

Criterion C: Development

The product is developed using Java in the NetBeans IDE. The Gym Management System is a database system which is developed to make the process of registering gym members and trainers easier for the front desk employee at the gym. The user needs to input the information in the given text boxes, and once it is added the information is stored in an online database; which means the user can access the information from other devices as long as they have the login information. To ensure security of the information, the program is equipped with a login page which would only allow access when the user enters both the correct username and password.

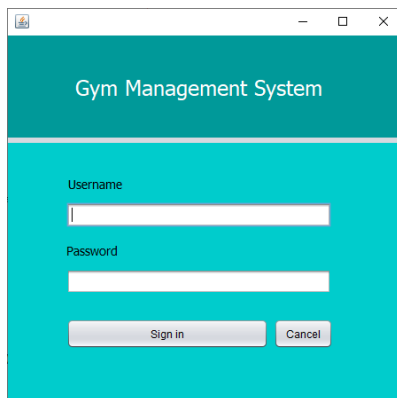


Image 1.0 When the program is run this is the first page, login page, users see.

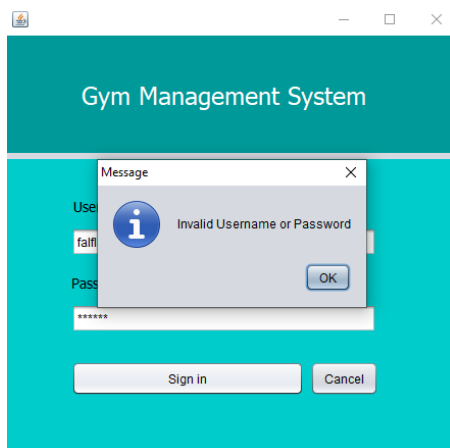


Image 1.1 Error message when either or both the username and password are wrong

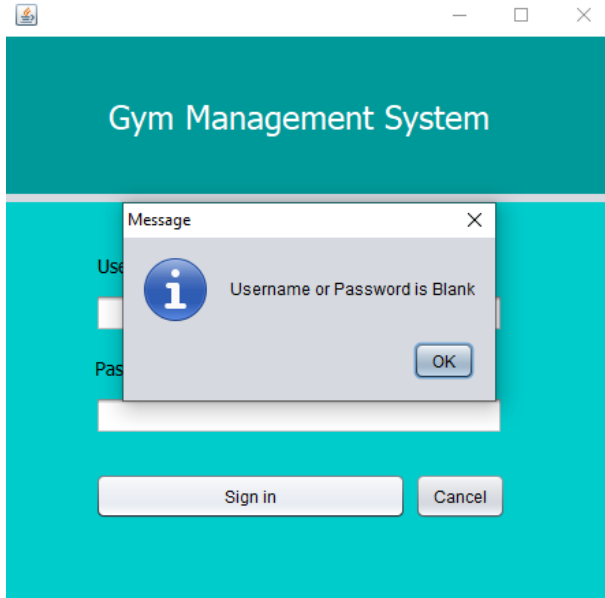


Image 1.2 Error message when one or both text field are empty

After the user is greeted with a login page, they must enter the username “admin” and password “123” to enter; failure to do either will result in the error messages shown in image 1.1 and image 1.2.

```

156
157 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
158     // TODO add your handling code here:
159     String uname = usernameText.getText();
160     String pass = passwordText.getText();
161
162     if(uname.isEmpty() && pass.isEmpty())
163     {
164         JOptionPane.showMessageDialog(this, "Username or Password is Blank");
165     }
166
167     else if(uname.equals("admin") && pass.equals("123"))
168     {
169         Main m = new Main();
170         this.hide();
171         m.setVisible(true);
172     }
173     else
174     {
175         JOptionPane.showMessageDialog(this, "Invalid Username or Password");
176     }
177 }
178

```

Code 1.0 the if-else condition for the login button

The login button utilizes the if-else function as shown in image 2.0. There are in fact three conditions 'if', 'else-if', and 'else'. The code can view to understand the conditions used; when the user enters the password right, the 'else-if' condition executes and the user logs in. When the user logs in they open up the next page which offers a selection between 'add trainer' and 'add member'. The user must select either one of the options to continue as shown in image 2.0.

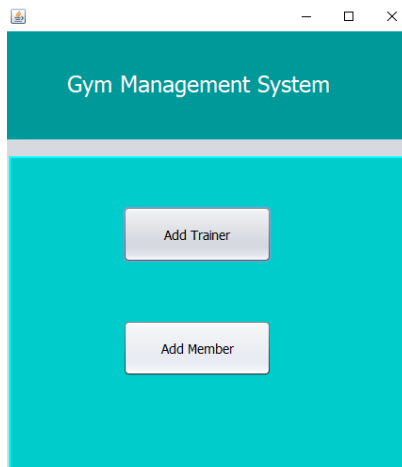


Image 2.0 the choice the program offers when the user login.

Adding Trainer Data into the database:

```

370
371 private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
372     // TODO add your handling code here:
373
374     String tname = trainerText.getText();
375     String age = ageText.getText();
376     String address = addressText.getText();
377     String mobile = mobileText.getText();
378     SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
379     String date = df.format(txtDateReg.getDate());
380
381
382     try {
383         pst = con.prepareStatement("insert into trainer(Name, Age, Address, Mobile, DateReg) values(?, ?, ?, ?, ?)");
384         pst.setString(1, tname);
385         pst.setString(2, age);
386         pst.setString(3, address);
387         pst.setString(4, mobile);
388         pst.setString(5, date);
389         pst.executeUpdate();
390         JOptionPane.showMessageDialog(this, "Trainer Added");
391
392         trainerText.setText("");
393         ageText.setText("");
394         addressText.setText("");
395         mobileText.setText("");
396         txtDateReg.setText("");
397         table_load();
398
399     } catch (SQLException ex) {
400         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
401     }
402
403 }

```

Code 2.0 Connecting to database and then adding data entry into it.

The Code 2.0 shows the programming done to enter data into the trainer table in image 3.0 and then sent to the database. The database used is an online database based on MySQL and requires the XAMPP software on the computer to connect to the database on the internet.

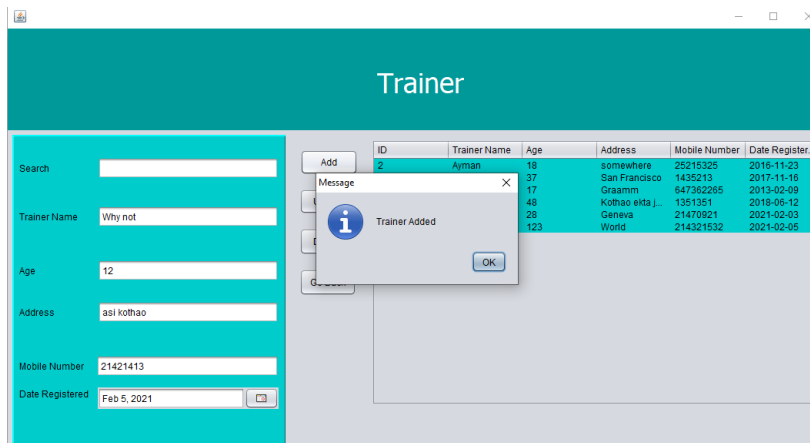


Image 3.0 a message confirms when a trainer is added.

Similarly, the code for updating and deleting are written.

Updating Trainer Data:

```

437 df = (DefaultTableModel)jTable1.getModel();
438 int selected = jTable1.getSelectedRow();
439 int id = Integer.parseInt(df.getValueAt(selected, 0).toString());
440
441 String tname = trainerText.getText();
442 String age = ageText.getText();
443 String address = addressText.getText();
444 String mobile = mobileText.getText();
445 SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
446 String date = df.format(txtDateReg.getDate());
447
448
449
450 try {
451     pst = con.prepareStatement("update trainer set Name = ?,Age = ?, Address = ?, Mobile = ?, DateReg = ? where ID = ?");
452     pst.setString(1,tname);
453     pst.setString(2,age);
454     pst.setString(3,address);
455     pst.setString(4,mobile);
456     pst.setString(5,date);
457     pst.setInt(6,id);
458     pst.executeUpdate();
459     JOptionPane.showMessageDialog(this, "Trainer Updated");
460
461     trainerText.setText("");
462     ageText.setText("");
463     addressText.setText("");
464     mobileText.setText("");
465     trainerText.requestFocus();
466
467     jButton1.setEnabled(true);
468     table_load();
469
470 } catch (SQLException ex) {
471     Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);

```

Code 3.0 Searching the chosen record and then modifying it according to the updates.

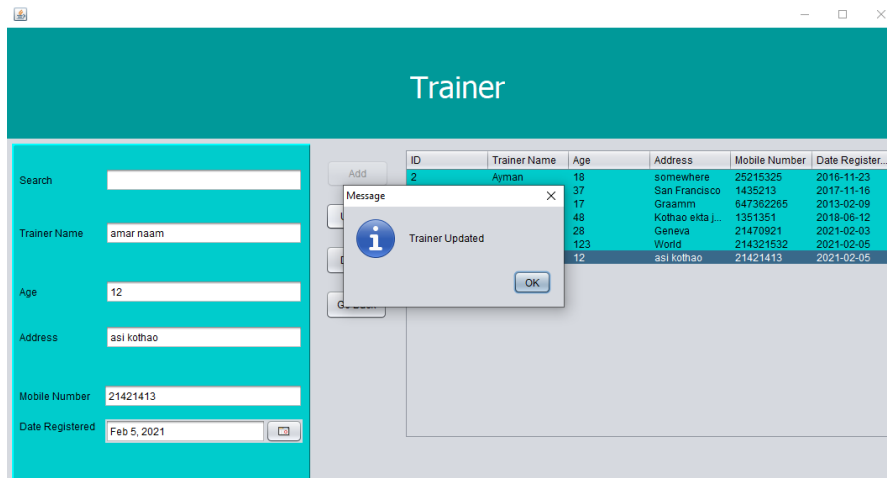


Image 4.0 A pop-up message is displayed when a record is updated.

Deleting Trainer Data:

```

477 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
478     // TODO add your handling code here:
479
480     df = (DefaultTableModel)jTable1.getModel();
481     int selected = jTable1.getSelectedRow();
482     int id = Integer.parseInt(df.getValueAt(selected, 0).toString());
483
484
485     try {
486         pst = con.prepareStatement("delete from trainer where ID = ?");
487
488         pst.setInt(1, id);
489         pst.executeUpdate();
490         JOptionPane.showMessageDialog(this, "Trainer Deleted");
491
492         trainerText.setText("");
493         ageText.setText("");
494         addressText.setText("");
495         mobileText.setText("");
496         trainerText.requestFocus();
497
498         jButton1.setEnabled(true);
499         table_load();
500
501     } catch (SQLException ex) {
502         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
503     }

```

Code 4.0 Deleting a record by looking up the chosen record's ID.

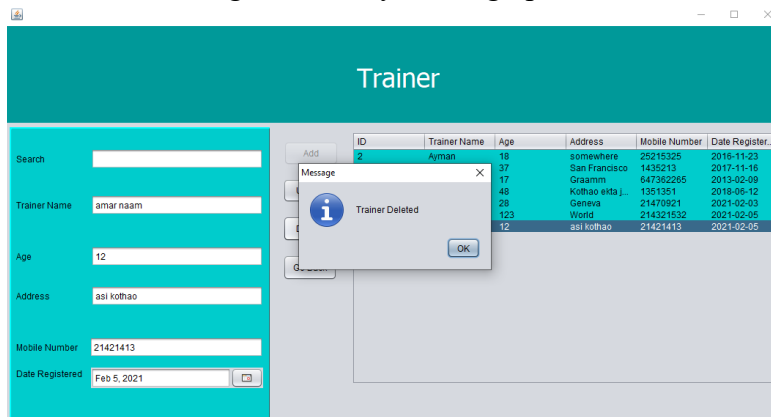


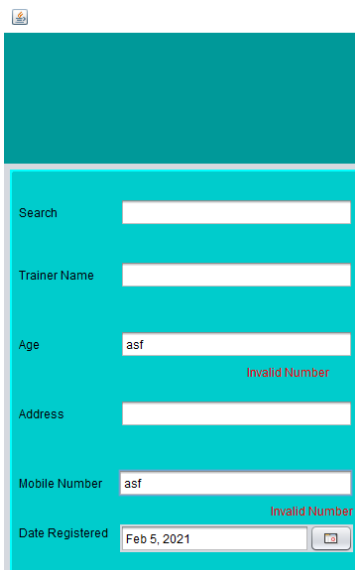
Image 5.0 A pop-up message is displayed when a record is deleted.

Data Validation:

Since in the trainer panel there are few text fields to enter, I have selected the numerical text fields to have data validation to ensure only numbers are inserted.

```
556 private void ageTextKeyTyped(java.awt.event.KeyEvent evt) {  
557     // TODO add your handling code here:  
558  
559     try(  
560         int i = Integer.parseInt(ageText.getText());  
561         ageValidation.setText("");  
562     }catch(NumberFormatException e1){  
563         ageValidation.setText("Invalid Number");  
564     }  
565  
566  
567 }  
568  
569 private void mobileTextKeyTyped(java.awt.event.KeyEvent evt) {  
570     // TODO add your handling code here:  
571  
572     try(  
573         int i = Integer.parseInt(mobileText.getText());  
574         mobileValidation.setText("");  
575     }catch(NumberFormatException e1){  
576         mobileValidation.setText("Invalid Number");  
577     }  
578 }
```

Code 5.0 Programming an error message to display when anything other than numbers is inserted in the relevant text fields.



The screenshot shows a Java Swing window titled "Trainer Panel" with a light blue background. It contains several text input fields with labels to their left: "Search", "Trainer Name", "Age", "Address", "Mobile Number", and "Date Registered". The "Age" and "Mobile Number" fields contain the text "asf". Below each of these fields, the text "Invalid Number" is displayed in red. The "Date Registered" field shows "Feb 5, 2021" and has a small calendar icon to its right.

Image 6.0 Error messages in red “Invalid Number” is displayed when the wrong details are inserted.

I have employed texts in red color “Invalid Number” underneath the numerical text field to show exactly where the user makes the error, instead of a popup error message which would require the user to find the error.

Selecting data from table:

```
405 private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {  
406     // TODO add your handling code here:  
407     try {  
408  
409         df = (DefaultTableModel)jTable1.getModel();  
410  
411         int selected = jTable1.getSelectedRow();  
412  
413         int id = Integer.parseInt(df.getValueAt(selected, 0).toString());  
414  
415         trainerText.setText(df.getValueAt(selected, 1).toString());  
416         ageText.setText(df.getValueAt(selected, 2).toString());  
417         addressText.setText(df.getValueAt(selected, 3).toString());  
418         mobileText.setText(df.getValueAt(selected, 4).toString());  
419  
420  
421         Date date = new SimpleDateFormat("yyyy-MM-dd").parse((String)df.getValueAt(selected, 5));  
422         txtDateReg.setDate(date);  
423  
424         jButton1.setEnabled(false);  
425  
426  
427     } catch (ParseException ex) {  
428         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);  
429     }  
430 }  
431  
432 }
```

Code 6.0 for selecting a record with a mouse click

The screenshot shows a Java Swing window titled "Trainer". On the left, there is a search bar and several input fields: "Trainer Name" (containing "Hello"), "Age" (containing "123"), "Address" (containing "World"), "Mobile Number" (containing "214321532"), and "Date Registered" (containing "Feb 5, 2021"). In the center, there are four buttons: "Add", "Update", "Delete", and "Go Back". On the right, there is a table with the following data:

ID	Trainer Name	Age	Address	Mobile Number	Date Register...
2	Ayman	18	somewhere	25215325	2016-11-23
3	Kyle	37	San Francisco	1435213	2017-11-16
6	Adi	17	Graamm	647362265	2013-02-09
7	Mahika	48	Kothao ehta j...	1351351	2018-08-12
12	Jack	28	Geneva	21470921	2021-02-03
13	Hello	123	World	214321532	2021-02-05

Image 7.0 A record has been selected with a mouse click and all the input boxes are filled with relevant information.

Search:

The product is equipped with a search engine to find names of the trainers, this saves the user time from searching through the table.

```
520 private void searchKeyReleased(java.awt.event.KeyEvent evt) {  
521     // TODO add your handling code here:  
522  
523     try {  
524         Class.forName("com.mysql.jdbc.Driver");  
525         con = DriverManager.getConnection("jdbc:mysql://localhost/gymmanagementsystem", "root", "");  
526         String sql = "Select * from trainer where Name =?";  
527         PreparedStatement pst = con.prepareStatement(sql);  
528         pst.setString(1, search.getText());  
529         ResultSet rs = pst.executeQuery();  
530         if(rs.next()){  
531  
532             String settxttrainer = rs.getString("Name");  
533             trainerText.setText(settxttrainer);  
534  
535             String settxtage = rs.getString("Age");  
536             ageText.setText(settxtage);  
537  
538             String settxtaddress = rs.getString("Address");  
539             addressText.setText(settxtaddress);  
540  
541             String settxtmobile = rs.getString("Mobile");  
542             mobileText.setText(settxtmobile);  
543  
544             txtDateReg.setDate(rs.getDate("DateReg"));  
545         }  
546  
547     } catch (ClassNotFoundException | SQLException ex) {  
548         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);  
549     }  
550 }  
551
```

Searching data:

Code 7.0 for searching data in database

The screenshot shows a Java Swing window titled "Trainer". On the left is a search form with fields for "Search" (containing "Ayman"), "Trainer Name" (containing "Ayman"), "Age" (containing "18"), "Address" (containing "somewhere"), "Mobile Number" (containing "25215325"), and "Date Registered" (containing "Nov 23, 2016"). To the right of the form are buttons for "Add", "Update", "Delete", and "Go Back". On the right side of the window is a table with the following data:

ID	Trainer Name	Age	Address	Mobile Number	Date Register...
2	Ayman	18	somewhere	25215325	2016-11-23
3	Kyle	37	San Francisco	1435213	2017-11-18
6	Adit	17	Graamm	647362265	2013-02-09
7	Mahika	48	Kohao ettsj...	1351351	2018-06-12
12	Jack	28	Ceneva	21470921	2021-02-03
13	Helio	123	World	214321532	2021-02-05

Image 8.0 Typing the name “Ayman” to search that record

Similarly, this was done for Member panel of the product.

Adding Member data in the database:

```
439 // TODO add your handling code here:
440 String fname = firstNameText.getText();
441 String lname = lastNameText.getText();
442 SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
443 String dob = df.format(DOBText.getDate());
444 String mobile = mobileText.getText();
445 String gender = genderText.getSelectedText().toString();
446 String address = addressText.getText();
447 String sub = subscriptionText.getSelectedText().toString();
448 String amount = amountText.getText();
449 SimpleDateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
450 String sdate = df1.format(StartDate.getDate());
451 String trainer = trainerText.getSelectedText().toString();
452
453
454 try {
455     pst = con.prepareStatement("insert into member(FirstName,LastName,DOB,Mobile,Gender,Address,Subscription,Amount,Date,Trainer)values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
456     pst.setString(1, fname);
457     pst.setString(2, lname);
458     pst.setString(3, dob);
459     pst.setString(4, mobile);
460     pst.setString(5, gender);
461     pst.setString(6, address);
462     pst.setString(7, sub);
463     pst.setString(8, amount);
464     pst.setString(9, sdate);
465     pst.setString(10, trainer);
466     pst.executeUpdate();
467     JOptionPane.showMessageDialog(this, "Member Added");
468
469 } catch (SQLException ex) {
470     Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
471 }
472
```

Code 8.0 Connecting to database and then adding data entry into it.

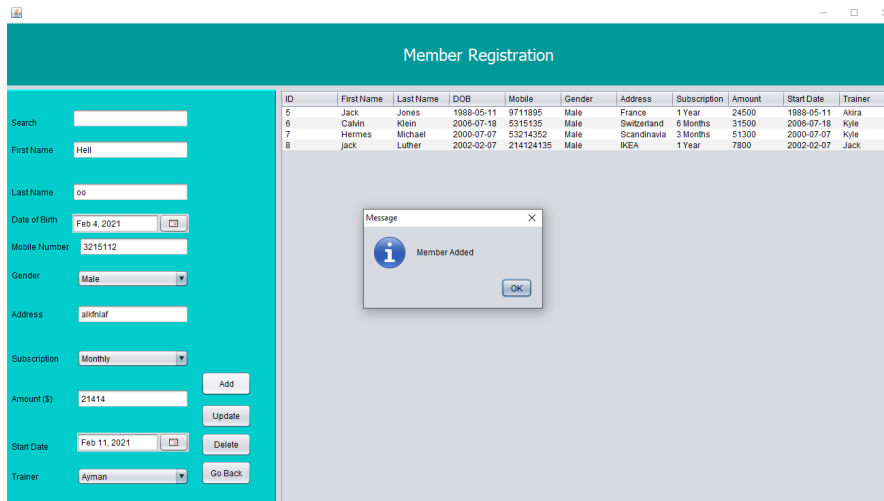


Image 9.0 A pop-up message is displayed when a record is Added.

Updating Member data:

```

518 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
519     // TODO add your handling code here:
520
521     df = (DefaultTableModel) jTable1.getModel();
522     int selected = jTable1.getSelectedRow();
523     int id = Integer.parseInt(df.getValueAt(selected, 0).toString());
524
525     String fname = firstNameText.getText();
526     String lname = lastNameText.getText();
527     SimpleDateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
528     String dob = df1.format(DOBText.getDate());
529     String mobile = mobileText.getText();
530     String gender = genderText.getSelectedText().toString();
531     String address = addressText.getText();
532     String sub = subscriptionText.getSelectedText().toString();
533     String amount = amountText.getText();
534     SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd");
535     String sdate = df2.format(StartDate.getDate());
536     String trainer = trainerText.getSelectedText().toString();
537
538
539
540
541     try {
542         pst = con.prepareStatement("update member set FirstName = ?, LastName = ?, DOB = ?, Mobile = ?, Gender = ?, Address = ?, Subscription = ?, Amount = ?, Date = ?, Trainer = ? where ID = ?");
543         pst.setString(1, fname);
544         pst.setString(2, lname);
545         pst.setString(3, dob);
546         pst.setString(4, mobile);
547         pst.setString(5, gender);
548         pst.setString(6, address);
549         pst.setString(7, sub);
550         pst.setString(8, amount);
551         pst.setString(9, sdate);
552         pst.setString(10, trainer);
553         pst.setInt(11, id);
554         pst.executeUpdate();
555         JOptionPane.showMessageDialog(this, "Member Updated");
556
557         firstNameText.setText("");
558         lastNameText.setText("");
559         mobileText.setText("");
560         genderText.setSelectedText("");
561         addressText.setText("");
562         subscriptionText.setSelectedText("");
563         amountText.setText("");
564         trainerText.setSelectedText("");
565         firstNameText.requestFocus();
566
567         jButton1.setEnabled(true);
568         table_load();
569
570     } catch (SQLException ex) {
571         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
572     }
573 }

```

Code 9.0 to find and update a record in the database when selected.

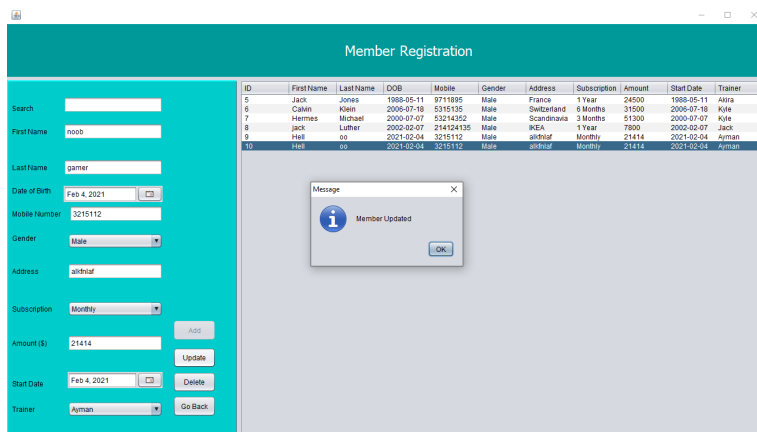


Image 10.0 A pop-up message is displayed when a record is updated.

Deleting Member Data:

```
476 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
477     // TODO add your handling code here:  
478  
479     df = (DefaultTableModel) jTable1.getModel();  
480     int selected = jTable1.getSelectedRow();  
481     int id = Integer.parseInt(df.getValueAt(selected, 0).toString());  
482  
483  
484     try {  
485         pst = con.prepareStatement("delete from member where ID = ?");  
486  
487         pst.setInt(1, id);  
488         pst.executeUpdate();  
489         JOptionPane.showMessageDialog(this, "Member Deleted");  
490  
491         firstNameText.setText("");  
492         lastNameText.setText("");  
493         //txtDOB.setText("");  
494         mobileText.setText("");  
495         genderText.setSelectedItem("");  
496         addressText.setText("");  
497         subscriptionText.setSelectedItem("");  
498         amountText.setText("");  
499         //txtDate.setText("");  
500         trainerText.setSelectedItem("");  
501         firstNameText.requestFocus();  
502  
503         jButton1.setEnabled(true);  
504         table_load();  
505  
506     } catch (SQLException ex) {  
507         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);  
508     }  
509 }
```

Code 10.0 Deleting a record by looking up the chosen record's ID.

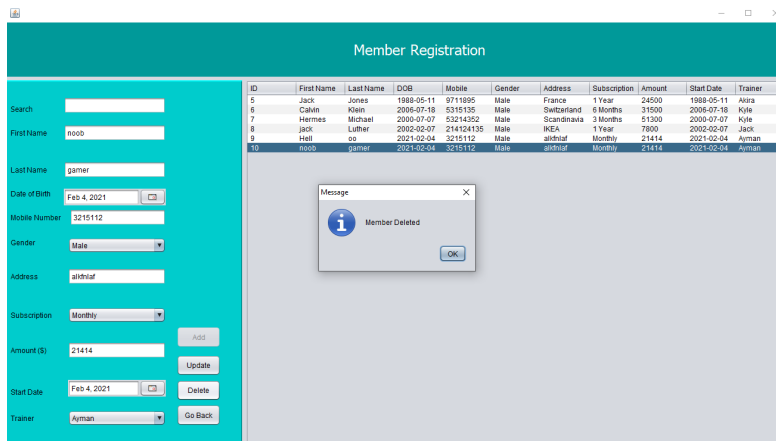


Image 11.0 A pop-up message is displayed when a record is deleted.

Data Validation:

```
666 private void mobileTextKeyTyped(java.awt.event.KeyEvent evt) {  
667     // TODO add your handling code here:  
668     try{  
669         int i = Integer.parseInt(mobileText.getText());  
670         mobileValidation.setText("");  
671     }catch(NumberFormatException e){  
672         mobileValidation.setText("Invalid Number");  
673     }  
674 }  
675 }  
676  
677 private void amountTextKeyTyped(java.awt.event.KeyEvent evt) {  
678     // TODO add your handling code here:  
679     try{  
680         int i = Integer.parseInt(amountText.getText());  
681         amountValidation.setText("");  
682     }catch(NumberFormatException e){  
683         amountValidation.setText("Invalid Number");  
684     }  
685 }
```

Code 11.0 Programming an error message to display when anything other than numbers is inserted in the relevant text fields.

The screenshot shows a user registration form on a teal background. The form includes the following fields and controls:

- Search:** A text input field.
- FirstName:** A text input field.
- LastName:** A text input field.
- Date of Birth:** A date picker showing "Feb 4, 2021".
- Mobile Number:** A text input field containing "afa". Below it, the text "Invalid Number" is displayed in red.
- Gender:** A dropdown menu with "Male" selected.
- Address:** A text input field.
- Subscription:** A dropdown menu with "Monthly" selected.
- Amount (\$):** A text input field containing "af". Below it, the text "Invalid Number" is displayed in red.
- Start Date:** A date picker showing "Feb 4, 2021".
- Trainer:** A dropdown menu with "Ayman" selected.

At the bottom right of the form, there are four buttons: "Add", "Update", "Delete", and "Go Back".

Image 12.0 Error messages in red “Invalid Number” is displayed when the wrong details are inserted.

Furthermore, in the Member panel of the product, drop-down-menus were employed to reduce errors made by the user and increase their productivity overall. The image shows the product has three drop-down-menus for Member panel. The task was done using NetBeans Swing GUI features.



Search

First Name

Last Name

Date of Birth 

Mobile Number
Invalid Number

Gender

Address

Subscription

Amount (\$)
Invalid Number

Start Date 

Trainer

Image 13.0 A drop-down-menu for gender is provided.

Selecting data from the Member table:

```

176 private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
177     // TODO add your handling code here:
178     try {
179         df = (DefaultTableModel) jTable1.getModel();
180
181         int selected = jTable1.getSelectedRow();
182
183         int id = Integer.parseInt(df.getValueAt(selected, 0).toString());
184
185         firstNameText.setText(df.getValueAt(selected, 1).toString());
186         lastNameText.setText(df.getValueAt(selected, 2).toString());
187         Date date = new SimpleDateFormat("yyyy-MM-dd").parse((String) df.getValueAt(selected, 3));
188         doDate.setText(date);
189         mobileText.setText(df.getValueAt(selected, 4).toString());
190         genderText.setSelectedItem(df.getValueAt(selected, 5).toString());
191         addressText.setText(df.getValueAt(selected, 6).toString());
192         subscriptionText.setSelectedItem(df.getValueAt(selected, 7).toString());
193         amountText.setText(df.getValueAt(selected, 8).toString());
194         Date sdate = new SimpleDateFormat("yyyy-MM-dd").parse((String) df.getValueAt(selected, 9));
195         txtDate.setDate(sdate);
196         trainerText.setSelectedItem(df.getValueAt(selected, 10).toString());
197
198
199
200
201
202
203         jButton1.setEnabled(false);
204
205
206
207     } catch (ParseException ex) {
208         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
209     }
210 }

```

Code 12.0 for selecting a record with a mouse click

The screenshot shows a Java Swing application titled "Member Registration". On the left is a registration form with various input fields. On the right is a table displaying a list of members. The table has columns for ID, First Name, Last Name, DOB, Mobile, Gender, Address, Subscription, Amount, Start Date, and Trainer. The record for ID 6 (Calvin Klein) is selected and highlighted in blue. The form fields on the left are populated with the data from this selected record: First Name is "Calvin", Last Name is "Klein", Date of Birth is "Jul 18, 2006", Mobile Number is "5315135", Gender is "Male", Address is "Switzerland", Subscription is "6 Months", Amount (\$) is "31500", Start Date is "Jul 18, 2006", and Trainer is "Kyle". Buttons for "Add", "Update", "Delete", and "Go Back" are also visible.

ID	First Name	Last Name	DOB	Mobile	Gender	Address	Subscription	Amount	Start Date	Trainer
5	Jack	Jones	1988-05-11	9711895	Male	France	1 Year	24500	1988-05-11	Akira
6	Calvin	Klein	2006-07-18	5315135	Male	Switzerland	6 Months	31500	2006-07-18	Kyle
7	Hermes	Michael	2000-07-07	53214352	Male	Scandinavia	3 Months	51300	2000-07-07	Kyle
8	Jack	Luther	2002-02-07	214124135	Male	IKEA	1 Year	7800	2002-02-07	Jack
9	Hell	oo	2021-02-04	3215112	Male	alKhiaf	Monthly	21414	2021-02-04	Ayman

Image 14.0 A record has been selected with a mouse click and all the input boxes are filled with relevant information.

Searching data in Members in table and database:

```

617 private void searchKeyReleased(java.awt.event.KeyEvent evt) {
618     // TODO add your handling code here:
619     try {
620         Class.forName("com.mysql.jdbc.Driver");
621         con = DriverManager.getConnection("jdbc:mysql://localhost/gymmanagementssystem", "root", "");
622         String sql = "Select * from member where FirstName ~?";
623         PreparedStatement pst = con.prepareStatement(sql);
624         pst.setString(1, search.getText());
625         ResultSet rs = pst.executeQuery();
626         if(rs.next()){
627
628             String settxtfname = rs.getString("FirstName");
629             firstNameText.setText(settxtfname);
630
631             String settxtlname = rs.getString("LastName");
632             lastNameText.setText(settxtlname);
633
634             DOBText.setDate(rs.getDate("DOB"));
635
636             String settxtmobile = rs.getString("Mobile");
637             mobileText.setText(settxtmobile);
638
639             /* String settxtgender = rs.getString("Gender");
640             genderText.setText(settxtgender); */
641
642             String settxtaddress = rs.getString("Address");
643             addressText.setText(settxtaddress);
644
645             //String settxtsub= rs.getString("Subscription");
646             //txtsub.setText(settxtsub);
647
648             String settxtamount = rs.getString("Amount");
649             amountText.setText(settxtamount);
650
651             txtdate.setDate(rs.getDate("Date"));
652
653             // String settxttrainer = rs.getString("Trainer");
654             //txttrainer.setText(settxttrainer);
655
656         }
657
658     } catch (ClassNotFoundException | SQLException ex) {
659         Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, null, ex);
660     }
661 }

```

Code 13.0 for searching data in database

ID	First Name	Last Name	DOB	Mobile	Gender	Address	Subscription	Amount	Start Date	Trainer
5	Jack	Jones	1988-05-11	9711895	Male	France	1 Year	24500	1988-05-11	Alira
6	Cake	Klen	2006-07-18	5315135	Male	Switzerland	6 Months	31500	2006-07-18	Kyle
7	Hermes	Michael	2000-07-07	5214352	Male	Scandinavia	3 Months	51300	2000-07-07	Kyle
8	Jack	Luther	2002-02-07	214124135	Male	IKEA	1 Year	7800	2002-02-07	Jack
9	Heli	oo	2021-02-04	3215112	Male	alstuf	Monthly	21414	2021-02-04	Ayman

Image 15.0 Typing the name “Hermes” to search that record

Classes and Methods employed in the product:

Class:

Class	Description
public class Main extends javax.swing.JFrame	This is the main class of the program; it handles the execution of the entire product and connects the whole thing. Furthermore, this contains the methods which are needed to present user with the option between 'Add Trainer' and 'Add Member'.
public class Trainer extends javax.swing.JFrame	The users see this page when they select 'Add Trainer', and much like the member class, this too contains all the necessary methods to register a new trainer.
public class login extends javax.swing.JFrame	This is the first page users see when they run the program; it contains the methods for username, password, and other buttons.
public class Member extends javax.swing.JFrame	This is page user see when they select the 'Add Member' option; it contains all the necessary methods (add, delete, update etc.) to register a new member.

Method:

Method	Description
public void table_load()	For inserting and viewing the table, made in the online database, in the program.
public void Connect()	To connect the online database with the program.
private void jButtonActionPerformed	To program different operations (add, delete, update, go back) for the buttons in the program.
private void jTableMouseClicked	When mouse clicked over a row in the table, it inserts all the relevant information in the input text fields.
private void searchKeyReleased	When user stops typing in the search box, the method looks for the name entered and fills up the input boxes with relevant information.
private void mobileTextKeyTyped	When anything other than numbers is typed, and error message is shown.
private void ageTextKeyTyped	When anything other than numbers is typed, the method shows an error message.
private void amountTextKeyTyped(The amount is numerical figure, so when anything other than numbers are typed in this field errors are shown.

<code>public void load_trainer()</code>	Brings in the trainer names from the trainer table in the database and then presents them in a drop-down-menu in the Member panel.
---	--

[Word Count: 1395]