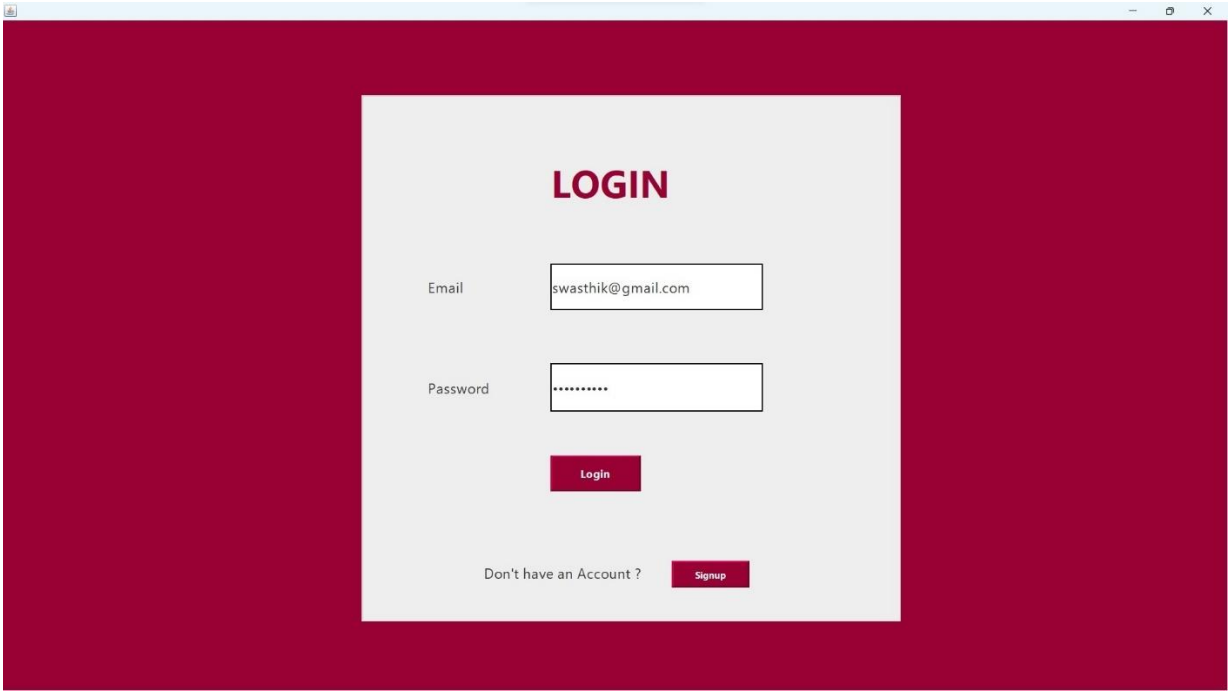


Fig.5.1 Login Page

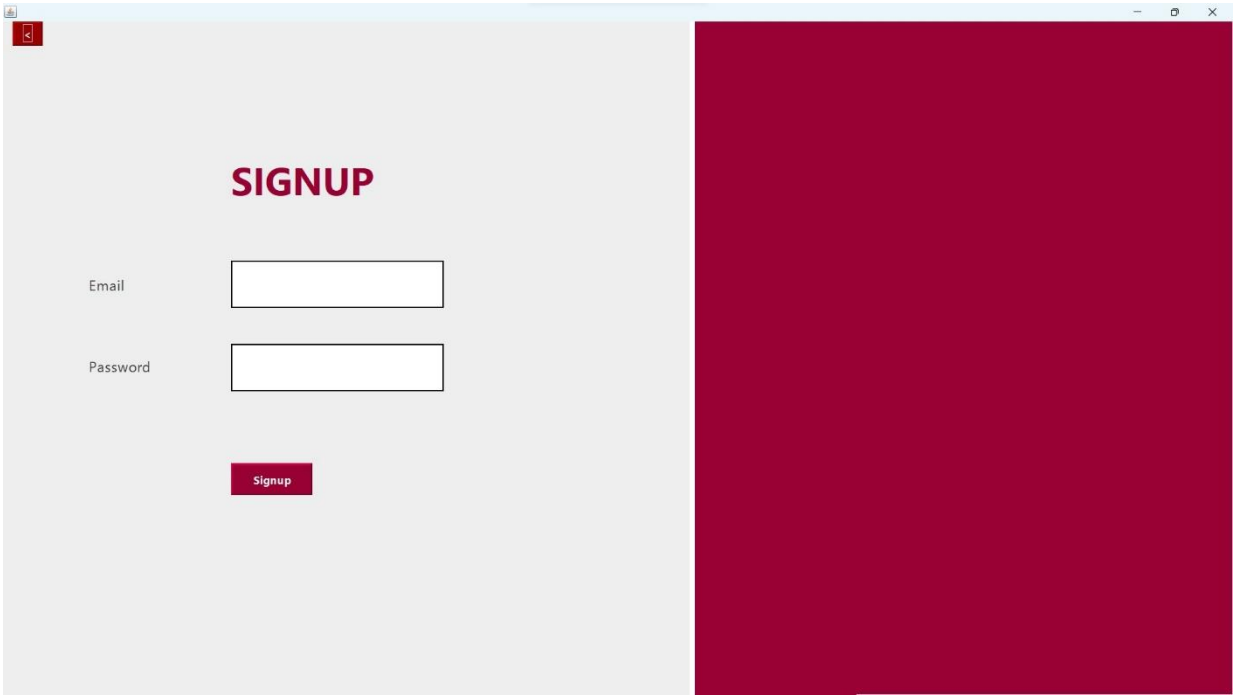


Fig.5.2 Signup Page

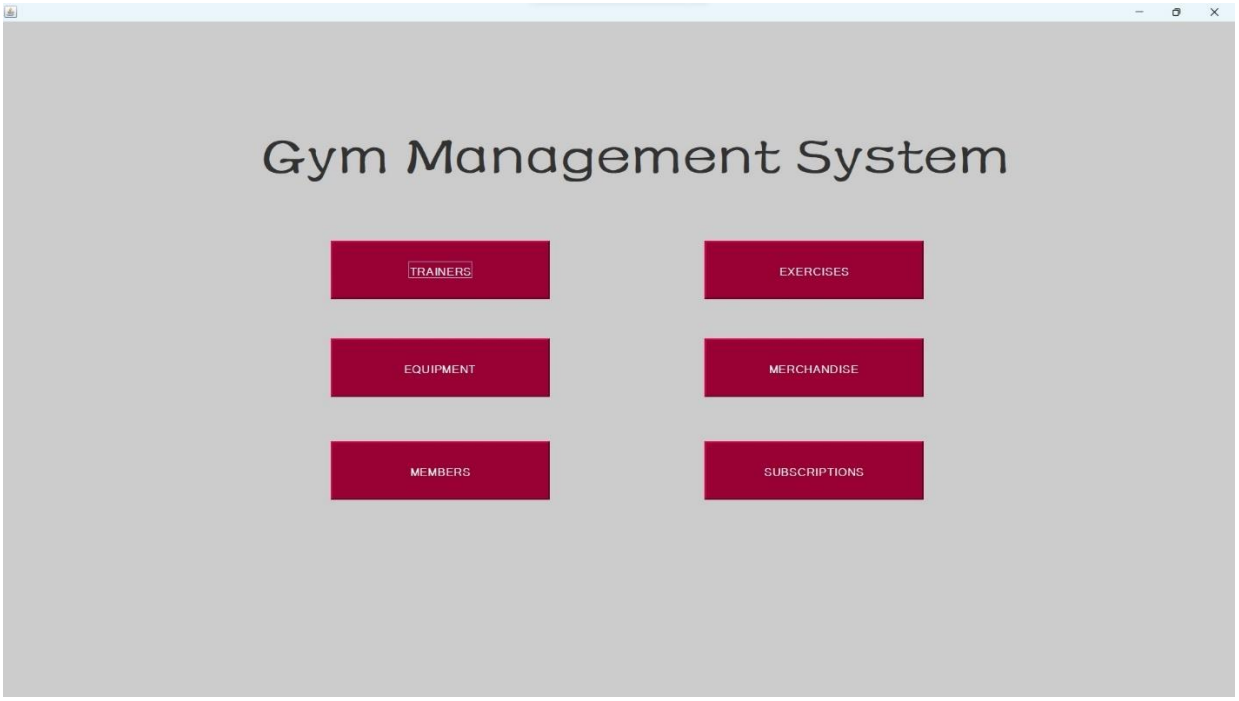


Fig.5.3. Homepage

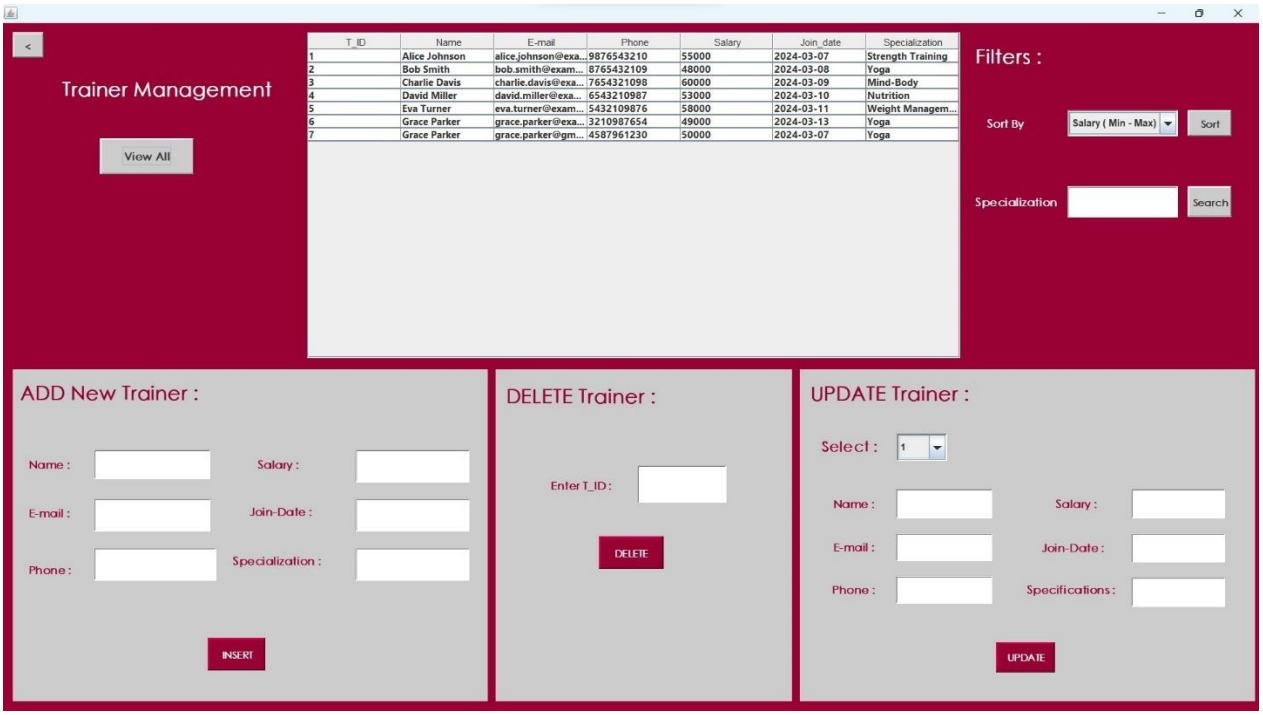


Fig.5.4. Trainer Management

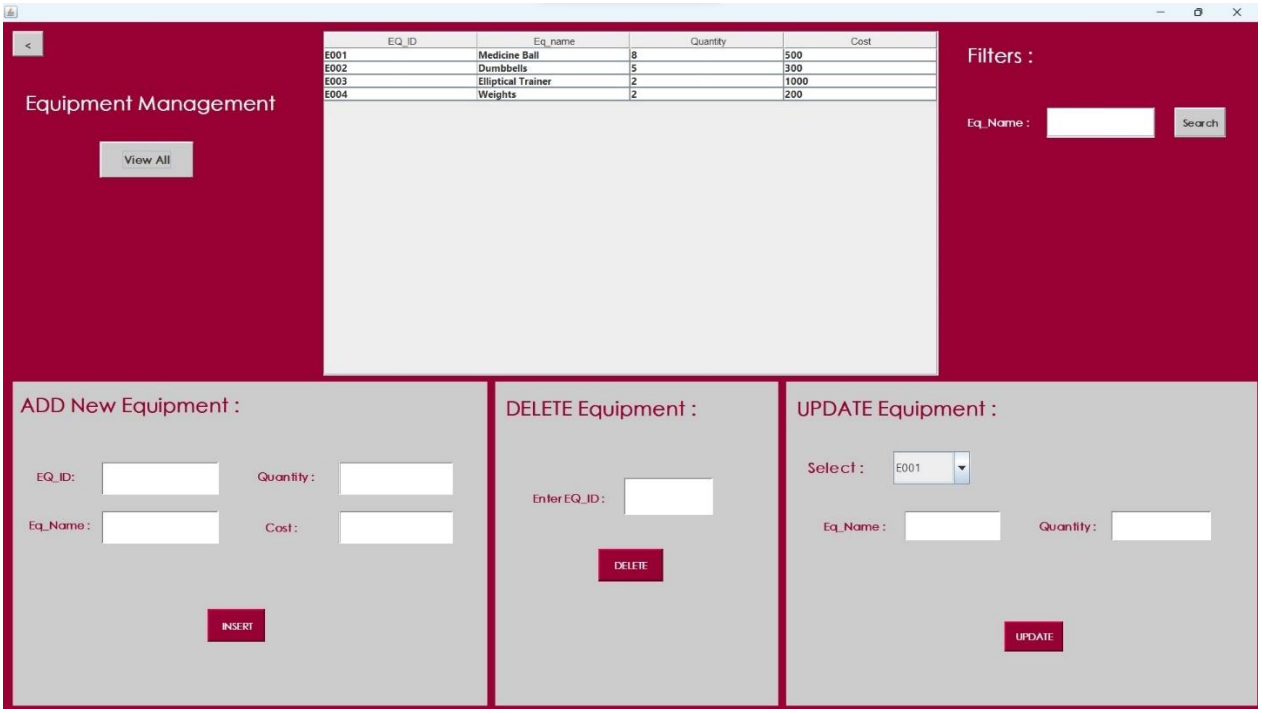


Fig.5.5. Equipment Management.

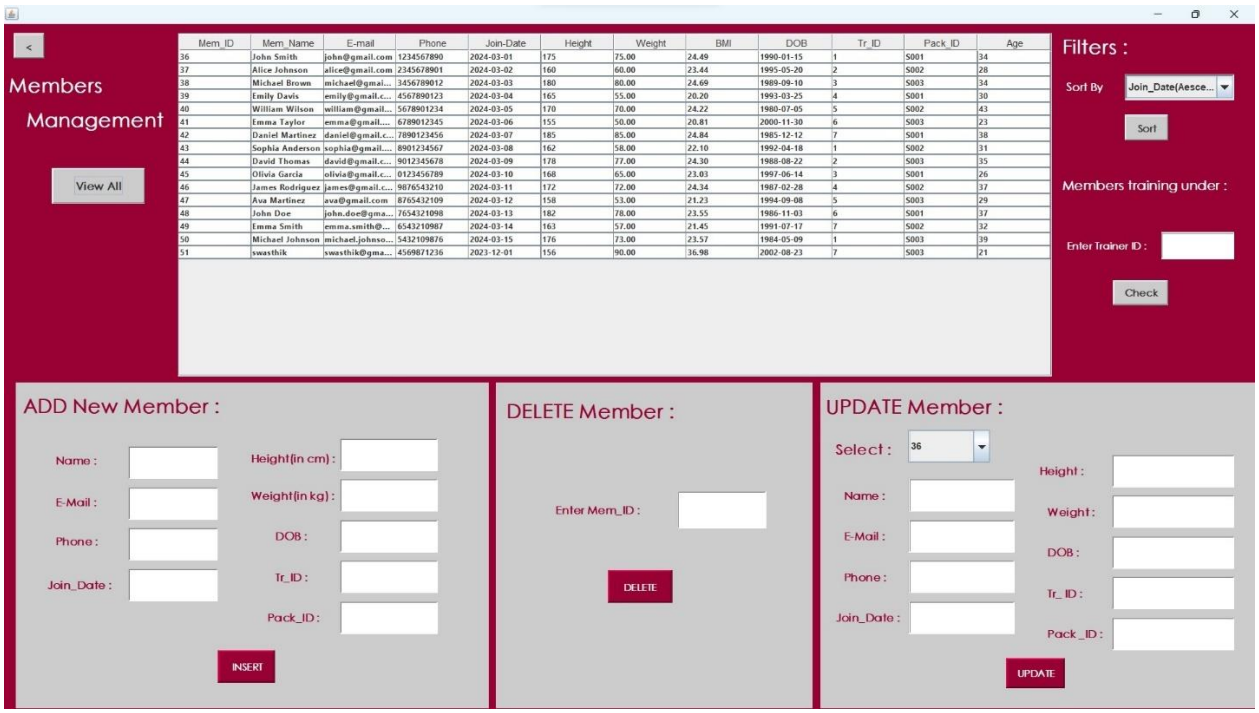
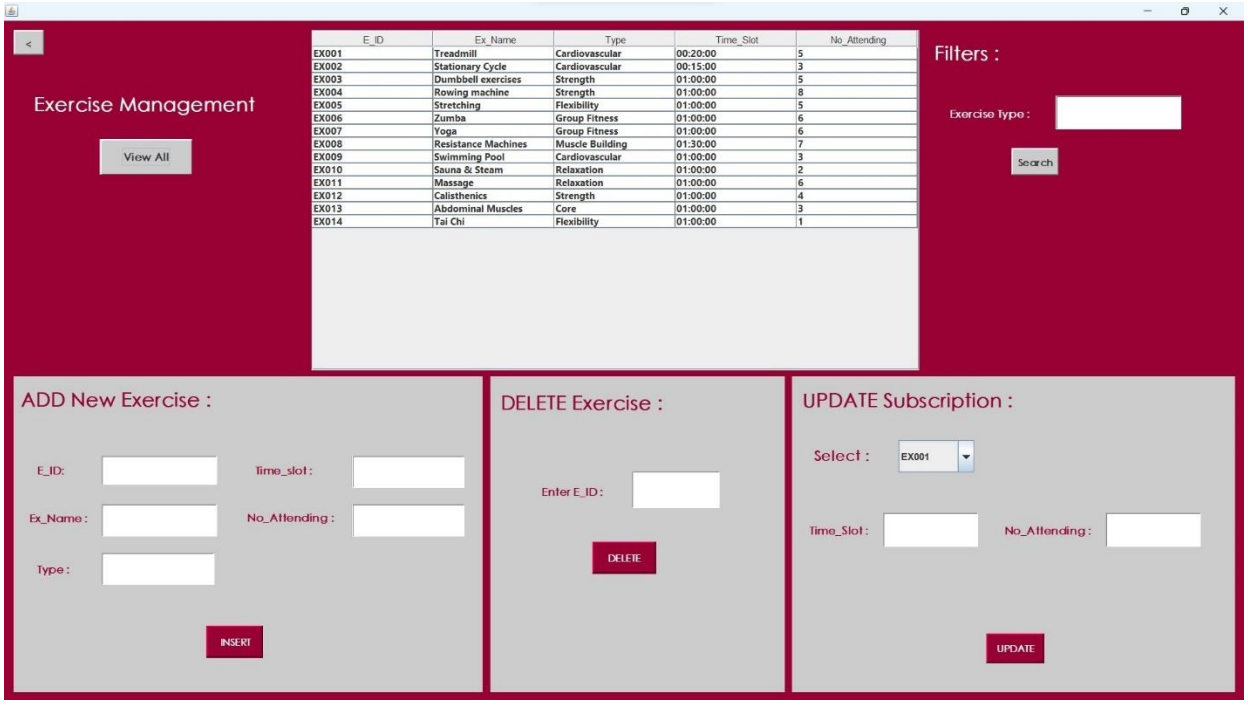
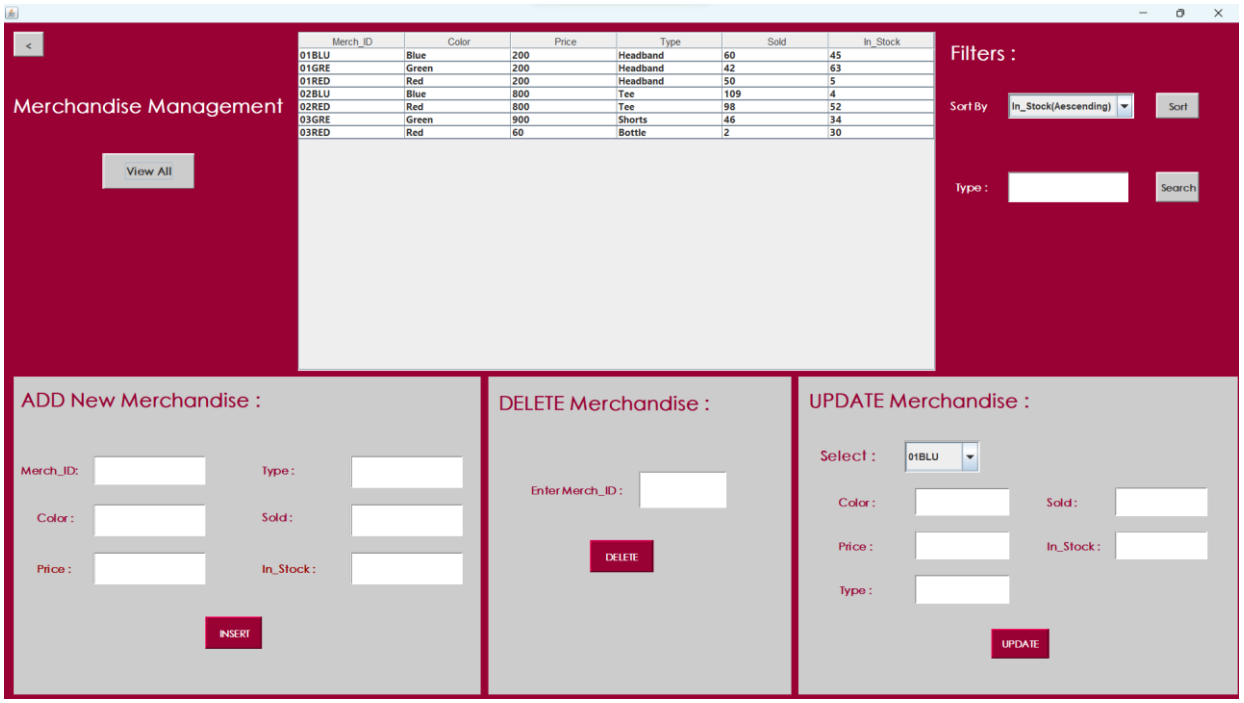


Fig.5.6. Member Management.



E_ID	Ex_Name	Type	Time_Slot	No_Attending
EX001	Treadmill	Cardiovascular	00:20:00	5
EX002	Stationary Cycle	Cardiovascular	00:15:00	3
EX003	Dumbbell exercises	Strength	01:00:00	5
EX004	Rowing machine	Strength	01:00:00	8
EX005	Stretching	Flexibility	01:00:00	5
EX006	Zumba	Group Fitness	01:00:00	6
EX007	Yoga	Group Fitness	01:00:00	6
EX008	Resistance Machines	Muscle Building	01:30:00	7
EX009	Swimming Pool	Cardiovascular	01:00:00	3
EX010	Sauna & Steam	Relaxation	01:00:00	2
EX011	Massage	Relaxation	01:00:00	6
EX012	Calisthenics	Strength	01:00:00	4
EX013	Abdominal Muscles	Core	01:00:00	3
EX014	Tai Chi	Flexibility	01:00:00	1

Fig.5.7. Exercise Management



Merch_ID	Color	Price	Type	Sold	In_Stock
01BLU	Blue	200	Headband	60	45
01GRE	Green	200	Headband	42	63
01RED	Red	200	Headband	50	5
02BLU	Blue	800	Tee	109	4
02RED	Red	800	Tee	98	52
03GRE	Green	900	Shorts	46	34
03RED	Red	60	Bottle	2	30

Fig.5.8. Merchandise Management.

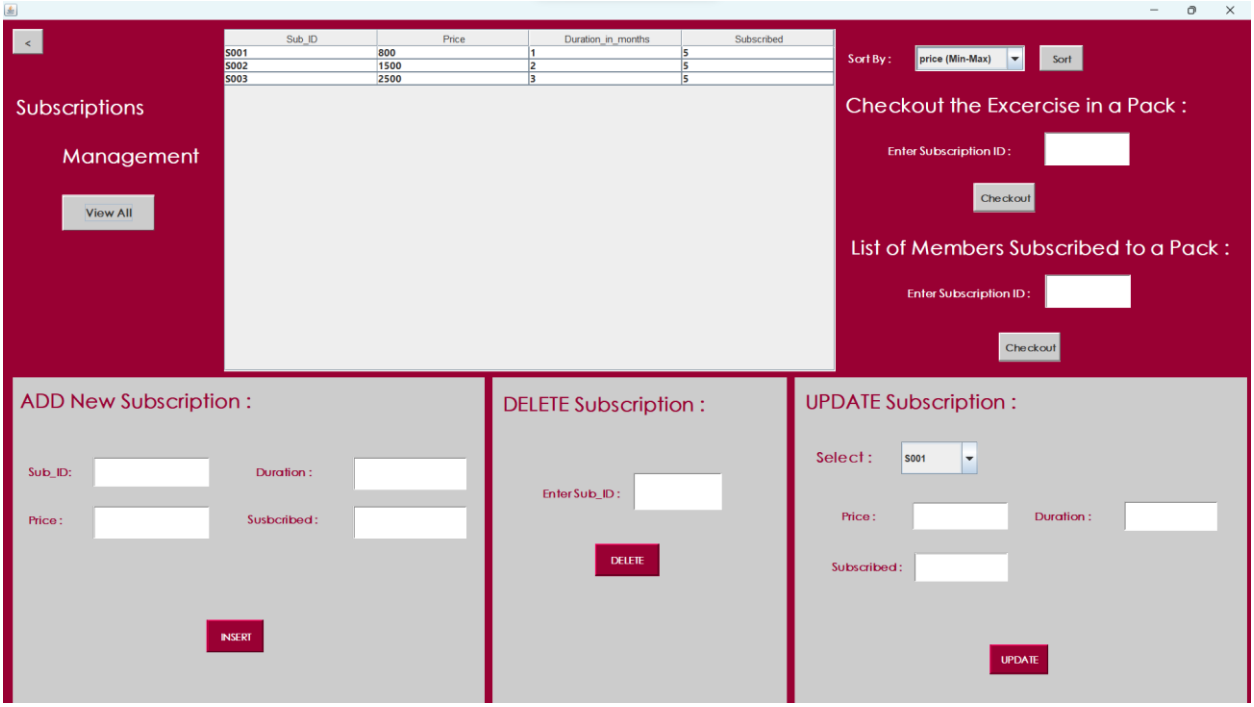


Fig.5.9. Subscription Management.

CHAPTER 6

CONCLUSION

In conclusion, the development of the Gym Management System marks a significant advancement in the domain of fitness and wellness management. By harnessing the power of Java NetBeans and MySQL integration, the system offers a comprehensive solution for gym owners and members alike, streamlining operations and enhancing user experience.

Through its intuitive user interface and robust backend functionality, the application simplifies tasks such as member registration, equipment management, subscription tracking, and exercise scheduling. This streamlined approach not only improves efficiency but also fosters a more engaging and interactive environment for gym members.

Moreover, by incorporating features such as subscription management and equipment tracking, the system promotes transparency and accountability in gym operations. Gym owners can efficiently manage resources, track member attendance, and optimize equipment usage, leading to improved business performance and customer satisfaction.

As the Gym Management System evolves, its data analytics capabilities will play a pivotal role in informing strategic decision-making and driving continuous improvement initiatives. By analyzing member engagement metrics, equipment utilization trends, and subscription patterns, gym owners can tailor their services to better meet the needs of their clientele and enhance overall member satisfaction.

In essence, the Gym Management System represents a transformative tool for gym owners and members alike, offering a seamless and efficient platform for managing fitness facilities and promoting healthy lifestyles. With its emphasis on usability, data-driven insights, and customer satisfaction, the system is poised to revolutionize the fitness industry and make a positive impact on the health and well-being of individuals

REFERENCES

[1] "SQL","Fundamentals of Database Systems", Elmasri Navathe, 3rd Edition, Pearson Edition, 2000.

[2] "MYSQL", https://en.wikipedia.org/wiki/MySQL

[3] "JAVA", https://www.w3schools.com/java

[4] "NETBEANS", https://en.wikipedia.org/wiki/Neteans