***SVV ASSIGNMENT 3***

***18-SE-86***

***WHITE BOX TESTING OF GYM MANAGEMENT SYSTEM***

***The system had following user defined functions***

***1 Splash.initialize()***

@FXML

    public void initialize() {

       // makeStageDrageable();

        FadeTransition.applyFadeTransition(parent, Duration.seconds(2), (e) -> {

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/login.fxml"));

            try {

                loader.load();

            } catch (IOException ex) {

                ex.printStackTrace();

            }

            System.out.println("user");

            Parent root = loader.getRoot();

            Stage stage = new Stage();

            stage.setScene(new Scene(root));

            stage.show();

            stage.setResizable(false);

            parent.getScene().getWindow().hide();

           // stage.showAndWait();

        });

    }

***2 DBQueries.getConnection()***

public Connection getDbConnection() throws ClassNotFoundException, SQLException {

//step1 load the driver class

            Class.forName ("oracle.jdbc.OracleDriver");

//step2 create  the connection object

             dbConnection= DriverManager.getConnection(

                    "jdbc:oracle:thin:@localhost:1521:orcl","SYSTEM","multansultan8");

             return dbConnection;

    }

***3 Signup employee()***

    public void signUpEmployee(Employee employee,String value) {

        String tableName= "AdminT";

        String id ="adminId";

         if(value.equals("Admin")){

            tableName = "AdminT";

            id = "adminId";

        }else if(value.equals("Trainer")){

            tableName = "TrainerT";

            id = "trainerId";

        }else if(value.equals("Nutriant")){

            tableName = "NutriantT";

            id = "nutriantId";

        }

        String insert;

         if(tableName.equals("TrainerT")) {

             insert = "INSERT INTO " + tableName + " ( " + id + " , " +

                     "firstName, lastName, userName , userPassword, " +

                     "qualification , experience,timeSlot) VALUES(?,?,?,?,?,?,?,?)";

         }else if(tableName.equals("NutriantT")){

             insert = "INSERT INTO " + tableName + " ( " + id + " , " +

                     "firstName, lastName, userName , userPassword, " +

                     "qualification , experience,timeSlot) VALUES(?,?,?,?,?,?,?,?)";

         }else {

            insert = "INSERT INTO " + tableName + " ( " + id + " , " +

                    "firstName, lastName, userName , userPassword, " +

                    "qualification , experience) VALUES(?,?,?,?,?,?,?)";

        }

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(insert);

            preparedStatement.setLong(1, employee.getIdNo());

            preparedStatement.setString(2, employee.getFirstName());

            preparedStatement.setString(3, employee.getLastName());

            preparedStatement.setString(4, employee.getName());

            preparedStatement.setString(5, employee.getPassword());

            preparedStatement.setString(6, employee.getQualification());

            preparedStatement.setString(7, employee.getExperience());

            if(tableName.equals("TrainerT") || tableName.equals("NutriantT")){

                preparedStatement.setString(8,employee.getSlotTime());

            }

            preparedStatement.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***4 Get Employee()***

 public ResultSet getEmployee(Employee employee , String value) {

        ResultSet resultSet = null;

        String tableName= "AdminT";

//        if(value.equals("User")){

//            tableName = "UserT";

//        }else

        if(value.equals("Admin")){

            tableName = "AdminT";

        }else if(value.equals("Trainer")){

            tableName = "TrainerT";

        }else if(value.equals("Nutriant")){

            tableName = "NutriantT";

        }

        if (!employee.getName().equals("") || !employee.getPassword().equals("")) {

            String query = "SELECT \* FROM " + tableName + " WHERE "

                    +  "userName=?" + " AND " + "userPassword=?";

            // select all from users where username="paulo" and password="password"

            try {

                PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

                preparedStatement.setString(1, employee.getName());

                preparedStatement.setString(2, employee.getPassword());

                resultSet = preparedStatement.executeQuery();

            } catch (SQLException e) {

                e.printStackTrace();

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

            }

        } else {

            System.out.println("Please enter your credentials");

        }

        return resultSet;

    }

***5 Signup user()***

public void signUpUser(User user) throws SQLException, ClassNotFoundException {

        CallableStatement insert = getDbConnection().prepareCall("{ call PROCEDURE2(?,?,?,?,?,?,?,?,?,?) }");

        insert.setLong(1, user.getIdNo());

        insert.setString(2, user.getFirstName());

        insert.setString(3, user.getLastName());

        insert.setString(4, user.getName());

        insert.setString(5, user.getPassword());

        insert.setLong(6, user.getUserAge());

        insert.setLong(7, user.getUserWeight());

        insert.setLong(8, user.getFee());

        insert.setString(9, user.getMuscle());

        insert.setString(10,user.getTimeSlot());

        insert.executeUpdate();

    }

***6 Get user()***

 public ResultSet getUser(User user) {

        ResultSet resultSet = null;

        if (!user.getName().equals("") || !user.getPassword().equals("")) {

            String query = "SELECT \* FROM  UserT WHERE "

                    +  "userName=?" + " AND " + "userPassword=?";

            try {

                PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

                preparedStatement.setString(1, user.getName());

                preparedStatement.setString(2, user.getPassword());

                resultSet = preparedStatement.executeQuery();

            } catch (SQLException e) {

                e.printStackTrace();

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

            }

        } else {

            System.out.println("Please enter your credentials");

        }

        return resultSet;

    }

***7 Get All Users()***

public ResultSet getAllUsers() throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  UserT";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***8 Delete user()***

 public void deleteUser(long usrId) throws SQLException, ClassNotFoundException {

        String query1 = "DELETE FROM ExerciseChart  WHERE userId=?";

        PreparedStatement ps = getDbConnection().prepareStatement(query1);

        ps.setLong(1, usrId);

        ps.execute();

        ps.close();

        String query2 = "DELETE FROM FoodChart  WHERE userId=?";

        PreparedStatement ps1 = getDbConnection().prepareStatement(query2);

        ps1.setLong(1, usrId);

        ps1.execute();

        ps1.close();

        String query3 = "DELETE FROM ExerciseT  WHERE userId=?";

        PreparedStatement ps2 = getDbConnection().prepareStatement(query3);

        ps2.setLong(1, usrId);

        ps2.execute();

        ps2.close();

        String query = "DELETE FROM UserT  WHERE userId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, usrId);

        preparedStatement.execute();

        preparedStatement.close();

    }

***9 Add Equipment()***

 public void addEquipment(Equipment equipment , FileInputStream fis , File file){

        String query = "INSERT INTO EquipT ( equId , image ," +

                " equName, equModel , equCast ," +

                " dateOfPurchase , dateOfMaintance )" +

                "VALUES(?,?,?,?,?,?,?)";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setInt(1,equipment.getId());

           // preparedStatement.setString(2,equipment.getImage());

            preparedStatement.setBinaryStream(2,fis,(int) file.length());

            preparedStatement.setString(3,equipment.getName());

            preparedStatement.setString(4,equipment.getModel());

            preparedStatement.setLong(5,equipment.getCost());

            preparedStatement.setDate(6,  equipment.getDateOfPurchase());

            preparedStatement.setDate(7, equipment.getLastMaintenance());

            preparedStatement.execute();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***10 Get Equipment()***

 public ResultSet getEquipment() throws SQLException,ClassNotFoundException{

        String query = "SELECT \* FROM  EquipT";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***11 Delete Equipment()***

   public void deleteEqu(long id)throws SQLException,ClassNotFoundException{

        String query = "DELETE FROM EquipT  WHERE equId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, id);

        preparedStatement.execute();

        preparedStatement.close();

    }

***12 Get All Trainers()***

public ResultSet getAllTrainers() throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  TrainerT";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***13 Delete Trainer()***

public void deleteTrainer(long trainerId) throws SQLException, ClassNotFoundException {

        String query1 = "DELETE FROM ExerciseChart  WHERE trainerId=?";

        PreparedStatement ps = getDbConnection().prepareStatement(query1);

        ps.setLong(1, trainerId);

        ps.execute();

        ps.close();

        String query = "DELETE FROM TrainerT  WHERE trainerId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, trainerId);

        preparedStatement.execute();

        preparedStatement.close();

    }

***14 Get All Nutritionists()***

ublic ResultSet getAllNutriants() throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  NutriantT";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***15 Delete Nutritionists()***

 public void deleteNutriant(long nutriantId) throws SQLException, ClassNotFoundException {

        String query2 = "DELETE FROM FoodChart  WHERE nutriantId=?";

        PreparedStatement ps1 = getDbConnection().prepareStatement(query2);

        ps1.setLong(1, nutriantId);

        ps1.execute();

        ps1.close();

        String query = "DELETE FROM NutriantT  WHERE nutriantId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, nutriantId);

        preparedStatement.execute();

        preparedStatement.close();

    }

***16 Get Exercise Chart()***

public ResultSet getExerciseChart() throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  ExerciseChart";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***17 Add Exercise()***

public void addExercise(ExerciseChart exerciseChart , ObservableList<String> list) throws SQLException, ClassNotFoundException {

        String query = "INSERT INTO ExerciseChart ( userId , trainerId , muscleType,timeSlot)" +

                "VALUES(?,?,?,?)";

        try{

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1,exerciseChart.getUserId());

            preparedStatement.setLong(2,exerciseChart.getTrainerId());

            preparedStatement.setString(3,exerciseChart.getMuscleType());

            preparedStatement.setString(4, LoginController.timeSlot);

            preparedStatement.execute();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        for(int i=0 ; i<list.size() ; i++){

            System.out.println(list.get(i));

            System.out.println(list.size());

                if(list.get(i).equals("Core")){

                    for(int j=0 ; j<3;j++){

                        String q1 = "INSERT INTO ExerciseT (muscleType , exerciseType,userId)" +

                                "VALUES(?,?,?)";

                        PreparedStatement preparedStatement = getDbConnection().prepareStatement(q1);

                        //preparedStatement.setLong(1,(j+10+exerciseChart.getUserId()));

                        preparedStatement.setString(1,"Core");

                        preparedStatement.setString(2,coreList[j]);

                        preparedStatement.setLong(3,exerciseChart.getUserId());

                        preparedStatement.execute();

                    }

                }else if(list.get(i).equals("Back")) {

                    for (int k = 0; k < 4; k++) {

                        String q2 = "INSERT INTO ExerciseT ( excId , muscleType , exerciseType,userId)" +

                                "VALUES(?,?,?,?)";

                        PreparedStatement p1 = getDbConnection().prepareStatement(q2);

                        p1.setLong(1, (k + 20+exerciseChart.getUserId()));

                        p1.setString(2, "Back");

                        p1.setString(3, backList[k]);

                        p1.setLong(4, exerciseChart.getUserId());

                        p1.execute();

                    }

                }else if(list.get(i).equals("Upper limbs")) {

                    for (int l = 0; l < 3; l++) {

                        String q3 = "INSERT INTO ExerciseT ( excId , muscleType , exerciseType,userId)" +

                                "VALUES(?,?,?,?)";

                        PreparedStatement p2 = getDbConnection().prepareStatement(q3);

                        p2.setLong(1, (l + 30+exerciseChart.getUserId()));

                        p2.setString(2, "Upper limbs");

                        p2.setString(3, upperList[l]);

                        p2.setLong(4, exerciseChart.getUserId());

                        p2.execute();

                    }

                }else if(list.get(i).equals("Lower limbs")) {

                    for (int m = 0; m < 5; m++) {

                        String q2 = "INSERT INTO ExerciseT ( excId , muscleType , exerciseType,userId)" +

                                "VALUES(?,?,?,?)";

                        PreparedStatement p1 = getDbConnection().prepareStatement(q2);

                        p1.setLong(1, (m + 40+exerciseChart.getUserId()));

                        p1.setString(2, "Lower limbs");

                        p1.setString(3, lowerList[m]);

                        p1.setLong(4, exerciseChart.getUserId());

                        p1.execute();

                    }

                }

        }

    }

***18 Get Exercise()***

public ResultSet getExercises(long userId) throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  ExerciseT WHERE userId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1,userId);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***19 Delete Exercise()***

public void deleteExercise(long exId) throws SQLException, ClassNotFoundException {

        String query = "DELETE FROM ExerciseT  WHERE excId=?";

        System.out.println("db"+exId);

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, exId);

        preparedStatement.execute();

        preparedStatement.close();

    }

***20 Update Exercise()***

 public void updateExercise(long id, String exerciseTyp) throws SQLException, ClassNotFoundException {

        String query = "UPDATE ExerciseT SET exerciseType=? WHERE excid=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setString(1, exerciseTyp);

        preparedStatement.setLong(2, id);

        preparedStatement.executeUpdate();

        preparedStatement.close();

    }

***21 Add Food()***

public void addFood(FoodChart foodChart){

        String query = "INSERT INTO FoodChart ( userId , nutriantId , food ,timeSlot )" +

                "VALUES(?,?,?,?)";

        try{

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1,foodChart.getUserId());

            preparedStatement.setLong(2,foodChart.getNutriantId());

            preparedStatement.setString(3,foodChart.getFood());

            preparedStatement.setString(4, LoginController.timeSlot);

            preparedStatement.execute();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***22 Get Food()***

public ResultSet getFood(String slot) throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  FoodChart WHERE timeSlot=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setString(1,slot);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***23 Get User Food()***

public ResultSet getUserFood(long userId) throws SQLException, ClassNotFoundException {

        String query = "SELECT \* FROM  FoodChart WHERE userId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1,userId);

        ResultSet resultSet = preparedStatement.executeQuery();

        return resultSet;

    }

***24 Get User Profile()***

public ResultSet getUserProfile(long id) {

        ResultSet resultSet = null;

            String query = "SELECT \* FROM  UserT WHERE "

                    +  "userId=?";

            try {

                PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

                preparedStatement.setLong(1, id);

                resultSet = preparedStatement.executeQuery();

            } catch (SQLException e) {

                e.printStackTrace();

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

            }

        return resultSet;

    }

***25 Update User()***

public void updateUser(long id, String fnm , String lnm , String unm , String password ,long age, long fee,long weight,String muscle,String time) throws SQLException, ClassNotFoundException {

        String query = "UPDATE UserT SET userId=?" +

                " ,firstName=?, lastName=?, userName=? , " +

                "userPassword=?, age=? , weight=? , fee=? , muscleType=? , timeSlot=?" +

                " WHERE userId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            preparedStatement.setString(2, fnm);

            preparedStatement.setString(3, lnm);

            preparedStatement.setString(4, unm);

            preparedStatement.setString(5, password);

            preparedStatement.setLong(6, age);

            preparedStatement.setLong(7, weight);

            preparedStatement.setLong(8, fee);

            preparedStatement.setString(9, muscle);

            preparedStatement.setString(10,time);

            preparedStatement.setLong(11, id);

            preparedStatement.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***26 Delete Food()***

public void deleteFood(long id) throws SQLException, ClassNotFoundException {

        String query = "DELETE FROM FoodChart  WHERE userId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setLong(1, id);

        preparedStatement.execute();

        preparedStatement.close();

    }

***27 Update Food()***

 public void updateFood(long id, String food) throws SQLException, ClassNotFoundException {

        String query = "UPDATE FoodChart SET food=? WHERE userId=?";

        PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

        preparedStatement.setString(1, food);

        preparedStatement.setLong(2, id);

        preparedStatement.executeUpdate();

        preparedStatement.close();

    }

***28 Get Trainer()***

 public ResultSet getTrainer(long id) {

        ResultSet resultSet = null;

        String query = "SELECT \* FROM  TrainerT WHERE "

                +  "trainerId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            resultSet = preparedStatement.executeQuery();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        return resultSet;

    }

***29 Update Employee()***

public void updateEmployee(long id,String fnm,String lnm,String nm,String password,String qual,String exp,String tms) throws SQLException, ClassNotFoundException {

        String query = "UPDATE TrainerT SET trainerId=?" +

                " ,firstName=?, lastName=?, userName=? , " +

                "userPassword=?, qualification=? , experience=? , timeSlot=?" +

                " WHERE trainerId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            preparedStatement.setString(2, fnm);

            preparedStatement.setString(3, lnm);

            preparedStatement.setString(4, nm);

            preparedStatement.setString(5, password);

            preparedStatement.setString(6,qual);

            preparedStatement.setString(7,exp);

            preparedStatement.setString(8,tms);

            preparedStatement.setLong(9, id);

            preparedStatement.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***30 Get Nutritionist()***

public ResultSet getNutriant(long id) {

        ResultSet resultSet = null;

        String query = "SELECT \* FROM  NutriantT WHERE "

                +  "nutriantId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            resultSet = preparedStatement.executeQuery();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        return resultSet;

    }

***31 Update Employee()***

public void updateEmployee2(long id,String fnm,String lnm,String nm,String password,String qual,String exp,String tms) throws SQLException, ClassNotFoundException {

        String query = "UPDATE NutriantT SET nutriantId=?" +

                " ,firstName=?, lastName=?, userName=? , " +

                "userPassword=?, qualification=? , experience=? , timeSlot=?" +

                " WHERE nutriantId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            preparedStatement.setString(2, fnm);

            preparedStatement.setString(3, lnm);

            preparedStatement.setString(4, nm);

            preparedStatement.setString(5, password);

            preparedStatement.setString(6,qual);

            preparedStatement.setString(7,exp);

            preparedStatement.setString(8,tms);

            preparedStatement.setLong(9, id);

            preparedStatement.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***32 UsersAllExercises.setTable()***

public void updateEmployee2(long id,String fnm,String lnm,String nm,String password,String qual,String exp,String tms) throws SQLException, ClassNotFoundException {

        String query = "UPDATE NutriantT SET nutriantId=?" +

                " ,firstName=?, lastName=?, userName=? , " +

                "userPassword=?, qualification=? , experience=? , timeSlot=?" +

                " WHERE nutriantId=?";

        try {

            PreparedStatement preparedStatement = getDbConnection().prepareStatement(query);

            preparedStatement.setLong(1, id);

            preparedStatement.setString(2, fnm);

            preparedStatement.setString(3, lnm);

            preparedStatement.setString(4, nm);

            preparedStatement.setString(5, password);

            preparedStatement.setString(6,qual);

            preparedStatement.setString(7,exp);

            preparedStatement.setString(8,tms);

            preparedStatement.setLong(9, id);

            preparedStatement.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

***33 UserHome.initialize()***

@FXML

    void initialize(){

        profile.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getUserProfile(LoginController.userId);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    user = new User();

                    try {

                        user.setIdNo(resultSet.getLong("userId"));

                        user.setFirstName(resultSet.getString("firstName"));

                        user.setLastName(resultSet.getString("lastName"));

                        user.setName(resultSet.getString("userName"));

                        user.setPassword(resultSet.getString("userPassword"));

                        user.setUserAge(resultSet.getLong("age"));

                        user.setUserWeight(resultSet.getLong("weight"));

                        fee = resultSet.getLong("fee");

                        user.setMuscle(resultSet.getString("muscleType"));

                        user.setTimeSlot(resultSet.getString("timeSlot"));

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                profile.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/userProfile.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UserProfile userProfile = loader.getController();

                userProfile.setId(user.getIdNo());

                userProfile.setfName(user.getFirstName());

                userProfile.setLname(user.getLastName());

                userProfile.setuName(user.getName());

                userProfile.setPassword(user.getPassword());

                userProfile.setAge(user.getUserAge());

                userProfile.setWeight(user.getUserWeight());

                userProfile.setMuscleType(user.getMuscle());

                userProfile.setTimeSlot(user.getTimeSlot());

                userProfile.update.setOnAction(event1 -> {

                    try {

                        dbQueries.updateUser(userProfile.getId(),userProfile.getfName(),userProfile.getLname()

                                ,userProfile.getuName(),userProfile.getPassword(),userProfile.getAge(),fee,

                                userProfile.getWeight(),userProfile.getMuscleType(),userProfile.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    userProfile.update.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

        logOut.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                Parent parent = null;

                try {

                    parent = FXMLLoader.load(getClass().getResource("/sample/view/Login.fxml"));

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Scene scene = new Scene(parent);

                Stage stage = (Stage) logOut.getScene().getWindow();

                stage.setScene(scene);

                stage.show();

            }

        });

    }

***34 Check Exercise()***

 void checkExercise(ActionEvent event) throws IOException {

        exc = FXMLLoader.load(getClass().getResource("/sample/view/allExercisesUser.fxml"));

        setNode(exc);

    }

***35 Check Food()***

   @FXML

    void checkFood(ActionEvent event) throws IOException {

        exc = FXMLLoader.load(getClass().getResource("/sample/view/userAllFood.fxml"));

        setNode(exc);

    }

***36 SetNode()***

public void setNode(Node node)

    {

        inner\_pane.getChildren().clear();

        inner\_pane.getChildren().add((Node)node);

        AnchorPane.setTopAnchor(node,4.0);

        AnchorPane.setBottomAnchor(node,0.0);

        AnchorPane.setRightAnchor(node,0.0);

        AnchorPane.setLeftAnchor(node,0.0);

        FadeTransition ft =new FadeTransition();

        ft.setDuration(Duration.millis(1500));

        ft.setNode(node);

        ft.setFromValue(0.1);

        ft.setToValue(1);

        ft.setAutoReverse(false);

        ft.play();

    }

***37 UserAllFood.SetTable()***

 private void setTable() throws SQLException {

        dbQueries =new DBQueries();

        list = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getUserFood(LoginController.userId);

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            FoodChart foodChart = new FoodChart();

                foodChart.setUserId(resultSet.getLong("userId"));

                foodChart.setNutriantId(resultSet.getLong("nutriantId"));

                foodChart.setFood(resultSet.getString("food"));

                list.addAll(foodChart);

        }

        food.setCellValueFactory(new PropertyValueFactory<FoodChart,String>("food"));

        table\_view.setItems(list);

    }

***38 TrainerHome.initialize()***

@FXML

    void initialize(){

        logout.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                Parent parent = null;

                try {

                    parent = FXMLLoader.load(getClass().getResource("/sample/view/Login.fxml"));

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Scene scene = new Scene(parent);

                Stage stage = (Stage) logout.getScene().getWindow();

                stage.setScene(scene);

                stage.show();

            }

        });

        update.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getTrainer(LoginController.trainerId);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    employee = new Employee();

                    try {

                        employee.setIdNo(resultSet.getLong("trainerId"));

                        employee.setFirstName(resultSet.getString("firstName"));

                        employee.setLastName(resultSet.getString("lastName"));

                        employee.setQualification(resultSet.getString("qualification"));

                        employee.setExperience(resultSet.getString("experience"));

                        employee.setSlotTime(resultSet.getString("timeSlot"));

                        employee.setPassword(resultSet.getString("userPassword"));

                        employee.setName(resultSet.getString("userName"));

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                update.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateTrainer.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateTrainer updateTrainer = loader.getController();

                updateTrainer.userName.setVisible(true);

                updateTrainer.password.setVisible(true);

                updateTrainer.setIdCard(employee.getIdNo());

                updateTrainer.setFirstname(employee.getFirstName());

                updateTrainer.setLastName(employee.getLastName());

                updateTrainer.setExperience(employee.getExperience());

                updateTrainer.setQualfication(employee.getQualification());

                updateTrainer.setTimeSlot(employee.getSlotTime());

                updateTrainer.setUserName(employee.getName());

                updateTrainer.setPassword(employee.getPassword());

                updateTrainer.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateEmployee(updateTrainer.getIdCard(),

                                updateTrainer.getFirstname(),

                                updateTrainer.getLastName(),

                                updateTrainer.getUserName(),updateTrainer.getPassword(),

                                updateTrainer.getQualfication(),updateTrainer.getExperience()

                                ,updateTrainer.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    updateTrainer.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

    }

***39 TrainerHome.SetNode()***

public void setNode(Node node)

    {

        main\_Pane.getChildren().clear();

        main\_Pane.getChildren().add((Node)node);

        AnchorPane.setTopAnchor(node,4.0);

        AnchorPane.setBottomAnchor(node,0.0);

        AnchorPane.setRightAnchor(node,0.0);

        AnchorPane.setLeftAnchor(node,0.0);

        FadeTransition ft =new FadeTransition();

        ft.setDuration(Duration.millis(1500));

        ft.setNode(node);

        ft.setFromValue(0.1);

        ft.setToValue(1);

        ft.setAutoReverse(false);

        ft.play();

    }

***40 Exercise.initial;ize()***

 @FXML

    void initialize() throws SQLException, ClassNotFoundException {

        exercises = FXCollections.observableArrayList();

        exercises = ExerciseChartController.exercises;

        TableColumn excId = new TableColumn("ExerciseId");

        TableColumn userId = new TableColumn("UserId");

        TableColumn muscleType = new TableColumn("Muscle Type");

        TableColumn exerciseType = new TableColumn("Exercise Type");

        excId.setPrefWidth(120);

        userId.setPrefWidth(120);

        muscleType.setPrefWidth(120);

        exerciseType.setPrefWidth(120);

        table\_View.getColumns().addAll(excId,userId,muscleType,exerciseType);

        setTable(excId,userId,muscleType,exerciseType);

        deleteButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long id = table\_View.getSelectionModel().getSelectedItem().getExId();

                System.out.println(id);

                try {

                    dbQueries.deleteExercise(id);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                table\_View.refresh();

            }

        });

        updateButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long id = table\_View.getSelectionModel().getSelectedItem().getExId();

                String muscle = table\_View.getSelectionModel().getSelectedItem().getMuscleType();

                String exs = table\_View.getSelectionModel().getSelectedItem().getExerciseType();

                System.out.println(id);

                updateButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateExercise.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateExercise updateExercise = loader.getController();

                updateExercise.setId(id);

                updateExercise.setMuscle(muscle);

                updateExercise.setExerciseType(exs);

                updateExercise.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateExercise(id , updateExercise.getExerciseType());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    try {

                        setTable(excId,userId,muscleType,exerciseType);

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    updateExercise.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

    }

***41 Exercise.SetTable()***

public void setTable(TableColumn excid , TableColumn userid, TableColumn muscleType,TableColumn exType) throws SQLException, ClassNotFoundException {

        dbQueries = new DBQueries();

        exercises = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        resultSet = dbQueries.getExercises(ExerciseChartController.userId);

        while (resultSet.next()){

            Exercise exercise = new Exercise();

            try {

                exercise.setExId(resultSet.getInt("excId"));

                exercise.setUserId(resultSet.getInt("userId"));

                exercise.setExerciseType(resultSet.getString("exerciseType"));

                exercise.setMuscleType(resultSet.getString("muscleType"));

            } catch (SQLException e) {

                e.printStackTrace();

            }

            exercises.addAll(exercise);

        }

        excid.setCellValueFactory(new PropertyValueFactory<Exercise,Integer>("exId"));

        userid.setCellValueFactory(new PropertyValueFactory<Exercise,Integer>("userId"));

        muscleType.setCellValueFactory(new PropertyValueFactory<Exercise,String>("muscleType"));

        exType.setCellValueFactory(new PropertyValueFactory<Exercise,String>("exerciseType"));

        table\_View.setItems(exercises);

    }

***42 ExerciseChartControler.initialize()***

 @FXML

    void initialize() {

        TableColumn id = new TableColumn("UserId");

        TableColumn trainerId = new TableColumn("TrainerId");

        TableColumn muscleType = new TableColumn("MuscleType");

        id.setPrefWidth(150);

        trainerId.setPrefWidth(150);

        muscleType.setPrefWidth(150);

        table\_View.getColumns().addAll(id,trainerId,muscleType);

        try {

            setTable(id,trainerId,muscleType);

        } catch (SQLException e) {

            e.printStackTrace();

        }

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                userId = table\_View.getSelectionModel().getSelectedItem().getUserId();

                System.out.println(userId);

            }

        });

    }

***43 ExerciseChartCOntroller.SetTable()***

  public void setTable(TableColumn userId,TableColumn trainerId,TableColumn muscleType) throws SQLException {

        dbQueries = new DBQueries();

        list = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getExerciseChart();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            ExerciseChart chart = new ExerciseChart();

            if(LoginController.timeSlot.equals(resultSet.getString("timeSlot"))) {

                chart.setUserId(resultSet.getInt("userId"));

                chart.setTrainerId(resultSet.getInt("trainerId"));

                chart.setMuscleType(resultSet.getString("muscleType"));

                list.addAll(chart);

            }

        }

        userId.setCellValueFactory(new PropertyValueFactory<User,Integer>("userId"));

        trainerId.setCellValueFactory(new PropertyValueFactory<User,Integer>("trainerId"));

        muscleType.setCellValueFactory(new PropertyValueFactory<User,String>("muscleType"));

        table\_View.setItems(list);

    }

***44 AddExerciseForUser.initialize()***

void initialize(){

        userId.setText(String.valueOf(UserController.userId));

        trainerId.setText(String.valueOf(LoginController.trainerId));

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                try {

                    addUser();

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

            }

        });

    }

***45 AddExerciseForUser.addUser()***

rivate void addUser() throws SQLException, ClassNotFoundException {

        if(core.isSelected()){

            checkBox.add(core.getText());}

        if(!core.isSelected()){

            if(checkBox.contains(core.getText())){

                checkBox.remove(core.getText());}

        }

        if(back.isSelected()){

            checkBox.add(back.getText());}

        if(!back.isSelected()){

            if(checkBox.contains(back.getText())){

                checkBox.remove(back.getText());}

        }

        if(upper\_limbe.isSelected()){

            checkBox.add(upper\_limbe.getText());}

        if(!upper\_limbe.isSelected()){

            if(checkBox.contains(upper\_limbe.getText())){

                checkBox.remove(upper\_limbe.getText());}

        }

        if(lower\_limbs.isSelected()){

            checkBox.add(lower\_limbs.getText());}

        if(!lower\_limbs.isSelected()){

            if(checkBox.contains(lower\_limbs.getText())) {

                checkBox.remove(lower\_limbs.getText());

            }

        }

        dbQueries = new DBQueries();

        ExerciseChart exerciseChart = new ExerciseChart(UserController.userId,LoginController.trainerId,checkBox.toString());

        for (int i=0 ;i<checkBox.size();i++){

            System.out.println(checkBox.get(i));

        }

        dbQueries.addExercise(exerciseChart , checkBox);

        addButton.getScene().getWindow().hide();

    }

***46 NutritionistHome.initialize()***

@FXML

    void initialize(){

        logout.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                Parent parent = null;

                try {

                    parent = FXMLLoader.load(getClass().getResource("/sample/view/Login.fxml"));

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Scene scene = new Scene(parent);

                Stage stage = (Stage) logout.getScene().getWindow();

                stage.setScene(scene);

                stage.show();

            }

        });

        update.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getNutriant(LoginController.nutriantId);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    employee = new Employee();

                    try {

                        employee.setIdNo(resultSet.getLong("nutriantId"));

                        employee.setFirstName(resultSet.getString("firstName"));

                        employee.setLastName(resultSet.getString("lastName"));

                        employee.setQualification(resultSet.getString("qualification"));

                        employee.setExperience(resultSet.getString("experience"));

                        employee.setSlotTime(resultSet.getString("timeSlot"));

                        employee.setPassword(resultSet.getString("userPassword"));

                        employee.setName(resultSet.getString("userName"));

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                update.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateTrainer.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateTrainer updateTrainer = loader.getController();

                updateTrainer.userName.setVisible(true);

                updateTrainer.password.setVisible(true);

                updateTrainer.setIdCard(employee.getIdNo());

                updateTrainer.setFirstname(employee.getFirstName());

                updateTrainer.setLastName(employee.getLastName());

                updateTrainer.setExperience(employee.getExperience());

                updateTrainer.setQualfication(employee.getQualification());

                updateTrainer.setTimeSlot(employee.getSlotTime());

                updateTrainer.setUserName(employee.getName());

                updateTrainer.setPassword(employee.getPassword());

                updateTrainer.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateEmployee2(updateTrainer.getIdCard(),

                                updateTrainer.getFirstname(),

                                updateTrainer.getLastName(),

                                updateTrainer.getUserName(),updateTrainer.getPassword(),

                                updateTrainer.getQualfication(),updateTrainer.getExperience()

                                ,updateTrainer.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    updateTrainer.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

    }

***47 NutritionistHome.SetNode()***

 public void setNode(Node node)

    {

        main\_pane.getChildren().clear();

        main\_pane.getChildren().add((Node)node);

        AnchorPane.setTopAnchor(node,4.0);

        AnchorPane.setBottomAnchor(node,0.0);

        AnchorPane.setRightAnchor(node,0.0);

        AnchorPane.setLeftAnchor(node,0.0);

        FadeTransition ft =new FadeTransition();

        ft.setDuration(Duration.millis(1500));

        ft.setNode(node);

        ft.setFromValue(0.1);

        ft.setToValue(1);

        ft.setAutoReverse(false);

        ft.play();

    }

***48 FoodChartController.initialize()***

 @FXML

    void initialize() throws SQLException {

        setTable();

        delete.setOnAction(new EventHandler<ActionEvent>(){

            @Override

            public void handle(ActionEvent event) {

                long id  = table\_view.getSelectionModel().getSelectedItem().getUserId();

                table\_view.getItems().remove(table\_view.getSelectionModel().getSelectedItem());

                try {

                    dbQueries.deleteFood(id);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                table\_view.refresh();

            }

        });

        update.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long id  = table\_view.getSelectionModel().getSelectedItem().getUserId();

                String food = table\_view.getSelectionModel().getSelectedItem().getFood();

                update.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateFood.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateFood updateFood = loader.getController();

                updateFood.setUserId(id);

                updateFood.setFood(food);

                updateFood.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateFood(id , updateFood.getFood());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    try {

                        setTable();

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    updateFood.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

    }

***49 FoodChartController.SetTable()***

private void setTable() throws SQLException {

        dbQueries =new DBQueries();

        list = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getFood(LoginController.timeSlot);

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

           FoodChart foodChart = new FoodChart();

            if (LoginController.timeSlot.equals(resultSet.getString("timeSlot"))) {

               foodChart.setUserId(resultSet.getLong("userId"));

               foodChart.setNutriantId(resultSet.getLong("nutriantId"));

               foodChart.setFood(resultSet.getString("food"));

                list.addAll(foodChart);

            }

        }

        userId.setCellValueFactory(new PropertyValueFactory<FoodChart,Long>("userId"));

        nutriantId.setCellValueFactory(new PropertyValueFactory<FoodChart,Long>("nutriantId"));

        food.setCellValueFactory(new PropertyValueFactory<FoodChart,String>("food"));

        table\_view.setItems(list);

    }

***50 ControlUser.initialize()***

 @FXML

    void initialize() throws SQLException {

        setTable();

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                User selectedItem = table\_view.getSelectionModel().getSelectedItem();

                 //table\_view.getSelectionModel().getSelectedItem().getParent().getChildren().remove(selectedItem);

                userId = selectedItem.getIdNo();

                System.out.println(userId);

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/addFood.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                stage.showAndWait();

            }

        });

    }

***51 ControlUser.SetTable()***

private void setTable() throws SQLException {

        dbQueries =new DBQueries();

        list = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getAllUsers();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            User user = new User();

            if (LoginController.timeSlot.equals(resultSet.getString("timeSlot"))) {

                user.setIdNo(resultSet.getLong("userId"));

                user.setFirstName(resultSet.getString("firstName"));

                user.setLastName(resultSet.getString("lastName"));

                user.setUserAge(resultSet.getLong("age"));

                user.setUserWeight(resultSet.getLong("weight"));

                user.setFee(resultSet.getLong("fee"));

                user.setMuscle(resultSet.getString("muscleType"));

                user.setTimeSlot(resultSet.getString("timeSlot"));

                list.addAll(user);

            }

        }

        id.setCellValueFactory(new PropertyValueFactory<User,Long>("idNo"));

        firstName.setCellValueFactory(new PropertyValueFactory<User,String>("firstName"));

        lastName.setCellValueFactory(new PropertyValueFactory<User,String>("LastName"));

        age.setCellValueFactory(new PropertyValueFactory<User,Long>("userAge"));

        weight.setCellValueFactory(new PropertyValueFactory<User,Long>("userWeight"));

        table\_view.setItems(list);

    }

***52 AddFood.initializec()***

void initialize(){

        userId.setText(String.valueOf(controllUser.userId));

        nutriantId.setText(String.valueOf(LoginController.nutriantId));

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addFood();

            }

        });

    }

***53 AddFood.addFood()***

private void addFood() {

        dbQueries = new DBQueries();

        FoodChart chart = new FoodChart(controllUser.userId,LoginController.nutriantId,textArea.getText());

        dbQueries.addFood(chart);

        addButton.getScene().getWindow().hide();

    }

***54 user()***

public User(String firstName, String lastName, String name, String password, String muscle, long idNo, long userAge, long userWeight, long fee,String timeSlot) {

        this.firstName = firstName;

        LastName = lastName;

        Name = name;

        Password = password;

        this.muscle = muscle;

        this.idNo = idNo;

        this.userAge = userAge;

        this.userWeight = userWeight;

        this.fee = fee;

        this.timeSlot = timeSlot;

    }

***55 FoodChart()***

ublic FoodChart(long userId, long nutriantId, String food) {

        this.userId = userId;

        this.nutriantId = nutriantId;

        this.food = food;

    }

***56 ExerciseChart()***

  public ExerciseChart(long userId, long trainerId, String muscleType) {

        this.userId = userId;

        this.trainerId = trainerId;

        this.muscleType = muscleType;

    }

***57 Exercise()***

public Exercise(int exId, int userId, String exerciseType, String muscleType) {

        this.exId = exId;

        this.userId = userId;

        this.exerciseType = exerciseType;

        this.muscleType = muscleType;

    }

***58 Equipment()***

 public Equipment(int id, String model, String name,  long cost, Date dateOfPurchase, Date lastMaintenance) {

        this.id = id;

        this.model = model;

        this.name = name;

        this.image = image;

        this.cost = cost;

        this.dateOfPurchase = dateOfPurchase;

        this.lastMaintenance = lastMaintenance;

    }

***59 Employee()***

public Employee(String name, String password, String experience, String qualification, String firstName, String lastName,long idNo) {

        this.name = name;

        this.password = password;

        this.experience = experience;

        this.qualification = qualification;

        this.firstName = firstName;

        this.lastName = lastName;

        this.idNo = idNo;

    }

***60 UserSignUp.initialize()***

 @FXML

    void initialize()

    {

        signUp.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                try {

                    signUp();

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

            }

        });

        ToggleGroup tg = new ToggleGroup();

        dayTime.setToggleGroup(tg);

        nightTime.setToggleGroup(tg);

        // add a change listener

        tg.selectedToggleProperty().addListener(new ChangeListener<Toggle>()

        {

            public void changed(ObservableValue<? extends Toggle> ob,

                                Toggle o, Toggle n)

            {

                RadioButton rb = (RadioButton)tg.getSelectedToggle();

                if (rb != null) {

                    s = rb.getText();

                    System.out.println(s);

                }

            }

        });

    }

***61 UserSignUp.signup()***

  private void signUp() throws SQLException, ClassNotFoundException {

        dbQueries = new DBQueries();

        fnm = firstName.getText();

        lnm = lastname.getText();

        nm =  name.getText();

        pas = password.getText();

        userAge = Long.parseLong(age.getText());

        userWeight = Long.parseLong(weight.getText());

        idNo = Long.parseLong(idCard.getText());

        if(ch1.isSelected()){

            checkBox.add(ch1.getText());}

        if(!ch1.isSelected()){

            if(checkBox.contains(ch1.getText())){

                checkBox.remove(ch1.getText());}

        }

        if(ch2.isSelected()){

            checkBox.add(ch2.getText());}

        if(!ch2.isSelected()){

            if(checkBox.contains(ch2.getText())){

                checkBox.remove(ch2.getText());}

        }

        if(ch3.isSelected()){

            checkBox.add(ch3.getText());}

        if(!ch3.isSelected()){

            if(checkBox.contains(ch3.getText())){

                checkBox.remove(ch3.getText());}

        }

        if(ch4.isSelected()){

            checkBox.add(ch4.getText());}

        if(!ch4.isSelected()){

            if(checkBox.contains(ch4.getText())) {

                checkBox.remove(ch4.getText());

            }

        }

        fee = checkBox.size() \* 1000;

        User user = new User(fnm,lnm,nm,pas,checkBox.toString(),idNo,userAge,userWeight,fee,s);

        dbQueries.signUpUser(user);

        signUp.getScene().getWindow().hide();

    }

***62 UserScreen.initialize()***

@FXML

    void initialize() throws SQLException {

        TableColumn id = new TableColumn("UserId");

        TableColumn firstName = new TableColumn("FirstName");

        TableColumn lastName = new TableColumn("LastName");

        TableColumn age = new TableColumn("Age");

        TableColumn weight = new TableColumn("Weight");

        TableColumn muscle = new TableColumn("Muscle");

        TableColumn fee = new TableColumn("Fee");

        TableColumn timeSlot = new TableColumn("TimeSlot");

        id.setPrefWidth(90);

        firstName.setPrefWidth(90);

        lastName.setPrefWidth(90);

        age.setPrefWidth(90);

        weight.setPrefWidth(90);

        muscle.setPrefWidth(220);

        fee.setPrefWidth(90);

        tableview.getColumns().addAll(id,firstName,lastName,age,weight,muscle,fee,timeSlot);

        setTable(id,firstName,lastName,age,weight,muscle,fee,timeSlot);

        FilteredList<User> users = new FilteredList<>(userList);

        search.textProperty().addListener(new ChangeListener<String>() {

            @Override

            public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

                users.setPredicate(new Predicate<User>() {

                    @Override

                    public boolean test(User user) {

                      if(newValue == null || newValue.isEmpty()){

                          return true;

                      }

                      String lower = newValue.toLowerCase();

                      if(user.getFirstName().toLowerCase().indexOf(lower) != -1){

                          return true;

                      }else if(user.getLastName().toLowerCase().indexOf(lower) != -1){

                          return true;

                      }

                      else

                          return false;

                    }

                });

            }

        });

        tableview.setItems(users);

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/addnewUser.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                stage.showAndWait();

               // tableview.refresh();

                try {

                    setTable(id,firstName,lastName,age,weight,muscle,fee,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        });

        updateButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long idd = tableview.getSelectionModel().getSelectedItem().getIdNo();

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getUserProfile(idd);

                System.out.println(idd);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    user = new User();

                    try {

                        user.setIdNo(resultSet.getLong("userId"));

                        user.setFirstName(resultSet.getString("firstName"));

                        user.setLastName(resultSet.getString("lastName"));

                        user.setName(resultSet.getString("userName"));

                        user.setPassword(resultSet.getString("userPassword"));

                        user.setUserAge(resultSet.getLong("age"));

                        user.setUserWeight(resultSet.getLong("weight"));

                        user.setFee(resultSet.getLong("fee"));

                        user.setMuscle(resultSet.getString("muscleType"));

                        user.setTimeSlot(resultSet.getString("timeSlot"));

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                updateButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/userProfile.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UserProfile userProfile = loader.getController();

                userProfile.feeLabel.setVisible(true);

                userProfile.fee.setVisible(true);

                userProfile.setId(user.getIdNo());

                userProfile.setfName(user.getFirstName());

                userProfile.setLname(user.getLastName());

                userProfile.setuName(user.getName());

                userProfile.setPassword(user.getPassword());

                userProfile.setAge(user.getUserAge());

                userProfile.setWeight(user.getUserWeight());

                userProfile.setMuscleType(user.getMuscle());

                userProfile.setTimeSlot(user.getTimeSlot());

                userProfile.setFee(user.getFee());

                userProfile.update.setOnAction(event1 -> {

                    try {

                        dbQueries.updateUser(userProfile.getId(),userProfile.getfName(),userProfile.getLname()

                                ,userProfile.getuName(),userProfile.getPassword(),userProfile.getAge(),userProfile.getFee(),

                                userProfile.getWeight(),userProfile.getMuscleType(),userProfile.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    try {

                        setTable(id,firstName,lastName,age,weight,muscle,fee,timeSlot);

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    userProfile.update.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

        deleteButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long idd = tableview.getSelectionModel().getSelectedItem().getIdNo();

              //  tableview.getItems().remove(tableview.getSelectionModel().getSelectedItem());

                System.out.println(idd);

                try {

                    dbQueries.deleteUser(idd);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                try {

                    setTable(id,firstName,lastName,age,weight,muscle,fee,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

                // tableview.refresh();

            }

        });

    }

***63 UserScreen.setTable()***

public void setTable(TableColumn id,TableColumn firstName,TableColumn lastName,TableColumn age,TableColumn weight , TableColumn muscle,TableColumn fee,TableColumn timeSlot) throws SQLException{

        dbQueries = new DBQueries();

        userList = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getAllUsers();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            User user = new User();

            user.setIdNo(resultSet.getLong("userId"));

            user.setFirstName(resultSet.getString("firstName"));

            user.setLastName(resultSet.getString("lastName"));

            user.setUserAge(resultSet.getLong("age"));

            user.setUserWeight(resultSet.getLong("weight"));

            user.setFee(resultSet.getLong("fee"));

            user.setMuscle(resultSet.getString("muscleType"));

            user.setTimeSlot(resultSet.getString("timeSlot"));

            userList.addAll(user);

        }

        id.setCellValueFactory(new PropertyValueFactory<User,Long>("idNo"));

        firstName.setCellValueFactory(new PropertyValueFactory<User,String>("firstName"));

        lastName.setCellValueFactory(new PropertyValueFactory<User,String>("LastName"));

        age.setCellValueFactory(new PropertyValueFactory<User,Long>("userAge"));

        weight.setCellValueFactory(new PropertyValueFactory<User,Long>("userWeight"));

        fee.setCellValueFactory(new PropertyValueFactory<User,Long>("fee"));

        muscle.setCellValueFactory(new PropertyValueFactory<User,Long>("muscle"));

        timeSlot.setCellValueFactory(new PropertyValueFactory<User,String>("timeSlot"));

     //   tableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);

        tableview.setItems(userList);

    }

***64 TrainerScreen.initialize()***

@FXML

    void initialize() throws SQLException {

        TableColumn id = new TableColumn("TrainerId");

        TableColumn firstName = new TableColumn("FirstName");

        TableColumn lastName = new TableColumn("LastName");

        TableColumn qualification = new TableColumn("Qualification");

        TableColumn experience = new TableColumn("Experience");

        TableColumn timeSlot = new TableColumn("TimeSlot");

        id.setPrefWidth(140);

        firstName.setPrefWidth(140);

        lastName.setPrefWidth(140);

        qualification.setPrefWidth(140);

        experience.setPrefWidth(140);

        timeSlot.setPrefWidth(140);

        table\_view.getColumns().addAll(id,firstName,lastName,qualification,experience,timeSlot);

        setTable(id,firstName,lastName,qualification,experience,timeSlot);

        FilteredList<Employee> employ = new FilteredList<>(employeesList);

        search.textProperty().addListener(new ChangeListener<String>() {

            @Override

            public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

                employ.setPredicate(new Predicate<Employee>() {

                    @Override

                    public boolean test(Employee employee) {

                        if(newValue == null || newValue.isEmpty()){

                            return true;

                        }

                        String lower = newValue.toLowerCase();

                        if(employee.getFirstName().toLowerCase().indexOf(lower) != -1){

                            return true;

                        }else if(employee.getLastName().toLowerCase().indexOf(lower) != -1){

                            return true;

                        }

                        else

                            return false;

                    }

                });

            }

        });

        table\_view.setItems(employ);

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/addNewTrainer.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                stage.showAndWait();

                try {

                    setTable(id,firstName,lastName,qualification,experience,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        });

        updateButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long idd = table\_view.getSelectionModel().getSelectedItem().getIdNo();

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getTrainer(idd);

                System.out.println(idd);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                   employee = new Employee();

                    try {

                        employee.setIdNo(resultSet.getLong("trainerId"));

                        employee.setFirstName(resultSet.getString("firstName"));

                        employee.setLastName(resultSet.getString("lastName"));

                        employee.setQualification(resultSet.getString("qualification"));

                        employee.setExperience(resultSet.getString("experience"));

                        employee.setSlotTime(resultSet.getString("timeSlot"));

                        password = resultSet.getString("userPassword");

                        name = resultSet.getString("userName");

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                updateButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateTrainer.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateTrainer updateTrainer = loader.getController();

                updateTrainer.setIdCard(employee.getIdNo());

                updateTrainer.setFirstname(employee.getFirstName());

                updateTrainer.setLastName(employee.getLastName());

                updateTrainer.setExperience(employee.getExperience());

                updateTrainer.setQualfication(employee.getQualification());

                updateTrainer.setTimeSlot(employee.getSlotTime());

                updateTrainer.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateEmployee(updateTrainer.getIdCard(),

                                updateTrainer.getFirstname(),

                                updateTrainer.getLastName(),

                                name,password,

                                updateTrainer.getQualfication(),updateTrainer.getExperience()

                        ,updateTrainer.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    try {

                        setTable(id,firstName,lastName,qualification,experience,timeSlot);

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    updateTrainer.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

        deleteButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

               // table\_view.getItems().remove(table\_view.getSelectionModel().getSelectedItem());

                long idd = table\_view.getSelectionModel().getSelectedItem().getIdNo();

                System.out.println(idd);

                try {

                    dbQueries.deleteTrainer(idd);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                try {

                    setTable(id,firstName,lastName,qualification,experience,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

                //table\_view.refresh();

            }

        });

    }

***65 TrainerScreen.setTable()***

public void setTable(TableColumn id,TableColumn firstName,TableColumn lastName,TableColumn qualfication , TableColumn experience,TableColumn timeSlot) throws SQLException{

        dbQueries = new DBQueries();

        employeesList = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getAllTrainers();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            Employee employee = new Employee();

            employee.setIdNo(resultSet.getLong("trainerId"));

            employee.setFirstName(resultSet.getString("firstName"));

            employee.setLastName(resultSet.getString("lastName"));

            employee.setQualification(resultSet.getString("qualification"));

            employee.setExperience(resultSet.getString("experience"));

            employee.setSlotTime(resultSet.getString("timeSlot"));

            employeesList.addAll(employee);

        }

        id.setCellValueFactory(new PropertyValueFactory<Employee,Long>("idNo"));

        firstName.setCellValueFactory(new PropertyValueFactory<EmptyMessageImpl,String>("firstName"));

        lastName.setCellValueFactory(new PropertyValueFactory<Employee,String>("lastName"));

        qualfication.setCellValueFactory(new PropertyValueFactory<Employee,String>("qualification"));

        experience.setCellValueFactory(new PropertyValueFactory<Employee,String>("experience"));

        timeSlot.setCellValueFactory(new PropertyValueFactory<Employee,String>("slotTime"));

        table\_view.setItems(employeesList);

    }

***66 NutritionistScreen.initialize()***

 @FXML

    void initialize() throws SQLException {

        TableColumn id = new TableColumn("TrainerId");

        TableColumn firstName = new TableColumn("FirstName");

        TableColumn lastName = new TableColumn("LastName");

        TableColumn qualification = new TableColumn("Qualification");

        TableColumn experience = new TableColumn("Experience");

        TableColumn timeSlot = new TableColumn("TimeSLot");

        id.setPrefWidth(140);

        firstName.setPrefWidth(140);

        lastName.setPrefWidth(140);

        qualification.setPrefWidth(140);

        experience.setPrefWidth(140);

        timeSlot.setPrefWidth(140);

        table\_view.getColumns().addAll(id,firstName,lastName,qualification,experience,timeSlot);

        setTable(id,firstName,lastName,qualification,experience,timeSlot);

        FilteredList<Employee> employ = new FilteredList<>(employeesList);

        search.textProperty().addListener(new ChangeListener<String>() {

            @Override

            public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

               employ.setPredicate(new Predicate<Employee>() {

                   @Override

                   public boolean test(Employee employee) {

                       if(newValue == null || newValue.isEmpty()){

                           return true;

                       }

                       String lower = newValue.toLowerCase();

                       if(employee.getFirstName().toLowerCase().indexOf(lower) != -1){

                           return true;

                       }else if(employee.getLastName().toLowerCase().indexOf(lower) != -1){

                           return true;

                       }

                       else

                           return false;

                   }

               });

            }

        });

        table\_view.setItems(employ);

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/addNewNutriant.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                stage.showAndWait();

                try {

                    setTable(id,firstName,lastName,qualification,experience,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        });

        updateButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long idd = table\_view.getSelectionModel().getSelectedItem().getIdNo();

                dbQueries = new DBQueries();

                ResultSet resultSet = dbQueries.getNutriant(idd);

                System.out.println(idd);

                while(true){

                    try {

                        if (!resultSet.next())

                            break;

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    employee = new Employee();

                    try {

                        employee.setIdNo(resultSet.getLong("nutriantId"));

                        employee.setFirstName(resultSet.getString("firstName"));

                        employee.setLastName(resultSet.getString("lastName"));

                        employee.setQualification(resultSet.getString("qualification"));

                        employee.setExperience(resultSet.getString("experience"));

                        employee.setSlotTime(resultSet.getString("timeSlot"));

                        password = resultSet.getString("userPassword");

                        name = resultSet.getString("userName");

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                }

                updateButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/updateTrainer.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                UpdateTrainer updateTrainer = loader.getController();

                updateTrainer.setIdCard(employee.getIdNo());

                updateTrainer.setFirstname(employee.getFirstName());

                updateTrainer.setLastName(employee.getLastName());

                updateTrainer.setExperience(employee.getExperience());

                updateTrainer.setQualfication(employee.getQualification());

                updateTrainer.setTimeSlot(employee.getSlotTime());

                updateTrainer.updateButton.setOnAction(event1 -> {

                    try {

                        dbQueries.updateEmployee2(updateTrainer.getIdCard(),

                                updateTrainer.getFirstname(),

                                updateTrainer.getLastName(),

                                name,password,

                                updateTrainer.getQualfication(),updateTrainer.getExperience()

                                ,updateTrainer.getTimeSlot());

                    } catch (SQLException e) {

                        e.printStackTrace();

                    } catch (ClassNotFoundException e) {

                        e.printStackTrace();

                    }

                    try {

                        setTable(id,firstName,lastName,qualification,experience,timeSlot);

                    } catch (SQLException e) {

                        e.printStackTrace();

                    }

                    updateTrainer.updateButton.getScene().getWindow().hide();

                });

                stage.showAndWait();

            }

        });

        deleteButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

              //  table\_view.getItems().remove(table\_view.getSelectionModel().getSelectedItem());

                long idd = table\_view.getSelectionModel().getSelectedItem().getIdNo();

                System.out.println(idd);

                try {

                    dbQueries.deleteNutriant(idd);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                try {

                    setTable(id,firstName,lastName,qualification,experience,timeSlot);

                } catch (SQLException e) {

                    e.printStackTrace();

                }

                //   table\_view.refresh();

            }

        });

    }

***67 NutritionistScreen.setTable()***

public void setTable(TableColumn id,TableColumn firstName,TableColumn lastName,TableColumn qualfication , TableColumn experience,TableColumn timeSlot) throws SQLException{

        dbQueries = new DBQueries();

        employeesList = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getAllNutriants();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            Employee employee = new Employee();

            employee.setIdNo(resultSet.getLong("nutriantId"));

            employee.setFirstName(resultSet.getString("firstName"));

            employee.setLastName(resultSet.getString("lastName"));

            employee.setQualification(resultSet.getString("qualification"));

            employee.setExperience(resultSet.getString("experience"));

            employee.setSlotTime(resultSet.getString("timeSlot"));

            employeesList.addAll(employee);

        }

        id.setCellValueFactory(new PropertyValueFactory<Employee,Long>("idNo"));

        firstName.setCellValueFactory(new PropertyValueFactory<Employee,String>("firstName"));

        lastName.setCellValueFactory(new PropertyValueFactory<Employee,String>("lastName"));

        qualfication.setCellValueFactory(new PropertyValueFactory<Employee,String>("qualification"));

        experience.setCellValueFactory(new PropertyValueFactory<Employee,String>("experience"));

        timeSlot.setCellValueFactory(new PropertyValueFactory<Employee,String>("slotTime"));

        table\_view.setItems(employeesList);

    }

***68 LoginControl.initialize()***

 void initialize(){

        comBox.getSelectionModel().selectFirst();

        login.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

               value = comBox.getValue().toString();

                System.out.println(value);

                System.out.println("login clicked");

                dbQueries = new DBQueries();

                String loginText = name.getText().trim();

                String loginPwd = password.getText().trim();

                Employee employee = new Employee();

                employee.setName(loginText);

                employee.setPassword(loginPwd);

                User user = new User();

                user.setName(loginText);

                user.setPassword(loginPwd);

                ResultSet userRow=null;

                if(value.equals("User")){

                    userRow = dbQueries.getUser(user);

                }else if(value.equals("Trainer")){

                    userRow = dbQueries.getEmployee(employee, value);

                }else if(value.equals("Admin")){

                    userRow = dbQueries.getEmployee(employee, value);

                }else if(value.equals("Nutriant")){

                    userRow = dbQueries.getEmployee(employee, value);

                }

                int counter = 0;

                try {

                    while (userRow.next()) {

                        counter++;

                        String name = userRow.getString("firstName");

                       if(value.equals("Trainer")){

                            trainerId = userRow.getInt("trainerId");

                            timeSlot = userRow.getString("timeSlot");

                        }else if(value.equals("Nutriant")){

                           nutriantId = userRow.getInt("nutriantId");

                           timeSlot = userRow.getString("timeSlot");

                       }else if(value.equals("User")){

                           userId = userRow.getInt("userId");

                       }

                        System.out.println("Welcome! " + name);

                    }

                    if (counter == 1) {

                        showAddItemScreen();

                    }else {

                        Shaker userNameShaker = new Shaker(name);

                        Shaker passwordShaker = new Shaker(password);

                        passwordShaker.shake();

                        userNameShaker.shake();

                        System.out.println("error");

                    }

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        });

        signUp.setOnMouseClicked(new EventHandler<MouseEvent>() {

            @Override

            public void handle(MouseEvent event) {

                System.out.println("signup");

                final Stage dialog = new Stage();

                dialog.initModality(Modality.APPLICATION\_MODAL);

                //dialog.initOwner(primaryStage);

                HBox dialogVbox = new HBox(20);

                Button user = new Button("User");

                Button employee = new Button("Employee");

                dialogVbox.getChildren().add(user);

                dialogVbox.getChildren().add(employee);

                dialog.setTitle("Select one of them");

                dialogVbox.setMargin(user,new Insets(16,16,16,16));

                dialogVbox.setMargin(employee,new Insets(16,16,16,16));

                Scene dialogScene = new Scene(dialogVbox, 350, 100);

                dialog.setScene(dialogScene);

                dialog.show();

                user.setOnAction(new EventHandler<ActionEvent>() {

                    @Override

                    public void handle(ActionEvent event) {

                        //Take users to signup screen

                        user.getScene().getWindow().hide();

                        FXMLLoader loader = new FXMLLoader();

                        loader.setLocation(getClass().getResource("/sample/view/userSignUp.fxml"));

                        try {

                            loader.load();

                        } catch (IOException e) {

                            e.printStackTrace();

                        }

                        System.out.println("user");

                        Parent root = loader.getRoot();

                        Stage stage = new Stage();

                        stage.setScene(new Scene(root));

                        stage.showAndWait();

                        //dialog.close();

                    }

                });

                employee.setOnAction(new EventHandler<ActionEvent>() {

                    @Override

                    public void handle(ActionEvent event) {

                        System.out.println("employee");

                        //Take users to signup screen

                        employee.getScene().getWindow().hide();

                        FXMLLoader loader = new FXMLLoader();

                        loader.setLocation(getClass().getResource("/sample/view/employeeSignUp.fxml"));

                        try {

                            loader.load();

                        } catch (IOException e) {

                            e.printStackTrace();

                        }

                        Parent root = loader.getRoot();

                        Stage stage = new Stage();

                        Scene scene = new Scene(root);

                        stage.setScene(scene);

                        stage.showAndWait();

                        dialog.close();

                    }

                });

            }

        });

    }

***69 LoginControl.showAddItemScreen()***

private void showAddItemScreen() {

        //Take users to signup screen

        login.getScene().getWindow().hide();

        FXMLLoader loader = new FXMLLoader();

        if(value.equals("Trainer"))

        {

            str = "trainerHome";

        }else if(value.equals("Admin")){

            str ="adminhome";

        }else if(value.equals("Nutriant")){

            str = "nutriantHome";

        }else if(value.equals("User")){

            str = "userHome";

        }

        loader.setLocation(getClass().getResource("/sample/view/"+str+".fxml"));

        try {

            loader.load();

        } catch (IOException e) {

            e.printStackTrace();

        }

        Parent root = loader.getRoot();

        Stage stage = new Stage();

        stage.setScene(new Scene(root));

        stage.showAndWait();

    }

***70 EquipmentController.initialize()***

 @FXML

    void initialize() throws SQLException {

        TableColumn image = new TableColumn("Image");

        TableColumn id = new TableColumn("Id");

        TableColumn name = new TableColumn("Name");

        TableColumn model = new TableColumn("Model");

        TableColumn cast = new TableColumn("cast");

        TableColumn dfp = new TableColumn("Date of purchase");

        TableColumn dlm = new TableColumn("Date of last maintenance");

        image.setPrefWidth(160);

        id.setPrefWidth(90);

        name.setPrefWidth(90);

        model.setPrefWidth(90);

        cast.setPrefWidth(90);

        dfp.setPrefWidth(150);

        dlm.setPrefWidth(200);

        table\_view.getColumns().addAll(image,id,name,model,cast,dfp,dlm);

        try {

            setTable(image,id,name,model,cast,dfp,dlm);

        } catch (IOException e) {

            e.printStackTrace();

        }

        FilteredList<Equipment> list = new FilteredList<>(equipmentList);

        search.textProperty().addListener(new ChangeListener<String>() {

            @Override

            public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

                list.setPredicate(new Predicate<Equipment>() {

                    @Override

                    public boolean test(Equipment equipment) {

                        if(newValue == null || newValue.isEmpty()){

                            return true;

                        }

                        String lower = newValue.toLowerCase();

                        if(equipment.getModel().toLowerCase().indexOf(lower) != -1){

                            return true;

                        }

                        else

                            return false;

                    }

                });

            }

        });

        table\_view.setItems(list);

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addButton.getScene().getWindow();

                FXMLLoader loader = new FXMLLoader();

                loader.setLocation(getClass().getResource("/sample/view/addEquipment.fxml"));

                try {

                    loader.load();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Parent root = loader.getRoot();

                Stage stage = new Stage();

                stage.setScene(new Scene(root));

                stage.showAndWait();

                try {

                    setTable(image,id,name,model,cast,dfp,dlm);

                } catch (SQLException | IOException e) {

                    e.printStackTrace();

                }

            }

        });

        deleteButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                long idd = table\_view.getSelectionModel().getSelectedItem().getId();

                try {

                    dbQueries.deleteEqu(idd);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

                try {

                    setTable(image,id,name,model,cast,dfp,dlm);

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                // table\_view.refresh();

            }

        });

    }

***71 EmployeeController.SetTable()***

 public void setTable(TableColumn image,TableColumn id,TableColumn name,TableColumn model, TableColumn cast,TableColumn dfp,TableColumn dlm) throws SQLException, IOException {

        dbQueries = new DBQueries();

        equipmentList = FXCollections.observableArrayList();

        ResultSet resultSet = null;

        try {

            resultSet = dbQueries.getEquipment();

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        while (resultSet.next()) {

            Equipment equipment = new Equipment();

            equipment.setId(resultSet.getInt("equId"));

            equipment.setName(resultSet.getString("equName"));

            equipment.setModel(resultSet.getString("equModel"));

            equipment.setCost(resultSet.getLong("equCast"));

            equipment.setDateOfPurchase(resultSet.getDate("dateOfPurchase"));

            equipment.setLastMaintenance(resultSet.getDate("dateOfMaintance"));

            File file =new File("photo.jpg");

            try {

                fos = new FileOutputStream(file);

            } catch (FileNotFoundException e) {

                e.printStackTrace();

            }

            byte b[];

            Blob blob;

            blob = resultSet.getBlob("image");

            System.out.println(blob);

            b = blob.getBytes(1,(int)blob.length());

            fos.write(b);

            ImageView imageView = new ImageView(new Image("file:photo.jpg",150,200,true,true));

            imageView.setFitHeight(200);

            imageView.setFitWidth(150);

            imageView.setPreserveRatio(true);

            equipment.setImage(imageView);

            equipmentList.addAll(equipment);

        }

        image.setCellValueFactory(new PropertyValueFactory<Equipment, ImageView>("image"));

        id.setCellValueFactory(new PropertyValueFactory<Equipment,Long>("id"));

        name.setCellValueFactory(new PropertyValueFactory<Equipment,String>("name"));

        model.setCellValueFactory(new PropertyValueFactory<Equipment,String>("model"));

        cast.setCellValueFactory(new PropertyValueFactory<Equipment,Long>("cost"));

        dfp.setCellValueFactory(new PropertyValueFactory<Equipment, Date>("dateOfPurchase"));

        dlm.setCellValueFactory(new PropertyValueFactory<Equipment,Date>("lastMaintenance"));

        table\_view.setItems(equipmentList);

    }

***72 EmployeeSignUP.initialize()***

 void initialize(){

        comBox.getSelectionModel().selectFirst();

        signUp.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                signUp();

            }

        });

    }

***73 EmployeeSignUP.SignUp()***

 private void signUp() {

        dbHandler = new DBQueries();

        idNo =Long.parseLong(idCard.getText());

        fnm = firstname.getText().toString();

        lnm = lastName.getText().toString();

        nm = name.getText().toString();

        pas = password.getText().toString();

        exp = experience.getText().toString();

        qua = qualfication.getText().toString();

        Employee employee = new Employee(nm,pas,exp,qua,fnm,lnm,idNo);

        employee.setSlotTime(s);

        dbHandler.signUpEmployee(employee,value);

        signUp.getScene().getWindow().hide();

    }

***74 EmployeeSignUP.ComboSelection()***

 public void comboSelection(ActionEvent actionEvent) {

        value = comBox.getValue().toString();

        System.out.println(value);

        if(value.equals("Trainer") || value.equals("Nutriant") ){

            timeLabel.setVisible(true);

            dayTime.setVisible(true);

            nightTime.setVisible(true);

            selectTimeSlot();

        }else{

            timeLabel.setVisible(false);

            dayTime.setVisible(false);

            nightTime.setVisible(false);

        }

    }

***75 EmployeeSignUp.setTimeSolot()***

 private void selectTimeSlot() {

        ToggleGroup tg = new ToggleGroup();

        dayTime.setToggleGroup(tg);

        nightTime.setToggleGroup(tg);

        // add a change listener

        tg.selectedToggleProperty().addListener(new ChangeListener<Toggle>()

        {

            public void changed(ObservableValue<? extends Toggle> ob,

                                Toggle o, Toggle n)

            {

                RadioButton rb = (RadioButton)tg.getSelectedToggle();

                if (rb != null) {

                    s = rb.getText();

                    System.out.println(s);

                }

            }

        });

    }

***76 AdminHome.initialize()***

 @FXML

    void initialize(){

        logout.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                Parent parent = null;

                try {

                    parent = FXMLLoader.load(getClass().getResource("/sample/view/login.fxml"));

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Scene scene = new Scene(parent,1200,700);

                Stage stage = (Stage) logout.getScene().getWindow();

                stage.setScene(scene);

                stage.show();

            }

        });

    }

***77 AdminHome.setNode()***

public void setNode(Node node)

    {

        main\_pane.getChildren().clear();

        main\_pane.getChildren().add((Node)node);

        FadeTransition ft =new FadeTransition();

        ft.setDuration(Duration.millis(1500));

        ft.setNode(node);

        ft.setFromValue(0.1);

        ft.setToValue(1);

        ft.setAutoReverse(false);

        ft.play();

    }

***78 AddnewUsrr.initialize()***

@FXML

    void initialize()

    {

        addUser.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                try {

                    addUser();

                } catch (SQLException e) {

                    e.printStackTrace();

                } catch (ClassNotFoundException e) {

                    e.printStackTrace();

                }

            }

        });

        ToggleGroup tg = new ToggleGroup();

        dayTime.setToggleGroup(tg);

        nightTime.setToggleGroup(tg);

        // add a change listener

        tg.selectedToggleProperty().addListener(new ChangeListener<Toggle>()

        {

            public void changed(ObservableValue<? extends Toggle> ob,

                                Toggle o, Toggle n)

            {

                RadioButton rb = (RadioButton)tg.getSelectedToggle();

                if (rb != null) {

                    s = rb.getText();

                    System.out.println(s);

                }

            }

        });

    }

***79 AddNewUser.addUser()***

 private void addUser() throws SQLException, ClassNotFoundException {

        dbQueries = new DBQueries();

        fnm = firstName.getText();

        lnm = lastname.getText();

        nm =  name.getText();

        pas = password.getText();

        userAge = Long.parseLong(age.getText());

        userWeight = Long.parseLong(weight.getText());

        idNo = Long.parseLong(idCard.getText());

        if(ch1.isSelected()){

            checkBox.add(ch1.getText());}

        if(!ch1.isSelected()){

            if(checkBox.contains(ch1.getText())){

                checkBox.remove(ch1.getText());}

        }

        if(ch2.isSelected()){

            checkBox.add(ch2.getText());}

        if(!ch2.isSelected()){

            if(checkBox.contains(ch2.getText())){

                checkBox.remove(ch2.getText());}

        }

        if(ch3.isSelected()){

            checkBox.add(ch3.getText());}

        if(!ch3.isSelected()){

            if(checkBox.contains(ch3.getText())){

                checkBox.remove(ch3.getText());}

        }

        if(ch4.isSelected()){

            checkBox.add(ch4.getText());}

        if(!ch4.isSelected()){

            if(checkBox.contains(ch4.getText())) {

                checkBox.remove(ch4.getText());

            }

        }

        fee = checkBox.size() \* 1000;

        User user = new User(fnm,lnm,nm,pas,checkBox.toString(),idNo,userAge,userWeight,fee,s);

        dbQueries.signUpUser(user);

        addUser.getScene().getWindow().hide();

    }

***80 AddNewTrainer.initialize()***

 @FXML

    void initialize() throws SQLException {

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

             addTrainer();

            }

        });

        ToggleGroup tg = new ToggleGroup();

        dayTime.setToggleGroup(tg);

        nightTime.setToggleGroup(tg);

        // add a change listener

        tg.selectedToggleProperty().addListener(new ChangeListener<Toggle>()

        {

            public void changed(ObservableValue<? extends Toggle> ob,

                                Toggle o, Toggle n)

            {

                RadioButton rb = (RadioButton)tg.getSelectedToggle();

                if (rb != null) {

                    s = rb.getText();

                    System.out.println(s);

                }

            }

        });

    }

***81 AddNewTrainer.addTrainer()***

 private void addTrainer(){

        fname = firstname.getText();

        lname = lastName.getText();

        nm = name.getText();

        pass = password.getText();

        exp = experience.getText();

        qua = qualfication.getText();

        id = Long.parseLong(idCard.getText());

        Employee employee = new Employee(nm,pass,exp,qua,fname,lname,id);

        employee.setSlotTime(s);

        dbQueries = new DBQueries();

        dbQueries.signUpEmployee(employee,"Trainer");

        addButton.getScene().getWindow().hide();

    }

***82 AddNewNutritionist.initialize()***

 @FXML

    void initialize() throws SQLException {

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addTrainer();

            }

        });

        ToggleGroup tg = new ToggleGroup();

        dayTime.setToggleGroup(tg);

        nightTime.setToggleGroup(tg);

        // add a change listener

        tg.selectedToggleProperty().addListener(new ChangeListener<Toggle>()

        {

            public void changed(ObservableValue<? extends Toggle> ob,

                                Toggle o, Toggle n)

            {

                RadioButton rb = (RadioButton)tg.getSelectedToggle();

                if (rb != null) {

                    s = rb.getText();

                    System.out.println(s);

                }

            }

        });

    }

***83 AddNewNutritionist.addNutritionist()***

 private void addNutruitiomnist(){

        fname = firstname.getText();

        lname = lastName.getText();

        nm = name.getText();

        pass = password.getText();

        exp = experience.getText();

        qua = qualfication.getText();

        id = Long.parseLong(idCard.getText());

        Employee employee = new Employee(nm,pass,exp,qua,fname,lname,id);

        employee.setSlotTime(s);

        dbQueries = new DBQueries();

        dbQueries.signUpEmployee(employee,"Nutriant");

        addButton.getScene().getWindow().hide();

    }

***84 AddEquipment.initialize()***

@FXML

    void initialize(){

        image.setOnMouseClicked(new EventHandler<MouseEvent>() {

            @Override

            public void handle(MouseEvent event) {

                FileChooser fileChooser = new FileChooser();

                //Set extension filter

                FileChooser.ExtensionFilter extFilterjpg =

                        new FileChooser.ExtensionFilter("jpg files (\*.jpg)", "\*.jpg");

                FileChooser.ExtensionFilter extFilterpng =

                        new FileChooser.ExtensionFilter("png files (\*.png)", "\*.png");

                fileChooser.getExtensionFilters()

                        .addAll( extFilterjpg, extFilterpng);

                //Show open file dialog

                file = fileChooser.showOpenDialog(null);

                if(file != null){

                    img = new Image(file.toURI().toString(),300,200,true,true);

                    image.setImage(img);

                    image.setFitHeight(200);

                    image.setFitWidth(300);

                    image.setPreserveRatio(true);

                    try {

                        fis = new FileInputStream(file);

                    } catch (FileNotFoundException e) {

                        e.printStackTrace();

                    }

                }

            }

        });

        addButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                addEquipment();

            }

        });

    }

***85 AddEquipment.addEquipment()***

 private void addEquipment() {

        eqId = Integer.parseInt(id.getText());

        eqModel = model.getText();

        eqName = name.getText();

        eqCost = Long.parseLong(cost.getText());

        eqDOP = Date.valueOf(dop.getValue());

        eqDLM = Date.valueOf(dlm.getValue());

        Equipment equipment = new Equipment(eqId,eqModel,eqName,eqCost,eqDOP,eqDLM);

        dbQueries = new DBQueries();

        dbQueries.addEquipment(equipment , fis , file);

        addButton.getScene().getWindow().hide();

    }

***FOLLOWING TYPES OF WHITE BOX TESTING WERE PERFORMED***

1. ***SIMPLE LOOP TESTING(LT)***
2. ***METHOD COVERAGE(MT)***
3. ***BRANCH COVERAGE (BT)***
4. ***STATEMENT COVERAGE(ST)***
5. ***ERROR HANDLING TESTING(ET)***

***FOLLOWING TEST RESULTS WERE OBSERVED***

***CONDITION COVERAGE***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TESTCASE NUMBER | SCENARIO | EXPECTED RESULT | PASS/FAIL | COVERAGE |
| TC1 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH NO CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD NOT BE REGISTERED AS MUSCLE TYPE IS ESSENCIAL FIELD | FAIL | 100% |
| TC2 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 91.66% |
| TC3 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST TWO CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST AND SECOND MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 83.33% |
| TC4 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST THREE CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST SECOND AND THIRD MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 75 |
| TC5 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ALL FOUR CHECKBOX SELECTED IN MUSCLE TYPE | USER WILL BE CREATED WITH ALL FOUR MUSCLE TYPES AS SELECTED | PASS | 66.66% |
| TC6 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS NEITHER TRAINER NOR NUTRITIONIST | THE TIMELABLE COMBOBOX SHOULD NOT BE SHOWN | PASS | 50% |
| TC7 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS TRAINER | THE TIMELABLE COMBOBOX SHOULD BE SHOWN | PASS | 50% |
| TC8 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS NUTRITIONIST | THE TIMELABLE COMBOBOX SHOULD BE SHOWN | PASS | 50% |
| TC9 | IN DBQUERIES CLASS FOR ADD EXERCISE METHOD IF ALL THREE MUSCLE TYPES ARE SELECTED | EXERCISES FOR ALL MUSCLE TYPES SHOULD BE ADDED TO SPECIFIC USER | PASS | 100% |
| TC10 | IN DBQUERIES FOR CLASS ADD EXERCISES IF NON OF THE MUSCLE TYPES ARE SELECTED | NO EXERCISE SHOULD BE ADDED IN THE EXERCISE CHART | PASS | 0% SINCE A THE CONDITIONS ARE ONLY PRESENT ON THE VALUES OF MUSCLE TYPES AND IF THERE IS NO MUSCLE TYPE THERE ARE NO CONDITIONS |

***FOR STATEMENT COVERAGE***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TESTCASE NUMBER | SCENARIO | EXPECTED RESULT | PASS/FAIL | COVERAGE |
| TC1 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH NO CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD NOT BE REGISTERED AS MUSCLE TYPE IS ESSENCIAL FIELD | FAIL | 32/32 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 100% |
| TC2 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 30/32 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 93.75% |
| TC3 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST TWO CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST AND SECOND MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 28/32 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 87.5% |
| TC4 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ONLY FIRST THREE CHECKBOX SELECTED IN MUSCLE TYPE | USER SHOULD BE CREATED WITH FIRST SECOND AND THIRD MUSCLE TYPE AS THE ONLY SELECTED MUSCLE TYPE | PASS | 26/32 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 81.25% |
| TC5 | IN ADD USERS FUNCTION OF ADD NEW USER CLASS, ADD A USER WITH ALL FOUR CHECKBOX SELECTED IN MUSCLE TYPE | USER WILL BE CREATED WITH ALL FOUR MUSCLE TYPES AS SELECTED | PASS | 24/32 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 75% |
| TC6 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS NEITHER TRAINER NOR NUTRITIONIST | THE TIMELABLE COMBOBOX SHOULD NOT BE SHOWN | PASS | 8/12 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 66.66% |
| TC7 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS TRAINER | THE TIMELABLE COMBOBOX SHOULD BE SHOWN | PASS | 8/12 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 66.66% |
| TC8 | IN COMBOSELECTION FUNCTION OF EMPLOYEE SIGNUP CLASS  IF EMPLOYEE IS NUTRITIONIST | THE TIMELABLE COMBOBOX SHOULD BE SHOWN | PASS | 9/12 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 75% |
| TC9 | IN DBQUERIES CLASS FOR GET EMPLOYEES FUNCTION FOR AN ADMIN WITH DATA PRESENT IN DATABASE | ADMIN TABLE IS SHOWN | PASS | 13/23 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 56.52% |
| TC10 | IN DBQUERIES CLASS FOR GET EMPLOYEES FUNCTION FOR A TRAINER WITH DATA NOT PRESENT IN DATABASE | ERROR IS SHOWN SAYING PLEASE ENTER VALID CREDENTIALS | PASS | 9/23 LINES ARE TRAVERSED SO STATEMENT COVERAGE IS 39.13% |

***METHOD COVERAGE***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TO LOG IN AS ADMIN | THE ADMIN SHOULD LOGG IN IF THEY ARE ALREADY SIGNED UP | PASS | 12/85 FUNCTIONS ARE CALLED TO PERFORM THIS TASK SO METHOD COVERAGE IS 14.11% |
| TC2 | TO LOG IN AS ADMIN AND ADD A CUSTOMER WITH 4 MUSCLE TYPES | ADMIN SHOULD BE ABLE TO ADD THE USER IF THEY HAVE APPROPRIATE LOGIN INFOR AND USER INFO | PASS | 40/85 FUNCTIONS ARE USED IN PROCESS SO METHOD COVERAGE IS 47.05% |
| TC3 | TO SIGN UP AS CUSTOMER WITH 3 MUSCLE TYPES THAN LOG IN AND CHECK NUTRITIONIST CHART AND THAN LOG OUT | CUSTOMER SHOULD BE ABLE TO DO ALL FUNCTIONS IF THEY HAVE THE SPECIFIED INFORMATION | PASS | 32/85 FUNCTIONS WERE CALLED IN THIS PROCESS SO METHOD COVERAGE FOR THIS PROCESS IS 37.64% |
| TC4 | TO ADD A TRAINER WITH EVENING TIMESLOT | ADMIN SHOULD BE ABLE TO DO THE SPECIFIED FUNCTION IF THEY HAVE APPROPRIATE INFORMATION | PASS | 29/85 FUNCTIONS WERE CALLED IN THIS PROCESS SO METHOD COMPLEXITY IS 33.12% |
| TC5 | TO DELETE A CUSTOMER | ADMIN SHOULD BE ABLE TO DO THIS IF THEY HAVE APPROPRIATE INFO | PASS | 12/85 METHODS WERE CLLED IN THIS PROCESS SO METHOD COVERAGE FOR THIS PROCESS IS 14.11% |
| TC6 | To add a new equipment | ADMIN SHOULD BE ABLE TO DO THIS TASK IFG THEY HAVE THE APPROPRIATE INFO | PASS | 21/85 FUNCTIONS ARE CALLED IN THIS PROCESS SO METHOD COVERAGE IS 24.7% |
| TC7 | TO DELETE AN EQUIPMENT | ADMIN SHOULD BE ABLE TYO DO THE TASK IGF THEY HAVE THE APPROPRIATE INFO | 0PASS | 7/85 METHODS WERE CALLED SO METHOD COVERAGE IS 8.23% |
| TC8 | TO REACH LOGIN WINDOW | ANY USER SHOULDBE ABLE TO DO THIS | PASS | 2/85 METHODS ARE CALLED SO METHOD COMPLEXITY IS 2.33% |
| TC9 | TO SIGNING UP FOR A NEW TRAINER | THE TRAINER SHOULD BE SIGNED UP IF APPROPRIATE DATA IS ENTERED | PASS | SINCE A TOTAL OF 17 FUNCTIONS ARE CALLED IN THE PROCESS SO COVERAGE IS 17/85=20% |
| TC10 | TO SIGNUP AS ADMIN | THE ADMIN SHOULD BE SIGNED UP IF NO PRIOR ADMIN IS PRESENT | FAIL | SINCE THE SYSTEM DOES NOT CHECK FOR ADMIN PRESENCE |