MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE CODE: CSE-220

COURSE NAME: Object Oriented Programming Sessional-II

MIST KickOff

Group No: E_4

<u>GROUP MEMBERS</u>:

Aunindya Prosad Saha 202114014

G.M.Fahim Tazwar 202114025

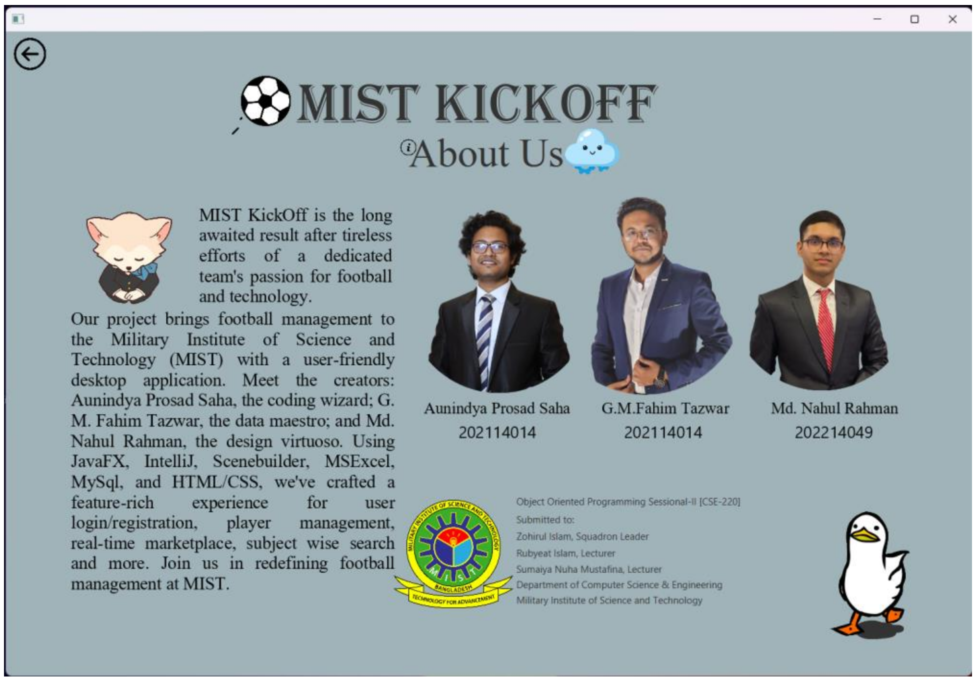
Md. Nahul Rahman 202214049

Contribution Matrix

No.	Name	ID	Contribution
01	Aunindya Prosad Saha	202114014	Project Idea √ Implemented Making GUI interactive, OOP Features, Categorical Search, Database Connection (MySQL), Transaction History Marketplace, Player Line Up (PDF).
02	G.M.Fahim Tazwar	202114025	Implemented Log In, Registration, Forgot Password, Database Connection (MySQL) Data Fetch, Insert, Update, Delete, Data Pictorial Representation, Custom Exception Handling.
03	Md. Nahul Rahman	202214049	Report Writing Implemented Frontend/GUI Design (Java FX) Making GUI responsive, OOP Implementation, Data Pictorial Representation, Database Connection, Player Line Up (PDF).

Introduction

The motive of the project is to make a desktop-based app to give each club in MIST Football Tournament a facility for more easy Player Data Management service, a virtual place to player buy/sell, and also a platform to keep an eye over all the factors of their own club.



MIST KickOff is the long awaited result after tireless efforts of a dedicated team's passion for football and technology.
Our project brings football management to the Military Institute of Science and Technology (MIST) with a user-friendly desktop application. Meet the creators: Aunindya Prosad Saha, the coding wizard; G. M. Fahim Tazwar, the data maestro; and Md. Nahul Rahman, the design virtuoso. Using JavaFX, IntelliJ, Scenebuilder, MSExcel, MySql, and HTML/CSS, we've crafted a feature-rich experience for user login/registration, player management, real-time marketplace, subject wise search and more. Join us in redefining football management at MIST.

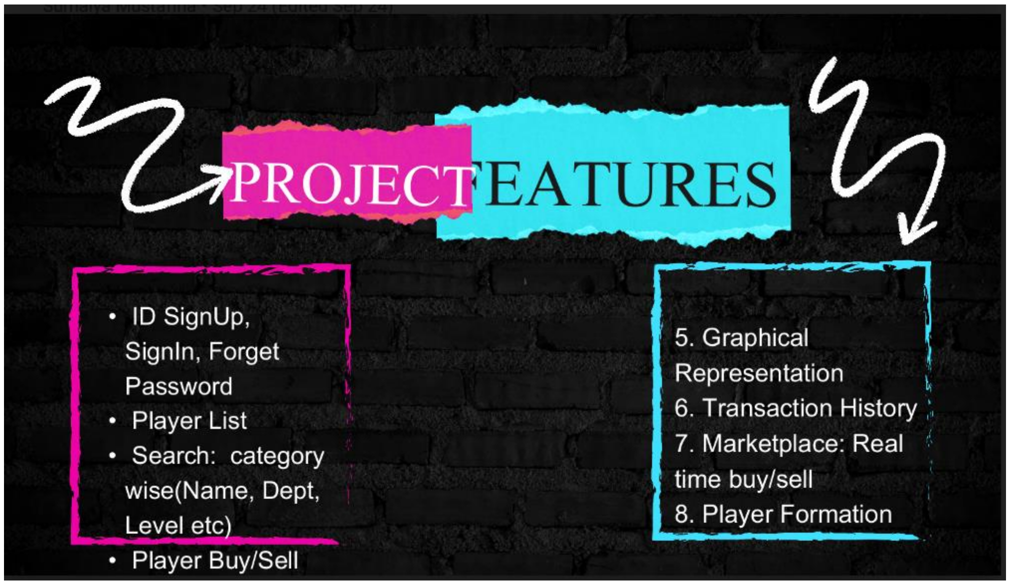
Aunindya Prosad Saha 202114014
G.M.Fahim Tazwar 202114014
Md. Nahul Rahman 202214049

Object Oriented Programming Sessional-II [CSE-220]
Submitted to:
Zohirul Islam, Squadron Leader
Rubyeat Islam, Lecturer
Sumaiya Nuha Mustafina, Lecturer
Department of Computer Science & Engineering
Military Institute of Science and Technology

To fulfill our target in order to create a user-interactive desktop app made in Java, we focused on the following features:

- User Login/Registration,
- Player List Table,
- A Real-time Marketplace to buy/sell,
- Categorical Search,
- Transaction History,
- Pictorial Representation of Data(Pie Chart, Line Chart) ,
- Player LineUp (PDF Format)

Our proposed features were,



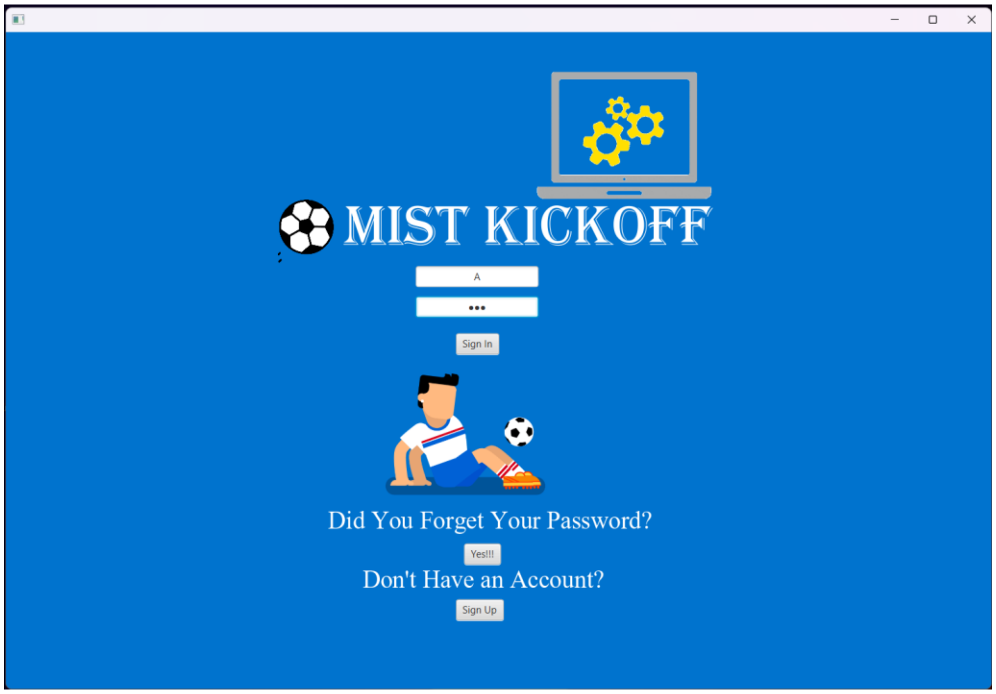
PROJECT FEATURES

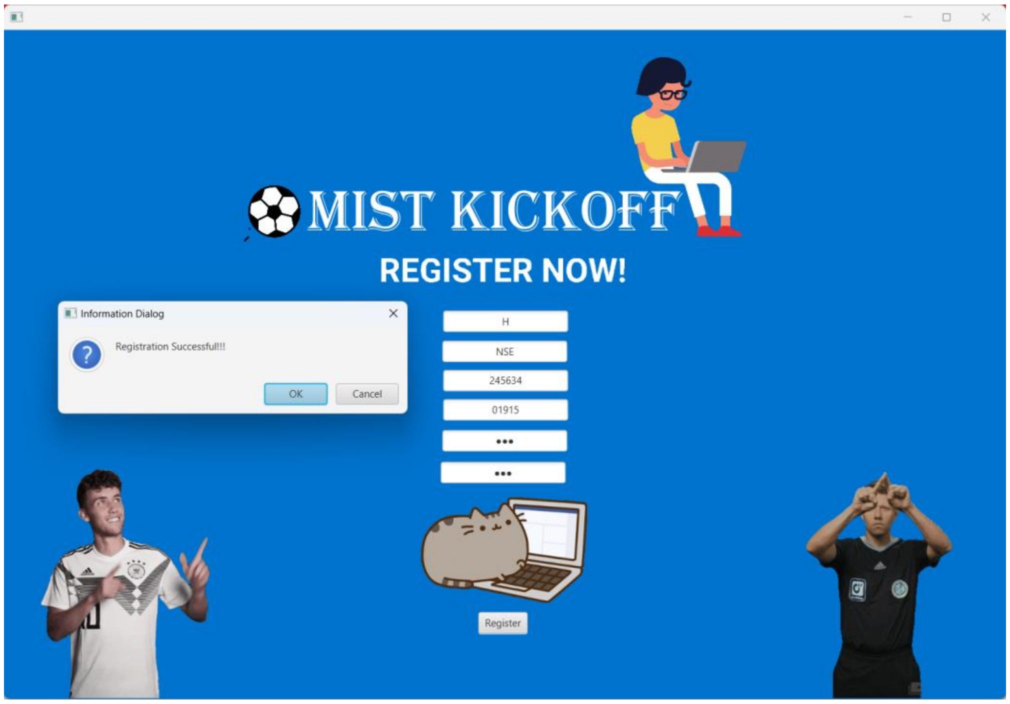
- ID SignUp, SignIn, Forget Password
- Player List
- Search: category wise(Name, Dept, Level etc)
- Player Buy/Sell

5. Graphical Representation
6. Transaction History
7. Marketplace: Real time buy/sell
8. Player Formation

Our Project implemented all the features described below:

1. User Login/Registration



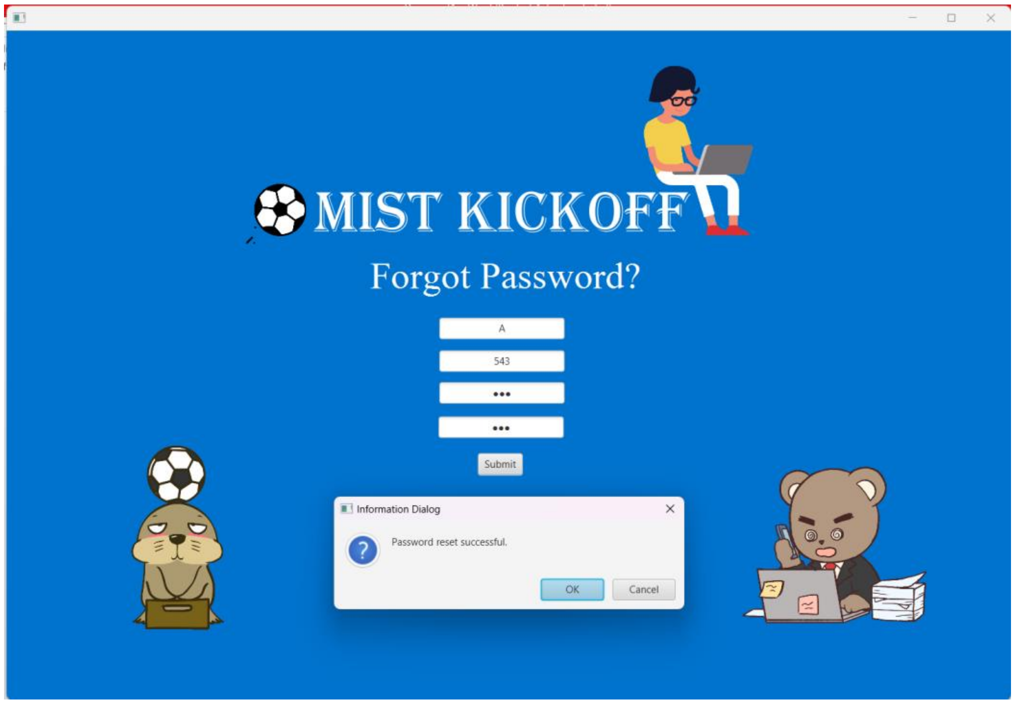


Information Dialog

Registration Successful!!!

OK Cancel

2. Forgot password handling

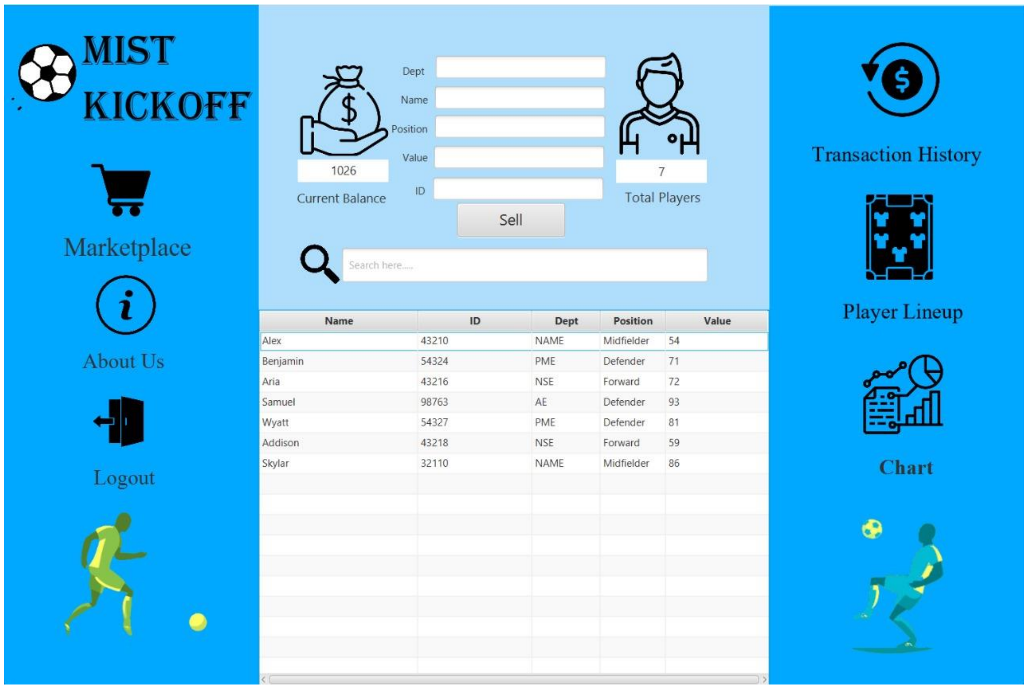


Information Dialog

Password reset successful.

OK Cancel

3. <u>Home interface housing main features and Categorical search:</u>



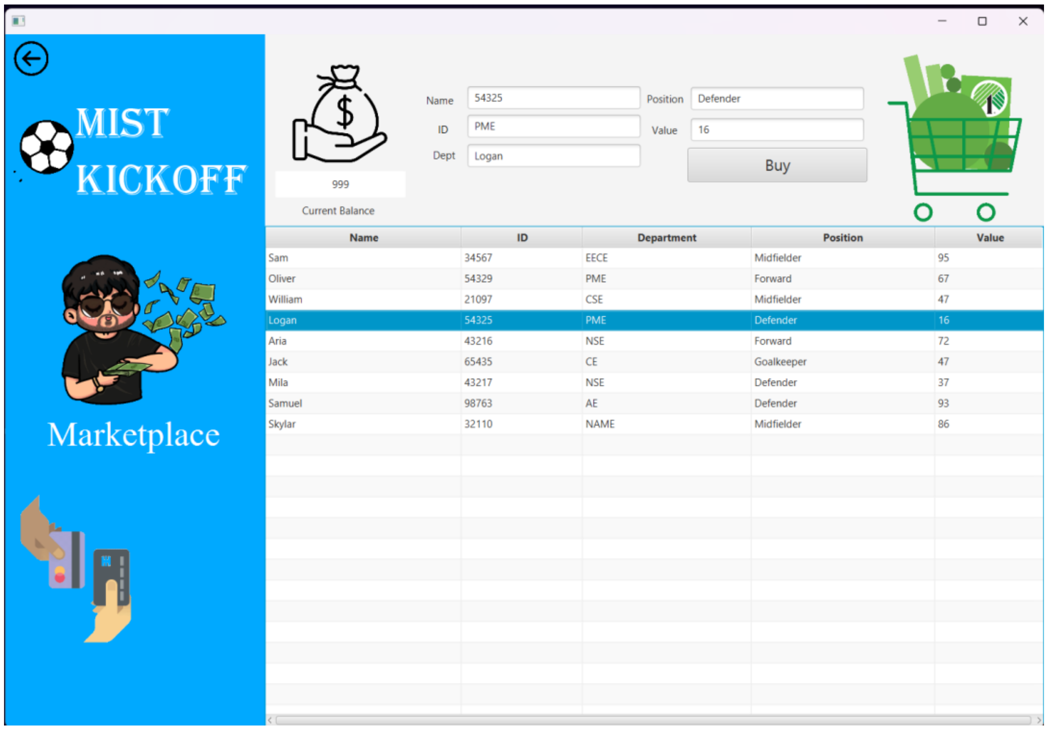
Dept
Name
Position
Value
ID

1026
Current Balance

7
Total Players

Name	ID	Dept	Position	Value
Alex	43210	NAME	Midfielder	54
Benjamin	54324	PME	Defender	71
Aria	43216	NSE	Forward	72
Samuel	98763	AE	Defender	93
Wyatt	54327	PME	Defender	81
Addison	43218	NSE	Forward	59
Skylar	32110	NAME	Midfielder	86

4. <u>Real-time marketplace for player buy and sell</u>

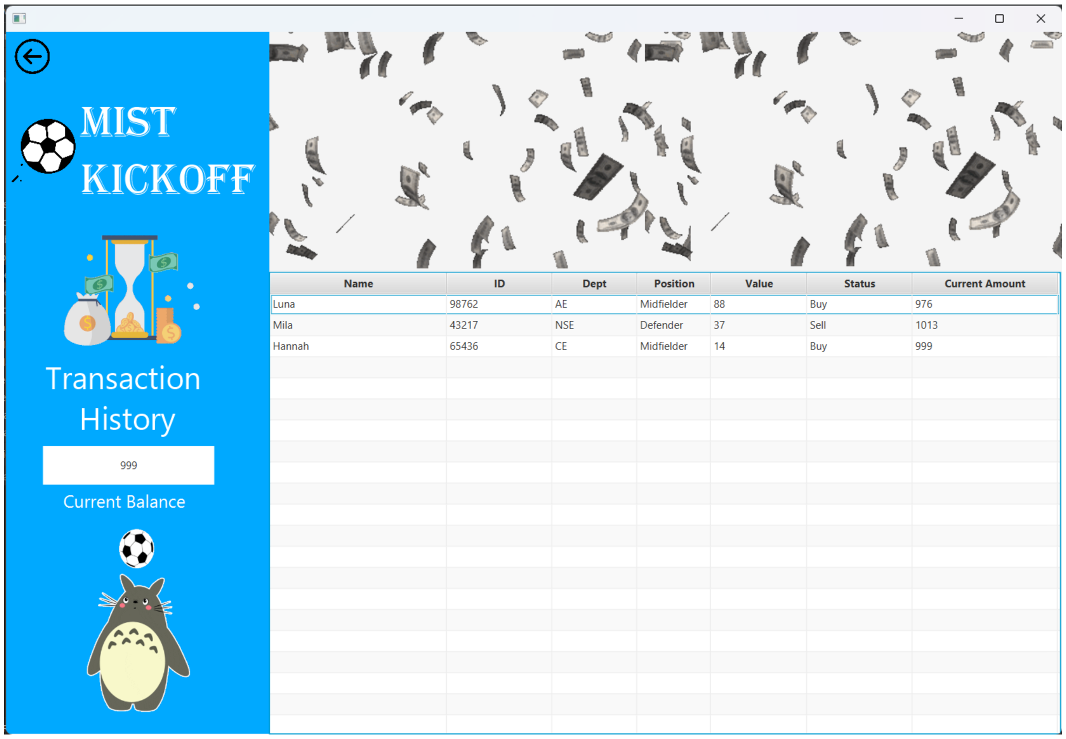


Name: 54325
ID: PME
Dept: Logan
Position: Defender
Value: 16

999
Current Balance

Name	ID	Department	Position	Value
Sam	34567	EECE	Midfielder	95
Oliver	54329	PME	Forward	67
William	21097	CSE	Midfielder	47
Logan	54325	PME	Defender	16
Aria	43216	NSE	Forward	72
Jack	65435	CE	Goalkeeper	47
Mila	43217	NSE	Defender	37
Samuel	98763	AE	Defender	93
Skylar	32110	NAME	Midfielder	86

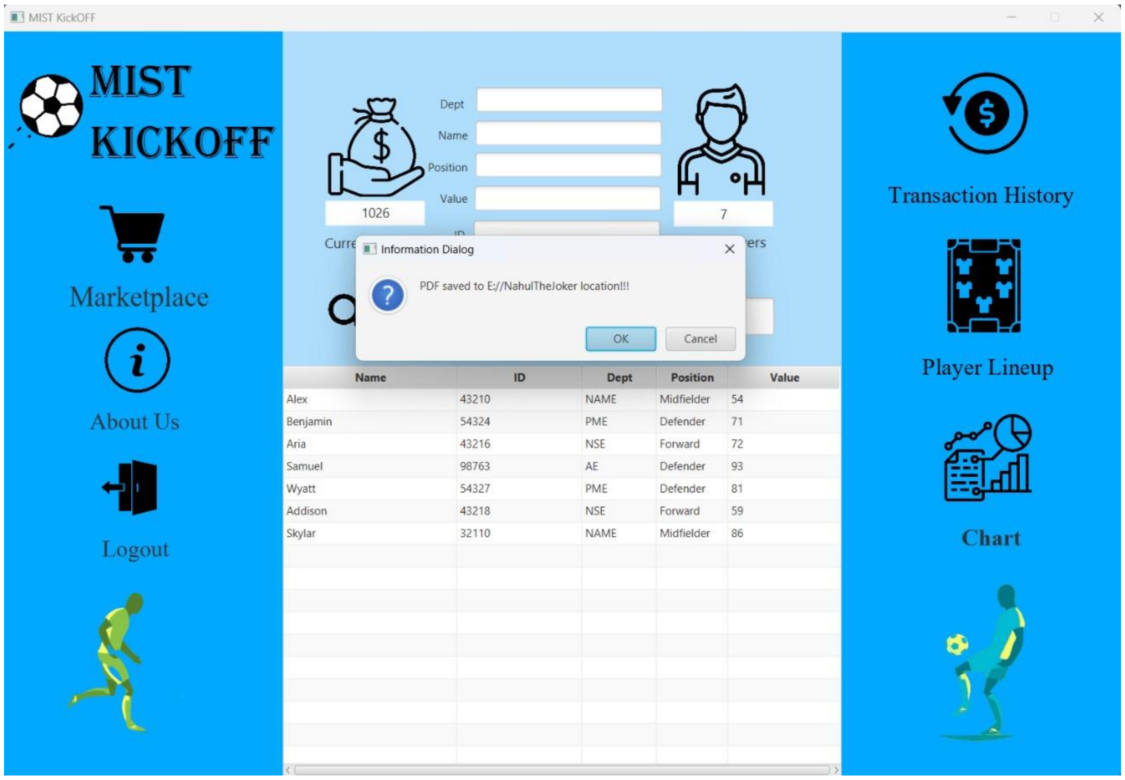
5. Transaction History to review all previous transaction records.



999

Name	ID	Dept	Position	Value	Status	Current Amount
Luna	98762	AE	Midfielder	88	Buy	976
Mila	43217	NSE	Defender	37	Sell	1013
Hannah	65436	CE	Midfielder	14	Buy	999

6. Player Line Up in PDF format

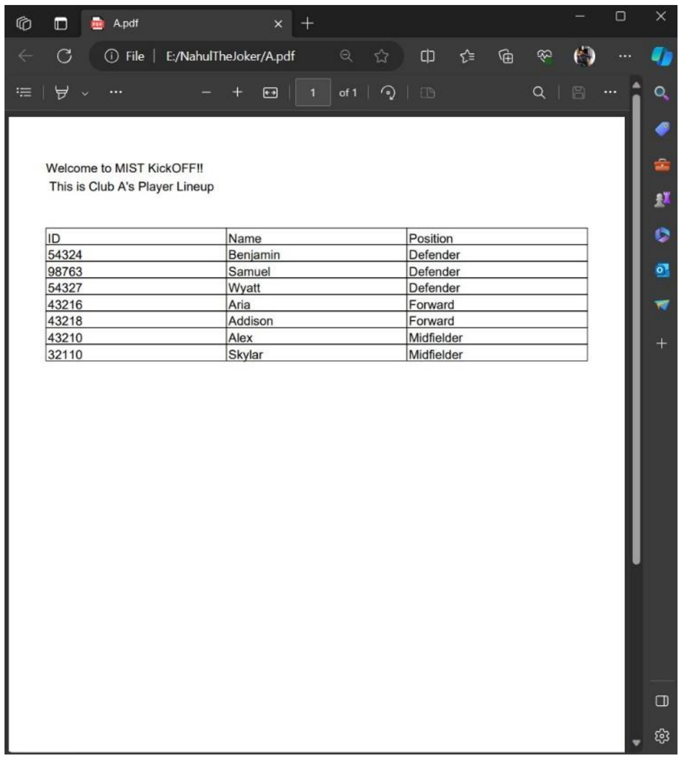


1026

7

PDF saved to E://NahulTheJoker location!!!

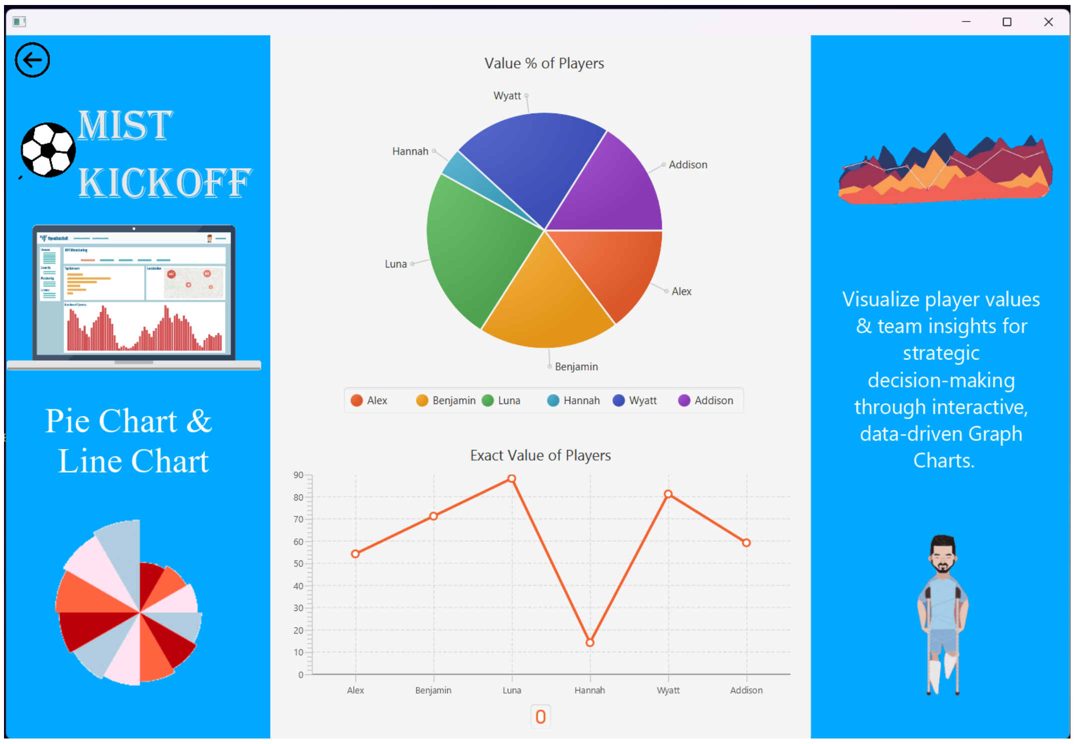
Name	ID	Dept	Position	Value
Alex	43210	NAME	Midfielder	54
Benjamin	54324	PME	Defender	71
Aria	43216	NSE	Forward	72
Samuel	98763	AE	Defender	93
Wyatt	54327	PME	Defender	81
Addison	43218	NSE	Forward	59
Skylar	32110	NAME	Midfielder	86



Welcome to MIST KickOFF!!
This is Club A's Player Lineup

ID	Name	Position
54324	Benjamin	Defender
98763	Samuel	Defender
54327	Wyatt	Defender
43216	Aria	Forward
43218	Addison	Forward
43210	Alex	Midfielder
32110	Skylar	Midfielder

7. <u>Graphical representation of data</u>


MIST KICKOFF
Pie Chart & Line Chart
Value % of Players
Wyatt
Hannah
Addison
Luna
Alex
Benjamin
● Alex ● Benjamin ● Luna ● Hannah ● Wyatt ● Addison
Exact Value of Players
0
Visualize player values & team insights for strategic decision-making through interactive, data-driven Graph Charts.

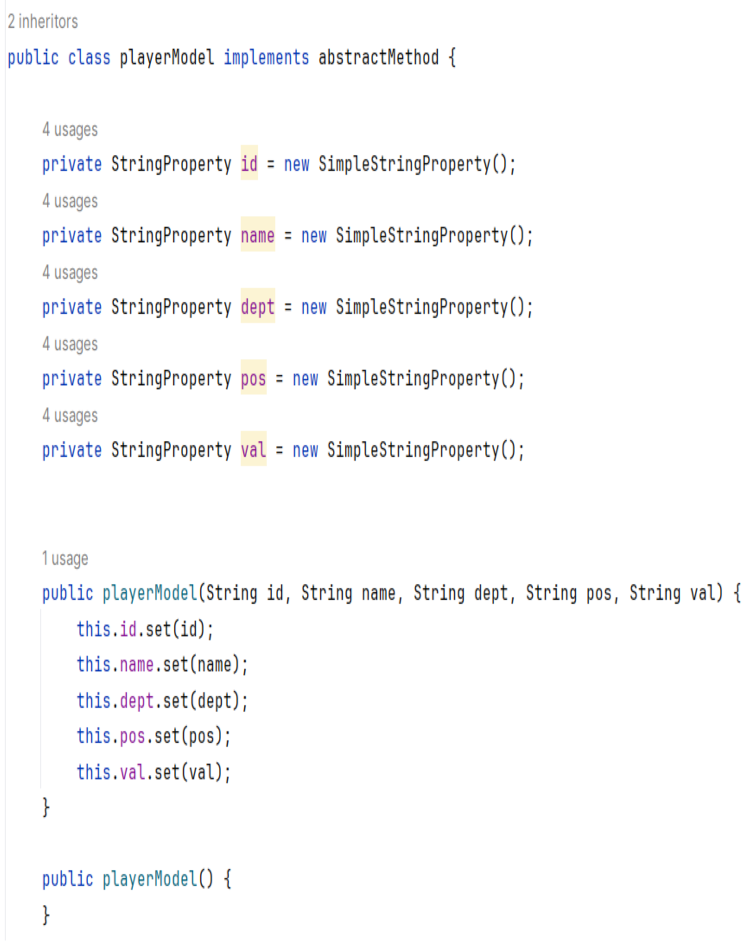
Join us on this innovative journey as we redefine the landscape of football management at MIST.

Features of OOP in JAVA

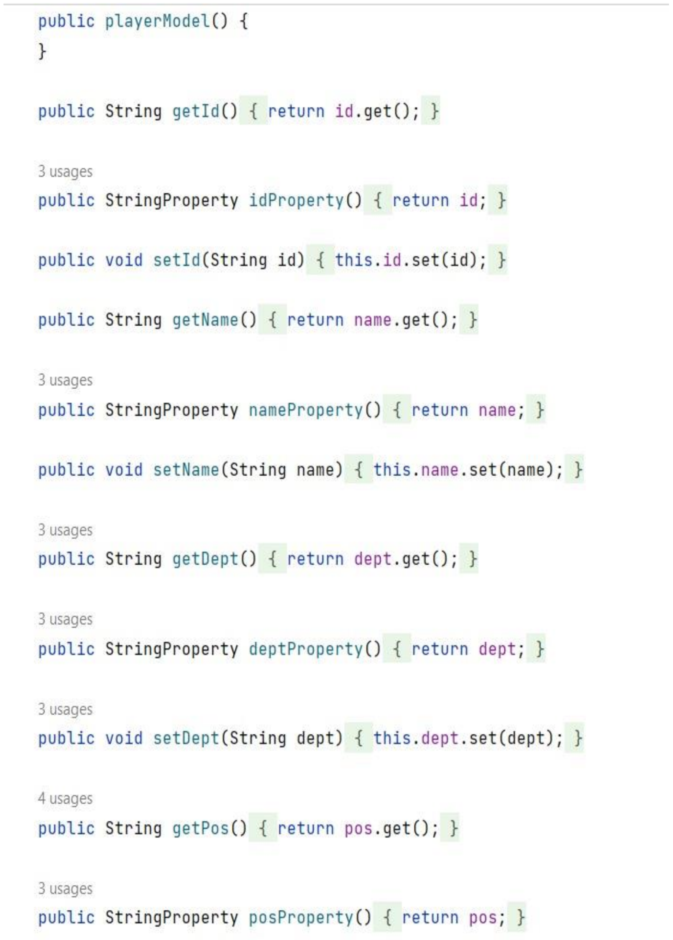
While explaining about the project that features the basic operations of object-oriented programming, all the codes can't be shown at a time but yes, some basic implementation can be shown.

Encapculation in OOP:

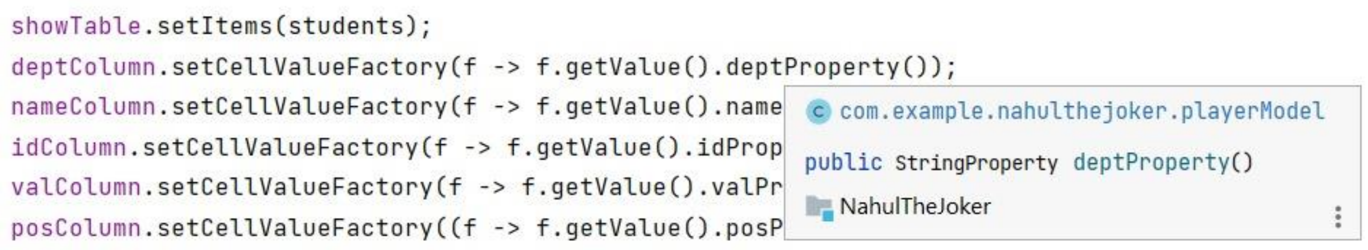
In playerModel class, the attributes are private. To access them getter setter has been declared in public mode. Later in homecontrol class to execute the table-view the attributes of playerModel was accessed through calling the getter,setter function of that class.



```
2 inheritors
public class playerModel implements abstractMethod {

    4 usages
    private StringProperty id = new SimpleStringProperty();
    4 usages
    private StringProperty name = new SimpleStringProperty();
    4 usages
    private StringProperty dept = new SimpleStringProperty();
    4 usages
    private StringProperty pos = new SimpleStringProperty();
    4 usages
    private StringProperty val = new SimpleStringProperty();


    1 usage
    public playerModel(String id, String name, String dept, String pos, String val) {
        this.id.set(id);
        this.name.set(name);
        this.dept.set(dept);
        this.pos.set(pos);
        this.val.set(val);
    }

    public playerModel() {
    }
```



```
public playerModel() {
}

public String getId() { return id.get(); }

3 usages
public StringProperty idProperty() { return id; }

public void setId(String id) { this.id.set(id); }

public String getName() { return name.get(); }

3 usages
public StringProperty nameProperty() { return name; }

public void setName(String name) { this.name.set(name); }

3 usages
public String getDept() { return dept.get(); }

3 usages
public StringProperty deptProperty() { return dept; }

3 usages
public void setDept(String dept) { this.dept.set(dept); }

4 usages
public String getPos() { return pos.get(); }

3 usages
public StringProperty posProperty() { return pos; }
```

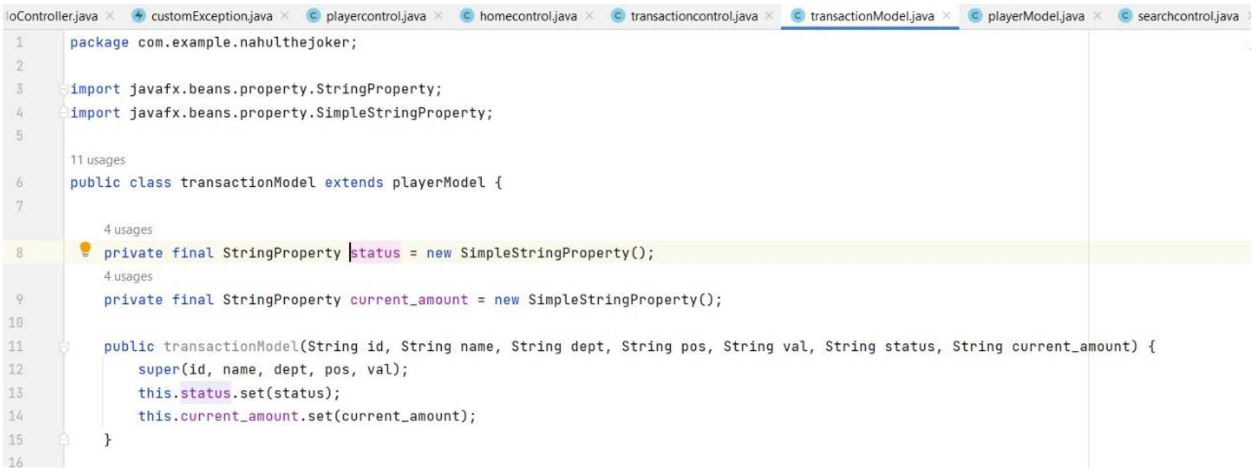


```
showTable.setItems(students);
deptColumn.setCellValueFactory(f -> f.getValue().deptProperty());
nameColumn.setCellValueFactory(f -> f.getValue().name
idColumn.setCellValueFactory(f -> f.getValue().idProp
valColumn.setCellValueFactory(f -> f.getValue().valPr
posColumn.setCellValueFactory((f -> f.getValue().posP
```

```
com.example.nahulthejoker.playerModel
public StringProperty deptProperty()
NahulTheJoker
```

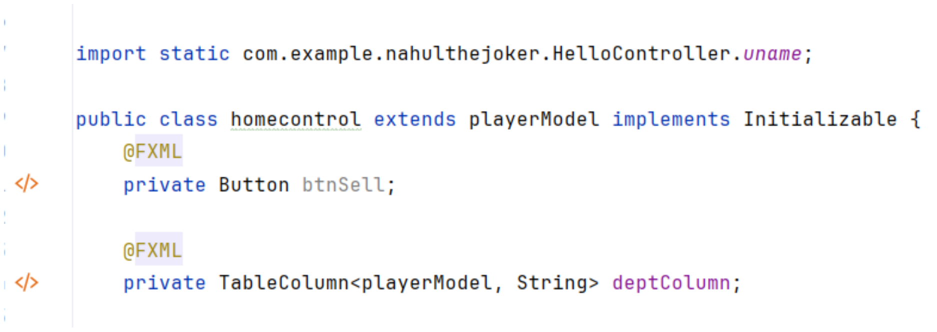
Inheritance in OOP:

transactionModel class inherited playerModel class. After creating transactionModel as the child class of playerModel class, necessary constructors, getter, setter functions have been declared.



```java
package com.example.nahulthejoker;

import javafx.beans.property.StringProperty;
import javafx.beans.property.SimpleStringProperty;

11 usages
public class transactionModel extends playerModel {

    4 usages
    private final StringProperty status = new SimpleStringProperty();
    4 usages
    private final StringProperty current_amount = new SimpleStringProperty();

    public transactionModel(String id, String name, String dept, String pos, String val, String status, String current_amount) {
        super(id, name, dept, pos, val);
        this.status.set(status);
        this.current_amount.set(current_amount);
    }
```

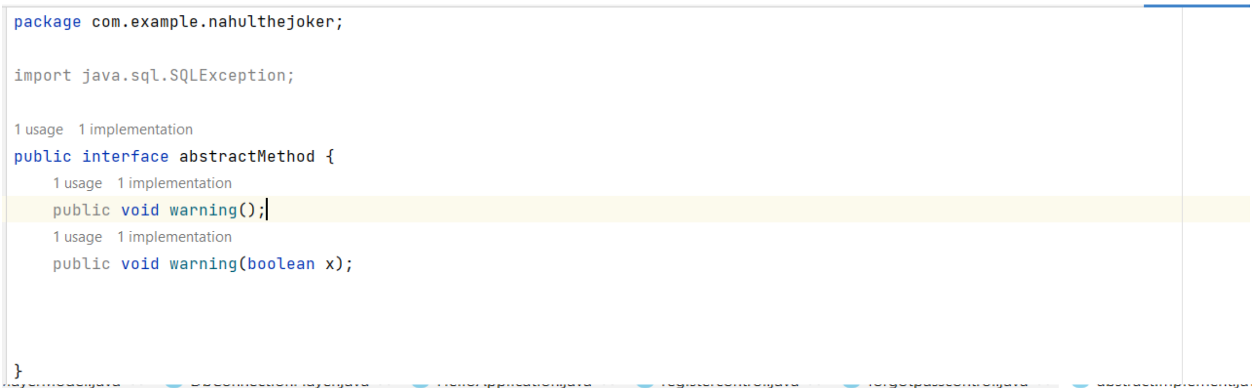
Homecontrol class inherited playerModel class.



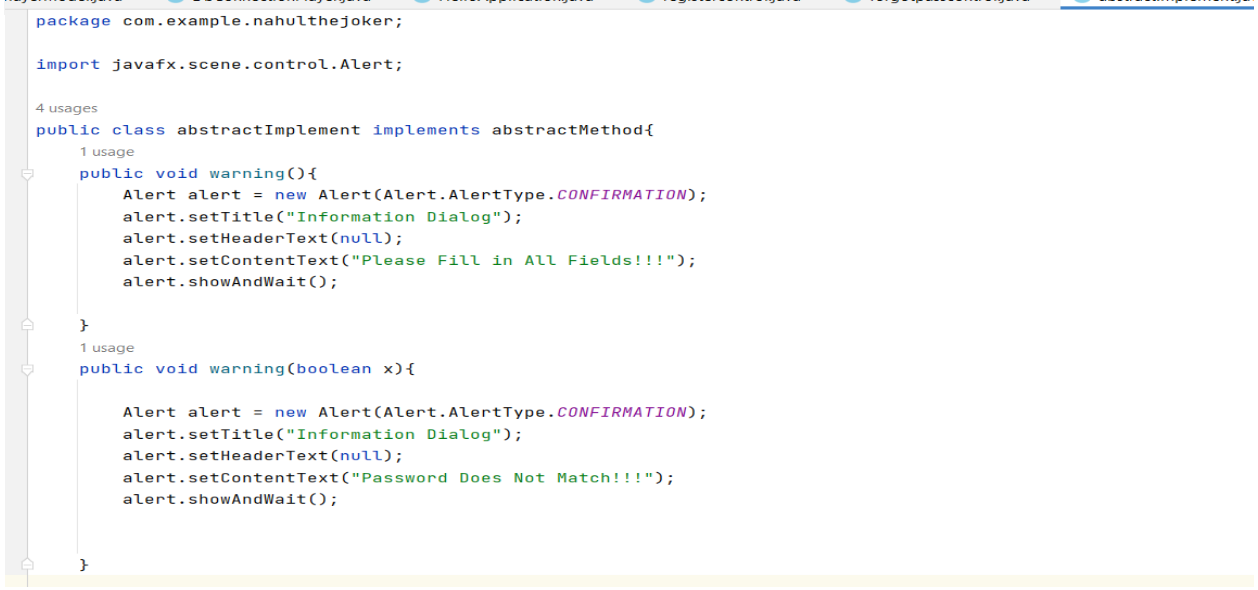
```java
import static com.example.nahulthejoker.HelloController.uname;

public class homecontrol extends playerModel implements Initializable {
    @FXML
    private Button btnSell;

    @FXML
    private TableColumn<playerModel, String> deptColumn;
```

Interface, Abstract and Polymorphism :

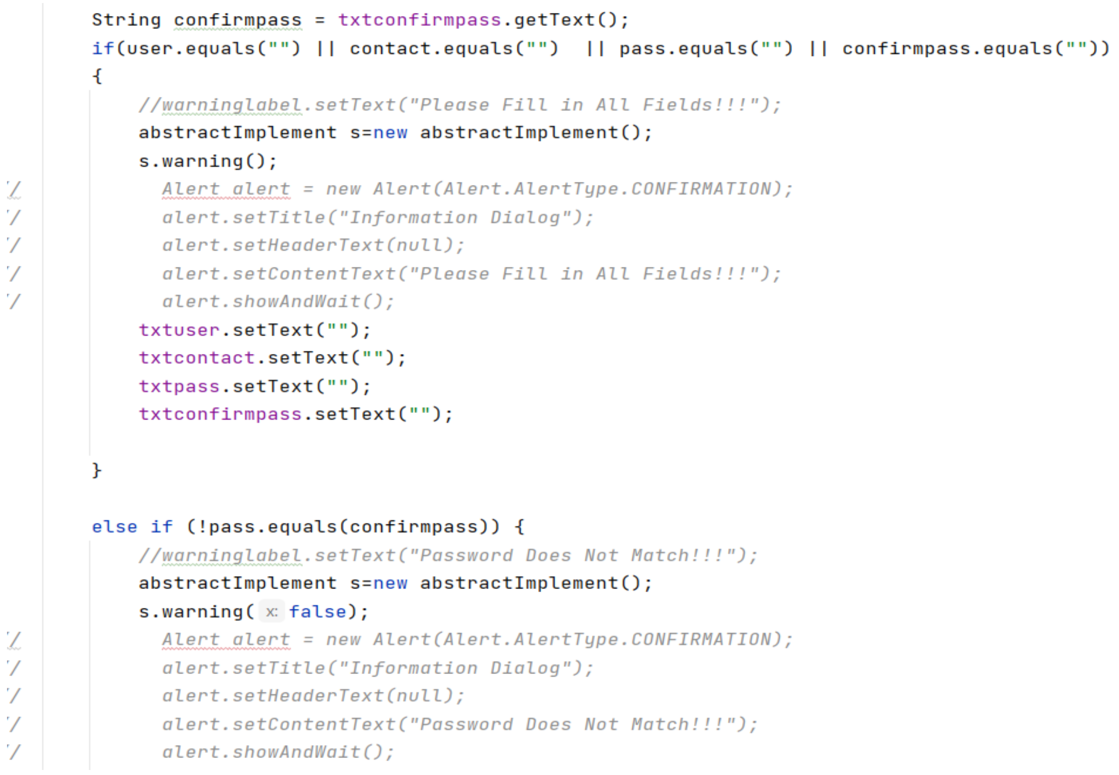
We took abstractMethod interface where warning() without parameter, warning() with parameter declared. Later in abstractImplement class the methods were defined. In forgotpass class two functions were called with(boolean) and without parameter. Code snippet is attached below.



```java
package com.example.nahulthejoker;

import java.sql.SQLException;

1 usage   1 implementation
public interface abstractMethod {
    1 usage   1 implementation
    public void warning();
    1 usage   1 implementation
    public void warning(boolean x);



}
```



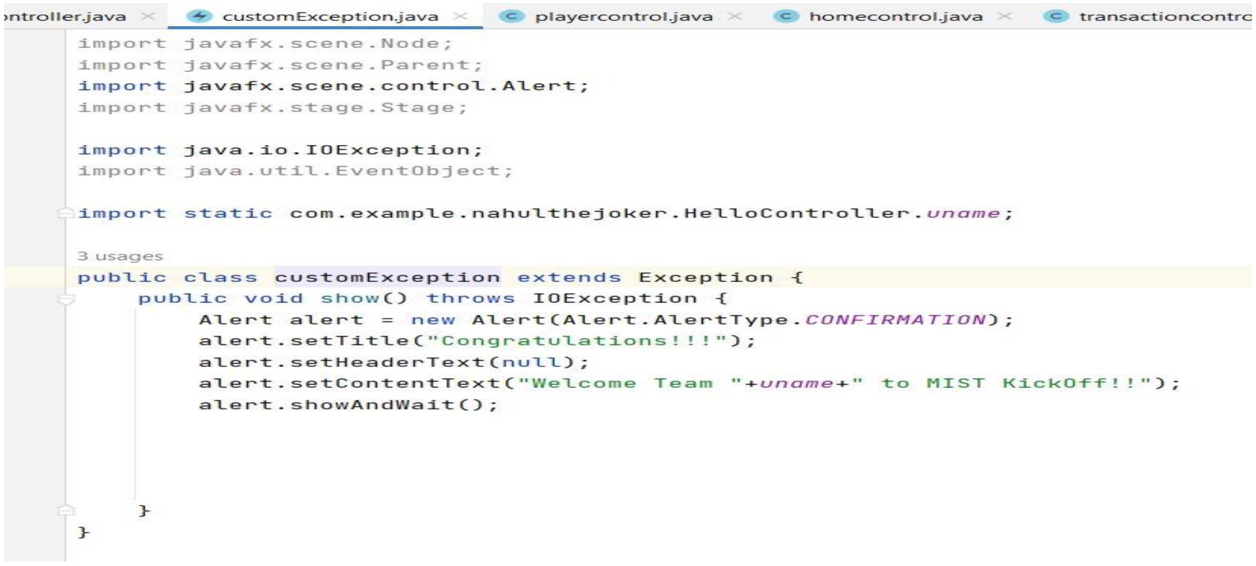
```java
package com.example.nahulthejoker;

import javafx.scene.control.Alert;

4 usages
public class abstractImplement implements abstractMethod{
    1 usage
    public void warning(){
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(null);
        alert.setContentText("Please Fill in All Fields!!!");
        alert.showAndWait();

    }
    1 usage
    public void warning(boolean x){

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(null);
        alert.setContentText("Password Does Not Match!!!");
        alert.showAndWait();


    }
```



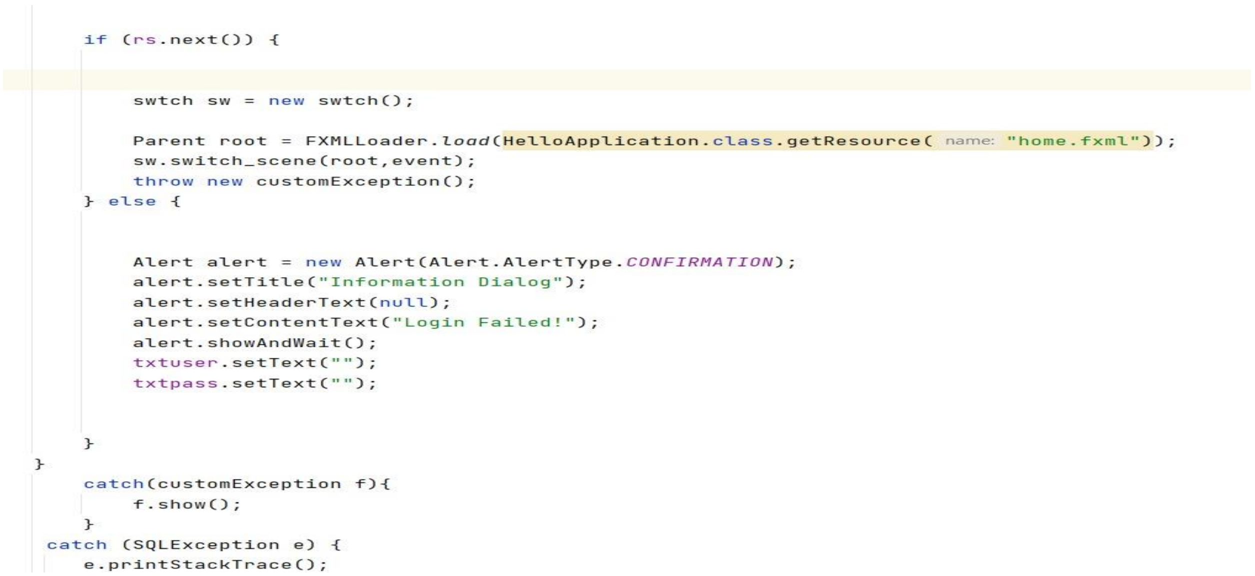
```java
        String confirmpass = txtconfirmpass.getText();
        if(user.equals("") || contact.equals("")  || pass.equals("") || confirmpass.equals(""))
        {
            //warninglabel.setText("Please Fill in All Fields!!!");
            abstractImplement s=new abstractImplement();
            s.warning();
/             Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
/             alert.setTitle("Information Dialog");
/             alert.setHeaderText(null);
/             alert.setContentText("Please Fill in All Fields!!!");
/             alert.showAndWait();
            txtuser.setText("");
            txtcontact.setText("");
            txtpass.setText("");
            txtconfirmpass.setText("");

        }

        else if (!pass.equals(confirmpass)) {
            //warninglabel.setText("Password Does Not Match!!!");
            abstractImplement s=new abstractImplement();
            s.warning( x: false);
/             Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
/             alert.setTitle("Information Dialog");
/             alert.setHeaderText(null);
/             alert.setContentText("Password Does Not Match!!!");
/             alert.showAndWait();
```

Custom Exception Handling:

We took a class named customException with a customized exception method(extends Exception class). In homecontrol class, if login is successful than it throws customException exception. Than in customException class an alert box instruction is executed.



```
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
import javafx.stage.Stage;

import java.io.IOException;
import java.util.EventObject;

import static com.example.nahulthejoker.HelloController.uname;

3 usages
public class customException extends Exception {
    public void show() throws IOException {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Congratulations!!!");
        alert.setHeaderText(null);
        alert.setContentText("Welcome Team "+uname+" to MIST KickOff!!");
        alert.showAndWait();



    }
}
```

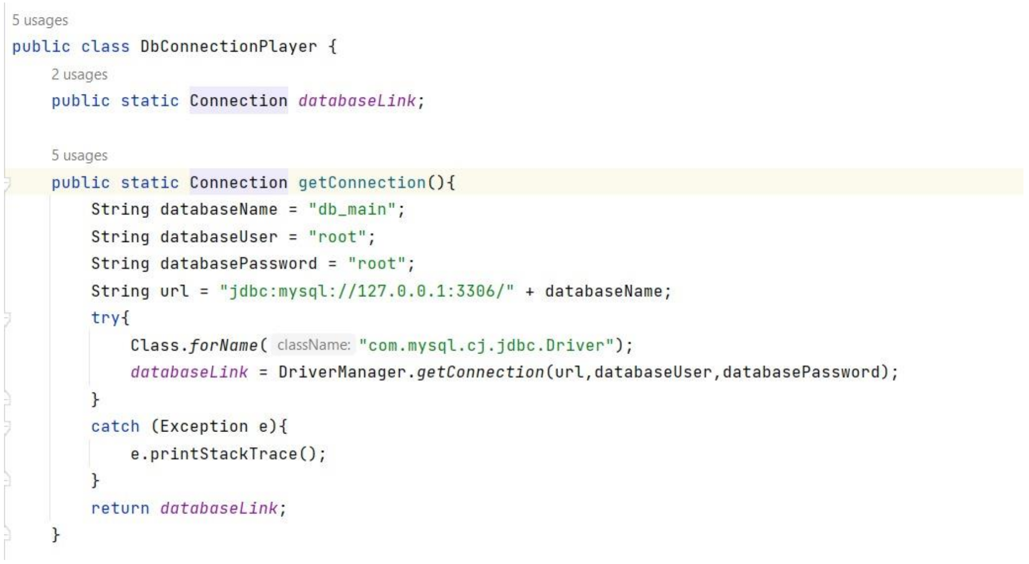


```
    if (rs.next()) {


        swtch sw = new swtch();

        Parent root = FXMLLoader.load(HelloApplication.class.getResource( name: "home.fxml"));
        sw.switch_scene(root,event);
        throw new customException();
    } else {


        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(null);
        alert.setContentText("Login Failed!");
        alert.showAndWait();
        txtuser.setText("");
        txtpass.setText("");


    }
}
    catch(customException f){
        f.show();
    }
 catch (SQLException e) {
    e.printStackTrace();
```

<u>Database Handling</u>

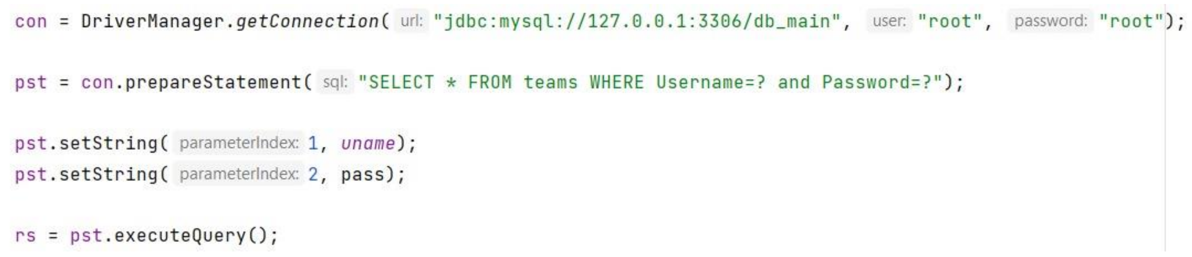
For Database connection and handling we used MySQL. For Data fetch, insert, update and delete operations; Queries used in the project are shown below.

<u>Database Connection</u>



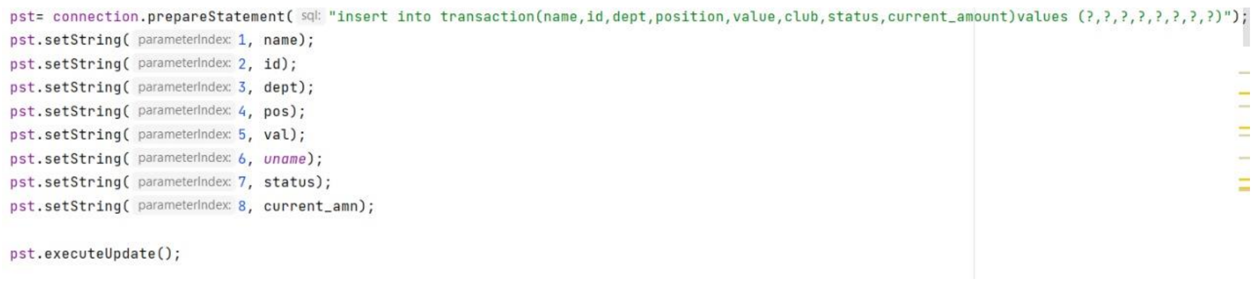
```
5 usages
public class DbConnectionPlayer {
    2 usages
    public static Connection databaseLink;

    5 usages
    public static Connection getConnection(){
        String databaseName = "db_main";
        String databaseUser = "root";
        String databasePassword = "root";
        String url = "jdbc:mysql://127.0.0.1:3306/" + databaseName;
        try{
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            databaseLink = DriverManager.getConnection(url,databaseUser,databasePassword);
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return databaseLink;
    }
```

<u>Data Fetch</u>



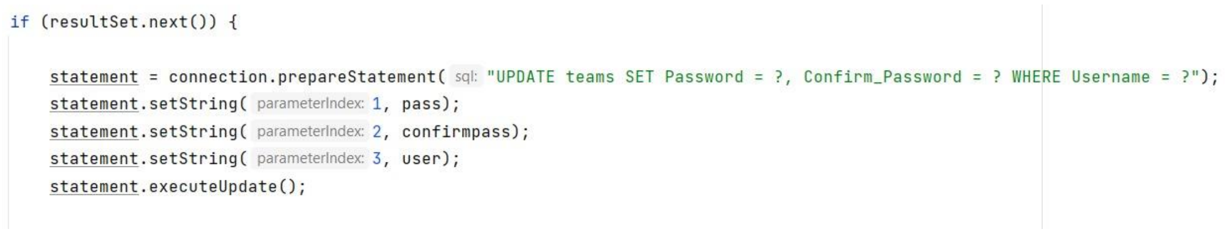
```
con = DriverManager.getConnection( url: "jdbc:mysql://127.0.0.1:3306/db_main",  user: "root",  password: "root");

pst = con.prepareStatement( sql: "SELECT * FROM teams WHERE Username=? and Password=?");

pst.setString( parameterIndex: 1, uname);
pst.setString( parameterIndex: 2, pass);

rs = pst.executeQuery();
```

Data Insertion

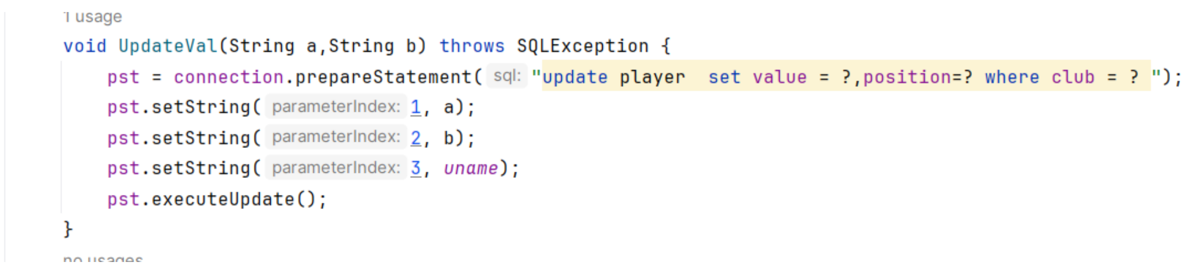


```
pst= connection.prepareStatement( sql: "insert into transaction(name,id,dept,position,value,club,status,current_amount)values (?,?,?,?,?,?,?,?)");
pst.setString( parameterIndex: 1, name);
pst.setString( parameterIndex: 2, id);
pst.setString( parameterIndex: 3, dept);
pst.setString( parameterIndex: 4, pos);
pst.setString( parameterIndex: 5, val);
pst.setString( parameterIndex: 6, uname);
pst.setString( parameterIndex: 7, status);
pst.setString( parameterIndex: 8, current_amn);

pst.executeUpdate();
```

Previous Data Delete Then Update

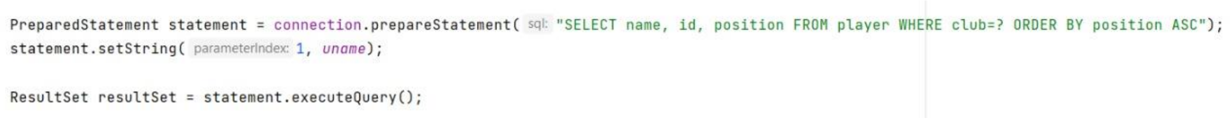


```
if (resultSet.next()) {

    statement = connection.prepareStatement( sql: "UPDATE teams SET Password = ?, Confirm_Password = ? WHERE Username = ?");
    statement.setString( parameterIndex: 1, pass);
    statement.setString( parameterIndex: 2, confirmpass);
    statement.setString( parameterIndex: 3, user);
    statement.executeUpdate();
```



```
    1 usage
    void UpdateVal(String a,String b) throws SQLException {
        pst = connection.prepareStatement( sql: "update player  set value = ?,position=? where club = ? ");
        pst.setString( parameterIndex: 1, a);
        pst.setString( parameterIndex: 2, b);
        pst.setString( parameterIndex: 3, uname);
        pst.executeUpdate();
    }
```

Data Ordering



```
PreparedStatement statement = connection.prepareStatement( sql: "SELECT name, id, position FROM player WHERE club=? ORDER BY position ASC");
statement.setString( parameterIndex: 1, uname);

ResultSet resultSet = statement.executeQuery();
```

Further Improvements:

- UI/UX design
- CSS implementation
- Individual player interface
- PDF enhancement
- Hyperlink Attachment
- Screen Responsive

Setbacks of the Project:

- While GUI implementing Intelij couldn't detect the imagre source path. When we kept the pictures in resource file than it was resolved.
- Connection with Database MYSQL was very sensitive. While implementing Query table name, column names needed to write cautiously.
- For design purpose use of CSS, Figma could be more helpful.
- ObservableArrayList keyword was unknown initially, so the use was of it was difficult.
- While implementing PDF, version of itextpdf needed to match with Intelij configuration.

In summary, MIST KickOff represents a successful amalgamation of fervor for football and technological ingenuity, introducing comprehensive football management to the Military Institute of Science and Technology (MIST). Employing a user-friendly desktop application, our collaborative team, comprised of Aunindya Prosad Saha, G. M. Fahim Tazwar, and Md. Nahul Rahman, leveraged JavaFX, IntelliJ, Scenebuilder, and MySQL to furnish a feature-rich interface. The project adheres to robust object-oriented programming principles, providing functionalities such as user login, a real-time marketplace, and sophisticated data representation. This endeavor not only redefines football management at MIST but also underscores our unwavering dedication to excellence in sports technology.

Reference and Links

1. https://www.flaticon.com/

2. https://html-color-codes.info/colors-from-image/

3. https://www.unscreen.com/upload

4. https://www.youtube.com/watch?v=DIZtUGfEW0Q&list=PLX5hCViO2ncTCdMt5ozw3UjjfxPqIxdOq