

ASSIGNMENT 1 FRONT SHEET

Qualification	BTEC Level 5 HND Diploma in Computing		
Unit number and title			
Submission date	22 nd December, 2022	Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	
Student Name	Truong Van Phuc Khang	Student ID	GCD210600
Class	GCD1002	Assessor name	Pham Thanh Son
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	Khang

Grading grid

Grade (0-10)

☐ **Summative Feedback:**

☐ **Resubmission Feedback:**

Grade:

Assessor Signature:

Date:

IV Signature:

Table of Contents

1. Introduction	4
2. Requirement	5
3. UI design	5
4. Implementation	8
5. Test.....	18
6. Result	18
7. Conclusion.....	26
Figure 1 Overall wireframe	5
Figure 2 Menu bar wireframe	5
Figure 3 File menu bar wireframe	6
Figure 4 About menu bar wireframe	6
Figure 5 Club search box without club name wireframe.....	6
Figure 6 Club search box with club name wireframe	6
Figure 7 Id search box without id wireframe.....	6
Figure 8 Id search box with id wireframe	7
Figure 9 Search and cancel search button wireframe	7
Figure 10 Footballer list table without info wireframe	7
Figure 11 Footballer list table with info wireframe	7
Figure 12 Footballer detail box without info wireframe	8
Figure 13 Footballer detail box with info wireframe	8
Figure 14 Add, delete, update, and cancel button wireframe	8
Figure 15 Footballer Manager app structure.....	8
Figure 16 Class Footballer	9
Figure 17 Class Club	9
Figure 18 Main function in FootballerManagerView.....	10
Figure 19 FootballerManagerView function.....	10
Figure 20 FootballerManagerModel.....	11
Figure 21 FootballManagerController class.....	11
Figure 22 Add or update footballer function.....	12
Figure 23 Update info into the detail box function	13
Figure 24 Delete Footballer function	13
Figure 25 Cancel function	13
Figure 26 Search Footballer function.....	14
Figure 27 Delete Searching info function	15

Figure 28 Save file function	16
Figure 29 Open file function	17
Figure 30 Exit the program function.....	17
Figure 31 An example about handling errors	18
Figure 32 Test case 1 before proceeding.....	19
Figure 33 Test case 1 after proceeding.....	19
Figure 34 Test case 2 before proceeding.....	20
Figure 35 Test case 2 after proceeding.....	20
Figure 36 Test case 3 before proceeding.....	21
Figure 37 Test case 4 after proceeding.....	21
Figure 38 Test case 4 before proceeding.....	22
Figure 39 Test case 4 after proceeding.....	22
Figure 40 Save file.....	23
Figure 41 Save file as data.txt.....	23
Figure 42 File data.txt	23
Figure 43 Exit the application	23
Figure 44 Open the app again, the table is reset.....	24
Figure 45 Open the file	24
Figure 46 Open the file data.txt.....	24
Figure 47 The file after proceeding, Test case 5 Success.....	25
Figure 48 Search players who are currently playing at Manchester City	25
Figure 49 Search players who Id is 2, and playing at Manchester City.....	25
Figure 50 Search player who Id is 4	26
Figure 51 The error occurs when I enter the information with missing date	26
Figure 52 The box appears as normal.....	26
Figure 53 Then I add the detail	26
Figure 54 But the data still not appears in the list.....	27
Figure 55 But then I clicked the search or cancel search button and the data appears	27
 Table 1 Test case.....	 18

1. Introduction

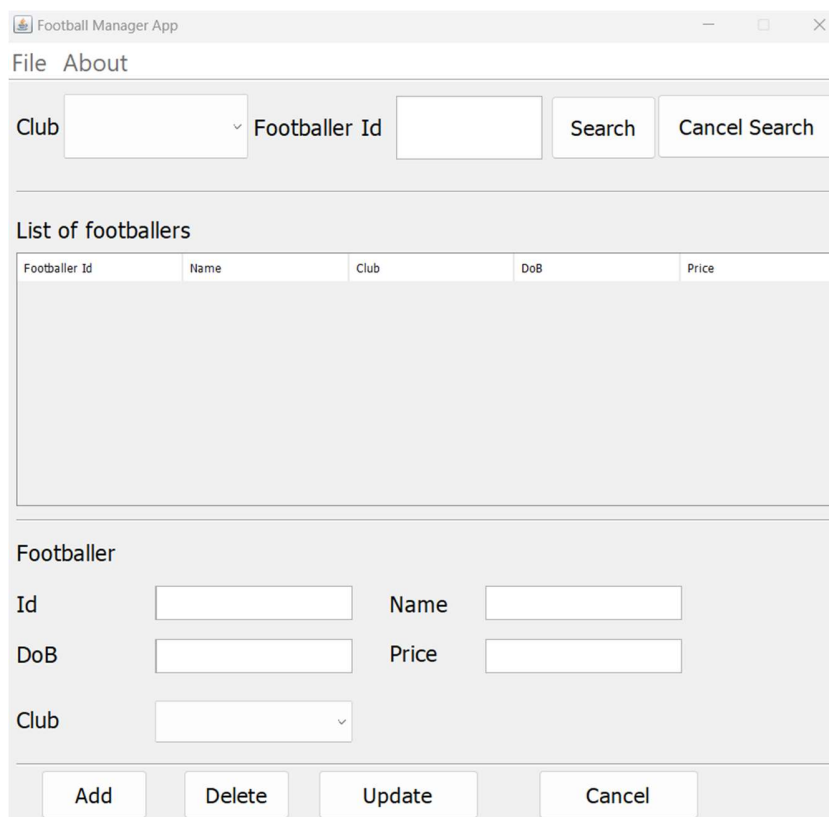
World Cup 2022 is coming, and many people want to search for information about players who are performing well in the tournament. Therefore, the company want me to create an app to store footballers information as well as their market price so that they can decide which footballer they will buy in the winter transfer window of the 2022-2023 season, which is going to start in January 2023.

2. Requirement

I have to develop a graphical user interface application to create, read, update and delete footballers. This application needs to save / open data from the text file, be able to searching for footballer base on clubs and id. It also have to handle some errors so that it will not crash at end user side. Finally, this also needs to be tested before the production phase start.

I also have to write a technical report about the development of this application, which is this assignment report, and then demo and explain my code and answer questions.

3. UI design



The wireframe shows a desktop application window titled "Football Manager App". It features a menu bar with "File" and "About". Below the menu bar is a search section with a "Club" dropdown, a "Footballer Id" input field, and "Search" and "Cancel Search" buttons. The main area is titled "List of footballers" and contains a table with columns: "Footballer Id", "Name", "Club", "DoB", and "Price". The table is currently empty. Below the table is a "Footballer" section with input fields for "Id", "Name", "DoB", "Price", and a "Club" dropdown. At the bottom are four buttons: "Add", "Delete", "Update", and "Cancel".

Figure 1 Overall wireframe

File About

Figure 2 Menu bar wireframe

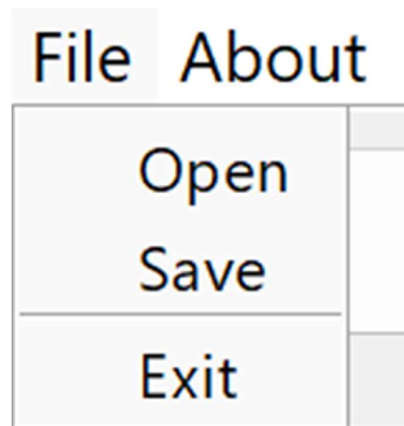


Figure 3 File menu bar wireframe

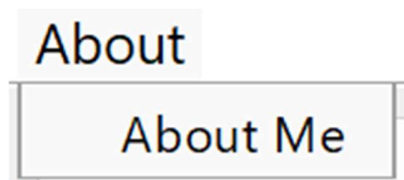


Figure 4 About menu bar wireframe

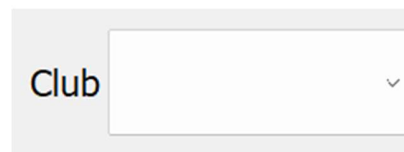


Figure 5 Club search box without club name wireframe

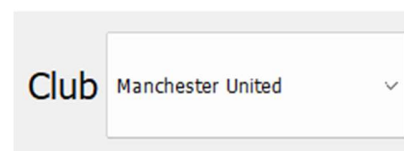


Figure 6 Club search box with club name wireframe



Figure 7 Id search box without id wireframe

Footballer Id

Figure 8 Id search box with id wireframe

Search

Cancel Search

Figure 9 Search and cancel search button wireframe

List of footballers

Footballer Id	Name	Club	DoB	Price

Figure 10 Footballer list table without info wireframe

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Antony	Manchester United	24/2/2000	75
3	Saka	Arsenal	5/9/2001	90
4	Mac Allister	Brighton & Hove...	24/12/1998	32
5	Rice	West Ham United	14/1/1999	80
6	Trippier	Newcastle United	19/9/1990	13
7	Zaha	Crystal Palace	10/11/1992	32
8	Fernandes	Manchester United	8/9/1994	75

Figure 11 Footballer list table with info wireframe

Footballer

Id	<input type="text"/>	Name	<input type="text"/>
DoB	<input type="text"/>	Price	<input type="text"/>
Club	<input type="text" value="v"/>		

Figure 12 Footballer detail box without info wireframe

Footballer

Id	<input type="text" value="1"/>	Name	<input type="text" value="Haaland"/>
DoB	<input type="text" value="21/7/2000"/>	Price	<input type="text" value="170"/>
Club	<input type="text" value="v"/>		

Figure 13 Footballer detail box with info wireframe

<input type="button" value="Add"/>	<input type="button" value="Delete"/>	<input type="button" value="Update"/>	<input type="button" value="Cancel"/>
------------------------------------	---------------------------------------	---------------------------------------	---------------------------------------

Figure 14 Add, delete, update, and cancel button wireframe

4. Implementation

Program structure contains two classes of data: Footballer and Club. A class contains the wireframes and its functions: FootballManagerView, a class consisting of the app data: FootballManagerModel and an class to handle the event when the user click buttons: FootballManagerController.

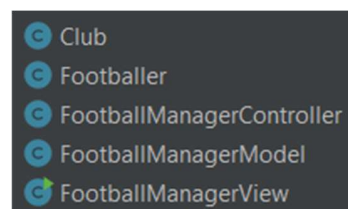


Figure 15 Footballer Manager app structure

In the Footballer program, data about footballer is contained in this class.


```
public class Footballer implements Serializable{
    7 usages
    private int footballerId;
    7 usages
    private String name;
    7 usages
    private Club club;
    9 usages
    private Date doB;
    7 usages
    private int price;

    2 usages
    public Footballer(int footballerId, String name, Club club, Date doB, int price) {
        this.footballerId = footballerId;
        this.name = name;
        this.club = club;
        this.doB = doB;
        this.price = price;
    }
}
```

Figure 16 Class Footballer

All the information of the Club is store at class Club.

```
27 usages
public class Club implements Serializable{
    5 usages
    private int clubId;
    8 usages
    private String name;

    1 usage
    public Club(int clubId, String name) {
        this.name = name;
        this.clubId = clubId;
    }
}
```

Figure 17 Class Club

In class FootballerManagerView contain the main function, JFrame function which is the wireframe of the project, and important function for the program to run.

```

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                FootballManagerView frame = new FootballManagerView();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

```

Figure 18 Main function in FootballerManagerView

```

/**
 * Create the frame.
 */
2 usages
public FootballManagerView() {
    this.model = new FootballManagerModel();
    setTitle("Football Manager App");
    setResizable(false);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener((WindowAdapter) windowClosing(e) → {
        JFrame jFrame = (JFrame) e.getSource();
        int result = JOptionPane.showConfirmDialog(jFrame, message: "Are you sure ?",
            title: "Exit Application", JOptionPane.YES_NO_OPTION);
        if (result == JOptionPane.YES_OPTION){
            setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
    });
    setBounds(x: 250, y: 25, width: 725, height: 700);

    Action action = new FootballManagerController(view: this);
}

```

Figure 19 FootballerManagerView function

While the FootballerManagerView class handle the overall the GUI, FootballerManagerModel class handle the class database of the project.

```
public class FootballManagerModel {  
    11 usages  
    private ArrayList<Footballer> footballers;  
    3 usages  
    private String name;  
  
    1 usage  
    public FootballManagerModel() {  
        this.footballers = new ArrayList<Footballer>();  
        this.name="";  
    }  
  
    public FootballManagerModel(ArrayList<Footballer> footballers) { this.footballers = footballers; }  
  
    2 usages  
    public ArrayList<Footballer> getFootballers() {  
        return footballers;  
    }  
  
    1 usage  
    public void setFootballers(ArrayList<Footballer> footballers) { this.footballers = footballers; }
```

Figure 20 FootballManagerModel

The last class is FootballManagerController, which handle the I/O interfaces.

```
public class FootballManagerController implements Action{  
    15 usages  
    public FootballManagerView view;  
  
    1 usage  
    public FootballManagerController(FootballManagerView view) { this.view = view; }
```

Figure 21 FootballManagerController class

In this project, I've also added some features so that the user can interact with the app in an accurate and convenient way. All the features are:

Add or Update a Footballer into the System:

```

public void addFootballer(Footballer footballer) {
    DefaultTableModel model_table = (DefaultTableModel) table.getModel();
    model_table.addRow(new Object[] { footballer.getFootballerId() + "",
        footballer.getName(),
        footballer.getCountry().getName(),
        footballer.getDoB().getDate() + "/"
            + (footballer.getDoB().getMonth() + 1)
            + "/" + (footballer.getDoB().getYear() + 1900),
        footballer.getPrice() });
}

1 usage
public void addOrUpdateFootballer(Footballer footballer) {
    DefaultTableModel model_table = (DefaultTableModel) table.getModel();
    if (!this.model.checkExist(footballer)) {
        this.model.insert(footballer);
        this.addFootballer(footballer);
    } else {
        this.model.update(footballer);
        int rowCount = model_table.getRowCount();
        for (int i = 0; i < rowCount; i++) {
            String id = model_table.getValueAt(i, column: 0) + "";
            if (id.equals(footballer.getFootballerId() + "")) {
                model_table.setValueAt(aValue: footballer.getFootballerId() + "", i, column: 0);
                model_table.setValueAt(aValue: footballer.getName() + "", i, column: 1);
                model_table.setValueAt(aValue: footballer.getCountry().getName() + "", i, column: 2);
                model_table.setValueAt(aValue: footballer.getDoB().getDate() + "/"
                    + (footballer.getDoB().getMonth() + 1) + "/"
                    + (footballer.getDoB().getYear() + 1900) + "", i, column: 3);
                model_table.setValueAt(aValue: footballer.getPrice() + "", i, column: 4);
            }
        }
    }
}
    
```

Figure 22 Add or update footballer function

Show the footballer info into the detail box:

```

public Footballer getChooseFootballer() {
    DefaultTableModel model_table = (DefaultTableModel) table.getModel();
    int i_row = table.getSelectedRow();

    int footballerId = Integer.valueOf(model_table.getValueAt(i_row, column: 0) + "");
    String name = model_table.getValueAt(i_row, column: 1) + "";
    Club club = Club.getClubByName(model_table.getValueAt(i_row, column: 2) + "");
    String date = model_table.getValueAt(i_row, column: 3) + "";
    Date doB = null;
    try {
        doB = new SimpleDateFormat(pattern: "dd/MM/yyyy").parse(date);
    } catch (ParseException e) {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Error " + e.getMessage());
    }
    int price = Integer.valueOf(model_table.getValueAt(i_row, column: 4) + "");

    Footballer footballer = new Footballer(footballerId, name, club, doB, price);
    return footballer;
}

```

Figure 23 Update info into the detail box function

Delete a Footballer from the database:

```

public void deleteFootballer() {
    DefaultTableModel model_table = (DefaultTableModel) table.getModel();
    int i_row = table.getSelectedRow();
    int choice = JOptionPane.showConfirmDialog(parentComponent: this, message: "Are you sure about that ?");
    if (choice == JOptionPane.YES_OPTION) {
        Footballer footballer = getChooseFootballer();
        this.model.delete(footballer);
        model_table.removeRow(i_row);
    }
}

```

Figure 24 Delete Footballer function

Delete the information in the detail box:

```

public void deleteForm() {
    textField_FootballerID.setText("");
    textField_Name.setText("");
    textField_DoB.setText("");
    textField_Price.setText("");
    comboBox_Club.setSelectedIndex(-1);
}

```

Figure 25 Cancel function

Search the footballer base on their club and/or Id:

```

public void searchFootballer() {
    this.reloadData();
    int club_id = this.comboBox_Club_Search.getSelectedIndex() - 1;
    String footballerIdSearch = this.textField_FootballerId_Search.getText();
    DefaultTableModel model_table = (DefaultTableModel) table.getModel();
    int rowCount = model_table.getRowCount();

    Set<Integer> footballerIdDelete = new TreeSet<>();
    if (club_id >= 0) {
        Club clubSelected = Club.getClubById(club_id);
        for (int i = 0; i < rowCount; i++) {
            String club = model_table.getValueAt(i, column: 2) + "";
            String id = model_table.getValueAt(i, column: 0) + "";
            if (!club.equals(clubSelected.getName())) {
                footballerIdDelete.add(Integer.valueOf(id));
            }
        }
    }
    if (footballerIdSearch.length() > 0) {
        for (int i = 0; i < rowCount; i++) {
            String club = model_table.getValueAt(i, column: 0) + "";
            if (!club.equals(footballerIdSearch)) {
                footballerIdDelete.add(Integer.valueOf(club));
            }
        }
    }
    for (Integer idDelete : footballerIdDelete) {

        rowCount = model_table.getRowCount();
        for (int i = 0; i < rowCount; i++) {
            String idInTable = model_table.getValueAt(i, column: 0) + "";
            if (idInTable.equals(idDelete.toString())) {
                try {
                    model_table.removeRow(i);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                break;
            }
        }
    }
}
    
```

Figure 26 Search Footballer function

Delete the data in the search box:

```
public void reloadData() {  
    while (true) {  
        DefaultTableModel model_table = (DefaultTableModel) table.getModel();  
        int rowCount = model_table.getRowCount();  
        if(rowCount == 0)  
            break;  
        else  
            try {  
                model_table.removeRow(0);  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
    }  
    for (Footballer footballer : this.model.getFootballers()) {  
        this.addFootballer(footballer);  
    }  
}  
  
1 usage  
public void deleteSearch() {  
    textField_FootballerId_Search.setText("");  
    comboBox_Club_Search.setSelectedIndex(-1);  
}
```

Figure 27 Delete Searching info function

Save File:


```
public void saveFile(String path) {
    try {
        this.model.setName(path);
        FileOutputStream fos = new FileOutputStream(path);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        for (Footballer footballer : this.model.getFootballers()) {
            oos.writeObject(footballer);
        }
        oos.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Cannot save file!");
    }
}

1 usage
public void saveFileAction() {
    JFileChooser fc = new JFileChooser();
    int returnVal = fc.showSaveDialog( parent: this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        saveFile(file.getAbsolutePath());
    }
}
```

Figure 28 Save file function

Open File:


```
public void openFile(File file) {
    ArrayList<Footballer> footballers = new ArrayList<>();
    try {
        this.model.setName(file.getAbsolutePath());
        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Footballer footballer = null;
        while((footballer = (Footballer) ois.readObject())!=null) {
            footballers.add(footballer);
        }
        ois.close();
    } catch (Exception e) {
        if(e.getMessage() != null)
            JOptionPane.showMessageDialog( parentComponent: this, message: "This file cannot be open!");
    }
    this.model.setFootballers(footballers);
}

1 usage
public void openFileAction() {
    JFileChooser fc = new JFileChooser();
    int returnVal = fc.showOpenDialog( parent: this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        openFile(file);
        reloadData();
    }
}
```

Figure 29 Open file function

Exit the app:

```
public void exitProgram() {
    int result = JOptionPane.showConfirmDialog(
        parentComponent: this,
        message: "Do you really want to exit ?",
        title: "Exit",
        JOptionPane.YES_NO_OPTION);
    if (result == JOptionPane.YES_OPTION) {
        System.exit( status: 0);
    }
}
```

Figure 30 Exit the program function

Finally, if the user is doing something that is going to make the program crash, I can handle the errors and show a box to warn them about that.

```

case "Add":
    try {
        this.view.addFootballer();
    } catch (Exception e1) {
        JOptionPane.showMessageDialog(this.view, message: "Please enter all field!");
    }
  
```

Figure 31 An example about handling errors

5. Test

Test case ID	Description	Test Step	Expected Result	Status
1	Add a footballer into the system	Add the information into the detail box, then click Add button	The list of footballers will have a row with the info we've just input	Success
2	Add a footballer with missing information into the system	Add the information into the detail box except the date detail, then click Add button	A box will appear to tell the user to enter all fields	Success
3	Update an existing footballer row in the list	Click on the selected row, click Update, then change the detail and click Add	The selected row will be updated with the new information	Success
4	Delete an existing footballer row in the list	Click on the selected row, click Delete, then click Yes	The selected row will be deleted	Success
5	Save and Open the file	Save the data into a file, then reopen the app and open the saved file	A file will appear on the screen when we save the file, and all the data will appear when we open it.	Success

Table 1 Test case

6. Result

Test 1:

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
---------------	------	------	-----	-------

Footballer

Id Name

DoB Price

Club

Add Delete Update Cancel

Figure 32 Test case 1 before proceeding

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170

Footballer

Id Name

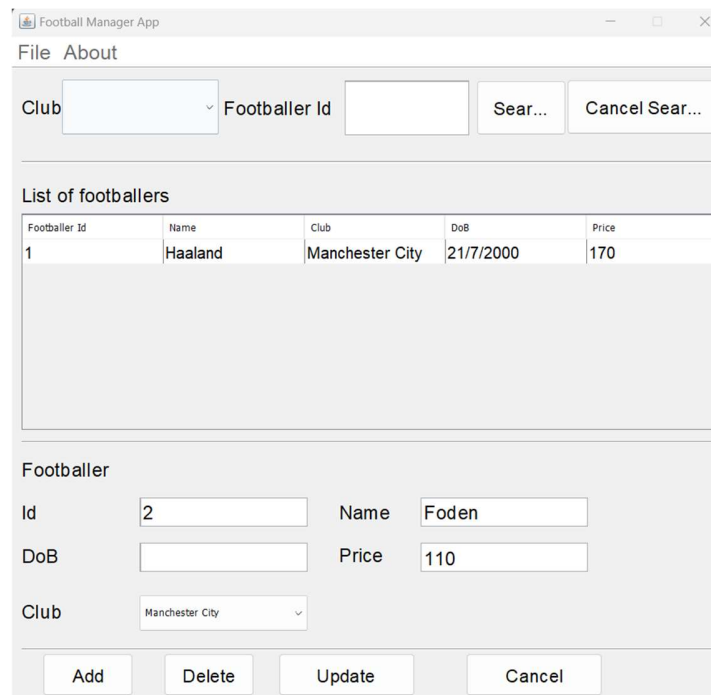
DoB Price

Club

Add Delete Update Cancel

Figure 33 Test case 1 after proceeding

Test 2:



The screenshot shows the Football Manager App interface. At the top, there is a menu bar with 'File' and 'About'. Below the menu bar, there is a search section with a 'Club' dropdown menu, a 'Footballer Id' input field, and two buttons: 'Sear...' and 'Cancel Sear...'. Below the search section, there is a 'List of footballers' table. The table has five columns: 'Footballer Id', 'Name', 'Club', 'DoB', and 'Price'. The first row of the table contains the following data: '1', 'Haaland', 'Manchester City', '21/7/2000', and '170'. Below the table, there is a 'Footballer' form with five input fields: 'Id' (containing '2'), 'Name' (containing 'Foden'), 'DoB' (empty), 'Price' (containing '110'), and 'Club' (a dropdown menu with 'Manchester City' selected). At the bottom of the form, there are four buttons: 'Add', 'Delete', 'Update', and 'Cancel'.

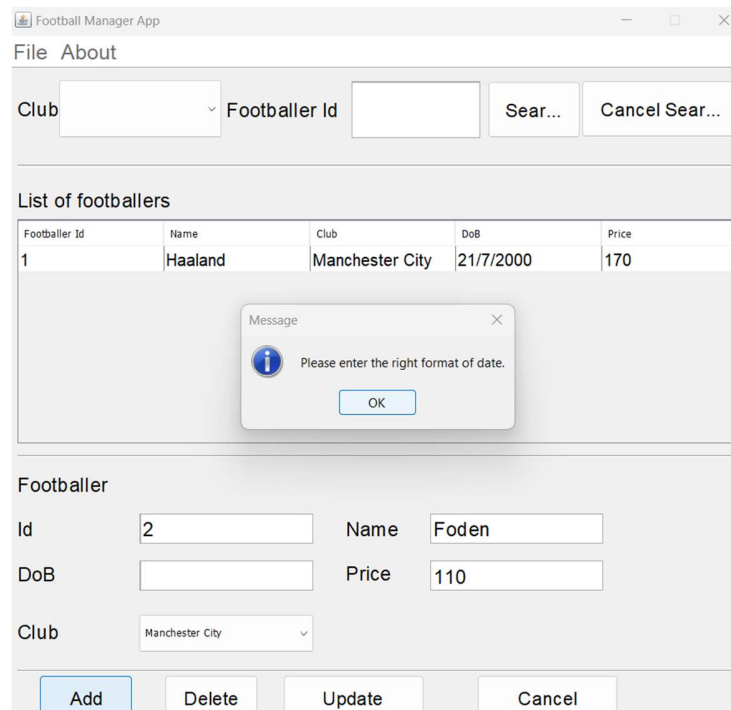
Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170

Footballer

Id: 2, Name: Foden, DoB: , Price: 110, Club: Manchester City

Buttons: Add, Delete, Update, Cancel

Figure 34 Test case 2 before proceeding



The screenshot shows the Football Manager App interface after proceeding with Test case 2. The interface is the same as in Figure 34, but with an error message dialog box displayed in the center. The dialog box has a title bar 'Message' and a close button 'X'. It contains an information icon and the text 'Please enter the right format of date.' Below the text is an 'OK' button. The 'Add' button in the 'Footballer' form is highlighted in blue.

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170

Message

Please enter the right format of date.

OK

Footballer

Id: 2, Name: Foden, DoB: , Price: 110, Club: Manchester City

Buttons: Add, Delete, Update, Cancel

Figure 35 Test case 2 after proceeding

Test 3:

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Foden	Arsenal	5/9/2001	90

Footballer

Id Name

DoB Price

Club

Add Delete Update Cancel

Figure 36 Test case 3 before proceeding

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90

Figure 37 Test case 4 after proceeding

Test 4:

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90
4	Kane	Tottenham Hotspur	28/7/1993	90

Footballer

Id Name

DoB Price

Club

Add Delete Update Cancel

Figure 38 Test case 4 before proceeding

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90

Footballer

Id Name

DoB Price

Club

Add Delete Update Cancel

Figure 39 Test case 4 after proceeding

Test 5:

Football Manager App

File About

Open Footballer Id Sear... Cancel Sear...

Save

Exit

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90
4	Kane	Tottenham Hotspur	28/7/1993	90
5	Salah	Liverpool	15/1/1992	80

Figure 40 Save file

File name: Save

Files of type: Cancel

Figure 41 Save file as data.txt

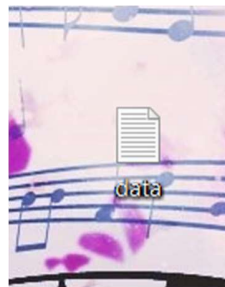


Figure 42 File data.txt

Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka			90
4	Kane		93	90
5	Salah		92	80

Exit Application

Are you sure ?

Yes No

Figure 43 Exit the application

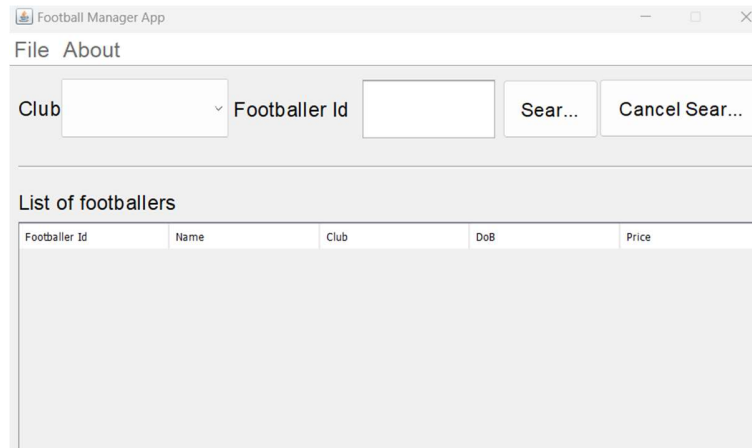


Figure 44 Open the app again, the table is reset

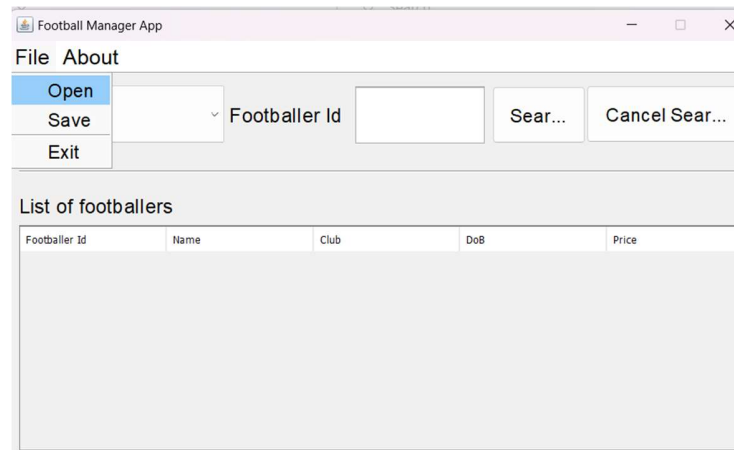
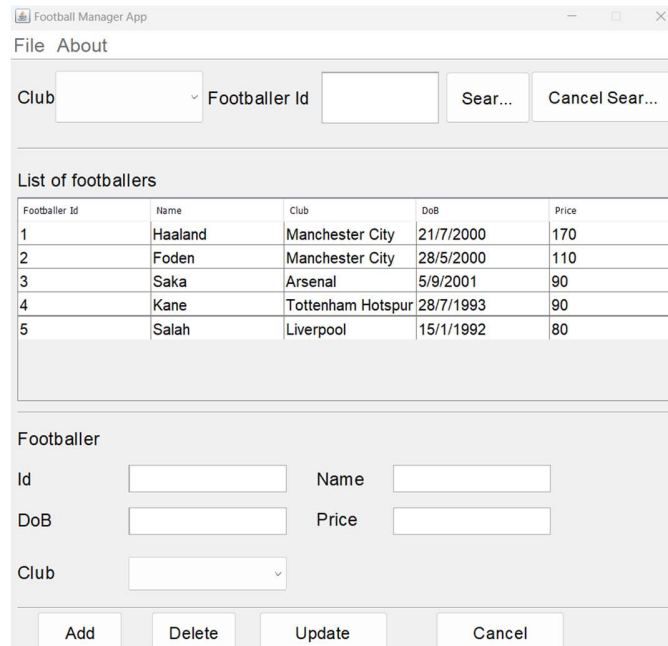


Figure 45 Open the file



Figure 46 Open the file data.txt



Football Manager App

File About

Club Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90
4	Kane	Tottenham Hotspur	28/7/1993	90
5	Salah	Liverpool	15/1/1992	80

Footballer

Id Name

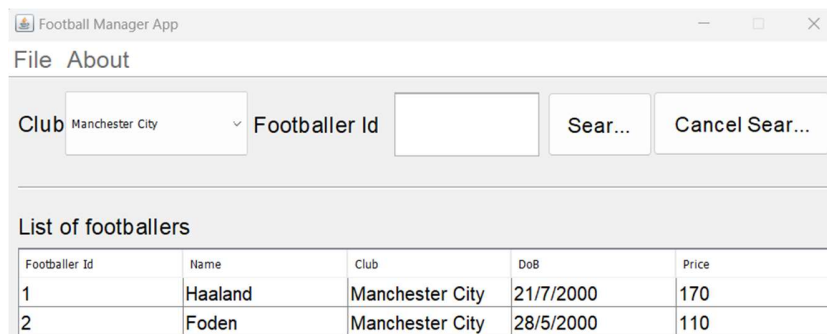
DoB Price

Club

Add Delete Update Cancel

Figure 47 The file after proceeding, Test case 5 Success

Some other functions running:



Football Manager App

File About

Club Manchester City Footballer Id Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110

Figure 48 Search players who are currently playing at Manchester City



Football Manager App

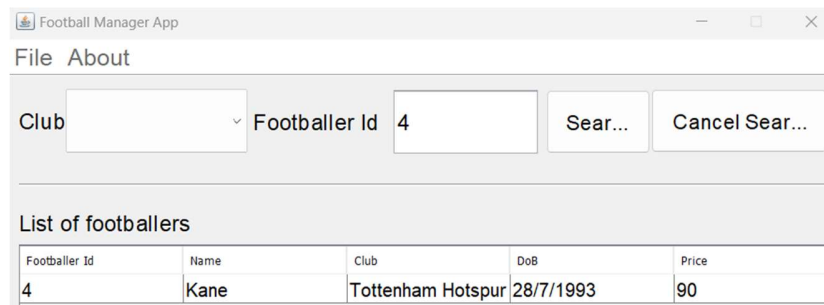
File About

Club Manchester City Footballer Id 2 Sear... Cancel Sear...

List of footballers

Footballer Id	Name	Club	DoB	Price
2	Foden	Manchester City	28/5/2000	110

Figure 49 Search players who Id is 2, and playing at Manchester City



Footballer Id	Name	Club	DoB	Price
4	Kane	Tottenham Hotspur	28/7/1993	90

Figure 50 Search player who Id is 4

7. Conclusion

Overall, I've created an application that allow the user to create, read, update, and delete the data of footballers. This app can also save and open .txt files. It can also search players based on playing club and id. I believe it can handle a variety of errors and change them into warning windows which help the user to realize the problem. However, there are still problems that I haven't handle properly yet which is show below. And yet, I confirms that the user can uses it as a tool for the 2023 window transfer as I mentioned at the introduction. Therefore, I will end this report here.

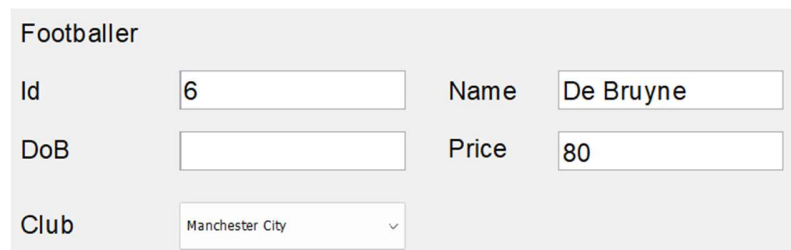


Figure 51 The error occurs when I enter the information with missing date

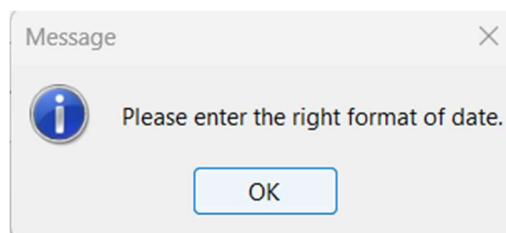


Figure 52 The box appears as normal

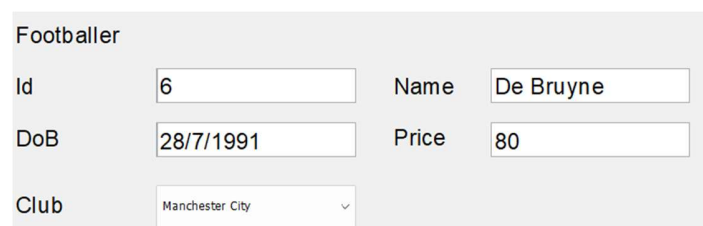


Figure 53 Then I add the detail

Club Footballer Id

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90
4	Kane	Tottenham Hotspur	28/7/1993	90
5	Salah	Liverpool	15/1/1992	80

Figure 54 But the data still not appears in the list

Club Footballer Id

List of footballers

Footballer Id	Name	Club	DoB	Price
1	Haaland	Manchester City	21/7/2000	170
2	Foden	Manchester City	28/5/2000	110
3	Saka	Arsenal	5/9/2001	90
4	Kane	Tottenham Hotspur	28/7/1993	90
5	Salah	Liverpool	15/1/1992	80
6	De Bruyne	Manchester City	28/7/1991	80

Figure 55 But then I clicked the search or cancel search button and the data appears