# SDT FISAC REPORT

**Prathmesh Sinha (200953220)**
**Rishit Upadhyay (200953184)**

## GYM MANAGEMENT - INTRODUCTION

A total exercise center administration application for the chief of exercise center to screen the subtleties of enrollments of individuals, recruiting of mentors, expense installment structure.

Backing of Java Swing and MongoDB and utilizing NodeJS on NetBeans.

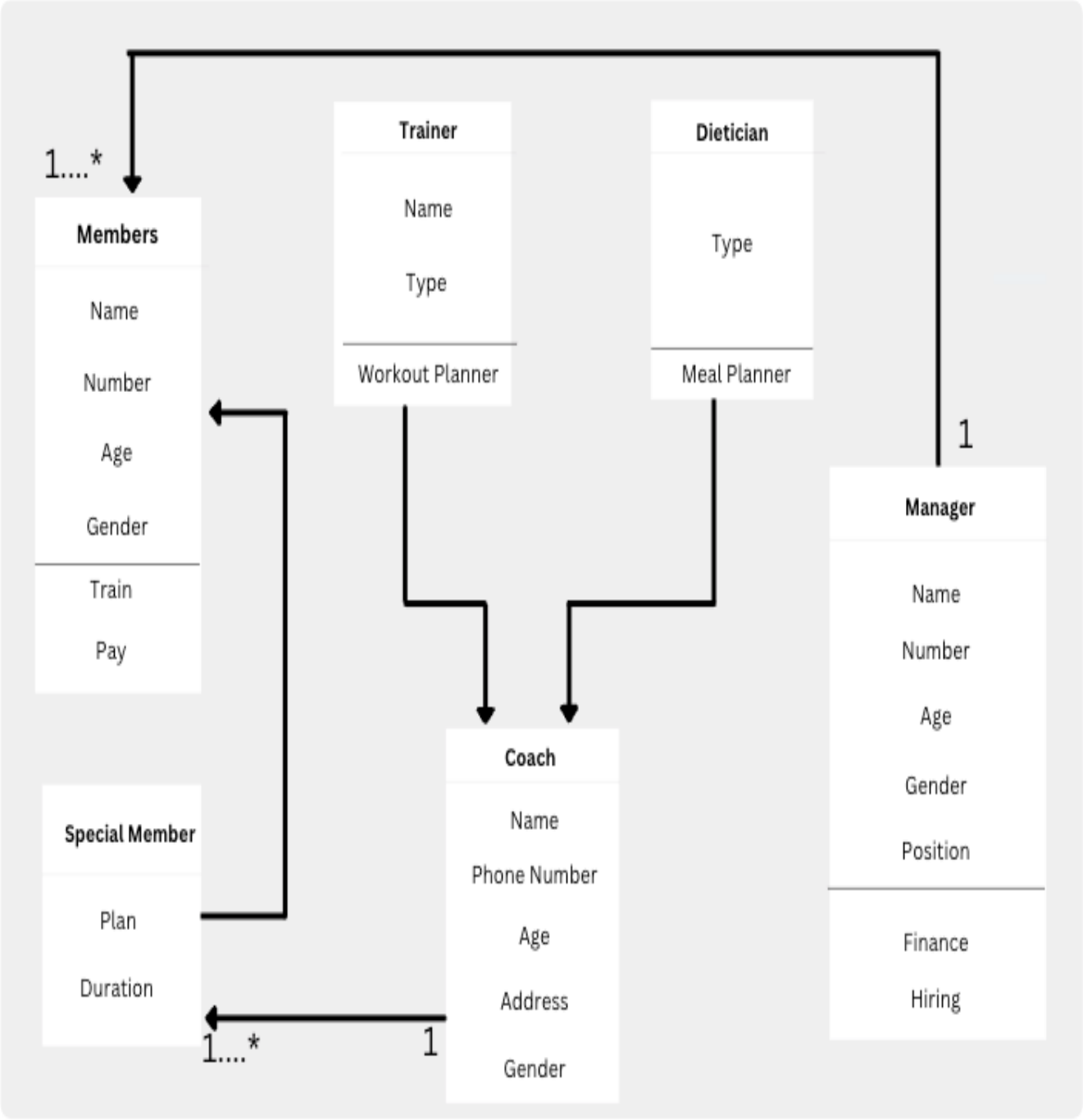
The client can sign up and log into his/her account using our interactive and highly intuitive GUI.

Our exercise center provides a state-of-the-art gym with modern equipment. We also have a wide variety of trainers and dieticians. Members can choose between two plans for a duration of 6 months or 1 year. They can either go for a trainer plan for the gym or a dietician plan for diet. We have some of the best junior and senior trainers with minimum 5 years of experience providing both conditioning and strength training. We have two types of dieticians according to plan, weight loss dieticians and sports dieticians who provide you with weekly personalized meal plans according to your body type.

We have a line of managers who are responsible for the finance department and the hiring process of staff. In the instalment area, clients can follow the expense instalment done by individuals and can print receipts. The subtleties expected for that are destined to be, Date of instalment, sum, and part.

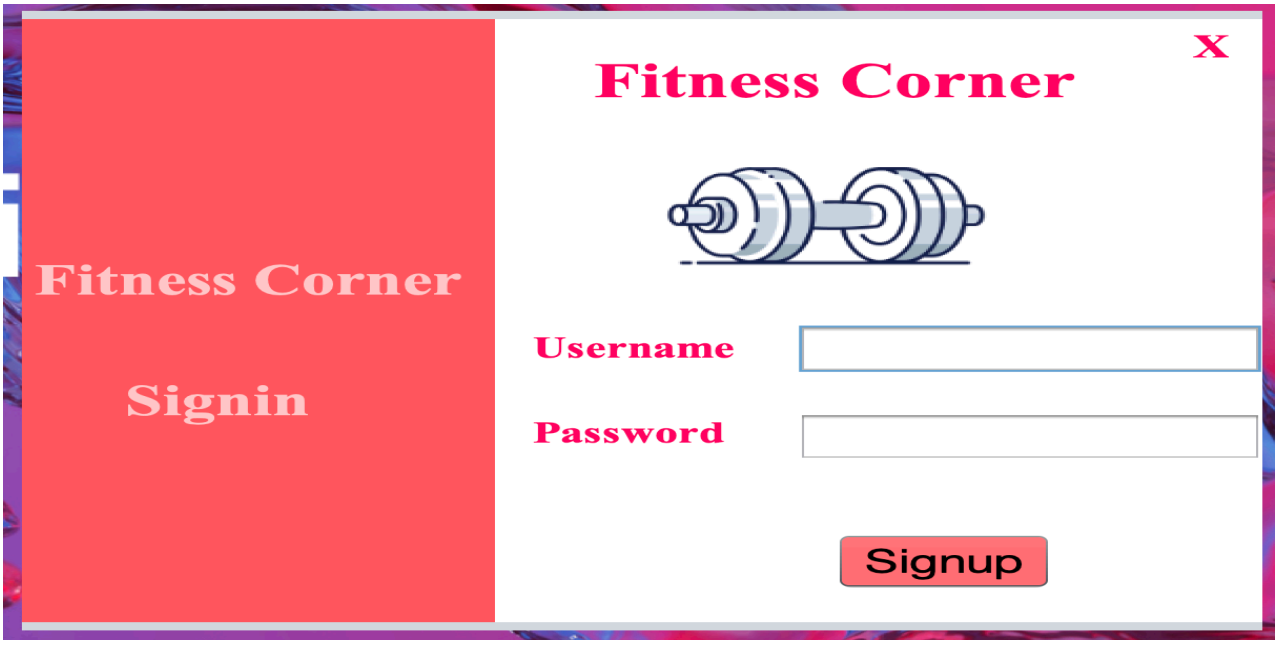
After the general insight into the application, the client can logout from his/her framework by tapping on the logout choice present on the left menu bar.

# Class Diagram



## Code Snippets

# Signup-page:




**Signup process is backed by mongo dB database.**

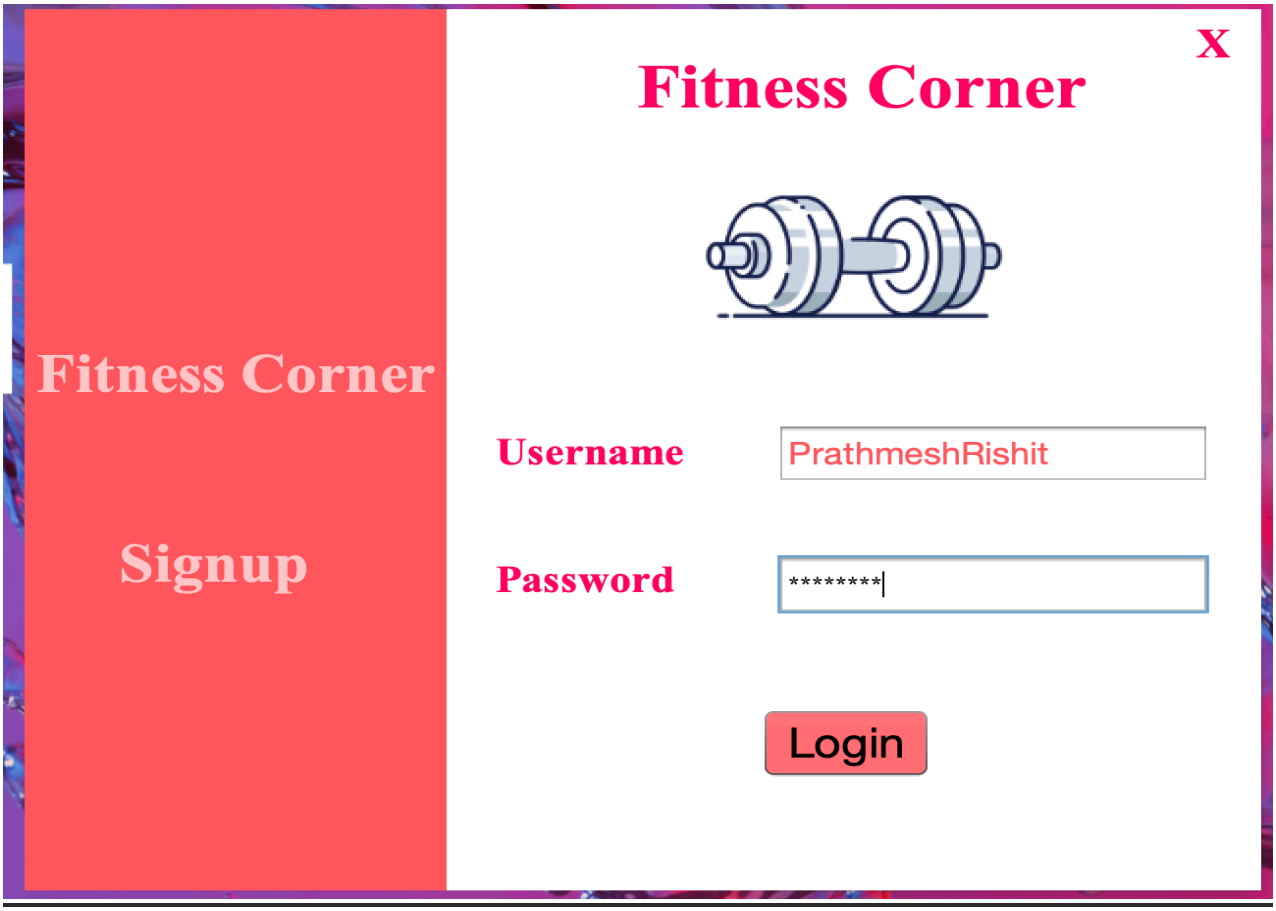
```
public final class Signup extends javax.swing.JFrame {
    /**
     * Creates new form Signup
     */
    public Signup() {
        initComponents();
        connect();
    }

    public String[] returntest(String username, String password){
        String[] s = {jTextField1.getText(),jPasswordField1.getText()};
        return s;
    }
    MongoClient mongoClient;
    MongoDatabase databasename;
    MongoCollection<org.bson.Document> collection;
    public void connect(){
        mongoClient = MongoClients.create( connectionString:"mongodb+srv://prathmeshD
        databasename = mongoClient.getDatabase( string:"SDT");
        collection = databasename.getCollection( string:"Admin");
        System.out.println( x:"Connected");
    }
```

**Adding Username and Password into database**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Document doc =new Document( key:"Username", value:jTextField1.getText());
    doc.append( key:"Password", value:jPasswordField1.getText());
    collection.insertOne( td:doc);
    JOptionPane.showMessageDialog(  parentComponent:this, message:"Signup Successfull");
    new Login().setVisible( b:true);
    this.dispose();
}
```

```
private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    new Login().setVisible( b:true);
    this.dispose();
}
```

# Login-page:



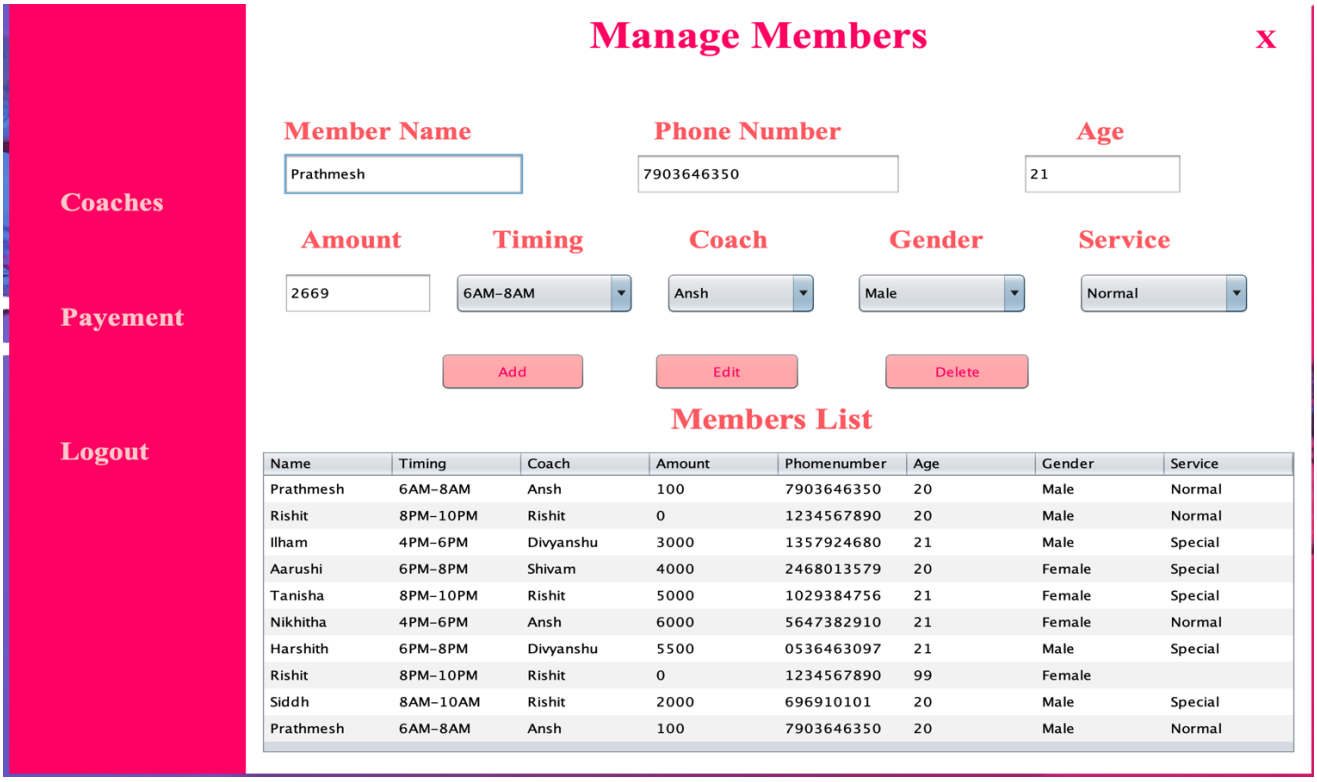
**Login process is also backed by mongo dB database.**

```
public final class Login extends javax.swing.JFrame {
    /**
     * Creates new form Login
     */
    public Login() {
        initComponents();
        connect();
    }
    MongoClient mongoClient;
    MongoDatabase databasename;
    MongoCollection<org.bson.Document> collection;

    public void connect(){
        mongoClient = MongoClients.create( connectionString:"mongodb+srv://pr
        databasename = mongoClient.getDatabase( string:"SDT");
        collection = databasename.getCollection( string:"Admin");
        System.out.println( x:"Connected");
    }
    public String[] returntest(String username, String password){
        String[] s = {jTextField1.getText(),jPasswordField1.getText()};
         return s;
    }
```

**Adding & Checking Username and Password from database for authentication.**

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        BasicDBObject searchQuery = new BasicDBObject( key:"Username", value:jTextField1.getText());
        searchQuery.append( key:"Password",  val:jPasswordField1.getText());
        System.out.println( x:"Retrieving specific Mongo Document");
        MongoCursor<Document> cursor = collection.find( bson:searchQuery).iterator();
        if(cursor.hasNext()){
            JOptionPane.showMessageDialog(  parentComponent:this, message:"Login Successfull");
            new Members().setVisible( b:true);
            this.dispose();
        }
        else{
            JOptionPane.showMessageDialog(  parentComponent:this, message:"Login Failed");
        }
    }
```

# Members-page:




```java
public final class Members extends javax.swing.JFrame {
    /**
     * Creates new form Members
     */
    public Members() {
        initComponents();
        connect();
        loadData();
    }
    MongoClient mongoClient;
    MongoDatabase databasename;
    MongoCollection<org.bson.Document> collection;

    public void connect() {
        mongoClient = MongoClients.create( connectionString:"mongodb+srv://prathmeshDro
        databasename = mongoClient.getDatabase( string:"SDT");
        collection = databasename.getCollection( string:"Member");
        System.out.println( x:"Connected");
    }
    public void loadData() {
        FindIterable<Document> iterDoc = collection.find();
        iterDoc.forEach(doc -> {
            String name = doc.getString( key:"Membername");
            String timing = doc.getString( key:"Timing");
            String coach = doc.getString( key:"Coach");
            String amount = doc.getString( key:"Amount");
            String phonenumber = doc.getString( key:"PhoneNumber");
            String age = doc.getString( key:"Age");
            String gender =  doc.getString( key:"Gender");
            String service = doc.getString( key:"Service");
            Object[] row = {name, timing, coach, amount, phonenumber, age, gender,
            DefaultTableModel model = (DefaultTableModel) listofmembers.getModel();
            model.addRow( rowData:row);
        });
    }
```

## Add Members Data:

```
private void addActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Document doc = new Document( key:"Membername",  value:membername.getText());
    doc.append( key:"PhoneNumber",  value:memberphonenumber.getText());
    doc.append( key:"Age",  value:memberage.getText());
    doc.append( key:"Amount",  value:amountpaidbymember.getText());
    doc.append( key:"Timing",  value:membertiming.getSelectedItem().toString());
    doc.append( key:"Coach",  value:memberCoachName.getSelectedItem().toString());
    doc.append( key:"Gender",  value:memberGender.getSelectedItem().toString());
    doc.append( key:"Service", value:memberservice.getSelectedItem().toString());
    collection.insertOne( td:doc);
    String name = membername.getText();
    String timing = membertiming.getSelectedItem().toString();
    String coach = memberCoachName.getSelectedItem().toString();
    String amount = amountpaidbymember.getText();
    String phonenumber = memberphonenumber.getText();
    String age = memberage.getText();
    String gender = memberGender.getSelectedItem().toString();
    String service = memberservice.getSelectedItem().toString();
    Object[] row = {name, timing, coach, amount, phonenumber, age, gender,service};
    DefaultTableModel model = (DefaultTableModel) listofmembers.getModel();
    model.addRow( rowData:row);
    JOptionPane.showMessageDialog( parentComponent:this,  message:"Member Added");
}
```

## List Update Members Data:

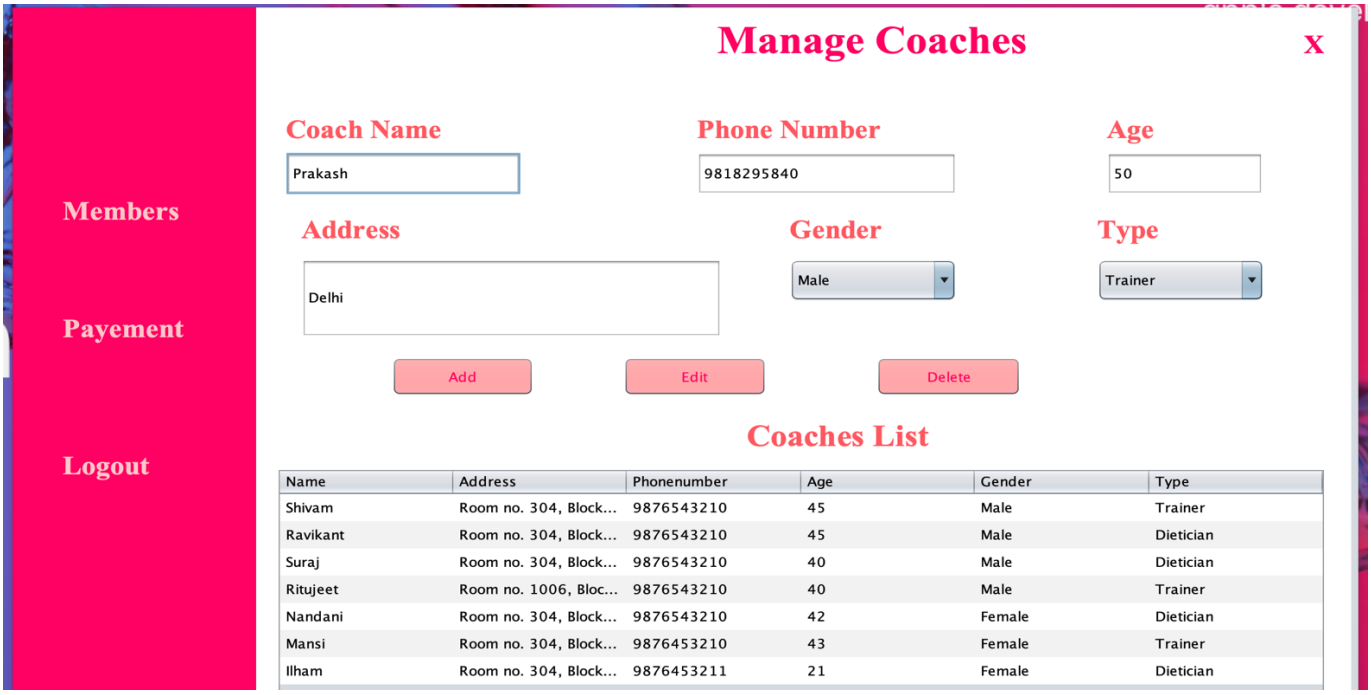
```
private void listofmembersMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel) listofmembers.getModel();
    int selectedRowIndex = listofmembers.getSelectedRow();
    membername.setText( t:model.getValueAt( row:selectedRowIndex,  column:0).toString());
    membertiming.setSelectedIndex( anIndex:timeindex( text:model.getValueAt( row:selectedRowIndex,  column:1).toString()));
    memberCoachName.setSelectedIndex( anIndex:coachindex( text:model.getValueAt( row:selectedRowIndex,  column:2).toString()));
    memberGender.setSelectedIndex( anIndex:genderindex( text:model.getValueAt( row:selectedRowIndex,  column:6).toString()));
    amountpaidbymember.setText( t:model.getValueAt( row:selectedRowIndex,  column:3).toString());
    memberphonenumber.setText( t:model.getValueAt( row:selectedRowIndex,  column:4).toString());
    memberage.setText( t:model.getValueAt( row:selectedRowIndex,  column:5).toString());
    memberservice.setSelectedIndex( anIndex:getService( text:model.getValueAt( row:selectedRowIndex,  column:7).toString()));
}
```

```
private void editActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Object selectedItem1 = membertiming.getSelectedItem();
    Object selectedItem2 = memberCoachName.getSelectedItem();
    collection.updateOne( bson:Filters.eq( fieldName:"Membername",  value:membername.getText()),  bson1:Updates.set( fieldName:"Amount",
    collection.updateOne( bson:Filters.eq( fieldName:"Membername",  value:membername.getText()),  bson1:Updates.set( fieldName:"Timing",
    collection.updateOne( bson:Filters.eq( fieldName:"Membername",  value:membername.getText()),  bson1:Updates.set( fieldName:"Coach",
    collection.updateOne( bson:Filters.eq( fieldName:"Membername", value:membername.getText()), bson1:Updates.set( fieldName:"Service",
    JOptionPane.showMessageDialog( parentComponent:this,  message:"Database Updated");
}
```

## Delete Item:

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    collection.deleteOne( bson:Filters.eq( fieldName:"Membername",  value:membername.getText()));
    DefaultTableModel model = (DefaultTableModel) listofmembers.getModel();
    int selectedRowIndex = listofmembers.getSelectedRow();
    model.removeRow( row:selectedRowIndex);
    membername.setText( t:" ");
    amountpaidbymember.setText( t:" ");
    memberphonenumber.setText( t:" ");
    memberage.setText( t:" ");
    JOptionPane.showMessageDialog( parentComponent:this,  message:"Data Deleted");
}
```

# Coaches-page:




## Connecting to Coach database:

```
public class Coaches extends javax.swing.JFrame {
    /**
     * Creates new form Coaches
     */
    public Coaches() {
        initComponents();
        connect();
        loadData();
    }

    MongoClient mongoClient;
    MongoDatabase databasename;
    MongoCollection<org.bson.Document> collection;

    public void connect() {
        mongoClient = MongoClients.create( connectionString:"mongodb+srv://p
        databasename = mongoClient.getDatabase( string:"SDT");
        collection = databasename.getCollection( string:"Coach");
        System.out.println( x:"Connected");
    }

    public void loadData() {
        FindIterable<Document> iterDoc = collection.find();
        iterDoc.forEach(doc -> {
            String name = doc.getString( key:"Coachname");
            String address = doc.getString( key:"Address");
            String gender = doc.getString( key:"Gender");
            String phonenumber = doc.getString( key:"PhoneNumber");
            String age = doc.getString( key:"Age");
            String type = doc.getString( key:"Type");
            Object[] row = {name, address,phonenumber, age,gender,type}
            DefaultTableModel model = (DefaultTableModel) listofcoaches
```

## Add Coach Data:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Object selectedItem1 = coachgender.getSelectedItem();
    Document doc = new Document( key:"Coachname",  value:coachname.getText());
    doc.append( key:"PhoneNumber",  value:coachphonenumber.getText());
    doc.append( key:"Age",  value:coachage.getText());
    doc.append( key:"Address",  value:coachaddress.getText());
    doc.append( key:"Gender",  value:selectedItem1.toString());
    doc.append( key:"Type", value:coachtype.getSelectedItem().toString());
    collection.insertOne( td:doc);
    String name = doc.getString( key:"Coachname");
    String address = doc.getString( key:"Address");
    String gender = doc.getString( key:"Gender");
    String phonenumber = doc.getString( key:"PhoneNumber");
    String age = doc.getString( key:"Age");
    String type = doc.getString( key:"Type");
    Object[] row = {name, address,phonenumber, age,gender,type};
    DefaultTableModel model = (DefaultTableModel) listofcoaches.getModel();
    model.addRow( rowData:row);
    JOptionPane.showMessageDialog( parentComponent:this,  message:"Member Added");
}
```

## List Updated Coach Data:

```
private void listofcoachesMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel) listofcoaches.getModel();
    int selectedRowIndex = listofcoaches.getSelectedRow();
    coachname.setText( t:model.getValueAt( row:selectedRowIndex,  column:0).toString());
    coachaddress.setText( t:model.getValueAt( row:selectedRowIndex,  column:1).toString());
    coachphonenumber.setText( t:model.getValueAt( row:selectedRowIndex,  column:2).toString());
    coachage.setText( t:model.getValueAt( row:selectedRowIndex,  column:3).toString());
    coachgender.setSelectedIndex( anIndex:genderindex( text:model.getValueAt( row:selectedRowIndex,  column:4).toString()));
    coachtype.setSelectedIndex( anIndex:getService( text:model.getValueAt( row:selectedRowIndex,  column:5).toString()));
}
```

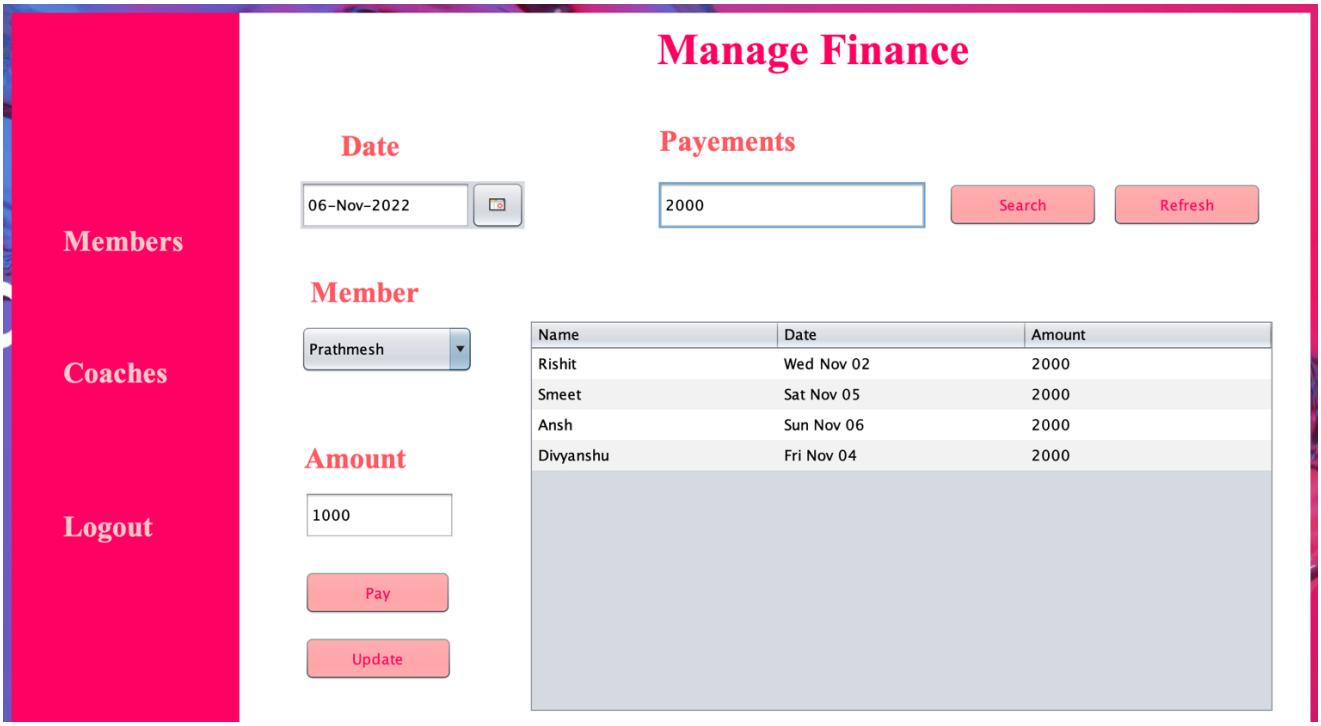
---

## Remove Coaches:

```
private void listofcoachesMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel) listofcoaches.getModel();
    int selectedRowIndex = listofcoaches.getSelectedRow();
    coachname.setText( t:model.getValueAt( row:selectedRowIndex,  column:0).toString());
    coachaddress.setText( t:model.getValueAt( row:selectedRowIndex,  column:1).toString());
    coachphonenumber.setText( t:model.getValueAt( row:selectedRowIndex,  column:2).toString());
    coachage.setText( t:model.getValueAt( row:selectedRowIndex,  column:3).toString());
    coachgender.setSelectedIndex( anIndex:genderindex( text:model.getValueAt( row:selectedRowIndex,  column:4).toString()));
    coachtype.setSelectedIndex( anIndex:getService( text:model.getValueAt( row:selectedRowIndex,  column:5).toString()));
}
```

---

# Payement-page:




## Connecting to Coach database:

```
public final class Payements extends javax.swing.JFrame {
    public Payements() {
        initComponents();
        connect();
        loadData();
    }
    MongoClient mongoClient;
    MongoDatabase databasename;
    MongoCollection<org.bson.Document> collection;

    public void connect() {
        mongoClient = MongoClients.create( connectionString:"mongodb+srv://pra
        databasename = mongoClient.getDatabase( string:"SDT");
        collection = databasename.getCollection( string:"Payement");
        System.out.println( x:"Connected");
    }
    public void loadData() {
        FindIterable<Document> iterDoc = collection.find();
        iterDoc.forEach(doc -> {
            String name = doc.getString( key:"Name");
            String date = doc.getString( key:"Date");
            String amount = doc.getString( key:"Amount");
            Object[] row = {name, date, amount};
            DefaultTableModel model = (DefaultTableModel) payementtable.g
```

## Actions Performed on Payment database:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String result = paydate.getDate().toString().split( regex:"T")[0];
    String realdate = result.split( regex:"00")[0];
    System.out.println( x: paydate.toString());
    Document doc = new Document( key:"Name",  value: paymember.getSelectedItem().toString());
    doc.append( key:"Date", value: realdate);
    doc.append( key:"Amount",  value: payamount.getText());
    collection.insertOne( td: doc);
    String name = doc.getString( key:"Name");
    String date = doc.getString( key:"Date");
    String amount = doc.getString( key:"Amount");
    Object[] row = {name, date, amount};
    DefaultTableModel model = (DefaultTableModel) payementtable.getModel();
    model.addRow( rowData: row);
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Payement Done");
}

private void jLabel4MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    new Coaches().setVisible( b: true);
    this.dispose();
}

private void payementtableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here
    DefaultTableModel model = (DefaultTableModel) payementtable.getModel();
    int selectedRowIndex = payementtable.getSelectedRow();
    paymember.setSelectedIndex( anIndex: coachindex( text: model.getValueAt( row: selectedRowIndex,
    payamount.setText( t: model.getValueAt( row: selectedRowIndex,  column: 2).toString());
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String result = paydate.getDate().toString().split( regex:"T")[0].split( regex:"00")[0];
    collection.updateOne( bson: Filters.eq( fieldName: "Name",  value: paymember.getSelectedItem().
    collection.updateOne( bson: Filters.eq( fieldName: "Name",  value: paymember.getSelectedItem().
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Database Updated");
}
```

# JUnit Testing:

## Signup Testing->

```
public class JUnitTest {

    public JUnitTest() {
    }

    @Test
    public void TestSignup(){
        Signup signup = new Signup();
        String [] li = {"Admin","1234"};
        Assert.assertArrayEquals( expecteds: li, actuals: signup.returntest( username: "Admin",  password: "1234"));
    }
```

## Results of JUnit Testing

```
-------------------------------------
 T E S T S
-------------------------------------
Running Test.JUnitTest
01:56:11.737 [main] INFO org.mongo
Connected
Tests run: 1, Failures: 0, Errors:

Results :

Tests run: 1, Failures: 0, Errors:

-------------------------------------
BUILD SUCCESS
-------------------------------------
Total time:  1.140 s
Finished at: 2022-11-09T01:56:11+0
-------------------------------------
```

Test Results

com.mycompany:GymSdtFisac2:jar:1.0-SNAPSHOT

Tests passed: 100.00 %

The test passed. (0.594 s)

Login Testing->

```java
public class JUnitTest {
    public JUnitTest() {
    }

    @Test
    public void TestLogin(){
      Login login = new Login();
      String [] li = {"Admin","1234"};
      Assert.assertArrayEquals( expecteds: li, actuals: login.returntest( username: "Admin",  password: "1234"));
    }
}
```

Results of JUnit Testing

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running Test.JUnitTest
02:02:36.908 [main] INFO org.mong
Connected
Tests run: 1, Failures: 0, Errors

Results :

Tests run: 1, Failures: 0, Errors

-------------------------------------------------------
BUILD SUCCESS
-------------------------------------------------------
Total time:  1.069 s
Finished at: 2022-11-09T02:02:36+
-------------------------------------------------------
```

Test Results

om.mycompany:GymSdtFisac2:jar:1.0-SNAPSHO

Tests passed: 100.00 %

The test passed. (0.65 s)

## Members Testing->

```
public class JUnitTest {
    public JUnitTest() {
    }
    @Test
    public void TestMembers(){
        Members mem = new Members();
        String [] li = {"Prathmesh","7903646350","21","2669","6AM-8AM","Ansh","Male","Normal"};
        Assert.assertArrayEquals( expecteds: li,  actuals: mem.returntest( name: "Prathmesh", phonenumber: "7903646350", age: "21",
                amount: "2669", timing: "6AM-8AM", coach: "Ansh", gender: "Male", service: "Normal"));
    }

}
```

## Results of JUnit Testing

```
-------------------------------------------
 T E S T S
-------------------------------------------
Running Test.JUnitTest
02:02:36.908 [main] INFO org.mong
Connected
Tests run: 1, Failures: 0, Errors

Results :

Tests run: 1, Failures: 0, Errors

-------------------------------------------
BUILD SUCCESS
-------------------------------------------
Total time:  1.069 s
Finished at: 2022-11-09T02:02:36+
-------------------------------------------
```

Test Results

om.mycompany:GymSdtFisac2:jar:1.0-SNAPSHO

Tests passed: 100.00 %

The test passed. (0.65 s)

Coaches Testing->

```
public class JUnitTest {
    public JUnitTest() {
    }

    @Test
    public void TestCoaches(){
        Coaches coach = new Coaches();
        String [] li = {"Prakash","9818295840","50","Delhi","Male","Trainer"};
        Assert.assertArrayEquals( expecteds: li,  actuals: coach.returntest( name: "Prakash", phonenumber: "9818295840",
                age: "50", address: "Delhi", gender: "Male", type: "Trainer"));
    }

}
```

Results of JUnit Testing

```
Results :

Tests run: 1, Failures: 0, Errors: 0, Skip

------------------------------------------
BUILD SUCCESS
------------------------------------------
Total time:  2.811 s
Finished at: 2022-11-09T02:19:11+05:30
------------------------------------------
```

Test Results ×

m.mycompany:GymSdtFisac2:jar:1.0-SNAPSHOT (Unit) ×

Tests passed: 100.00 %

The test passed. (2.081 s)

Payements Testing->

```
public class JUnitTest {
    public JUnitTest() {
    }
    @Test

     public void TestPayment(){
        Payements pay = new Payements();
        String [] li = {"1000","Prathmesh","1000"};
        Assert.assertArrayEquals( expecteds: li,  actuals: pay.returntest( payment: "1000",
                name: "Prathmesh", amount: "1000"));
    }
}
```

Results of JUnit Testing

```
Results :

Tests run: 1, Failures: 0, Errors: 0, Skip

------------------------------------------
BUILD SUCCESS
------------------------------------------
Total time:  1.938 s
Finished at: 2022-11-09T02:21:46+05:30
------------------------------------------
```

Test Results ×

om.mycompany:GymSdtFisac2:jar:1.0-SNAPSHOT (Unit) ×

Tests passed: 100.00 %

The test passed. (1.369 s)