# Example Technical Solution

## This section needs to contain your annotated listing. It is important that you annotate the complex sections of code.

**You could also add a brief introduction which shows how your units/forms are linked.**

**A mini contents page could also be included which indicates where any complex sections of code can be found in your program.**

## Introduction

The project consists of 8 units and 7 forms and is linked to an Access database which consists of 4 tables.

The units are linked as shown below.

UMenu

UEnterScores

UNewSeason

UPlayer

Utilites

UFixtures

USetUpTeam

UAddTeam

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unit** | **Associated Form** | **Uses (Linked to)** | **Tables used** | **Description** |
| Umenu | FmMenu | UNewSeason, UEnterscores | Users, Player, Team, Fixtures | The main form of the project. |
| UNewSeason | FmNewSeason | UMenu, UFixtures, Uplayer, UAddTeam, USetUpTeam | Player, Users, Team | Displays new season menu |
| UFixtures | FmFixtures | UNewSeason | Users, Team, Fixtures | Works out and prints fixtures |
| UAddTeam | FmAddTeam | UNewSeason, Utilities | Team, Player | Allows new teams and players to be added. |
| USetUpTeam | FmSetUpTeam | UNewSeason, Utilities | Team, Player | Allows teams to be registered for new season |
| UEnterScores | FmEnterScores | UMenu, Utilities | Users, Fxitures, Team, Player | Allows weekly scores to be entered |
| Utilities | - | - | - | Library of general routines |

### Complex Procedures

|  |  |  |
| --- | --- | --- |
| Identifier | Description | Page |
| Procedure FormCreate | Checks if the procedure exists - if it doesn't it creates it. | A2 |
| Procedure BtnPrintClick | Prints out the team sheets and, if it is not the first week, the league table including details of prize winners. | A4 |
| Procedure FindPrizeWinners | Searches the player table to find the individual prize winners | A4 |
| Procedure PrintTeamSheets | Prints out the team sheets for each match. | A5 |
| Procedure PrintTeam(Team, Position) | Prints out the particular team and players with their handicaps in a grid at the given position on the sheet. | A5 |
| Procedure PrintLeagueStandings | Prints out the league table and prize winners | A7 |
| Procedure CbxTeamNameClick | Displays details of the team selected and adds existing players' details to GridOld. | A18 |
| Procedure GridOldSelectCell | Transfers the player selected to GridOld | A18 |
| Function PlayerOK(r:integer):Boolean | Returns true if the player selected is not already in GridNew | A19 |
| Procedure BtnOKClick | Used to validate then store edited team details. Also finds and displays current players. | A19 |
| Procedure ProduceFixtures | Works out and stores the fixtures for the season | A24 |
| Procedure BtnPrintClick | Prints the fixture list | A26 |
| Procedure FormActivate | Finds the fixtures for this week and displays details for the first fixture. | A28 |
| Procedure DisplayTeams(Team1,Team2) | Displays the names of the teams for the fixture, set up the grids and gets the players for the teams. | A28 |
| Procedure GetPlayers(Team, Cbx) | Finds the players for the particular team and loads them into the combo box cbx. | A29 |
| Procedure BtnCalculateClick | Caluclate the scores and the points awarded to each team. | A31 |
| Procedure GetPoints(score1, score2, pts1, pts2) | Calculates the points awarded (pts1, pts2) for the scores score1, score2 | A32 |
| Procedure AddUp(grid) | Adds up the score in the grid. | A33 |
| Function NewHandicap(TotalPinfall, NoOfGames):integer | Calculates a player's new handicap based on the TotalPinfall and the NoOfGames. | A33 |
| Procedure BtnConfirmClic | Updates the team and players tables with the new details and gets details for next match. | A34 |
| Procedure SaveTeamScores(grid1, grid2, Pts, Team) | Updates the team's record in the team table | A34 |
| Procedure SavePlayersScores(Grid) | Updates the players' records. | A34 |
| Unit - Utilities | Contains validation routines | A35 |

## 

## Appendix - Annotated listings of program code

I have not listed the entire program here - just some sections to show the types of comment that you should include.

This is an old program - some sections need improvement as there are better ways of doing things.

**Unit UMenu**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls, jpeg, DB, ADODB, Grids, DBGrids,printers, Utilities, ComObj;

type

TFmMenu = class(TForm)

BtnExit: TButton;

GbxLogOn: TGroupBox;

BtnNewSeason: TButton;

BtnScores: TButton;

EdUserNAme: TLabeledEdit;

EdPassword: TLabeledEdit;

BtnLogOn: TButton;

BtnPrint: TButton;

Label1: TLabel;

Image1: TImage;

ADOUsers: TADOTable;

EdConfirm: TLabeledEdit;

BtnSetUpPassword: TButton;

AdoPlayer: TADOTable;

ADOTeam: TADOTable;

AdoFixtures: TADOTable;

ADOQLeague: TADOQuery;

AdoQFindThisWeeksFixtures: TADOQuery;

ADOQFindPlayers: TADOQuery;

procedure BtnNewSeasonClick(Sender: TObject);

procedure BtnExitClick(Sender: TObject);

procedure BtnScoresClick(Sender: TObject);

procedure BtnLogOnClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure BtnSetUpPasswordClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure BtnPrintClick(Sender: TObject);

Procedure PrintLeagueStandings;

Procedure PrintTeamSheets;

Procedure PrintTeam(Team:string; position:integer);

Procedure FindPrizeWinners;

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmMenu: TFmMenu;

PrizeScore:array [1..4] of integer; PrizeName:array [1..4] of string;

implementation

uses UNewSeason, UEnterScores;

{$R \*.dfm}

procedure TFmMenu.BtnNewSeasonClick(Sender: TObject);

begin

FmMenu.Hide;

FmNewSeason.show;

end;

procedure TFmMenu.BtnExitClick(Sender: TObject);

begin

close;

end;

procedure TFmMenu.BtnScoresClick(Sender: TObject);

begin

fmmenu.Hide;

fmenterscores.show;

end;

procedure TFmMenu.BtnLogOnClick(Sender: TObject);

begin

if (EdPassword.text='') or (EdUserName.text='') then

showmessage('Please enter a username and password.')

else

begin **//If user name and password found**

if adousers.Locate('Username;Password',vararrayof ([EdUserName.Text,EdPassword.Text]),[]) then

GbxLogOn.Hide //show menu

else

showmessage('Invalid username or password');

end;

end;

Procedure TFmMenu.FormCreate(Sender: TObject);

Var cat:OLEVariant;

begin

Cat:=CreateOleObject('ADOX.Catalog');

if not FileExists('PhoenixLeague.accdb') then **//If the database doesn't exist, create it**

begin

cat.create('Provider=Microsoft.ACE.OLEDB.12.0; Data Source= PhoenixLeague.accdb');

ADOCommand1.ConnectionString:=ConnStr;

ADOCommand1.CommandText:='CREATE TABLE Team(TeamName VARCHAR(15), '+

'Captain VARCHAR(30), ContactNo CHAR(11), Email VARCHAR(30), Played INTEGER, '+

'Points FLOAT, TotalPinFall INTEGER, TotalPinfallAgainst INTEGER, HighestGame INTEGER,'+ 'HighestSeries INTEGER, NoOfPlayers INTEGER, PRIMARY KEY (TeamName))';

ADOCommand1.execute;

ADOCommand1.CommandText:='CREATE TABLE Player(FullName VARCHAR(30), ID INTEGER,'+ 'TeamName VARCHAR(15), GamesPlayed INTEGER, Handicap INTEGER, TotalPinfall INTEGER,'+ 'HighestHcpGame INTEGER, HighestGame INTEGER, HighestHcpSeries INTEGER, '+

'HighestSeries INTEGER, FOREIGN KEY (TeamName) REFERENCES Team(TeamName), '+

'PRIMARY KEY (ID))';

ADOCommand1.execute;

ADOCommand1.CommandText:='CREATE TABLE Fixtures(WeekNo INTEGER, '+

'Team1 VARCHAR(15), Team2 VARCHAR(15), Played BIT, LaneNo INTEGER, '+

'FOREIGN KEY (Team1) REFERENCES Team(TeamName), '+

'FOREIGN KEY (Team2) REFERENCES Team(TeamName), PRIMARY KEY (WeekNo, Team1))';

ADOCommand1.execute;

ADOCommand1.CommandText:='CREATE TABLE Users(UserName VARCHAR(10), '+

'Password VARCHAR(10), WeekNo INTEGER, WeeksToBePlayed INTEGER, '+

'PRIMARY KEY (UserName))';

ADOCommand1.execute;

**//Add user initial user details**

ADOCommand1.CommandText:='INSERT INTO Users VALUES ("Admin", "Tenp1n", 0, 0)';

ADOCommand1.execute;

end;

end;

procedure TFmMenu.FormActivate(Sender: TObject);

begin

adoUsers.ConnectionString:=ConnStr;

adoUsers.TableName:='Users';

adoUsers.open;

adoTeam.ConnectionString:=ConnStr;

adoTeam.TableName:='Team';

adoTeam.open;

if (adoTeam.recordcount<6) then **//If less than the minimum number of teams**

begin **//disable enter scores and print button**

BtnScores.Enabled:=false;

BtnPrint.Enabled:=false;

end

else

begin **//If more than the minimum number of teams**

BtnScores.Enabled:=true; **//Enable the enter scores and print buttons**

BtnPrint.Enabled:=true;

end;

adoteam.close;

end;

procedure TFmMenu.BtnPrintClick(Sender: TObject);

begin

adoFixtures.ConnectionString:=ConnStr;

adoFixtures.TableName:='Fixtures';

adoFixtures.Open;

If adofixtures.recordcount=0 then **//If the fixtures have not been set up**

showmessage('Can''t print team sheets until fixtures have been set up.')

else

begin

PrintTeamSheets;

FindPrizeWinners;

If adoUsers['weekno']>0 then **//If at least one week has been played**

PrintLeagueStandings; **//Print out league positions**

Showmessage('Team sheets printed');

end;

adoFixtures.Close;

end;

Procedure TfmMenu.FindPrizeWinners;

var player:integer;

begin

**//Find prize winners**

adoPlayer.ConnectionString:=ConnStr;

adoPlayer.TableName:='Players';

adoPlayer.Open;

adoPlayer.First; **//Move to first player record**

for player:=1 to adoPlayer.recordcount do

begin

**//Check if a prize winner for highest series**

if adoPlayer['HighestSeries']>PrizeScore[1] then

begin

PrizeScore[1]:=adoPlayer['HighestSeries'];

PrizeName[1]:=adoPlayer['FullName'];

end

else

begin

If adoPlayer['HighestSeries']=PrizeScore[1] then

begin **//If score matches highest add name to list.**

PrizeName[1]:=PrizeName[1]+' & '+adoPlayer['FullName'];

end

else **//Players cannot win both scratch and handicap prizes**

begin

if adoPlayer['HighestHcpSeries']>PrizeScore[2] then

begin

PrizeScore[2]:=adoPlayer['HighestHcpSeries'];

PrizeName[2]:=adoPlayer['FullName'];

end

else

begin **//If score matches highest add name to list.**

If adoPlayer['HighestHcpSeries']=PrizeScore[2] then

begin

PrizeName[2]:=PrizeName[2]+' & '+adoPlayer['FullName'];

end

end;

end;

end;

**//Check if prize winner for highest game**

if adoPlayer['HighestGame']>PrizeScore[3] then

begin

PrizeScore[3]:=adoPlayer['HighestGame'];

PrizeName[3]:=adoPlayer['FullName'];

end

else

begin **//If score matches highest add name to list.**

If adoPlayer['HighestGame']=PrizeScore[3] then

begin

PrizeName[3]:=PrizeName[3]+' & '+adoPlayer['FullName'];

end

else **//Players cannot win both scratch and handicap prizes**

begin

if adoPlayer['HighestHcpGame']>PrizeScore[4] then

begin

PrizeScore[4]:=adoPlayer['HighestHcpGame'];

PrizeName[4]:=adoPlayer['FullName'];

end

else

begin **//If score matches highest add name to list.**

If adoPlayer['HighestHcpGame']=PrizeScore[4] then

begin

PrizeName[4]:=PrizeName[4]+' & '+adoPlayer['FullName'];

end

end;

end;

end;

adoPlayer.Next; **//Move to next player record**

end;

adoPlayers.close;

end;

Procedure TFmMenu.PrintTeamSheets;

var i, dpmm, match:integer;

begin

**//clear prize arrays**

for i:=1 to 4 do

PrizeScore[i]:=0;

**//Find week number**

adoUsers.Open;

weekNo:=adoUsers['WeekNo']+1;

adoQFindThisWeeksFixtures.ConnectionString:=ConnStr;

adoQFindThisWeeksFixtures.SQL.Clear; **//Query to find this week's fixtures**

adoQFindThisWeeksFixtures.SQL.Add('SELECT team1, team2, LaneNos');

adoQFindThisWeeksFixtures.SQL.Add('FROM Fixtures');

adoQFindThisWeeksFixtures.SQL.Add('WHERE WeekNo='+IntToStr(WeekNo));

adoQFindThisWeeksFixtures.SQL.Add('ORDER BY LaneNos');

adoQFindThisWeeksFixtures.Open;

adoQFindThisWeeksFixtures.First;  **//Move to first fixture**

with printer do **//set up printer**

begin

with canvas do

begin

dpmm:=pagewidth div 210;

For match:=1 to adoQFindThisWeeksFixtures.RecordCount do **//For each match**

begin

begindoc; **//Each match is on a separate sheet of paper.**

font.Size:=14;

font.Name:='Georgia';

font.Style:=[fsbold];

textout(40\*dpmm, 20\*dpmm, 'Pioneer League - Week Number :'+IntToStr(WeekNo));

font.Size:=12;

textout(15\*dpmm, 35\*dpmm, 'Lanes : '+adoQFindThisWeeksFixtures['LaneNos']);

PrintTeam(adoQFindThisWeeksFixtures['team1'], 50); //Print out first team's details

PrintTeam(adoQFindThisWeeksFixtures['team2'], 180); //Print out second team's details

enddoc;

adoQFindThisWeeksFixtures.Next; **//Move to next fixture**

end;

end;

adoQFindThisWeeksFixtures.close;

end;

end;

Procedure TFmMenu.PrintTeam(Team:string; position:integer);

**//Find and print the players in the team**

var dpmm, ypos, player:integer; Hcp:string;

begin

**//Run query to find team's players**

adoQfindPlayers.ConnectionString:=ConnStr;

adoQfindPlayers.SQL.Clear;

adoQfindPlayers.SQL.Add('SELECT FullName, Handicap');

adoQfindPlayers.SQL.Add('FROM Player');

adoQfindPlayers.SQL.Add('WHERE TeamName="'+Team+'"');

adoQFindPlayers.Open;

adoQFindPlayers.First; **//Moveto first player**

**//Print out details**

with printer do

begin

with canvas do

begin

dpmm:=pagewidth div 210;

**//Print team name**

font.Size:=12;

font.Style:=[fsbold];

TextOut(15\*dpmm, (position-10)\*dpmm, 'Team: '+team);

TextOut(115\*dpmm, (position-10)\*dpmm, 'Won: Lost: ');

**//Draw outside box**

pen.Width:=3;

Rectangle(10\*dpmm,(position-3)\*dpmm, 180\*dpmm,(90+position+8)\*dpmm);

**//Draw horizontal lines**

pen.Width:=1;

For ypos:=1 to 10 do

begin

if ypos = 8 then

pen.Width:= 3

else

pen.Width:=1;

MoveTo(10\*dpmm,(ypos\*9+position-2)\*dpmm); LineTo(180\*dpmm, (ypos\*9+position-2)\*dpmm);

end;

**//Draw vertical grid lines**

Ypos:=position;

MoveTo(28\*dpmm,(ypos-2)\*dpmm); LineTo(28\*dpmm,(90+position+8)\*dpmm);

MoveTo(83\*dpmm,(ypos-2)\*dpmm); LineTo(83\*dpmm,(90+position+8)\*dpmm);

MoveTo(108\*dpmm,(ypos-2)\*dpmm); LineTo(108\*dpmm,(90+position+8)\*dpmm);

MoveTo(133\*dpmm,(ypos-2)\*dpmm); LineTo(133\*dpmm,(90+position+8)\*dpmm);

MoveTo(158\*dpmm,(ypos-2)\*dpmm); LineTo(158\*dpmm,(90+position+8)\*dpmm);

**//Print out captions**

font.Size:=10;

font.Style:=[];

TextOut(13\*dpmm, position\*dpmm, 'HCP');

TextOut(30\*dpmm, position\*dpmm, 'Player');

TextOut(85\*dpmm, position\*dpmm, 'Game 1');

TextOut(110\*dpmm, position\*dpmm, 'Game 2');

TextOut(135\*dpmm, position\*dpmm, 'Game 3');

TextOut(160\*dpmm, position\*dpmm, 'Series');

TextOut(30\*dpmm,(position+72)\*dpmm,'Team Total');

TextOut(30\*dpmm,(position+81)\*dpmm,'Team HCP');

TextOut(30\*dpmm,(position+90)\*dpmm,'Total + HCP');

**//Print out players' name and handicaps**

for player:=1 to adoQFindPlayers.recordcount do

begin

ypos:=ypos+9;

If adoQFindPlayers['Played']=0 then **//If player hasn't played**

Hcp:='-' **//Display - for handicap to indicate not set**

else

Hcp:=IntToStr(adoQFindPlayers['Handicap']);

**//print out player's details.**

textout(15\*dpmm, ypos\*dpmm, Hcp);

textout(30\*dpmm, ypos\*dpmm, adoQFindPlayers['FullName']);

AdoQfindPlayers.Next; **//Move to next player**

end;

end;

end;

end;

Procedure TFmMenu.PrintLeagueStandings;

var dpmm, ypos, team, TeamHighGame, TeamHighSeries:integer; TeamHighGameName, TeamHighSeriesName:string;

begin

adoQleague.ConnectionString:=ConnStr; **//Query to sort teams into league order**

adoQLeague.SQL.Clear;

adoQLeague.SQL.Add('SELECT TeamName, Played, Points, TotalPinFall, TotalPinsAgainst, HighestGame, HighestSeries');

adoQLeague.SQL.Add('FROM Team');

adoQLeague.SQL.Add('ORDER BY Points DESC, TotalPinfall DESC');

adoQleague.Open;

with printer do

begin

with canvas do

begin

dpmm:=pagewidth div 210;

begindoc; **//Start of document**

font.Size:=18;

font.Name:='Georgia';

font.Style:=[fsbold];

textout(40\*dpmm, 20\*dpmm, 'The Pioneer League - League Standings');

font.Size:=10; **//Print table headings**

TextOut(15\*dpmm, 40\*dpmm, 'Team Name');

TextOut(55\*dpmm, 40\*dpmm, 'Points');

TextOut(75\*dpmm, 40\*dpmm, 'Played');

TextOut(95\*dpmm, 40\*dpmm, 'Pins For');

TextOut(115\*dpmm, 40\*dpmm, 'Against');

TextOut(135\*dpmm, 40\*dpmm, 'High Game');

TextOut(160\*dpmm, 40\*dpmm, 'High Series');

font.Style:=[];

adoQleague.First;  **//Move to first team record**

TeamHighGame:=adoQLeague['HighestGame']; **//Set team prizes to first team's scores**

TeamHighSeriesName:=adoQLeague['TeamName'];

TeamHighGameName:=adoQLeague['TeamName'];

TeamHighSeries:=adoQLeague['HighestSeries'];

For team:=1 to adoQLeague.recordcount do **//For each team**

begin

**//Print out team's results**

TextOut(15\*dpmm, (team\*7+40)\*dpmm, adoQleague['TeamName']);

TextOut(60\*dpmm, (team\*7+40)\*dpmm, adoQleague['Points']);

TextOut(78\*dpmm, (team\*7+40)\*dpmm, adoQleague['Played']);

TextOut(98\*dpmm, (team\*7+40)\*dpmm, adoQleague['TotalPinfall']);

TextOut(117\*dpmm, (team\*7+40)\*dpmm, adoQleague['TotalPinsAgainst']);

TextOut(140\*dpmm, (team\*7+40)\*dpmm, adoQleague['HighestGame']);

TextOut(160\*dpmm, (team\*7+40)\*dpmm, adoQleague['HighestSeries']);

**//Check if high game winner**

If (AdoQLeague['HighestGame']>TeamHighGame) then

begin

TeamHighGame:= AdoQLeague['HighestGame'];

TeamHighGameName:=adoQLeague['TeamName'];

end

else

begin

if (AdoQLeague['HighestGame']=TeamHighGame) then **//If Game = current high score**

begin **//Add team name to list**

TeamHighGameName:=TeamHighGameName+' & '+ adoQLeague['TeamName'];

end;

end;

**//Check if high series winner**

If (adoQLeague['HighestSeries']>TeamHighSeries) then

begin

TeamHighSeriesName:=adoQLeague['TeamName'];

TeamHighSeries:=adoQLeague['HighestSeries'];

end

else

begin

if (adoQLeague['HighestSeries']=TeamHighSeries) then **//If series = current high score**

begin **//Add team name to list**

TeamHighSeriesName:=TeamHighSeriesName+' & '+adoQLeague['TeamName'];

end;

end;

adoQleague.Next; **//Move to next team record**

end;

font.Style:=[fsbold];

font.Size:=11; **//Print out prize winners.**

TextOut(15\*dpmm, 200\*dpmm, 'Team Prizes');

font.Style:=[];

TextOut(15\*dpmm, 207\*dpmm, 'Team High Game: '+TeamHighGameName+' Score: '+IntToStr(TeamHighGame));

TextOut(100\*dpmm, 207\*dpmm, 'Team High series: '+TeamHighSeriesName+' Score: '+IntToStr(TeamHighSeries));

font.Style:=[fsbold];

TextOut(15\*dpmm, 220\*dpmm, 'Individual Prizes');

font.Style:=[];

TextOut(15\*dpmm, 227\*dpmm, 'Scratch Series: '+PrizeName[1]+' Score: '+IntToStr(PrizeScore[1]));

TextOut(15\*dpmm, 234\*dpmm, 'Handicap Series: '+PrizeName[2]+' Score: '+IntToStr(PrizeScore[2]));

TextOut(15\*dpmm, 241\*dpmm, 'Scratch Game: '+PrizeName[3]+' Score: '+IntToStr(PrizeScore[3]));

TextOut(15\*dpmm, 248\*dpmm, 'Handicap Game: '+PrizeName[4]+' Score: '+IntToStr(PrizeScore[4]));

enddoc;

end;

end;

adoQleague.close;

end;

end.

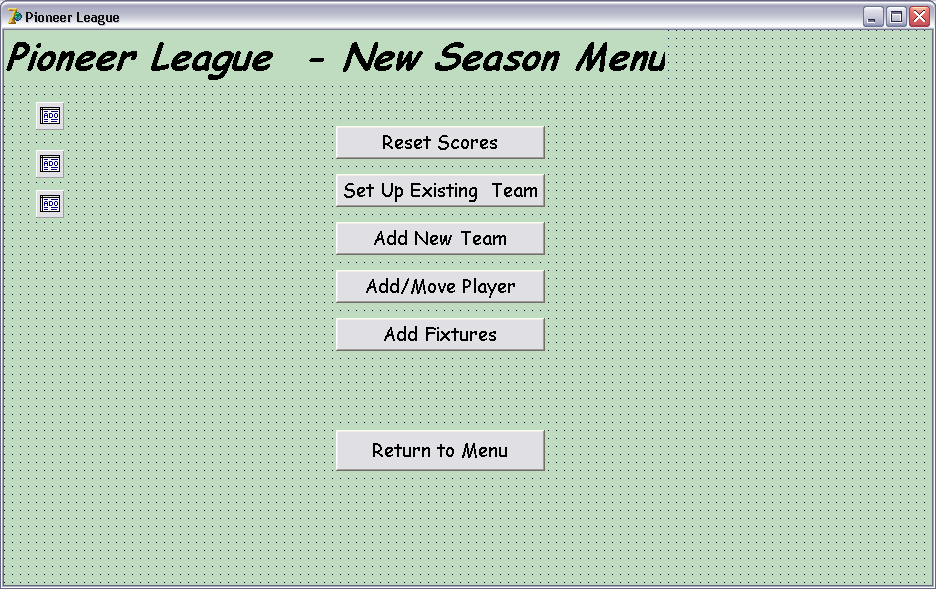
**Form FmNewSeason**

AdoTables

AdoUsers

AdoPlayers

AdoTeams



Buttons

BtnResetScores

BtnSetUpTeam

BtnAddNewteam

BtnPlayer

BtnAddFixtures

BtnReturn

**unit UNewSeason;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, DB, ADODB;

type

TFmNewSeason = class(TForm)

BtnAddFixtures: TButton;

BtnAddNewTeam: TButton;

BtnSetUpTeam: TButton;

BtnReturn: TButton;

BtnPlayer: TButton;

BtnResetScores: TButton;

ADOPlayer: TADOTable;

AdoScore: TADOTable;

ADOUser: TADOTable;

Label1: TLabel;

ADOTeam: TADOTable;

procedure BtnAddNewTeamClick(Sender: TObject);

procedure BtnReturnClick(Sender: TObject);

procedure BtnSetUpTeamClick(Sender: TObject);

procedure BtnAddFixturesClick(Sender: TObject);

procedure BtnPlayerClick(Sender: TObject);

procedure BtnResetScoresClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmNewSeason: TFmNewSeason;

implementation

uses UAddTeam, Ufixtures, USetUpTeam, UMenu, Uplayer;

{$R \*.dfm}

procedure TFmNewSeason.BtnAddNewTeamClick(Sender: TObject);

begin

FmNewSeason.Hide;

FmAddTeam.show;

end;

procedure TFmNewSeason.BtnReturnClick(Sender: TObject);

begin

FmMenu.show;

Close;

end;

procedure TFmNewSeason.BtnSetUpTeamClick(Sender: TObject);

begin

FmNewSeason.Hide;

FmSetUpTeam.show;

end;

procedure TFmNewSeason.BtnAddFixturesClick(Sender: TObject);

begin

FmNewSeason.Hide;

FmFixtures.show;

end;

procedure TFmNewSeason.BtnPlayerClick(Sender: TObject);

begin

FmPlayer.show;

FmNewSeason.Hide;

end;

procedure TFmNewSeason.BtnResetScoresClick(Sender: TObject);

var i:integer;

begin

AdoUser.First;

If adoUser['WeekNo']>0 then **//If the season has already started**

Showmessage('You cannot reset the scores mid-season.')

else

begin

adoPlayer.Open;

if adoplayer.recordcount>0 then **//If there are players stored**

begin

adoplayer.First; **//Move to first player record**

while not adoplayer.Eof do **//While not end of players table**

begin

adoplayer.edit; **//Reset player's scores**

adoplayer['TotalPinFall']:=0;

adoplayer['GamesPlayed']:=0;

adoplayer['HighestHcpGame']:=0;

adoplayer['HighestGame']:=0;

adoplayer['HighestHcpSeries']:=0;

adoplayer['HighestSeries']:=0;

adoplayer.post;

adoplayer.next; **//Move to next player record.**

end;

showmessage('done');

end;

end;

end;

procedure TFmNewSeason.FormActivate(Sender: TObject);

begin

AdoUser.ConnectionString:=ConnStr;

AdoUser.TableName:='Users';

AdoUser.Open;

AdoPlayer.ConnectionString:=ConnStr;

AdoPlayer.TableName:='Players';

adoplayer.Open;

AdoTeam.ConnectionString:=ConnStr;

AdoTeam.TeamName:='Team';

adoteam.Open;

If (adoteam.RecordCount<6) then **//If the minimum number of teams has not been added.**

begin

BtnAddFixtures.Enabled:=false; **//Disable all buttons except add teams**

BtnResetScores.Enabled:=false;

BtnExistingTeam.Enabled:=false;

BtnPlayer.Enabled:=false;

end

else

begin

BtnAddFixtures.Enabled:=true;  **//Enable all options**

BtnResetScores.Enabled:=true;

BtnExistingTeam.Enabled:=true;

BtnPlayer.Enabled:=true;

end;

end;

end.

**Form FmAddTeam**

AdoTables:

AdoTeam

AdoPlayer

Edit Boxes:

EdTeamName

EdCaptain

EdContact

EdEMail

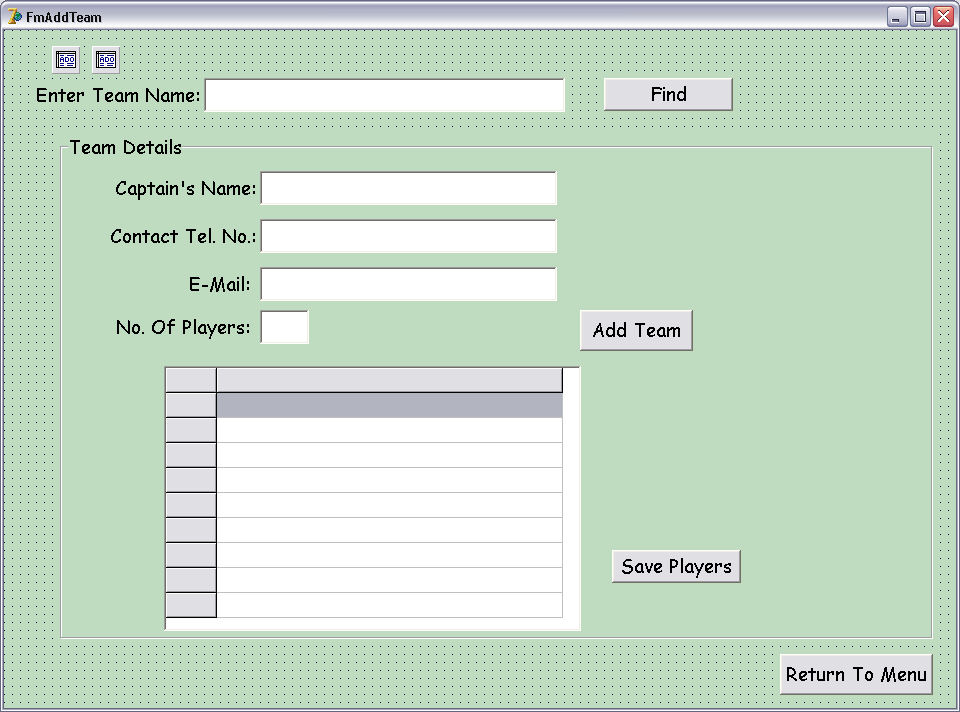
EdNoOfPlayers

StringGrid:

Grid

Group Box:

GbxTeam



Buttons

BtnFind

BtnAddTeam

BtnSave

BtnReturn

**unit UAddTeam;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DB, ADODB, StdCtrls, ExtCtrls, Grids, utilities;

type

TFmAddTeam = class(TForm)

EdTeamName: TLabeledEdit;

BtnFind: TButton;

ADOTeam: TADOTable;

GbxTeam: TGroupBox;

EdCaptain: TLabeledEdit;

EdContact: TLabeledEdit;

EdEMail: TLabeledEdit;

BtnAddTeam: TButton;

Grid: TStringGrid;

BtnSave: TButton;

ADOPlayer: TADOTable;

EdNoPlayers: TLabeledEdit;

BtnReturn: TButton;

procedure FormActivate(Sender: TObject);

procedure BtnFindClick(Sender: TObject);

procedure BtnAddTeamClick(Sender: TObject);

procedure BtnSaveClick(Sender: TObject);

Procedure ClearBoxes;

procedure BtnReturnClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmAddTeam: TFmAddTeam;

LastId:integer;

implementation

uses UMenu;

{$R \*.dfm}

procedure TFmAddTeam.FormActivate(Sender: TObject);

var i:integer;

begin

AdoPlayer.ConnectionString:=ConnStr;

AdoPlayer.TableName:='Players';

adoplayer.Open;

AdoTeam.ConnectionString:=ConnStr;

AdoTeam.TeamName:='Team';

adoTeam.Open;

if adoPlayer.recordcount=0 then **//If there are no players in the table**

LastId:=0 **//LastId is 0**

else

begin **//If there are players in the table**

adoplayer.Sort:='ID'; **//Sort the players into ID order**

adoplayer.last; **//Move to last player**

LastId:=adoPlayer['ID']; **//LastId is set to ID of last player.**

end;

GbxTeam.hide; **//Hide Team group box**

Grid.Hide; **//Hide players' grid**

**//Put captions in the grid**

Grid.Cells[1,0]:='Player Name';

For i:=1 to 8 do

Grid.Cells[0,i]:=IntToStr(i);

end;

procedure TFmAddTeam.BtnFindClick(Sender: TObject);

begin

If EdTeamName.Text <>'' then **//If a name has been entered**

begin

If adoTeam.Locate('TeamName', EdTeamName.Text,[]) then  **//If name found in database**

showmessage('Team name already exists. Either use transfer or choose another name')

else

begin

GbxTeam.show; **//Show Team group box**

end;

end

else

showmessage('Please enter a team name');

end;

procedure TFmAddTeam.BtnAddTeamClick(Sender: TObject);

begin

if EdCaptain.text<>'' Then **//If the captain's name has been entered**

begin

If (EdContact.text<>'') Or (EdEMail.text<>'') Then **//If a contact no or an e-mail address has been entered**

begin

if ValidInteger(EdNoPlayers.Text,3,7) then **//If the number of players is valid**

begin

adoTeam.Append; **//Put team table in append mode**

adoTeam['TeamName']:=EdTeamName.Text;

adoTeam['Captain']:=EdCaptain.Text;

adoTeam['ContactNo']:=EdContact.Text;

adoTeam['EMail']:=EdEMail.Text;

adoTeam['Points']:=0;

adoTeam['TotalPinfall']:=0;

adoTeam['TotalPinsAgainst']:=0;

adoTeam['HighestGame']:=0;

adoTeam['HighestSeries']:=0;

adoTeam['NoOfPlayers']:=StrToInt(EdNoPlayers.Text);

adoTeam.Post; **//Add team record to table**

grid.Show; **// show players' grid**

grid.RowCount:=StrToInt(EdNoPlayers.Text)+1;

grid.Cells[1,1]:=EdCaptain.Text; **//Put captain's name in the players's grid**

BtnSave.enabled:=true; **//Enable save players button**

end

end

else

showmessage('You must enter either a contact number or an e-mail address.');

end

else

begin

showmessage('You must enter the captain''s name.');

EdCaptain.SetFocus;

end;

end;

procedure TFmAddTeam.BtnSaveClick(Sender: TObject);

var i:integer;

begin

If EmptyCell(grid,1,StrToInt(EdNoPlayers.Text),1,1) then **//if one of the cells in the grid has been left empty**

showmessage('Please enter all the players'' names')

else

begin

for i:=1 to StrToInt(EdNoPlayers.Text) do **//Save all the players**

begin

adoPlayer.Append;

adoPlayer['FullName']:=grid.Cells[1,i];

adoPlayer['TeamName']:=EdTeamName.Text;

adoPlayer['TotalPinfall']:=0;

adoPlayer['GamesPlayed']:=0;

adoPlayer['HighestHcpGame']:=0;

adoPlayer['HighestHcpSeries']:=0;

adoPlayer['TotalPinfall']:=0;

adoPlayer['HighestGame']:=0;

adoPlayer['HighestSeries']:=0;

adoPlayer['HighestHcpGame']:=0;

LastId:=LastId+1;

adoPlayer['ID']:=LastID;

adoPlayer.Post;

end;

Showmessage('Team and players'' details have been saved.');

ClearBoxes;

end;

end;

Procedure TfmAddTeam.ClearBoxes;

begin

edTeamName.clear;

EdContact.clear;

EdEMail.clear;

EdNoPlayers.clear;

clearGrid(grid,1,1);

GbxTeam.hide;

end;

procedure TFmAddTeam.BtnReturnClick(Sender: TObject);

begin

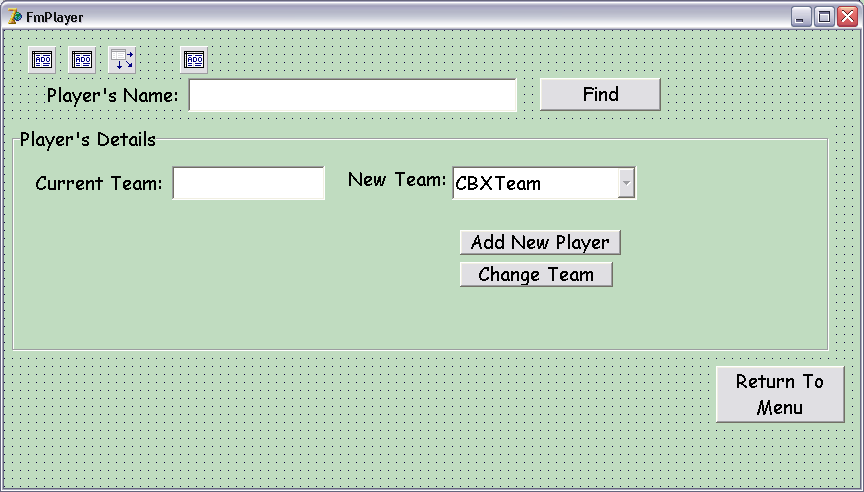
Fmmenu.show;

close;

end;

end.

**FmPlayer**



DataSource: DSTeam AdoTable: AdoUsers

AdoTables:

AdoTeam

AdoPlayer

Edit Boxes:

EdName

EdTeam

Group Box:

GbxDetails

Button:

BtnFind

DbLookUpComboBox:

CbxTeam - linked to DSTeam

Buttons:

BtnAddNew

BtnChangeTeam

BtnReturn

**unit Uplayer;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, DBCtrls, DB, ADODB;

type

TFmPlayer = class(TForm)

GbxDetails: TGroupBox;

EdName: TLabeledEdit;

BtnFind: TButton;

EdTeam: TLabeledEdit;

ADOPlayer: TADOTable;

ADOTeam: TADOTable;

DSTeam: TDataSource;

CBXTeam: TDBLookupComboBox;

Label1: TLabel;

BtnAddNew: TButton;

BtnSave: TButton;

BtnReturn: TButton;

ADOUsers: TADOTable;

procedure FormActivate(Sender: TObject);

procedure BtnFindClick(Sender: TObject);

procedure clear\_all;

procedure BtnAddNewClick(Sender: TObject);

procedure BtnReturnClick(Sender: TObject);

procedure BtnSaveClick(Sender: TObject);

Function PlayerSaved:Boolean;

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmPlayer: TFmPlayer;

implementation

uses UNewSeason;

{$R \*.dfm}

procedure TFmPlayer.FormActivate(Sender: TObject);

begin

AdoUser.ConnectionString:=ConnStr;

AdoUser.TableName:='Users';

AdoUser.Open;

AdoPlayer.ConnectionString:=ConnStr;

AdoPlayer.TableName:='Players';

adoplayer.Open;

AdoTeam.ConnectionString:=ConnStr;

AdoTeam.TeamName:='Team';

adoteam.Open;

clear\_all;

end;

procedure TFmPlayer.BtnFindClick(Sender: TObject);

begin

If EdName.text='' then **//If a player's name has not been entered**

showmessage('Please enter player''s name.')

else

begin

if adoPlayer.locate('fullname',edname.Text,[]) then **//If the player is found**

begin

showmessage('Player found. Select new team name then click on Change Team.');

EdTeam.text:=adoPlayer['TeamName'];

If adoPlayer['GamesPlayed']<> 0 then **//If player has already played this season, can't be transferred**

begin

showmessage('This player has already played for a team so can''t be transferred.');

end

else

begin

EdTeam.Show;

BtnSave.Show;

EdName.Show;

BtnAddNew.Hide;

GbxDetails.Show;

end;

end

else

begin

showmessage('Select team name then click on Add New.');

EdTeam.hide;

BtnSave.hide;

BtnAddNew.show;

GbxDetails.Show;

end;

end;

end;

procedure TFmPlayer.clear\_all;

begin

Gbxdetails.hide;

EdTeam.clear;

EdName.clear;

end;

procedure TFmPlayer.BtnAddNewClick(Sender: TObject);

var LastId:integer;

begin

**//Find ID of last player entered**

adoplayer.Sort:='ID';

adoplayer.last;

LastId:=adoPlayer['ID'];

adoPlayer.Append;

LastId:=LastId+1;

adoPlayer['ID']:=LastID;

adoPlayer['FullName']:=EdName.text;

adoPlayer['TotalPinfall']:=0; **//Set scores to 0 for new player**

adoPlayer['GamesPlayed']:=0;

adoPlayer['HighestHcpGame']:=0;

adoPlayer['HighestHcpSeries']:=0;

adoPlayer['TotalPinfall']:=0;

adoPlayer['HighestGame']:=0;

adoPlayer['HighestSeries']:=0;

adoPlayer['HighestHcpGame']:=0;

If playersaved then **//If player successfully saved**

begin

Showmessage('New player added.');

Clear\_All;

end;

end;

procedure TFmPlayer.BtnChangeTeamClick(Sender: TObject);

begin

adoPlayer.Edit;

If PlayerSaved then **//If player successfully edited**

begin

**//Subtract 1 from number of players in old team**

adoTeam.locate('TeamName',EdTeam.Text,[]);

adoTeam.Edit;

adoTeam['NoOfPlayers']:=adoTeam['NoOfPlayers']-1;

adoTeam.post;

Showmessage('Player transferred.');

Clear\_all;

end;

end;

Function TFmPlayer.PlayerSaved:Boolean;

begin

if adoTeam['NoOfPlayers']=7 then **//If team already has 7 players**

begin

showmessage('This team already has 7 players. Delete a player before adding a new one.');

PlayerSaved:=false;

end

else

begin

PlayerSaved:=true;

adoPlayer['TeamName']:=CbxTeam.Text;

adoPlayer.Post;

**//Add 1 to number of players in new team**

adoTeam.Edit;

adoTeam['NoOfPlayers']:=adoTeam['NoOfPlayers']+1;

adoTeam.post;

end;

end;

procedure TFmPlayer.BtnReturnClick(Sender: TObject);

begin

FmNewSeason.show;

close;

end;

end.

**FmSetUpTeam**

ADOTables:

ADOTeam

ADOPlayer

Data Source:

DSTeam

Edit Boxes:

EdContact

EdNoPlayers

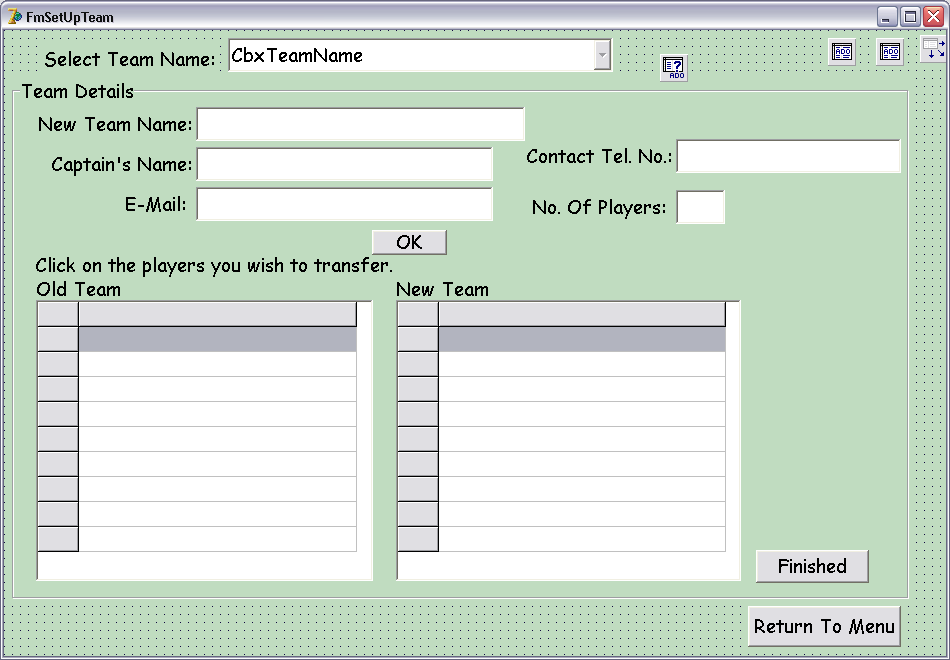
StringGrid:

GridNew

Buttons:

BtnFinished

BtnReturn



DBLookupComboBox:

CbxTeamName

linked to DSTeam

Edit Boxes:

EdNewTeam

EdCaptain

EdEMail

Button:

BtnOK

Group Box:

GbxDetails

StringGrid:

GridOld

AdoQuery: ADOFindPlayers

**unit USetUpTeam;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DB, ADODB, ComCtrls, StdCtrls, DBCtrls, Grids, ExtCtrls, utilities;

type

TFmSetUpTeam = class(TForm)

ADOTeam: TADOTable;

ADOPlayer: TADOTable;

DSTeam: TDataSource;

CbxTeamName: TDBLookupComboBox;

Label1: TLabel;

ADOFindPlayers: TADOQuery;

GbxTeam: TGroupBox;

EdCaptain: TLabeledEdit;

EdContact: TLabeledEdit;

EdEMail: TLabeledEdit;

EdNoPlayers: TLabeledEdit;

EdNewTeam: TLabeledEdit;

LblOld: TLabel;

GridOld: TStringGrid;

GridNew: TStringGrid;

LbLNew: TLabel;

LbLClick: TLabel;

BtnOK: TButton;

BtnFinished: TButton;

BtnReturn: TButton;

procedure FormActivate(Sender: TObject);

procedure CbxTeamNameClick(Sender: TObject);

procedure GridOldSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

Function PlayerOk(r:integer):boolean;

procedure BtnOKClick(Sender: TObject);

procedure BtnReturnClick(Sender: TObject);

procedure BtnFinishedClick(Sender: TObject);

procedure SavePlayerDetails;

Procedure ClearBoxes;

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmSetUpTeam: TFmSetUpTeam;

NoOfPlayers, LastID:integer;

implementation

uses UNewSeason;

{$R \*.dfm}

procedure TFmSetUpTeam.FormActivate(Sender: TObject);

begin

AdoPlayer.ConnectionString:=ConnStr;

AdoPlayer.TableName:='Players';

adoplayer.Open;

AdoTeam.ConnectionString:=ConnStr;

AdoTeam.TeamName:='Team';

adoteam.Open;

if adoPlayer.recordcount=0 then **//If there are no players currently in the table**

LastId:=0 **//Set LastId to 0**

else

begin

adoplayer.Sort:='ID'; **//Sort players into ID order**

adoplayer.last; **//Move to last player**

LastId:=adoPlayer['ID']; **//Set LastID to ID of the last player**

end;

ClearBoxes;

end;

procedure TFmSetUpTeam.CbxTeamNameClick(Sender: TObject);

var i:integer;

begin

EdNewTeam.Text:=adoTeam['TeamName'];

EdCaptain.Text:=adoTeam['Captain'];

EdContact.Text:=adoTeam['ContactNo'];

EdEmail.Text:=adoTeam['EMail'];

EdNoPlayers.text:=IntToStr(AdoTeam['NoOfPlayers']);

GbxTeam.Show;

showmessage('Make any required changes then click on the OK button.');

end;

procedure TFmSetUpTeam.GridOldSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

**//Transfers player clicked to the new grid.**

begin

If (acol=0) or (Arow=0) then **//Stops user clicking in first column of grid**

canselect:=false

else

begin

**//If the current number of players is less than the number entered**

if NoOfPlayers<StrToInt(EdNoPlayers.text) then

begin

If PlayerOk(Arow) Then **//If the player clicked is not already in the new grid**

begin

NoOfPlayers:=NoOfPlayers+1;

GridNew.Cells[1,NoOfPlayers]:=GridOld.Cells[1,Arow]; **//Copy ID and player name to new grid**

GridNew.Cells[0,NoOfPlayers]:=GridOld.Cells[0,Arow];

end;

end

else

showmessage('You already have '+ EdNoPlayers.text+' players - remove one before adding any more.');

end;

end;

Function TFmSetUpTeam.PlayerOk(r:integer):boolean;

var i:integer;

begin **//Check if selected player is already in the new grid**

PlayerOk:=true;

For i:=1 to NoOfPlayers do

begin

if GridOld.cells[1,r]=GridNew.cells[1,i] then

begin

showmessage('This name has already been added.');

PlayerOk:=false;

end;

end;

end;

procedure TFmSetUpTeam.BtnOKClick(Sender: TObject);

var i:integer;

begin

if EdNewTeam.text<>'' Then **//If a new team name has been entered**

begin

if EdCaptain.text<>'' Then **//If the captain's name has been entered**

begin

If (EdContact.text<>'') Or (EdEMail.text<>'') Then **//If contact no or e-mail address has been entered**

begin

if ValidInteger(EdNoPlayers.Text,3,7) then **//If the number of players is valid**

begin

adoTeam.Edit; **//Put team table in edit mode**

adoTeam['TeamName']:=EdNewTeam.Text;

adoTeam['Captain']:=EdCaptain.Text;

adoTeam['ContactNo']:=EdContact.Text;

adoTeam['EMail']:=EdEMail.Text;

adoTeam['NoOfPlayers']:=StrToInt(EdNoPlayers.Text);

adoTeam['Points']:=0;

adoTeam['TotalPinfall']:=0;

adoTeam['TotalPinsAgainst']:=0;

adoTeam['HighestGame']:=0;

adoTeam['HighestSeries']:=0;

adoTeam.Post;  **//Save changes to team record to table**

adoFindPLayers.Close; **//Run query to find players registered with this team**

adoFindPlayers.Parameters[0].value:=CbxTeamName.text;

adoFindPlayers.Open;

adoFindPlayers.First;

if adoFindPlayers.recordcount=0 then **//If no registered players have been found**

begin

showmessage('No players have been found for '+CbxTeamName.Text);

showmessage('Please add players in grid.');

GridNew.RowCount:=StrToInt(EdNoPlayers.text)+1;

GridNew.Show; LblNew.Show;

end

else

begin

gridOld.RowCount:=adoFindPlayers.recordcount+1;

For i:=1 to adoFindPlayers.RecordCount do **//Display current players in old grid**

begin

gridOld.Cells[1,i]:=adofindPlayers['FullName'];

gridOld.Cells[0,i]:=adoFindPlayers['ID'];

adoFindPlayers.Next;

end;

GridOld.Show;

GridNew.Show;

LblClick.Show;

LblOld.Show;

LblNew.Show;

GridNew.RowCount:=StrToInt(EdNoPlayers.text)+1; **//Set size of new grid to required size**

NoOfPlayers:=0;

end;

end

end

else

showmessage('You must enter either a contact number of an e-mail address.');

end

else

begin

showmessage('You must enter the captain''s name.');

EdCaptain.SetFocus;

end;

end

else

begin

showmessage('You cannot leave the new team name blank.');

EdNewTeam.setfocus;

end;

end;

procedure TFmSetUpTeam.BtnReturnClick(Sender: TObject);

begin

FmNewSeason.show;

Close;

end;

procedure TFmSetUpTeam.BtnFinishedClick(Sender: TObject);

var i, playersSaved:integer;

begin

If EmptyCell(GridNew,1,StrToInt(EdNoPlayers.text),1,1) then **//If a grid cell has been left empty**

showmessage('Please enter all players including new players.')

else

begin

PlayersSaved:=0;

for i:=1 to StrToInt(EdNoPlayers.text) do **//Save all the player's details**

begin

if gridNew.Cells[0,i]<>'-' then **//If player ID is known**

begin

adoPlayer.Locate('ID',StrToInt(gridNew.Cells[0,i]),[]); **//Find player**

adoPlayer.Edit; **//Save edited record**

savePlayerDetails;

adoPlayer.post;

PlayersSaved:=PlayersSaved+1;

end

else

begin

If adoPlayer.Locate('FullName',gridNew.Cells[1,i],[]) then **//If player name found in table**

begin

If messageDlg('A player with this name currently plays for '+adoPlayer['TeamName']+

' Do you want to transfer this player?',MtConfirmation,[MbYes,MbNo],0)=MrYes then

**//Confirm that this is the correct player**

begin

adoPlayer.Edit; **//Save edited record**

savePlayerDetails;

adoPlayer.post;

PlayersSaved:=PlayersSaved+1;

end

else

showmessage('Amend the name and click on finished again.');

end

else

begin

**//Confirm that this is a new player**

If messageDlg('New player found. Add?', mtconfirmation,[mbyes,mbno],0) =mryes then

begin

adoPlayer.Append;

adoPlayer['FullName']:=GridNew.cells[1,i];

LastId:=LastID+1; **//Set player ID to lastID +1**

adoPlayer['ID']:=LastID;

savePlayerDetails; **//Add new player's details to table**

adoPlayer.post;

PlayersSaved:=PlayersSaved+1;

end

else

showmessage('Amend the name and click on finished again.');

end;

end;

end;

If PlayersSaved=StrToInt(EdNoPlayers.text) then

begin

showmessage('Details transferred');

ClearBoxes;

end

end;

end;

procedure TFmSetUpTeam.SavePlayerDetails;

begin

adoPlayer['TeamName']:=EdNewTeam.Text;

adoPlayer['TotalPinFall']:=0;

adoPlayer['GamesPlayed']:=0;

adoPlayer['Handicap']:=0;

adoPlayer['HighestHcpGame']:=0;

adoPlayer['HighestHcpSeries']:=0;

adoPlayer['HighestGame']:=0;

adoPlayer['HighestSeries']:=0;

end;

Procedure TFmSetUpTeam.ClearBoxes;

var i:integer;

begin

ClearGrid(GridOld,1,0);

ClearGrid(GridNew,1,0);

GbxTeam.Hide;

GridOld.Hide;

GridNew.Hide;

LblClick.Hide;

LblOld.Hide;

LblNew.Hide;

gridold.cells[0,0]:='ID';

gridOld.Cells[1,0]:='Player Name';

gridNew.cells[0,0]:='ID';

gridNew.Cells[1,0]:='Player Name';

for i:=1 to 7 do

begin

gridOld.cells[0,i]:='-';

gridNew.cells[0,i]:='-';

end;

end;

end.

**FmFixtures**

**unit Ufixtures;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DB, ADODB, StdCtrls, Spin, Grids, DBGrids, printers;

type

TFmFixtures = class(TForm)

ADOFixtures: TADOTable;

ADOTeam: TADOTable;

BtnFind: TButton;

GbXTeams: TGroupBox;

AdoFindTeams: TADOQuery;

DSFixtures: TDataSource;

GbxFixtures: TGroupBox;

btnFixtures: TButton;

SpEdtimes: TSpinEdit;

LblTimes: TLabel;

BtnReturn: TButton;

ADOUser: TADOTable;

BtnPrint: TButton;

ADOQSortFixtures: TADOQuery;

TeamGrid: TStringGrid;

procedure BtnFindClick(Sender: TObject);

Procedure ProduceFixtures;

procedure btnFixturesClick(Sender: TObject);

procedure SpEdtimesChange(Sender: TObject);

procedure DeleteOldTeams;

procedure FormActivate(Sender: TObject);

procedure BtnReturnClick(Sender: TObject);

procedure BtnPrintClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmFixtures: TFmFixtures;

NoOfTeams, TimesRound, NoOfLanes:integer;

Fixture: array [1..23, 1..12, 1..12] of integer;

implementation

uses UNewSeason;

{$R \*.dfm}

procedure TFmFixtures.FormActivate(Sender: TObject);

begin

AdoUser.ConnectionString:=ConnStr;

AdoUser.TableName:='Users';

AdoUser.Open;

AdoPlayer.ConnectionString:=ConnStr;

AdoPlayer.TableName:='Players';

adoplayer.Open;

AdoTeam.ConnectionString:=ConnStr;

AdoTeam.TeamName:='Team';

adoteam.Open;

GbxTeams.Hide;

GbxFixtures.Hide;

BtnPrint.Enabled:=false;

If adouser['weekno']<>adoUser['WeeksToBePlayed'] then **//If season not finished**

begin

showmessage('You can''t reset the fixtures mid-season. Return to menu or print current fixtures.');

BtnFind.Enabled:=false;

BtnPrint.Enabled:=true;

end

else

begin

BtnFind.Enabled:=True;

BtnPrint.Enabled:=false;

if adoteam.Locate('teamName','BYE',[]) then **//Delete BYE from team table if it exists**

adoTeam.Delete;

adoteam.Close;

end;

end;

procedure TFmFixtures.BtnFindClick(Sender: TObject);

var TeamNo:integer;

begin

AdoFindTeams.ConnectionString:=ConnStr;

AdoFindTeams.SQL.Clear;

AdoFindTeams.SQL.Add('SELECT TeamName, TeamNo, Captain');

AdoFindTeams.SQL.Add('FROM Team');

AdoFindTeams.SQL.Add('WHERE Played=0');

adoFindTeams.Open; **//Run query to find the teams playing this season**

NoOfTeams:=adoFindTeams.RecordCount;

if NoOfTeams<6 then **//Check at least 6 teams have been set up to play.**

showmessage('Not enough teams have been found. Please add and set up teams first.')

else

begin

TeamGrid.RowCount:=NoOfTeams+1; **//Set gird size to number of teams**

adofindteams.First; **//Move to first record**

For TeamNo:=1 to NoOfTeams do **//For each team**

begin

TeamGrid.cells[0,TeamNo]:=IntToStr(TeamNo); **//Display team details in grid**

TeamGrid.cells[1,TeamNo]:=adoFindTeams['teamname'];

TeamGrid.cells[2,TeamNo]:=adofindTeams['Captain'];

adoFindTeams.Edit;

adoFindTeams['TeamNo']:=IntToStr(TeamNo);

adoFindTeams.post;

adofindteams.Next; **//Move to next record**

end;

If messageDlg(IntToStr(adoFindTeams.RecordCount)+' teams found. Is this correct?',

mtconfirmation,[mbyes,mbno],0)=mryes then

begin

**//delete any teams not required**

DeleteOldTeams;

NoOfLanes:=NoOfTeams;

**//Add a bye if the number of teams is odd**

if NoOfTeams mod 2 <>0 then

begin

NoOfLanes:=NoOfLanes-1;

NoOfTeams:=NoOfTeams+1;

TeamGrid.RowCount:=NoOfTeams+1;

TeamGrid.cells[0,NoOfTeams]:=IntToStr(NoOfTeams);

TeamGrid.cells[1,NoOfTeams]:='BYE';

TeamGrid.cells[2,NoOfTeams]:='BYE';

adoFindTeams.append;

adoFindTeams['teamname']:='BYE';

adofindTeams['Captain']:='BYE';

adoFindTeams['TeamNo']:=IntToStr(NoOfTeams+1);

adoFindTeams.post;

adofindteams.Next;

end;

timesRound:=1;

GbxFixtures.Show;

GbxTeams.Show;

if NoOfTeams<21 then **//If less than 21 teams let user select times round**

begin

showmessage('Select times round');

spedtimes.MaxValue:=44 div NoOfTeams; **//Set maximum times round that can be selected**

spedTimes.show;

LblTimes.show;

end;

end

else

begin

showmessage('Use ''ADD NEW TEAM'' to add any new teams');

end;

end;

end;

Procedure TFmFixtures.ProduceFixtures;

var TeamNumber, TimeRound, WeekNo, match, NoOfMatches, lane:integer;

begin

**//Set up week one**

NoOfmatches:=NoOfTeams div 2;

TeamNumber:=1;

for match:=1 to NoOfMatches do

begin

Fixture[1,match,1]:=TeamNumber;

Fixture[1,match,2]:=TeamNumber+1;

TeamNumber:=TeamNumber+2;

end;

**//set up remaining weeks**

for WeekNo:=2 to NoOfTeams-1 do

begin

Fixture[WeekNo,1,1]:=1;

Fixture[WeekNo,1,2]:=Fixture[WeekNo-1,2,1];

Fixture[WeekNo,NoOfMatches,1]:=Fixture[WeekNo-1,NoOfMatches,2];

for Match:=2 to NoOfMatches-1 do

Fixture[WeekNo,match,1]:=Fixture[WeekNo-1, match+1,1]; **//get team 1**

for Match:=NoOfMatches downto 2 do

Fixture[WeekNo,Match,2]:=Fixture[WeekNo-1,Match-1,2]; **//get team 2**

end;

**//Now create and save fixtures**

adofixtures.open;

for TimeRound:= 1 to timesround do **//For each time round**

begin

for WeekNo:=1 to NoOfTeams-1 do **//For each week in a 'time round'**

begin

lane:=(2\*(WeekNo-1)+1) mod NoOflanes;

for Match:=1 to NoOfMatches do **//For each match played in a week**

begin

adofixtures.Append;

adofixtures['weekno']:=WeekNo+(TimeRound-1)\*(NoOfTeams-1); **//get the week number**

**//This bit of code swaps round the lanes so that the teams play on a mix of odd and even lane no.s**

if WeekNo<NoOfTeams Div 2 then

begin

adofixtures['team1']:=TeamGrid.cells[1,Fixture[WeekNo,Match,1]];

adofixtures['team2']:=TeamGrid.cells[1,Fixture[WeekNo,Match,2]];

end

else

begin

adofixtures['team2']:=TeamGrid.cells[1,Fixture[WeekNo,Match,1]];

adofixtures['team1']:=TeamGrid.cells[1,Fixture[WeekNo,Match,2]];

end;

adofixtures['played']:=false;

if (adofixtures['team1']<>'BYE') and (adofixtures['team2']<>'BYE') then **//If not a BYE find the lanes**

begin

adofixtures['laneNos']:=intToStr(lane)+'-'+inttostr(lane+1);

lane:=lane+2;

if lane>NoOfLanes then lane:=1;

end

else

adofixtures['laneNos']:=' - ';

adofixtures.Post; **//Save fixture**

end;

end;

end;

end;

procedure TFmFixtures.DeleteOldTeams;

begin

adoteam.open;

adoteam.First; **//Move to first team record**

while not adoteam.Eof do

begin

if adoteam['played']<> 0 then **//If matches played not reset to 0**

begin

adoteam.Delete; **//Delete the team**

end;

adoteam.Next; **//Move to next team record**

end;

adoteam.Close;

end;

procedure TFmFixtures.btnFixturesClick(Sender: TObject);

begin

producefixtures;

showmessage('Fixtures now set up - print fixtures then return to menu');

adouser.open;

adouser.First; **//Reset weekno to 0 and set matches to be played**

adouser.Edit;

adouser['weekno']:=0;

adouser['WeeksToBePlayed']:=timesround\*(NoOfTeams-1);

adouser.Post;

adouser.close;

BtnPrint.Enabled:=true;

end;

procedure TFmFixtures.SpEdtimesChange(Sender: TObject);

begin

timesround:=spedTimes.Value;

end;

procedure TFmFixtures.BtnReturnClick(Sender: TObject);

begin

FmNewSeason.show;

Close;

end;

procedure TFmFixtures.BtnPrintClick(Sender: TObject);

var dpmm, ypos, weekno, NoOfMatches, MatchNo:integer;

begin

adoqSortFixtures.ConnectionString:=ConnStr;

adoqSortFixtures.SQL.Clear;

adoqSortFixtures.SQL.Add('SELECT weekNo, team1, team2, laneNos');

adoqSortFixtures.SQL.Add('FROM fixtures');

adoqSortFixtures.SQL.Add('ORDER BY weekno, lanenos');

adoqsortfixtures.Open; **//Run query to sort fixtures into week number order**

adoqsortfixtures.First;

adouser.Open;

adouser.First;

**//Calculate number of matches played each week**

NoOfMatches:=adoqsortfixtures.RecordCount div adouser['WeeksToBePLayed'];

with printer do **//Set up printer**

begin

dpmm:=pagewidth div 210;

begindoc; **//Start of document**

with canvas do

begin

font.Size:=18;

font.Name:='Georgia';

font.Style:=[fsbold];

textout(40\*dpmm, 20\*dpmm, 'The Pioneer League - Fixture List');

font.Size:=11;

ypos:=40;

For weekno:=1 to adouser['WeeksToBePLayed'] do **//For each week to be played**

begin

font.Style:=[fsbold];

TextOut(20\*dpmm, ypos\*dpmm,'Week No:'+IntToStr(weekno));

font.Style:=[];

For MatchNo:=1 to NoOfMatches do **//For each match played in a week**

Begin

ypos:=ypos+10; **//Print out match details**

TextOut(20\*dpmm, ypos\*dpmm,adoqsortfixtures['team1']+' v '+adoqsortfixtures['team2']);

TextOut(100\*dpmm, ypos\*dpmm,'Lanes: '+adoqsortFixtures['LaneNos']);

Adoqsortfixtures.Next; **//Move to next fixture**

end;

If ypos>(250-(NoOfMatches\*10))then **//If got to end of page**

begin

ypos:=20; **//Start a new page**

enddoc;

begindoc;

end

else

ypos:=ypos+20;

end;

end;

enddoc;

end;

end;

end.**FmEnterScores**

Label:

LbTeam2

Combo Box:

CbxTeam2

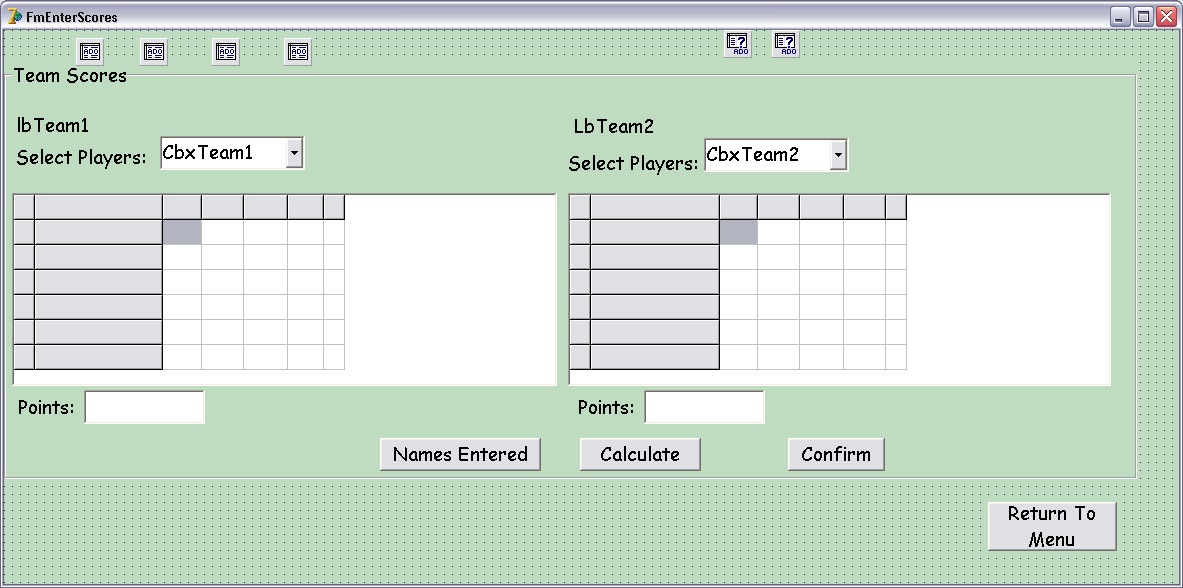
StringGrid:

Team2Grid

Edit Box:

Edpoints2

AdoTables: AdoUser, AdoFixtures, AdoPlayer, AdoTeam AdoQueries: AdoQFindFixtures, AdoQgetPlayers



Buttons: BtnNamesEntered, BtnCalculate, BtnConfirm, BtnReturn

Label:

LbTeam1

Combo Box:

CbxTeam1

StringGrid:

Team1Grid

Edit Box:

Edpoints1

GroupBox:

GbxPlayers

**unit UEnterScores;**

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DB, ADODB, DBCtrls, Grids, StdCtrls, DBGrids, ExtCtrls;

type

TFmEnterScores = class(TForm)

ADOuser: TADOTable;

ADOFixtures: TADOTable;

GbxPLayers: TGroupBox;

ADOQFindFixtures: TADOQuery;

Grid1: TStringGrid;

Grid2: TStringGrid;

lbTeam1: TLabel;

LbTeam2: TLabel;

AdoQTeam1: TADOQuery;

Label1: TLabel;

ADOQTeam2: TADOQuery;

Label2: TLabel;

BtnNamesEntered: TButton;

CbxTeam1: TComboBox;

CbxTeam2: TComboBox;

BtnCalculate: TButton;

AdoPlayer: TADOTable;

EdPoints1: TLabeledEdit;

EdPoints2: TLabeledEdit;

BtnConfirm: TButton;

ADOTeam: TADOTable;

BtnReturn: TButton;

procedure FormActivate(Sender: TObject);

procedure SetUpGrid(var grid:tstringgrid);

procedure GetPlayers(team:string; var Cbx:TcomboBox);

procedure CbxTeam1Click(Sender: TObject);

Function InList(grid:tstringgrid; player:string):boolean;

procedure CBxTeam2Click(Sender: TObject);

Procedure AddToGrid(grid:tstringgrid; cbx:TComboBox; var NoOfPlayers:Integer);

procedure Grid1SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

procedure SetUpScoreGrid(var grid:tstringgrid);

procedure Grid2SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

procedure BtnNamesEnteredClick(Sender: TObject);

procedure BtnCalculateClick(Sender: TObject);

Function valid(grid:Tstringgrid):boolean;

Procedure EnterBlinds(var grid:TstringGrid);

Function NewHandicap(Totalpinfall,NoOfGames:integer):integer;

Procedure AddUp(var grid:tstringgrid);

Procedure getPoints(score1, score2:integer; var pts1, pts2:real);

procedure BtnConfirmClick(Sender: TObject);

procedure SaveTeamScores(grid1,grid2:TstringGrid; Pts:real; team:string);

procedure SavePlayersScores(grid:TStringGrid);

procedure displayTeams(Team1,Team2:string);

procedure BtnReturnClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FmEnterScores: TFmEnterScores;

weekno, Team1Players, Team2Players:integer;

namesEntered,finished:boolean;

implementation

uses UMenu, utilities;

{$R \*.dfm}

procedure TFmEnterScores.FormActivate(Sender: TObject);

begin

AdoUser.ConnectionString:=ConnStr;

AdoUser.TableName:='Users';

AdoUser.Open;

AdoFixtures.ConnectionString:=ConnStr;

AdoFixtures.TableName:='Fixtures';

adoFixtures.Open;

AdoqFindFixtures.ConnectionString:=ConnStr;

GbxPlayers.Hide;

adouser.First;

weekno:=adouser['weekno']+1; **//Find the week number for this week**

adofixtures.Open;

finished:=false;

if weekno>adouser['WeeksToBePlayed'] then **//If the season has finished**

begin

showmessage('This season has finished. Please return to menu.');

finished:=true;

end

else

begin

adofixtures.close;

adoqfindfixtures.SQL.clear; **//Query to find this week's fixtures**

adoqfindfixtures.SQL.add('SELECT team1, team2, played');

adoqfindfixtures.SQL.add('FROM fixtures');

adoqfindfixtures.SQL.add('WHERE weekNo='+IntToStr(WeekNo));

adoqfindfixtures.open;

adoqfindfixtures.first; **//Move to first fixture**

If (adoqfindFixtures['Team1']='BYE') or (adoqfindFixtures['Team2']='BYE') then **//If fixture is a BYE**

adoqfindfixtures.next; **//Move to next fixture**

displayTeams(adoqFindFixtures['team1'],adoqFindFixtures['team2']); **//Get and display team details**

GbxPlayers.show;

namesEntered:=false;

end;

end;

procedure TFmEnterScores.displayTeams(Team1,Team2:string);

begin

lbTeam1.Caption:=Team1; **//Display team names**

LbTeam2.caption:=Team2;

Team1Players:=0;

Team2Players:=0;

SetUpGrid(grid1); **//Set up grid headings**

SetUpGrid(grid2);

GetPlayers(Team1, cbxteam1); **//Load team1's players' names into Team1's combo box**

GetPlayers(Team2, cbxteam2); **//Load team2's players' names into Team2's combo box**

NamesEntered:=false;

BtnConfirm.Enabled:=false;

Btncalculate.Enabled:=false;

end;

procedure TfmEnterScores.GetPlayers(team:string; var Cbx:TcomboBox);

begin

Cbx.Clear;

adoQGetPlayers.ConnectionString:=ConnStr;

adoQGetPlayers.SQL.Clear;

adoQGetPlayers.SQL.Add('SELECT FullName, ID');

adoQGetPlayers.SQL.Add('FROM player');

adoQGetPlayers.SQL.Add('WHERE teamname="'+Team+'"');

adoQGetPlayers.Open; **//Run the query to get the players for this team**

adoQGetPlayers.First;

while Not adoQGetPlayers.Eof do

begin

cbx.Items.Add(adoQGetPlayers ['FullName']); **//Put names in combobox**

adoQGetPlayers.Next;

end;

cbx.Items.add('Blind'); **//Add blind option to combo box**

end;

procedure TFMEnterScores.SetUpGrid(var grid:tstringgrid);

begin **//Set up the grid to enter the names**

grid.ColCount:=3;

grid.ColWidths[0]:=30;

grid.Cells[0,0]:='Id';

grid.ColWidths[1]:=50;

grid.Cells[1,0]:='Hcp';

grid.ColWidths[2]:=175;

grid.Cells[2,0]:='Name';

end;

procedure TFMEnterScores.SetUpScoreGrid(var grid:tstringgrid);

var i:integer; **//Set up grid to allow scores to be entered**

begin

grid.ColCount:=7;

for i:=1 to 3 do

begin

grid.ColWidths[i+2]:=60;

grid.Cells[i+2,0]:='Game'+IntToStr(i);

end;

grid.ColWidths[6]:=60;

grid.Cells[7,0]:='Series';

grid.Cells[1,4]:='Total';

end;

procedure TFmEnterScores.CbxTeam1Click(Sender: TObject);

begin

AddToGrid(grid1,cbxTeam1,Team1Players); **//Add selected player to score grid**

end;

Function TFmEnterScores.InList(grid:tstringgrid; player:string):boolean;

var i:integer; there:boolean;

begin

there:=false; **//Check selected player is not already in the scrore grid**

for i:=1 to 3 do

if grid.cells[2,i]=player then

there:=true;

InList:=there;

end;

procedure TFmEnterScores.CBxTeam2Click(Sender: TObject);

begin

AddToGrid(grid2,cbxTeam2,Team2Players); **//Add selected player to score grid**

end;

Procedure TFmEnterScores.AddToGrid(grid:tstringgrid; cbx:TComboBox; var NoOfPlayers:Integer);

var i:integer;

begin

if Not InList(grid,cbx.Text) then **//If selected player is not already in the grid**

begin

if NoOfplayers<3 then **//If less than three player have been added**

begin

NoOfPlayers:=NoOfplayers+1; **// Add one to number of players**

i:=1;

while (i<4) and (grid.cells[2,i]<>'') do **//Put name of player in first empty slot in grid**

i:=i+1;

grid.cells[2,i]:=cbx.text;

if cbx.text<>'Blind' then **//If not a 'Blind'**

begin

adoplayer.locate('FullName',cbx.Text,[]); **//Find player record**

grid.cells[0,i]:=adoPlayer['ID'];

if adoPlayer['gamesPlayed']>0 then **//If the player has played previously**

grid.Cells[1,i]:=adoPlayer['Handicap'] **//Add handicap to grid**

else //**else**

grid.Cells[1,i]:='-'; **//Display - in grid to indicate no handicap**

end;

end

else

showmessage('You already have three players - delete one before adding another.');

end

else

showmessage('Already chosen');

end;

procedure TFmEnterScores.Grid1SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

begin

if (acol=2)then **//If user has clicked on the name column**

begin

If (arow>0) and (arow<4) and Not NamesEntered then **//If names entered has not been clicked**

begin

if (grid1.Cells[2,arow]<>'') then **//If cell clicked on contains a name**

begin

grid1.Cells[2,arow]:=''; **//Remove name from grid**

Team1Players:=Team1Players-1; **//reduce the number of players added to grid**

end;

end

else

canselect:=false;

end

else

begin

if NamesEntered then **//If all the names have been entered**

begin

if (acol in [3..5]) and (arow in [1..3]) Then **//Restrict the user to entering data in required cells**

canselect:=true

else

canselect:=false;

end;

end;

end;

procedure TFmEnterScores.Grid2SelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

if (acol=2)then **//If user has clicked on the name column**

begin

If (arow>0) and (arow<4) and Not NamesEntered then **//If names entered has not been clicked**

begin

if (grid2.Cells[2,arow]<>'') then **//If cell clicked on contains a name**

begin

grid2.Cells[2,arow]:=''; **//Remove name from grid**

Team2Players:=Team2Players-1; **//reduce the number of players added to grid**

end;

end

else

canselect:=false;

end

else

begin

if NamesEntered then **//If all the names have been entered**

begin

if (acol in [3..5]) and (arow in [1..3]) Then **//Restrict the user to entering data in required cells**

canselect:=true

else

canselect:=false;

end;

end;

end;

procedure TFmEnterScores.BtnNamesEnteredClick(Sender: TObject);

begin

if (team1players<>3) or (team2Players<>3) then **//If less than 3 players entered**

showmessage('Please enter three players for each team.')

else

begin

NamesEntered:=true;

SetUpScoreGrid(grid1); **//set up grid headings**

SetUpScoreGrid(grid2);

showmessage('Please enter scores then click Calculate');

**//Enter scores for blinds**

EnterBlinds(grid1);

EnterBlinds(grid2);

Btncalculate.Enabled:=true;

end;

end;

Procedure TFmEnterScores.EnterBlinds(var grid:TstringGrid);

var r,c:integer;

begin

for r:=1 to 3 do

begin

if grid.cells[2,r]='Blind' Then **//If player is a 'blind'**

begin

grid.cells[1,r]:='0'; **//set handicap to 0**

for c:=3 to 5 do

grid.Cells[c,r]:='150' **//set scores to 150**

end;

end;

end;

procedure TFmEnterScores.BtnCalculateClick(Sender: TObject);

var r,c,Hcp1,Hcp2, score1, score2:integer; pts1, pts2, TotalPts1, TotalPts2:real;

begin

if (Not EmptyCell(grid1,1,3,3,5)) and (not emptyCell(grid2,1,3,3,5)) then **//If all scores entered**

begin

if valid(grid1) then **//If all grid1 scores are valid**

begin

if valid(grid2) then **//If all grid 2 scores are valid**

begin

AddUp(grid1); **//Add up grid 1 scores**

AddUp(grid2); **//Add up grid 2 scores**

TotalPts1:=0;

TotalPts2:=0;

**//Work out points for each player**

for r:=1 to 3 do

begin

hcp1:=strToInt(grid1.Cells[1,r]);

hcp2:=strToInt(grid2.Cells[1,r]);

for c:=3 to 6 do

begin

if c=6 then **//If calculating player's series score**

begin

hcp1:=hcp1\*3; **//Multiply the handicap by 3**

hcp2:=hcp2\*3;

end;

score1:=StrToInt(grid1.Cells[c,r])+hcp1; **//Work out team1 player's handicap score**

score2:=StrToInt(grid2.Cells[c,r])+hcp2; **//Work out team2 player's handicap score**

getPoints(score1,score2,pts1,pts2); **//Calculate the points for this game**

totalPts1:=TotalPts1+pts1; **//Add points to running total**

totalpts2:=TotalPts2+pts2;

end;

end;

**//Work out team points**

for c:=3 to 6 do

begin

score1:=StrToInt(grid1.Cells[c,6]); **//Work out team1's game score**

score2:=StrToInt(grid2.Cells[c,6]); **//Work out team2's game score**

getPoints(score1,score2,pts1,pts2); **//Calculate the points**

**//Multiply pts by 4 to get team points**

totalPts1:=TotalPts1+4\*pts1; **//Add points to running total**

totalpts2:=TotalPts2+4\*pts2;

end;

Edpoints1.Text:=FloatToStrf(totalpts1,fffixed,5,1); **//Display points**

Edpoints2.Text:=FloatToStrf(totalpts2,fffixed,5,1);

showmessage('Check the scores are correct then click on confirm.');

BtnConfirm.enabled:=true;

end;

end;

end

else

showmessage('Score missing.');

end;

Procedure TFMenterScores.getPoints(score1, score2:integer; var pts1, pts2:real);

begin

if score1>score2 then

begin

pts1:=1; **//Player 1 wins**

pts2:=0;

end

else

begin

if Score1<Score2 then

begin

pts2:=1; **//Players 2 wins**

pts1:=0;

end

else

begin

pts1:=0.5; **//It's a draw.**

pts2:=0.5;

end;

end;

end;

Procedure TFmEnterScores.AddUp(var grid:tstringgrid);

var player,c, PlayerTotal, TeamScratch, TeamHcp:integer;

begin

**//Calculate each player's total score**

TeamHcp:=0;

for player:=1 to 3 do

begin

PlayerTotal:=0;

for c:=3 to 5 do

begin

PlayerTotal:=PlayerTotal+StrToInt(grid.Cells[c,player]);

end;

grid.Cells[6,Player]:=IntToStr(PlayerTotal);

**//If a player does not have a current handicap - calculate it from these scores**

If grid.Cells[1,player]='-' then

begin

grid.Cells[1,player]:=IntToStr(NewHandicap(PlayerTotal,3));

end;

**//Calculate total team handicap**

TeamHcp:=TeamHcp+(StrToInt(grid.Cells[1,player]));

end;

**//Calculate team totals**

for c:=3 to 6 do

begin

teamScratch:=0;

for player:=1 to 3 do

begin

teamscratch:=TeamScratch+(StrToInt(grid.Cells[c,player]));

end;

grid.Cells[c,4]:=IntToStr(TeamScratch);

If c<>6 then

grid.Cells[c,5]:=IntToStr(TeamHcp)

else

grid.Cells[c,5]:=IntToStr(TeamHcp\*3);

grid.cells[c,6]:=IntToStr(TeamScratch+StrToInt(grid.Cells[c,5]));

end;

end;

Function TFmEnterScores.NewHandicap(Totalpinfall,NoOfGames:integer):integer;

var average, difference, hcp:real;

begin

average:=TotalPinfall/NoOfGames;

difference:=200-average;

Hcp:=Int(difference\*2/3);

NewHandicap:=Trunc(hcp);

end;

Function TFmEnterScores.valid(grid:Tstringgrid):boolean;

var r,c:integer;

begin

result:=true;

for r:=1 to 3 do

for c:=3 to 5 do

if not validInteger(grid.Cells[c,r],1,300) then **//if number entered is not an integer between 1 and 300**

result:=false;

end;

procedure TFmEnterScores.BtnConfirmClick(Sender: TObject);

var found:boolean;

begin

**//Update team records with new scores and points**

SaveTeamScores(grid1, grid2,StrToFloat(edpoints1.text),LbTeam1.caption);

SaveTeamScores(grid2,grid1,StrToFloat(edpoints2.text),LbTeam2.caption);

**//Update Player records with new totals**

savePlayersScores(grid1);

savePlayersScores(grid2);

adoqfindfixtures.Edit;

adoqfindfixtures['played']:=true; **//Record that the fixture has been played**

adoqfindfixtures.Post;

**//Clear grids**

ClearGrid(grid1,1,0);

ClearGrid(grid2,1,0);

**//find and display next match**

found:=false;

while Not(adoqfindfixtures.Eof) and Not found do

begin

adoqfindfixtures.next; **//Move to next fixture record**

if (adoqFindFixtures['team1']<>'BYE') and (adoqFindFixtures['team2']<>'BYE') then

found:=true;

end;

if Not(adoqfindfixtures.Eof) then **//If a fixture has been found**

displayTeams(adoqFindFixtures['team1'],adoqFindFixtures['team2'])

else

begin

showmessage('Finished'); **//If finished entering this week's scores**

finished:=true;

adouser.edit;

adouser['weekno']:=weekno; **//Update user table to indicate this week's scores have been saved**

adouser.Post;

adouser.Close;

end;

end;

procedure TfmEnterScores.SaveTeamScores(grid1,grid2:TstringGrid; Pts:real; team:string);

var i:integer;

begin

**//save the team scores**

adoteam.Open;

adoteam.Locate('TeamName',team,[]);

adoteam.edit;

adoteam['points']:=adoteam['points']+Pts;

adoTeam['TotalPinfall']:=adoTeam['TotalPinfall']+StrToInt(grid1.cells[6,6]);

adoTeam['TotalPinsAgainst']:=adoTeam['TotalPinsAgainst']+StrToInt(grid2.cells[6,6]);

adoTeam['Played']:=adoTeam['Played']+1;

If StrToInt(grid1.cells[6,6])>adoTeam['HighestSeries'] Then **//Check if new highest series**

adoTeam['HighestSeries']:= StrToInt(grid1.cells[6,6]);

for i:=3 to 5 do

begin

if StrToInt(grid1.cells[i,6])> adoTeam['HighestGame'] then **//Check if new highest game**

adoTeam['HighestGame']:=StrToInt(grid1.cells[i,6]);

end;

adoteam.post;

adoteam.Close;

end;

procedure TFmEnterScores.SavePlayersScores(grid:TStringGrid);

var i, hcp,c:integer;

begin

AdoPlayer.open;

for i:=1 to 3 do

begin

if grid.cells[2,i]<>'Blind' then **//If player is not a blind**

begin

adoPlayer.Locate('ID',grid.Cells[0,i],[]); **//Locate player record**

adoPlayer.edit;

hcp:=StrToInt(grid.cells[1,i]);

for c:=3 to 5 do

begin

if StrToInt(grid.Cells[c,i])>adoPlayer['HighestGame'] then **//Check if new highest scratch game**

adoPlayer['HighestGame']:=StrToInt(grid.Cells[c,i]);

If StrToInt(grid.Cells[c,i])+Hcp>adoPlayer['HighestHcpGame'] then **//Check if new highest hcp game**

adoPlayer['HighestHcpGame']:=StrToInt(grid.Cells[c,i])+Hcp;

end;

if StrToInt(grid.Cells[6,i])>adoPlayer['HighestSeries'] then **//Check if new highest scratch series**

adoPlayer['HighestSeries']:=StrToInt(grid.Cells[6,i]);

If StrToInt(grid.Cells[6,i])+Hcp>adoPlayer['HighestHcpSeries'] then **//Check if new highest hcp series**

adoPlayer['HighestHcpSeries']:=StrToInt(grid.Cells[6,i])+Hcp;

**//Update number of games played and total pinfall**

adoPlayer['GamesPlayed']:=adoPlayer['GamesPlayed']+3;

adoplayer['TotalPinfall']:=adoPlayer['TotalPinfall']+StrToInt(grid.Cells[6,i]);

**//Calculate new handicap**

adoPlayer['Handicap']:=NewHandicap(adoPlayer['TotalPinfall'],adoPlayer['GamesPlayed']);

adoPlayer.Post; **//Save updated player record**

end;

end;

end;

procedure TFmEnterScores.BtnReturnClick(Sender: TObject);

begin

if finished then **//If finished entering this week's scores**

begin

fmmenu.show; **//return to menu**

close;

end

else **//else**

showmessage('Finish entering all the scores first.') ; **//Display error message**

end;

end.

**unit Utilities;**

interface

uses Messages, SysUtils,Dialogs, grids;

Function ValidInteger(TxtNum:string; min,max:integer):boolean;

Function EmptyCell(grid:Tstringgrid; firstRow,LastRow, FirstCol,LastCol:integer):boolean;

Procedure ClearGrid(var grid:Tstringgrid; FirstRow, FirstCol:integer);

Const ConnStr='Provider=Microsoft.ACE.OLEDB.12.0; Data Source=PhoenixLeague.accdb;' +

'Persist Security Info = False;';

implementation

Function ValidInteger(TxtNum:string; min,max:integer):boolean;

var num:integer;

begin

ValidInteger:=true;

If TryStrToInt(TxtNum, num) then **//If value entered is an integer**

begin

If (num<min) or (num>max) then **//If value entered is outside the required range**

begin

showmessage('Please enter a number between '+IntToStr(min)+' and '+IntToStr(max));

ValidInteger:=false;

end;

end

else

begin

showmessage('Please enter digits only.');

ValidInteger:=false;

end;

end;

Function EmptyCell(grid:Tstringgrid; firstRow,LastRow, FirstCol,LastCol:integer):boolean;

var r,c:integer;

begin

EmptyCell:=False;

For c:=FirstCol to LastCol do

for r:=FirstRow to LastRow do

If grid.cells[c,r]='' then **//If cell in grid is empty**

emptyCell:=true;

end;

Procedure ClearGrid(var grid:Tstringgrid; FirstRow, FirstCol:integer);

var r,c:integer;

begin

for c:=FirstCol to grid.ColCount-1 do

for r:=FirstRow to grid.RowCount-1 do

grid.cells[c,r]:='' **//Set contents of cell in grid to empty string**

end;

end.