Tournament Planner

IT PAT

Johan Nel – Gr 11-5

HOËRSKOOL SECUNDA

Contents

Task 0: Research

Topic

The topic of my PAT is the planning and scoring of an eisteddfod tournament.

Purpose of program

I will use Delphi 2010 to write a program that will allow the tournament organisers (the teachers) to plan the tournament, and the tournament participants to view when the next event will be taking place.

Possible solution

My program will allow participants to register for a certain eisteddfod item (like Music). This will automatically add them to the database that the tournament organisers have access to. The tournament organisers will be able to set dates for the next events, enter participant scores, and determine who will go through to the next round.

Scope

My program won't allow participants to sign up and log in on the internet. Organisers and participants can only access the information on the computer with the database.

Task 1: Task definition and Users

Task 1A: Define the Task

Event organisers at my school have always used manual methods to plan and host tournaments. My aim is to code a program that will allow the organisers of the school's eisteddfod to easily plan and manage their tournament. It will allow them to save time as the program does most of the processing.

I will be designing and coding a program for the school which the event organisers can use to organize an eisteddfod tournament. It will allow participants to log in and enter the eisteddfod. It will also allow the organisers to make a timetable/schedule and let them enter the participant's scores after they have performed. After all participants participated in a certain round, the program will allow the organisers to enter a criteria. This will allow the program to automatically inform the participants who made it into the next round. After multiple rounds, the eisteddfod winner will be decided. The organiser can determine the winner by retrieving the results.

Even though this program has a large scope, some functionality will not be implemented at this stage. This includes but is not limited to:

- Users will not be able to edit their profiles after it has been created.
- The program will not be able to send information to external devices such as other computers, web browsers and phones to make it easier for the user to sign up.

The program will require me to use the skills I've learnt with Delphi programming to ensure the school can easily manage the eisteddfod tournament. It will also ensure that the participants are aware of their current scores and how the tournament has been progressing.

Task 1B: User story and acceptance test

- **User: Organiser**

Description	
Role	The event organiser will use this program to log in using an automatically generated username and a custom password. When logged in, they will be able to use the program to organise the tournament. This way they can see who has been registered, and give dates on rounds, and create schedules.
Activity	The event organiser will be able to: • Create an account (sign up) • Log in • View data in databases • Enter participant scores/comments • Give criteria that decides who will go through to the next round • Create and print schedules • Retrieve results • Eliminate participants • Delete items after all rounds are finished
Value	We need the event organisers to keep track of the tournament's progress and to have an electronic copy of the tournament's results. It will also make it easier to organise and manage tournaments.
Limitations	Only one tournament can be organised at a time. Organizers cannot edit the participant's personal information. Other limitations are defined in the scope of the project.

- **User: Participant**

Description	
Role	Participants will use this program to log in using an automatically generated username and a custom password. When logged in, they will be able to use the program to enter an item into the eisteddfod tournament. They will also be able to see how the tournament has been progressing, when the next round is, and how much they scored.
Activity	The event participant will be able to: • Create an account (sign up) • Log in • Enter an item • View their scores for the latest round • View their latest comments • See if they made it through to the next round or not • See when they will participate next (if a schedule has been created by the organiser)
Value	We need the event participants to keep track the participant scores and event progress. Without participants the event will not

	be a success. The program makes it easier to view each participant's progress.
Limitations	Participants cannot edit their own information. They also cannot view other's data or manage the tournament. Participants cannot see what position they are on the view item screen, they must use the results the organisers printed out.

- **"User": Admin**

Please note: Even though the help button and documentation speak of admins, they do not have a specific GUI. They are event organisers who also know how to use the program from top to bottom. They know how to manipulate the database, and how to traverse the program text files without causing runtime errors.

- **Use cases:**

The program will be able to:

- Let the user create an account.
- Let the user log in with their existing account.
- Encrypt the password so that the tournament organisers do not have access to it.
- Allow participants to fill in their information to enter an item and participate.
- Allow participants to view their information, when the next round will take place, and how much they scored on the previous round, and that round's comments.
- Allow participants to see if they made it through to the next round.
- Allow organisers to view and edit the data in the database.
- Allow organisers to add/change participant scores, comments, amount paid, etc. into the database.
- Allow organisers to give criteria that decides who will go through to the next round.
- Allow organisers to create and print schedules. Participants can access the printed schedules, not load the schedules in the program.
- Allow organisers eliminate participants with criteria.
- Allow organiser to delete items after the winner has been announced.

Task 2: Database Design

tblUsers

Field Name	Data Type	
Username	Short Text	Username used to log in
Password	Short Text	Password used to log in
First_Name	Short Text	Name
Last_Name	Short Text	Surname
Birth_Date	Date/Time	Date birthed on
User_Type	Short Text	M: Tournament Manager or V: Tournament Viewer
Paid	Currency	Amount paid by participant - R250 per round participated

Field Name	Data Type	Field size	Required?
Username 🔑	Short Text / String	42	Yes
Password	Short Text / String	30	Yes
First_Name	Short Text / String	20	Yes
Last_Name	Short Text / String	20	Yes
Birth_Date	Date	Format: Short Date yyyy/MM/dd	Yes
User_Type	Short Text / String	1	Yes
Paid	Currency / Real	Format: R "# ##0.00	No

Extract:

Username	Password	First_Name	Last_Name	Birth_Date	User_Type	Paid
AmyCole17	0|ŧŧ≀	Amy	Cole	2004/12/20	V	R 300.00
BarbaraFod13	0|ŧŧ≀	Barbara	Fod	2009/01/31	V	R 600.00
BelindaLee16	0|ŧŧ≀	Belinda	Lee	2005/10/30	V	R 475.00

tblScores

Field Name	Data Type	Descript
ItemCode	Short Text	The unique code of the item - items can be found in text file
Username	Short Text	The username in both tables are linked
ItemName	Short Text	The name of the item that will be performed
Score	Number	The participant's latest score
Participating	Yes/No	If the participant is still participating
PartTime	Number	How much time the participant's item will take
TimeRound	Number	The time is linked to this round. Ensures that the same time doesn't display after the user participated.
NextTime	Date/Time	The time that the participant will participate next
LastRoundPart	Number	The last round the user participated in
Comments	Long Text	The judge's latest comment on the item

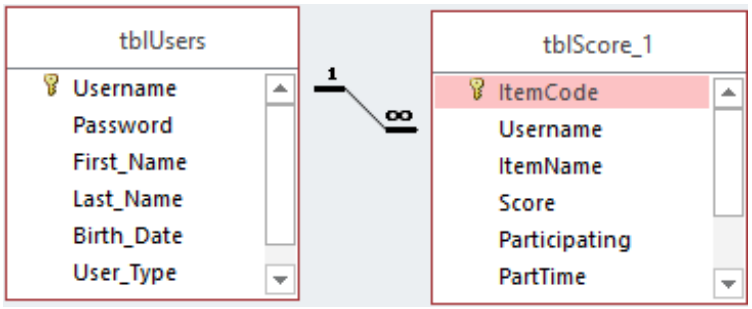
Field Name	Data Type	Field size	Required?
ItemCode 🔑	Short Text / String	5	Yes
Username 🗝	Short Text / String	42	Yes
ItemName	Short Text / String	50	No
Score	Number / Integer	Size: Integer	No
Participating	Yes/No / Boolean	True/False	No

PartTime	Number / Integer	Size: Byte	No
TimeRound	Number / Integer	Size: Byte	No
NextTime	Number / Integer	Format: Short Time hh:MM	No
LastRoundPart	Number / Integer	Size: Byte	No
Comments	Long Text / String	No size	No

Extract:

ItemCode	Username	ItemName	Score	Participatin	PartTime	TimeRound	NextTime	LastRoundPart	
REA01	PietCoetzee18	Gone series #1	78	☐	9	2	09:00	2	You didn't improve much
REA02	WendyCollins21	Paper Town	71	☐	15	2	09:09	2	I hoped you would do bette
REA03	AmyCole17	Harry Potter and the Half blood prince	80	☐	9	2	09:24	2	Amy, I'm dissapointed. You
REA04	SusanBoyle17	LOTR: Fellowship of the Ring	43	☐	11	1	10:33	1	When you walked on stage,

Relationship



Username: Primary key 🔑 in tblUsers and Foreign key🗝 in tblScore (relationship).

ItemCode: Primary key 🔑 in tblScore.

Database Manipulation

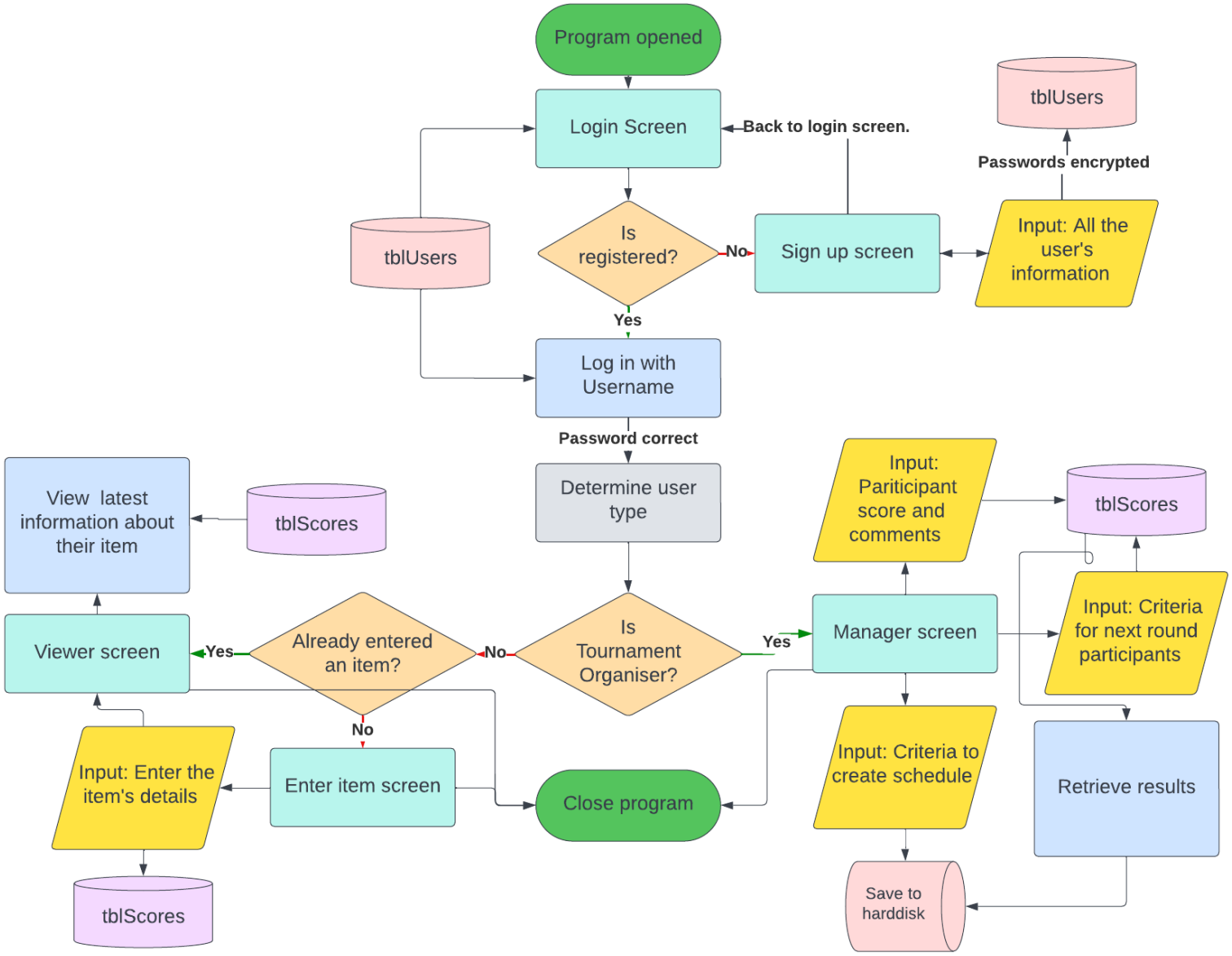
The database will be used and manipulated appropriately to ensure that the program's requirements are met.

- A user record, containing the username, name, surname, date of birth, user type and amount paid, will be created when the user signs up – **tblUsers**.
- The program will use **tblUsers** to check if the user entered the correct information when logging in.
- When a participant enters the eisteddfod, their item profile, containing the item code, username, item name, latest score, if they are still participating, approximate duration, next participation time, last round participated in, and comments, will be created – **tblScore.**
- The username is the primary key 🔑 in **tblUsers**, and foreign key🗝 in **tblScore**. This links the user's personal information to their item.
- The organiser will be able to make a schedule. The program will automatically sort the database according to item code, and then use every participant's item profile to create a schedule and allocate their next participating time (*NextTime*). It will also log what round this time represents (*TimeRound*).
- When a round starts, the organisers will be able to manipulate the user's item record (**tblScore**). They will be able to enter score the user obtained, and the judge's comments. The user's *LastRoundPart* field will also be updated to match the round's number. This will allow the user to view this information when they log in.
- The organiser will also be given the opportunity to enter the amount the user has paid so far. The tariff is R250.00 per round participated.

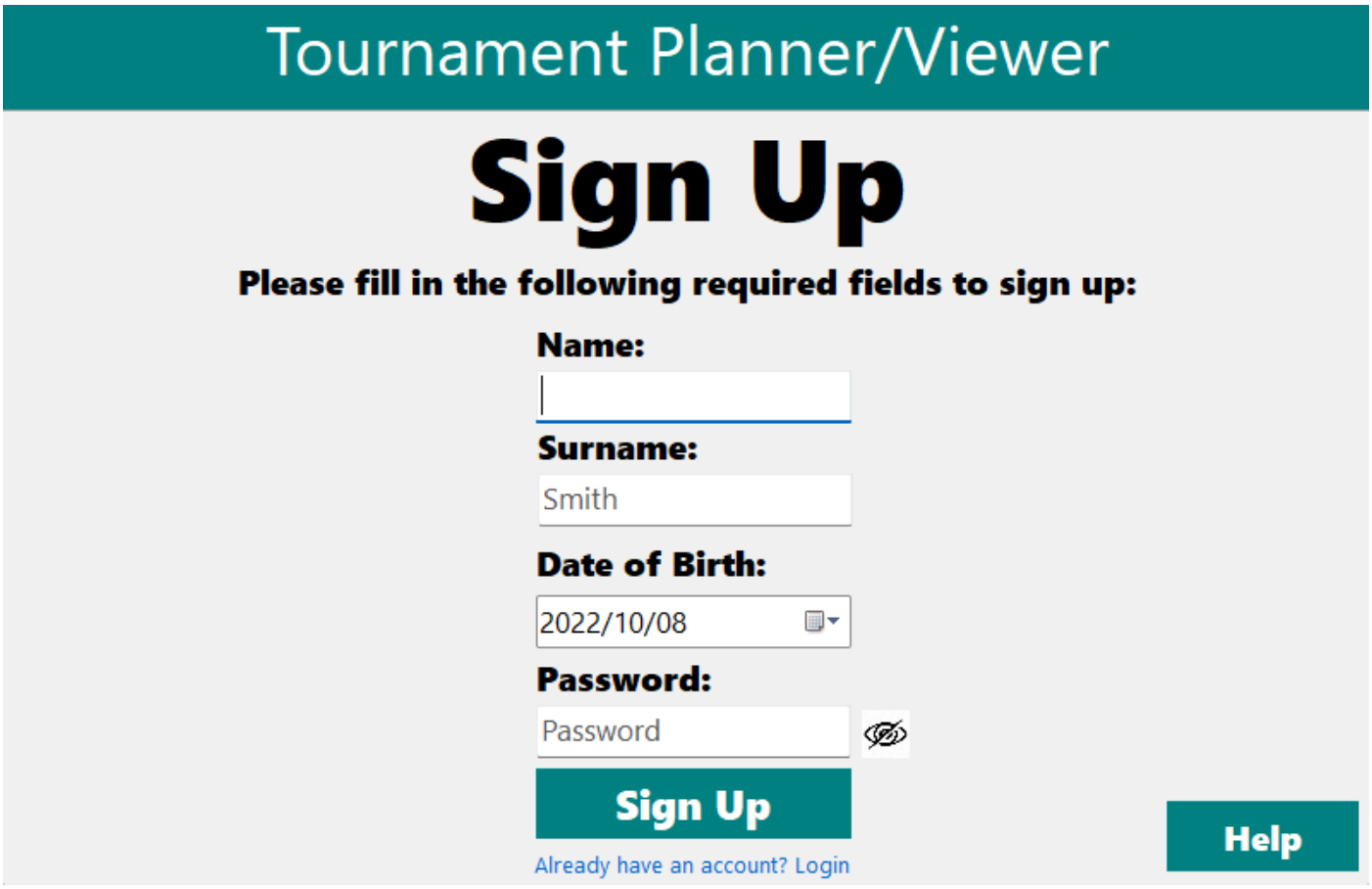
```
Owed = Rounds participated * 250 – Amount paid
```

- After every round, the organisers can give criteria to eliminate certain participants. The database will be used to determine if the user will be eliminated, by looking at the user's score. If the user is eliminated, their Participating field will be made False/No.

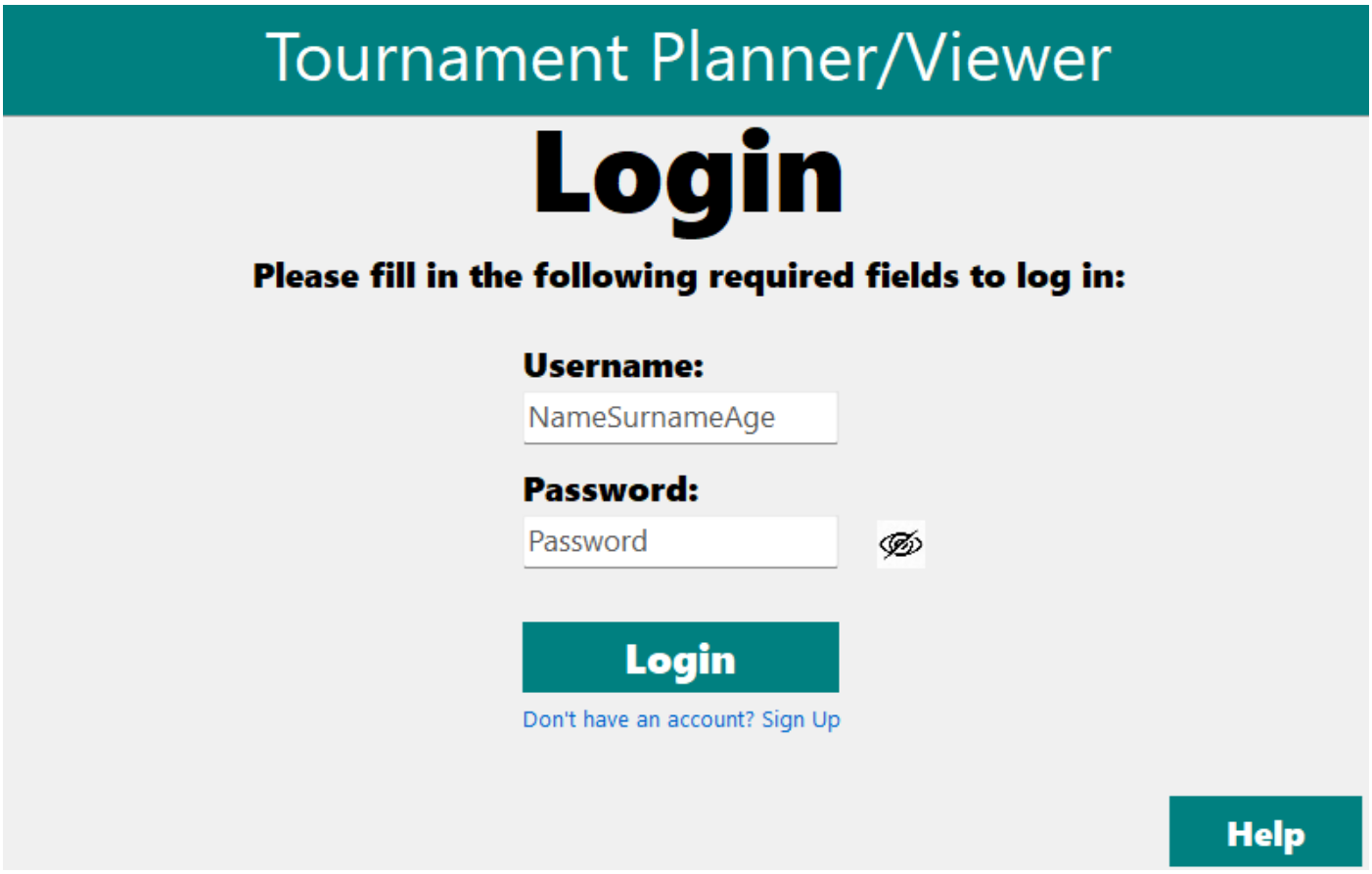
Task 3: Flow diagram and Navigation



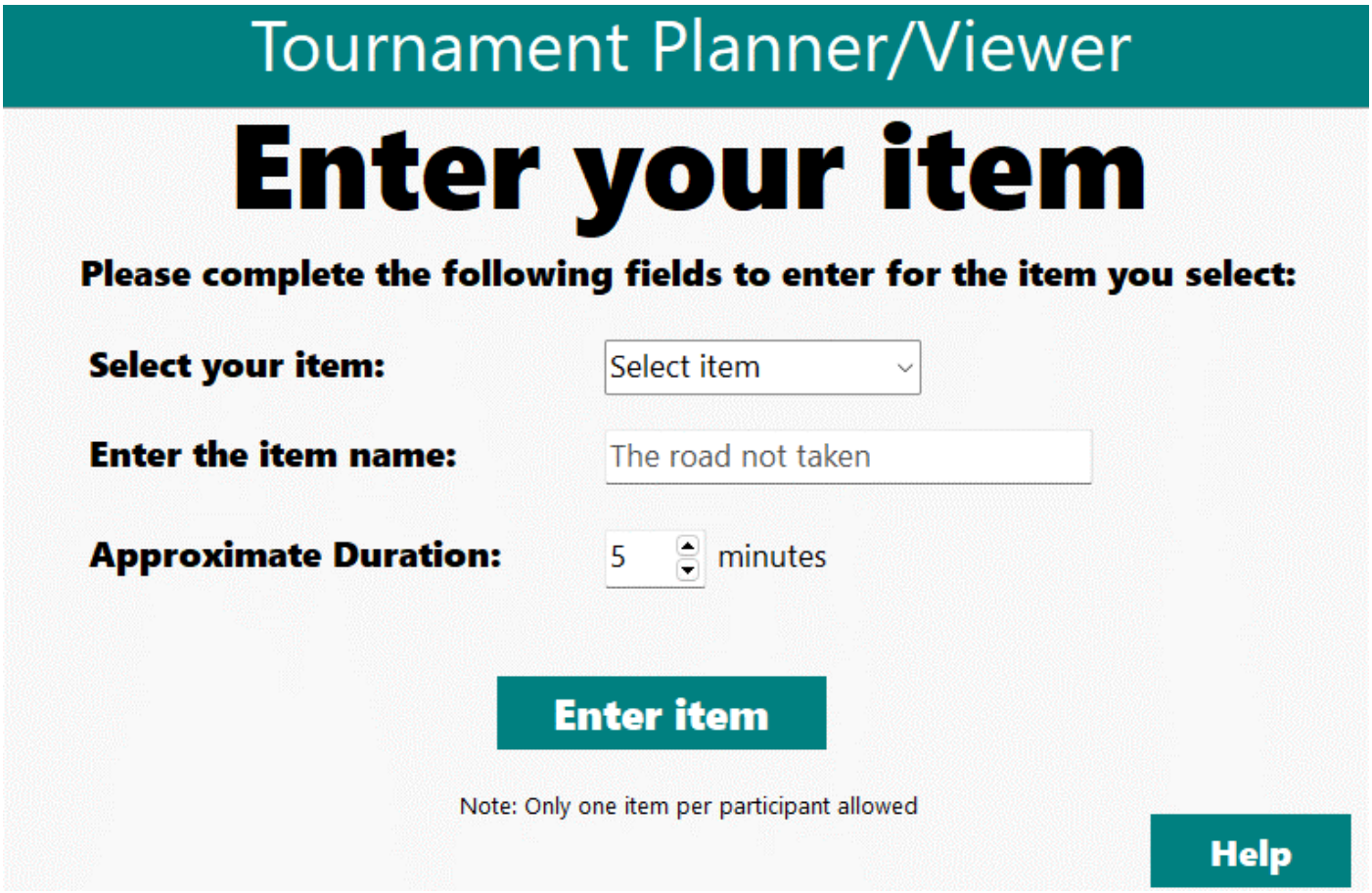
frmLogin – pnlSignUp



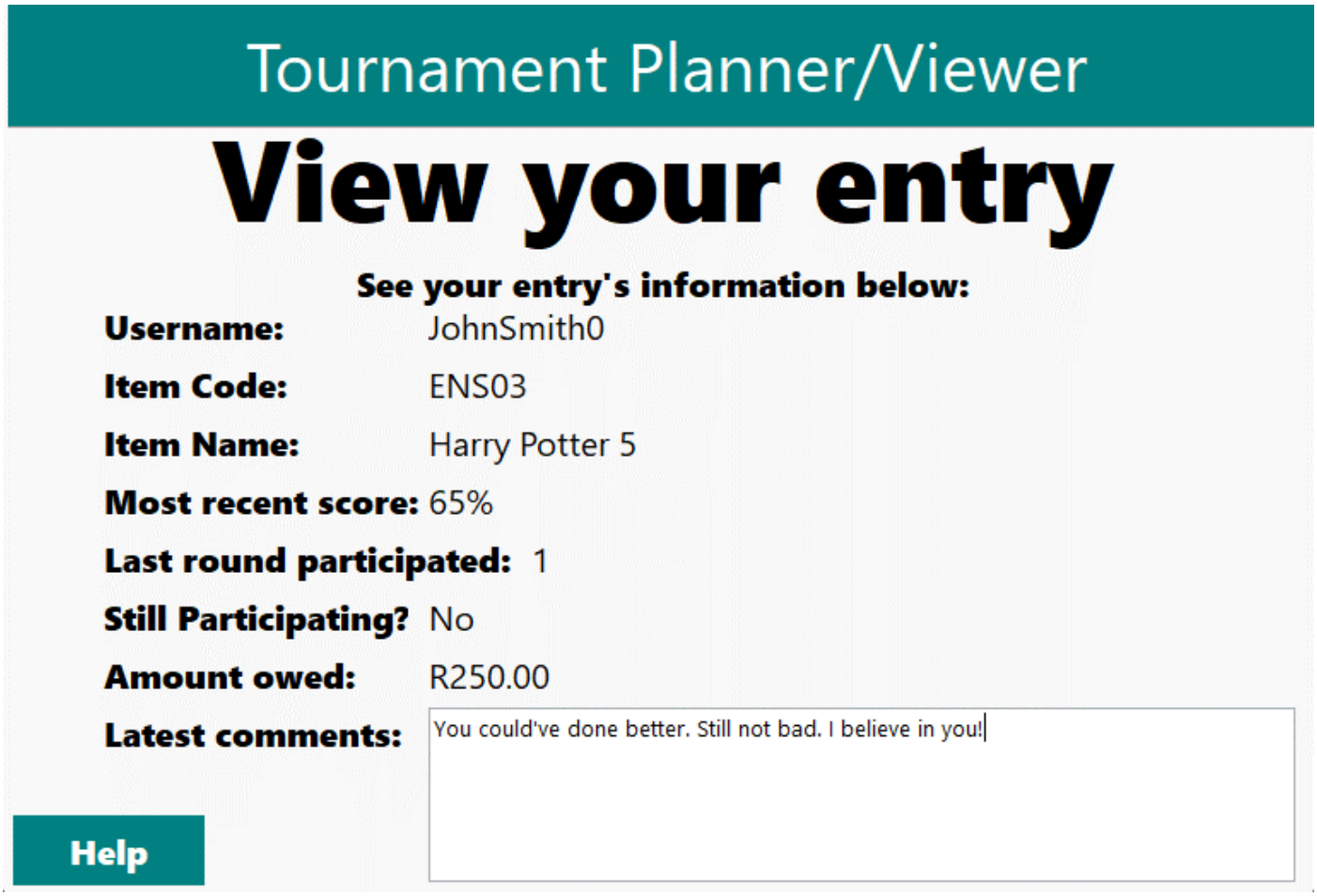
frmLogin – pnlLogin



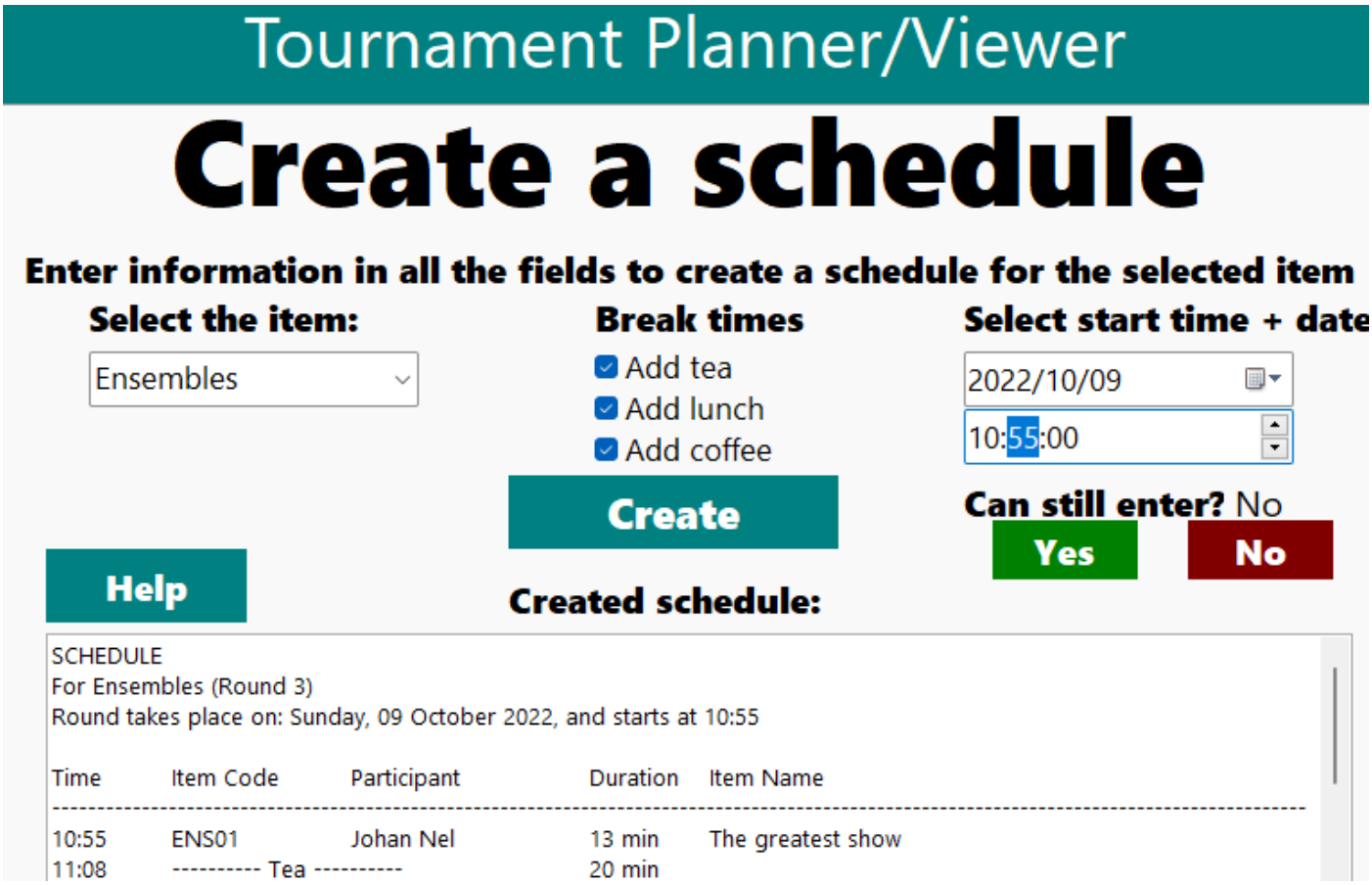
frmView – tbsEnter



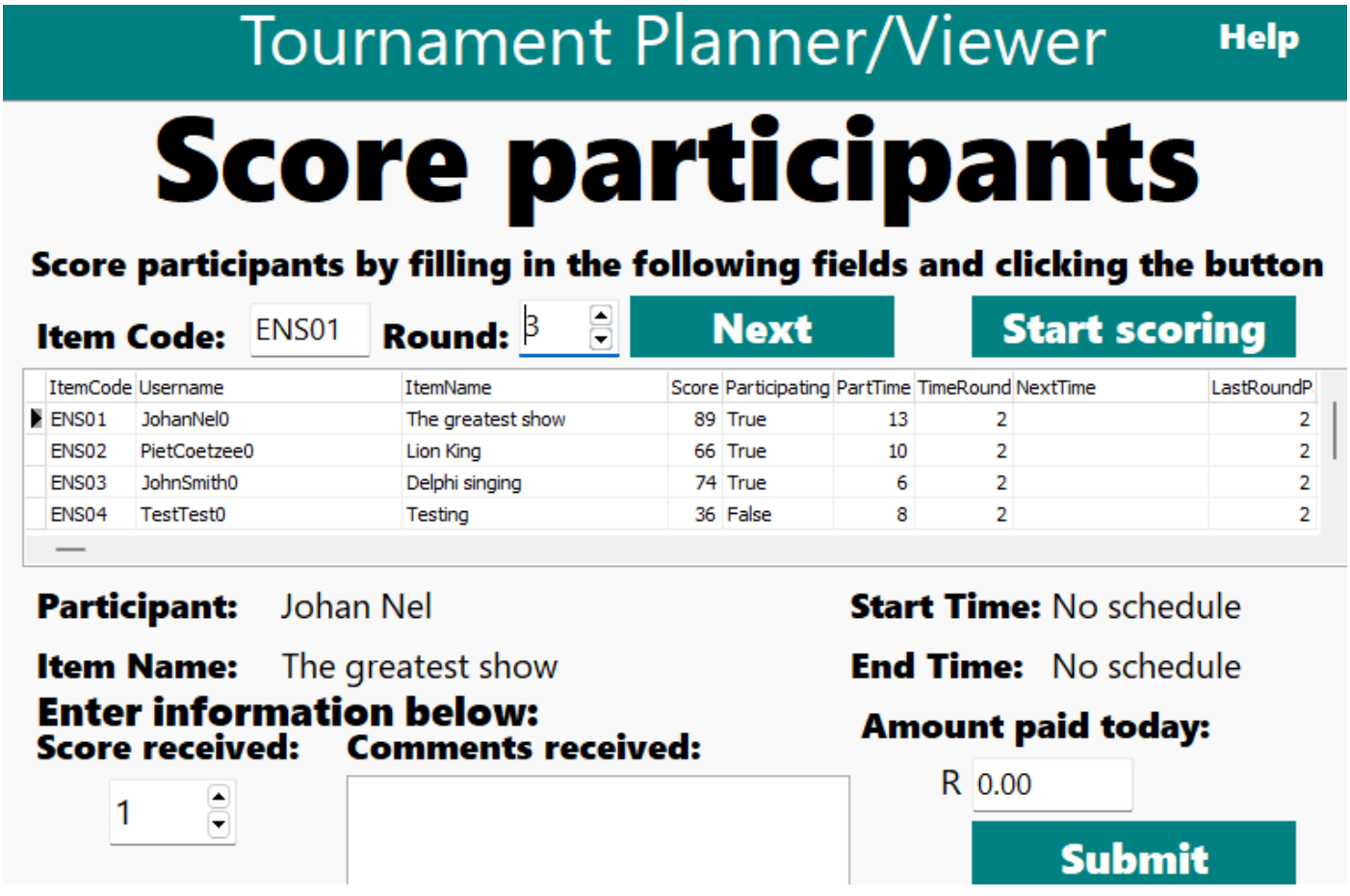
frmView – tbsView



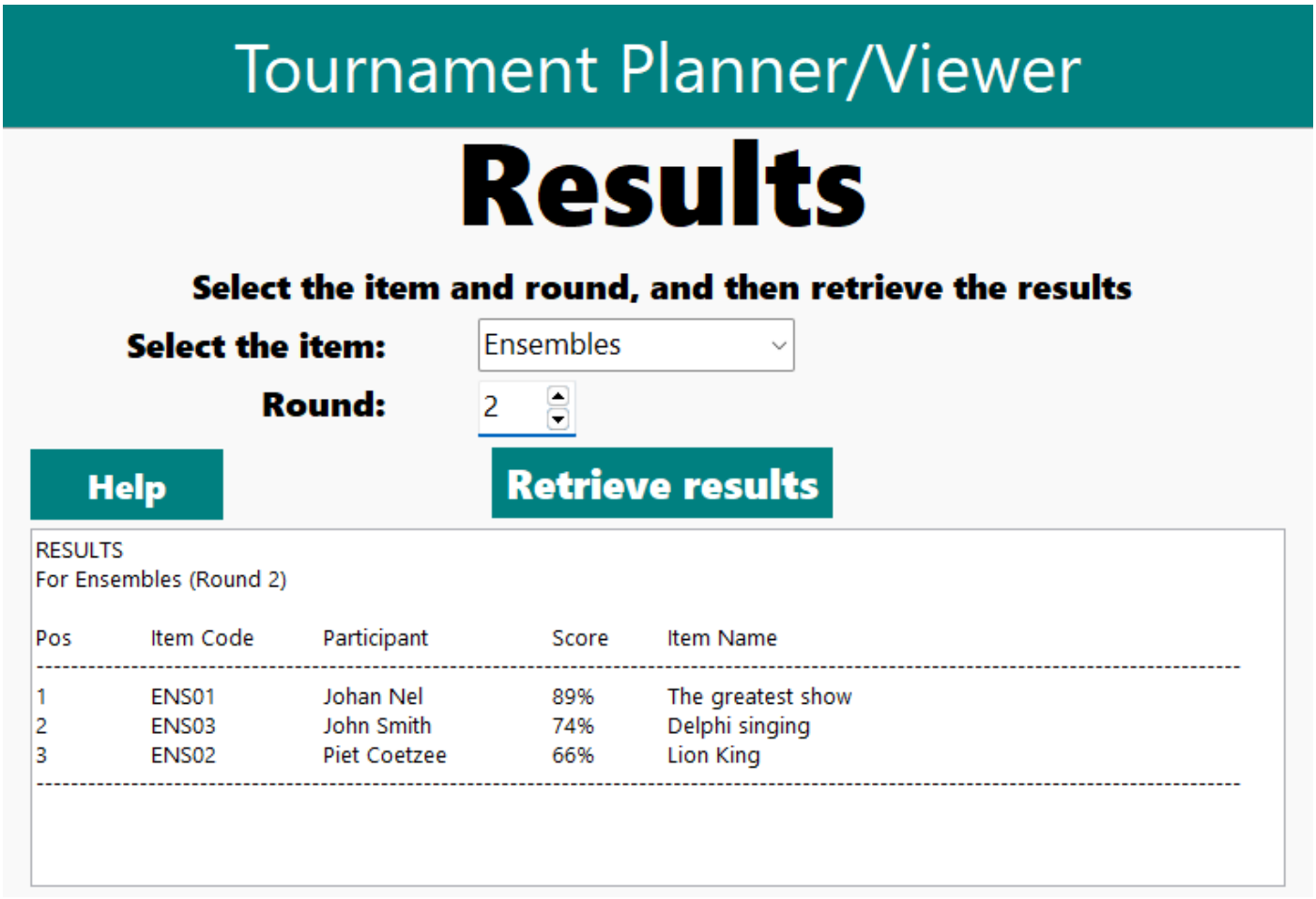
frmManage – tbsSchedule



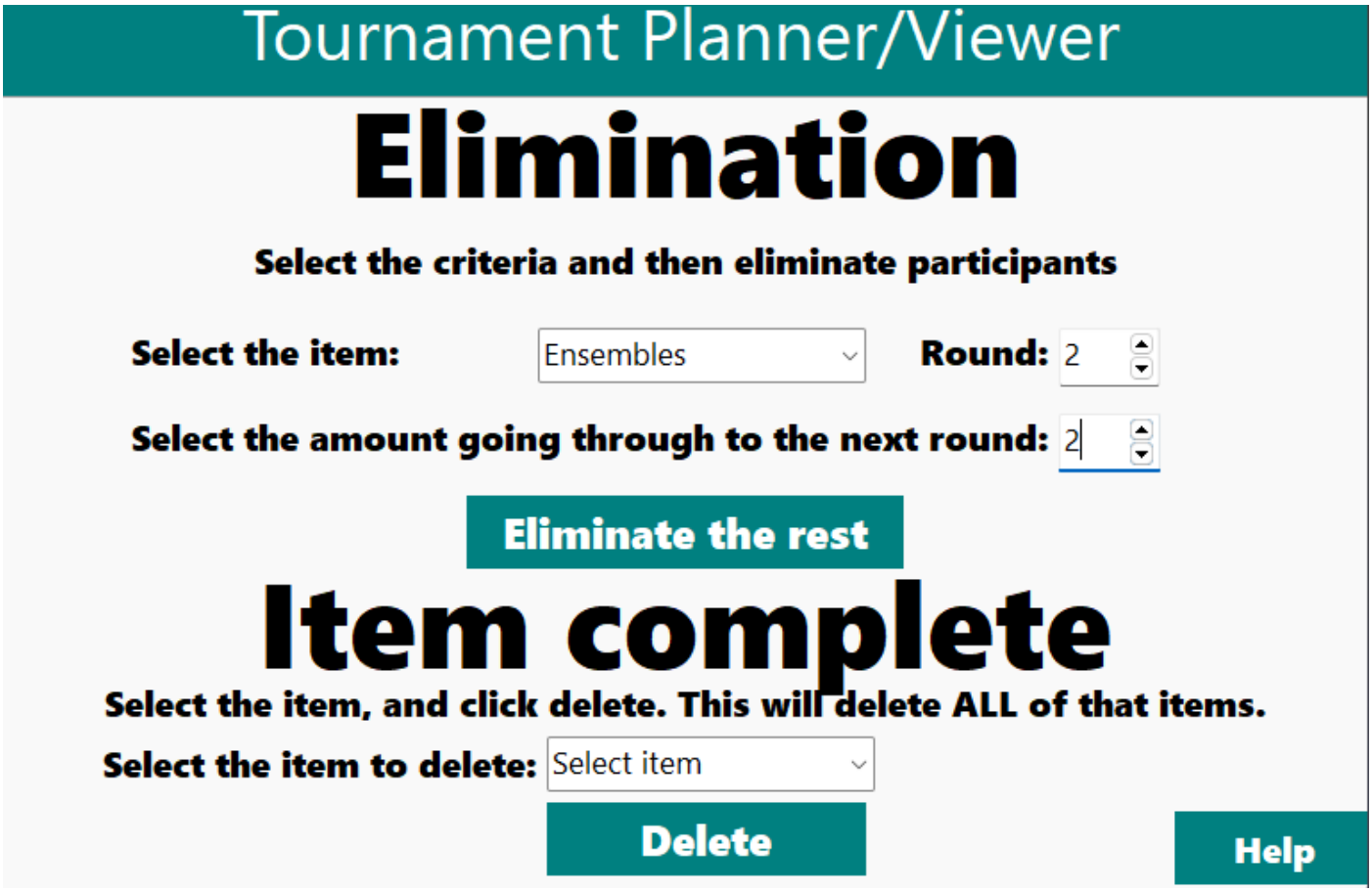
frmManage – tbsScore



frmManage – tbsResults



frmManage – tbsEliminate



Task 4: Data structures

Note: Variables and components for each GUI can be found in Task 5. I grouped all of the information for each panel/tab together to make the document easier to understand.

Text files – Includes Input, Processing and Output

Multiple text files will be used in my program to store data for later use.

- **Teachers.txt**

  Format: [Name] + [Surname]

  When a user signs up, the program will first loop through the text file. If the name and surname the user entered when signing up corresponds with a name and surname in the text file, it means that they are a teacher at the school. When signup is complete, the program will give the user the option to be a tournament manager/organiser or tournament viewer.

  Extract:

```
1  Maria Dobbin
2  Eloise Dorais
3  Sandra Jackson
```

- **Items.txt**

  Format: [Item name]

  This text file stores all the items that participants can enter. When logging in, the program loads all the items in this text file into the combo boxes that can be used to enter, and that organisers use to manage the tournament. The items are sorted alphabetically in the combo boxes

  It is not recommended to change anything in this file while the eisteddfod is in progress. It is however handy to change these items when a new eisteddfod, with different items, starts.

  Extract:

```
1  Comedy
2  Debating
3  Ensembles
```

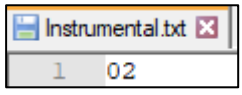
- **Items\[ItemName].txt**

  The program will automatically create a text file with the item name in the correct folder when a user enters a certain item for the first time. This text file keeps track of the latest item code for each specific item.

  For example: The first user enters Poetry. The program automatically creates Poetry.txt and adds code 00 in its first line. In the same process it will read the text file, take the existing code and add 1 in the first line. Now, Poetry.txt contains only 01. The user's ItemCode will be POE01. Now, a second user enters Poetry. Since Poetry.txt now exists, it will read the line, which is 01, add one, which makes it 02, and then save it. Now, Poetry.txt contains only 02, and this user's item code is now POE02. When the user deletes all of the participant's item info, the relevant ItemName file will also be reset.

  It is not recommended to change anything in these files while the eisteddfod is in progress. It is however recommended to delete these files once the eisteddfod is over to ensure that each new eisteddfod uses brand new item codes. With code, the program does this automatically.

  Extract:



  Format: [Last number used for item]

- **`Current.txt`**

  Format: [Username]

  This text file has a simple purpose. It stores the username of the currently logged in user, or the user that last logged in.

  Extract:

  Current.txt

  ```
  1  JohanNel16
  ```

- **`Scores\[Username].txt`**

  These text files are created after the first eisteddfod round. It keeps track of all the user's previous scores, and each score's respective comments.

  It is recommended to delete these files once the eisteddfod is over so that these do not unnecessarily take up device storage.

  Extract:

  ```
  1  Round 3
  2  Score: 65
  3  Comments: You could've done better. Still not bad.
  ```

- **`Help\[FormName].txt`**

  These text files are used to load information for the help button. Instead of storing help information in the program code, it is stored in these files. When a help button is clicked, the program will use a procedure to load the text file and display its contents in a MessageDlg. It is used to inform the user on what they should do if they struggle.

  Admins can change the content of these files whenever they want.

  Extract:

  SignUp.txt

  ```
  1  The purpose of the signup
  2  Ensure that you have enter
  3  After that, click the Sign
  ```

- **`Schedules\[ItemName][Round].txt`**

  These text files are created to store a copy of all the schedules that the organiser created with the built-in tool.

  Extract:

  Ensembles3.txt

  ```
  1  SCHEDULE
  2  For Ensembles. Round 3
  3  Round takes place on: Wednesday, 05 October 2022, and starts at 08:00
  4
  5  Time     Item Code    Participant Duration     Item Name
  6  ------------------------------------------------------------------------
  7  08:00    ENS01    Johan Nel    5 min    The Lord of the Rings
  ```

- **`Results\[ItemName][Round].txt`**

  These text files are created to store a copy of all the results that the organiser retrieved from the database with the built-in tool.

Extract:

Ensembles2.txt

```
RESULTS
For Ensembles (Round 2)

Pos Item Code   Participant Score   Item Name
-----------------------------------------------
1   ENS01   Johan Nel   89% The greatest show
```

- **Enter.txt**

  This text file contains either a 'Yes', or a 'No'. If it contains a 'No', participants can no longer enter the eisteddfod.

Extract:

Enter.txt

```
No
```

- **Documentation**

  To ensure that you know which files you may edit, delete, and use, please read through the application documentation, found here.

Arrays – Includes Input, Processing and Output

- **Arrays to sort the Items.txt file**

  Purpose

  To ensure that users can easily find the item they want to enter, the program will always sort the contents of the Items.txt text file before it loads it into all the item combo boxes.

  Populate the array

  I will use code to retrieve all the item names in the Items.txt text file, to be loaded into the array.

  Example:

Pos	1	2	3	4	5	…30
arrItems	Mime	Instrumental	Poetry	Debating	Prose	…

  Declaration

```
arrItems : array[1..30] of string
```

- **Arrays to determine who will be eliminated**

  Purpose

  To sort scores from high to low, parallel arrays will be used. This ensures that the participants with the highest scores, inputted by the organiser, will go through to the next round.

  Populate the array

  I will use tblScores to populate the arrays. Only still participating users, that participate in the selected item will be loaded into the array.

Pos	1	2	3	4	5	…n
arrScore	89	74	66	56	…	n
arrUser	JohanNel0	JohnSmith0	PietCoetzee0	JanGrobler0	…	n

Declaration

```
arrScore : array[1..1000] of Integer;
arrUser : array[1..1000] of String;
```

Subroutines – Mostly Processing

Multiple subroutines will be used to make programming easier:

- **Encrypt**

Purpose:

This simple **function** will be used to encrypt and decrypt passwords. Its purpose is to prevent the tournament organisers from seeing the user's original password. The function receives the password string as input, and then returns a string of the password encrypted or decrypted password.

Copy of code:

```
Result := sPW;
for k := 1 to Length(sPW) do
   Result[k] := chr(NOT(ord(sPW[k])));
```

Explanation:

The line of code is executed for every character of the password. First, it retrieves the character's ASCII value using *Ord* (a built-in function). It then uses the *NOT* operator to 'invert' this value. When you use the *NOT* operator on a number, it reruns the negative of the number, and minuses one. After it 'inverted' the character's ASCII code, it uses *Chr* (a built-in function) to return a new and encrypted or decrypted character. These characters are usually Chinese or Arabic when used for encryption.

Example:

`sPW[k] = A;`

The ASCII value of A is 65. The program performs the NOT operator.

Formula: –(65) –1 = –66

After it 'inverted' the ASCII value, it returns a character which cannot be understood. This new and encrypted character is ŧ. The character is now encrypted.

Now, in rerverse:

`sPW[k] = ŧ;`

In reverse, the same process happens. The ASCII value, -66, gets 'inverted' with NOT.

Formula: – (–66) –1 = 65.

The character corresponding to the ASCII value of 65 character is A. The character is now decrypted.

Declaration:

```
function Encrypt(sPW:String):String;
```

- **DisplayItem**

Purpose:

This simple **procedure** changes the captions of the labels on the view tab sheet of frmView. It allows the user to see information about their item.

Explanation:

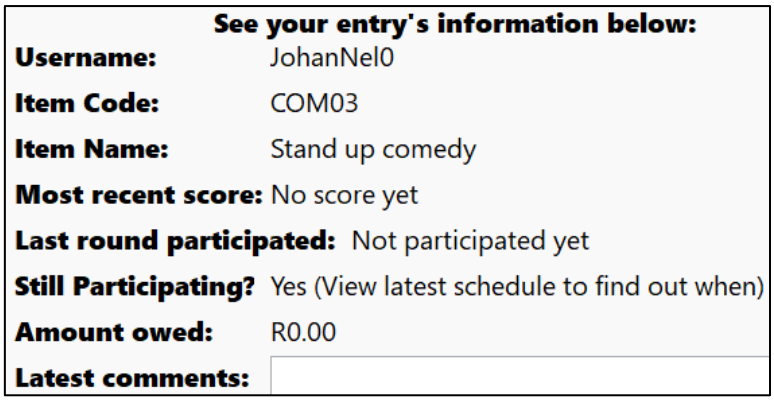
It retrieves information from the **tblScore** database table and displays it in the labels on tbsView.

Example:

When you run the procedure, the labels on tbsView change to reflect the currently selected user's item:

```
DisplayItem(JohanNel0);
```

Output:



Declaration:

```
procedure DisplayItem(sUsername:String);
```

- **Help**

Purpose:

This simple **procedure** is used to load help message from the Help text files. It shows the help message as a Information MessageDlg.

Copy of code:

```
//Test if the help text file is there
if FileExists('Help\'+sFile+'.txt') = False then
begin //if not show error
    iBut := MessageDlg('Help file not found! Contact an admin to fix this
error.',mtError,[mbAbort],0);
    Exit;
end;
  //If it is, assign and reset file pointer
  sMessage := '';
  AssignFile(FHelp,'Help\'+sFile+'.txt');
  Reset(FHelp);
  //generate a string help message
while NOT EOF(FHelp) do
begin
  ReadLn(FHelp,sLine);
```

```
  if sMessage = '' then //if it is the first line in the message, don't add
a 'line' in the message
  begin
    sMessage := sLine;
  end//if
  else
  begin
    sMessage := sMessage + #10 + sLine;
  end; //else
  end;//while
  //show the retrieved help message
  iBut := MessageDlg(sMessage,mtInformation,[mbOk],0);
  Closefile(FHelp);
```

Explanation:

When this procedure is called, it will first test if the help text file exists. If it does not, an error message it showed, and the procedure is exited.

If it does exist, the procedure assigns the text file to a variable. It then starts to generate the help message using a while loop. If the message is empty, it will not add a line/paragraph marker. If it contains text, it will add a paragraph marker for each new line that it read.

Afterwards it will show the help message with a MessageDlg, and close the text file.

Declaration:

```
procedure Help(sFile:String);
```

- **LoadItem**

Purpose:

This **procedure** loads the currently selected item's information into the relevant label's captions.

Explanation:

It retrieves the information about the currently entered ItemCode from **tblScore** and displays it in the captions on tbsScore

Output:

Participant:	Johan Nel	**Start Time:**	No schedule
Item Name:	The greatest show	**End Time:**	No schedule

Declaration:

```
procedure LoadItem(sItemCode: String);
```

- **Variables and components**

To organise this document, I put each form's variables and components underneath the form name. Find the Sign-Up panel variables and components here.

Task 5

Sign Up panel:

GUI can be found here.

- **Components and variables**

  Panel:
  pnlTitleSignUp – This panel is on all of the forms/panels/tab sheets.

  Labels:
  lblSignUp and lblSignUpInfo – These are displayed below the panel above. It gives the user instructions on what to do.

  lblGoLogin – This blue label allows the user to go back to the login form/panel if they already have an account. It serves as a button.

  lblDTP – Above the date time picker.

  Labelled edits:
  pnledtName, pnledtSurname, lbledtRewPW – The user inputs their information in these labelled edits.

  Date-time picker:
  dtpDOB – The user inputs their birth date in this to ensure they input it in the correct format.

  Image:
  imgRegisterEye – If the user clicks this image, it changes the password character of the password edit.

  Panel button:
  pnlbtnReg – This panel that serves as a button executes the code that runs when the user signs up. Variables in the OnClick procedure include:

  - **String:** sName, sSurname, sPassword, sUsername, sTest, sLine
  - **Date:** dDOB
  - **Integer:** iAge, iBut, k, iDateComp
  - **Boolean:** bFound, bTest, bSpace
  - **TextFile:** FTeach

- **Input and Validation**

Input variable	Source	Data type	Format	GUI Component	Validation
sName	Keyboard	String	General text (between 2 and 20 chars)	lbledtName	If the *sName, sSurname, sPassword* is blank. **Error:** You need to enter information in all of the fields to sign up. If *sName, sSurname, sPassword* contains space. **Error:** One of the fields you entered contains a space. Remove the space to continue.
sSurname	Keyboard	String	General text (between 2 and 20 chars)	lbledtSurname	
sPassword	Keyboard	String	General text (between 5	lbledtRegPW	

			and 30 chars)		If *sName* or *sSurname* is longer than 20 chars, or *sPassword* is longer than 30 chars. **Error:** The name and surname fields must have 20 characters or less, and the password must have 30 characters or less. Please make sure your fields aren't too long. If *sName* or *sSurname* is shorter than 2 chars, and *sPassword* is shorter than 5 chars **Error:** The name and surname fields must have a length of at least 2 characters, and the password must have at least 5 characters. Please make sure your fields have enough characters.
dDOB	Mouse/ Keyboard and DTP	Date	Short Date: yyyy/MM/dd	dtpDOB	If the date is a future date **Error:** The date you entered is a future date. Please enter a date that has already passed to sign up.
sLine	Text file	String	[Name] + [Surname]	None (Text File)	If the Teachers.txt file is not found **Error:** The Teachers.txt file needed to complete sign up could not be found. Contact an admin to fix this error.
iBut (Determines if the user wants to be an organiser or viewer)	Mouse clicking button	Integer	Click of a Yes or No button	MessageDlg	If the user's name is found in the Teachers.txt file, the message with Yes and No buttons will be displayed. (No error as user clicks predefined buttons)

- **Processing**

What processing will be done?	How will processing be done?
Calculate age	`iAge := YearsBetween(dDOB, Date);`
Generate username: sUsername **Format:** Name + Surname + Age	`sUsername := sName+sSurname+ IntToStr(iAge);`
Add user information to database tblUsers	`if bFound = False then` `begin` `  tblUsers.Insert;` `  tblUsers['Username'] := sUsername;` `  tblUsers['Password'] := Encrypt (sPassword);` `  tblUsers['First_Name'] := sName;` `  tblUsers['Last_Name'] := sSurname;` `  tblUsers['Birth_Date'] := dDOB;` `  tblUsers['User_Type'] := 'V';//default: V - tournament viewer`

	tblUsers.Post //Go to login panel pnlLoginForm.Show; pnlSignUpForm.Hide; pnlLoginForm.Top := 0; pnlLoginForm.Left := 0; lbledtUsername.SetFocus; lbledtName.Clear; lbledtSurname.Clear; dtpDOB.Date := Date; lbedtRegPW.Clear; end;
Test if the user is a teacher	sTest := sName+' '+sSurname; bTest := False; while (NOT EOF(FTeach)) AND (bTest = False) do begin readln(FTeach, sLine); if sLine = sTest then begin bTest := True; iBut := MessageDlg('We found your name in the school''s teacher list. Do you want to be a tournament organiser?' ,mtConfirmation,[mbYes,mbNo],0); if iBut = mrYes then begin tblUsers.Edit; tblUsers['User_Type'] := 'M'; tblUsers.Post; ShowMessage('We have updated your profile and you are now a tournament Manager!'); End //if yes pressed else begin ShowMessage('You will now be a tournament viewer. Contact the admin to change your user type.'); end; //else - clicked no end; //if teacher found end; //while closefile(FTeach);
Change Password edit char type if clicked on eye image	//Eye image on Register form clicked if bRegisterShow = False then //If Password isn't visible - make it visible begin imgRegisterEye.Picture.LoadFromFile ('imgShow.jpg'); lbedtRegPW.PasswordChar := #0; bRegisterShow := True; end

	else //If password is visible - make it invisible begin imgRegisterEye.Picture.LoadFromFile ('imgHide.jpg'); lbedtRegPW.PasswordChar := '*'; bRegisterShow := False; end; end;

After the user inputs all of their information and it has been validated, the program will need to process the input to store it in the database.

The program will also use the *YearsBetween* function to determine the user's age.

The program will then create a username. The username is created with the following format: Name]+[Surname]+[Age as string]

Once a username has been generated, the program will check if it is already in the database. If it is, the program will give an error message which says that the user much change a characteristic of their username and try again. This is to ensure that each user's username is completely unique, as it is the primary key of tblUsers.

If the username isn't found in the database, the program will insert a record which contains the Username, Password, FirstName, LastName, BirthDate and UserType (default = 'V'). It will also display a message confirming that the data has been stored. This message will also show the generated username.

After the message has been displayed, the program will automatically clear the sign up panel and take you to the login panel. It will also check if your name is in the Teachers.txt text file. If it is, the program will show another message asking the user if they want to be a tournament Organiser or Viewer. If they want to be an organiser, the program will automatically change the UserType field to 'M'.

- **Output**

Data	Format	GUI Component
sUsername when signed up successfully. User will be taken to pnlLogin.	Text displayed in message: We have recorded your information and it has been stored. To log in, use the username: '*[sUsername]*' (without apostrophes), and the password you entered.	MessageDlg of type Confirmation with OK button GUI changes automatically to pnlLogin.
A message if the user's name is found in the Teachers.txt text file	Text displayed in message: We found your name in the school's teacher list. Do you want to be a tournament manager? (This doubles as Input)	MessageDlg of type Confirmation with yes and no buttons

Login Panel

GUI can be found here.

- ## Components and variables

  ### *Panel:*

  pnlTitleLogin – This panel is on all of the forms/panels/tab sheets.

  ### *Labels:*

  lblLogin and lblLoginInfo – These are displayed below the panel above. It gives the user instructions on what to do.

  lblGoSignUp – This blue label allows the user to go to the sign-up form/panel if they do not an account. It serves as a button.

  ### *Labelled edits:*

  pnledtUsername, pnledtPassword – The user inputs their login information in these labelled edits.

  ### *Image:*

  imgLoginEye – If the user clicks this image, it changes the password character of the password edit.

  ### *Panel button:*

  pnlbtnLogin – This panel that serves as a button executes the code that runs when the user attempts to log in. Variables in the OnClick procedure include:

  - **String:** sPassword, sUsername
  - **Integer:** iBut (used to determine which button the user clicked with *MessageDlg*)
  - **Boolean:** bFound
  - **TextFile:** FCurrent

- ## Input and Validation

Input variable	Source	Data type	Format	GUI Component	Validation
sPassword	Keyboard	String	General text (between 5 and 30 chars)	lbledtPassword	If the password is less than 5 chars **Error:** The password must be at least 5 characters long.
sUsername	Keyboard	String	String	lbledtUsername	If the username is less than 6 chars **Error:** The username must be at least 6 characters long.
iBut (Determines what button the user clicked when a MessageDlg is displayed)	Mouse clicking button	Integer	Click of a Yes or No button	MessageDlg	No error/validation as user clicks predefined buttons. Message only displayed to Organisers. Asks if they want to view or manage the tournament.

- **Processing**

What processing will be done?	How will processing be done?
Search for the username in the database tblUsers	bFound := False tblUsers.First while NOT (tblUsers.EOF) AND (bFound = False) do begin if tblUsers[‘Username’] = sUsername //If the username is found begin bFound := True if tblUsers[‘Password’] = sPassword then begin lbledtUsername.Clear; lbledtPassword.Clear; *Store username in Current.txt* *Give managers the option to either view or manage* end//if password matches else MessageDlg('The password entered does not match the username - Wrong password.',mtError,[mbAbort],0); tblUsers.Next end;//while
If the user logs in, check if they are a manager.	if dmTournamentData.tblUsers['User_Type'] = 'M' then begin //If the user type is manager iBut := MessageDlg('Manager, You now have the option to select if you want to manage, or if you want to view the tournament.'+#10+'Click ''Yes'' to manage, and ''No'' to view.',mtConfirmation,[mbYes,mbNo],0); if iBut = mrYes then begin begin //If manage, go to manage form frmLogin.Hide; frmManage.Show; end else begin frmLogin.Hide; frmView.Show; end;
Change Password edit char type if clicked on eye image	//Eye image on Register form clicked if bLoginShow = False then //If Password isn't visible - make it visible begin

	imgLoginEye.Picture.LoadFromFile ('imgShow.jpg'); lbedtPassword.PasswordChar := #0; bLoginShow := True; end else //If password is visible - make it invisible begin imgLoginEye.Picture.LoadFromFile ('imgHide.jpg'); lbedtPassword.PasswordChar := '*'; bLoginShow := False; end; end;

- **Output**

Data	Format	GUI Component
Message when logged in successfully. User will be taken to frmView.	Text displayed in message: You've logged in successfully. Welcome '+ tblUsers ['First_Name']	MessageDlg of type Confirmation with OK button GUI changes automatically to frmView.

Enter item tab sheet

GUI can be found here.

- **Components and variables**

Panel:

pnlTitleEnter – This panel is on all of the forms/panels/tab sheets.

Labels:

lblEnter and lblEnterInfo – These are displayed below the panel above. It gives the user instructions on what to do.

lblItem, lblItemName, lblApprozTime – These labels show what the user should enter into the edits next to them.

lblOne – This smaller label informs the user that they can only enter one item.

Combo box:

cmbItem – The user uses this combo box to select the item they want to participate in. Items are loaded from the Items.txt text file.

Spin Edit:

spnApproxT – The user can use this to enter the approximate duration of their item (in minutes). Minimum of 5 and maximum of 15 minutes.

Panel button:

pnlbtnEnter – This panel that serves as a button executes the code that runs when the user attempts to enter their item. Variables in the OnClick procedure include:

- **String:** sItem, sItemName, sLine, sCode
- **Real:** rOwed

- **Integer:** iApproxT, iBut, k, iNum
- **Boolean:** bFound
- **TextFile:** FScore

- **Input and Validation**

Input variable	Source	Data type	Format	GUI Component	Validation
sItem (Item Type)	Mouse/ Keyboard and Combo box	String	General text from one of the Combo box's items	cmbItem	If the user did not select an item **Error:** You need to enter information in all of the fields to enter.
sItemName	Keyboard	String	General text no longer than 50 chars	lbledtItemName	If the item name is blank **Error:** You need to enter information in all of the fields to enter. If the item name is longer than 50 chars. **Error:** The item name cannot be longer than 50 characters. Please make sure yours isn't too long.
iBut (Determines what button the user clicked when a MessageDlg is displayed)	Mouse clicking button	Integer	Click of a Yes or No button	MessageDlg	No error/validation as user clicks predefined buttons. Message only displayed if an error occurs.
iApproxT	Mouse/ Keyboard	Integer	Number from Spin edit	spnApproxT	None. The default value is 5, and the user can change this if they need to.
rOwed	Database	Real	Currency	Database tblScore	(No validation as data is retrieved from database)
sUsername	TextFile	String	General text no longer than 42 chars	Current.txt	(No validation as data is retrieved from Text file) If the username is already in the tblScore database table **Error:** You have already entered an item. You cannot do so twice.

- **Processing**

What processing will be done?	How will processing be done?
Search for the username in the database tblScore	bFound := False tblUsers.First while NOT (tblUsers.EOF) AND (bFound = False) do begin if tblUsers[‘Username’] = sUsername //If the username is found begin bFound := True iBut := MessageDlg('You have already entered an item. You cannot do so twice.',mtError,[mbAbort],0); Exit; end else tblScore.Next; end;//while
Retrieve and change the most recent item code for the specific item when the item is entered	//Create the file if it doesn’t exist, otherwise edit it AssignFile(FScore,'Items\'+sItem+'.txt'); if FileExists('Items\'+sItem+'.txt') then reset(FScore) else begin Rewrite(FScore); WriteLn(FScore,'00'); end; reset(FScore); readln(FScore,sLine); //Assigns code variable to the line in the text file sCode := sLine; //Convert that code to a number iNum := StrToInt(sCode); inc(iNum);//Increases code sCode := IntToStr(iNum);//convert it back to a string If the code is only 1 digit (ex 2), then it will add a 0 (now 02) before it to ensure that the item code is always 5 chars if Length(sCode)=1 then sCode := '0'+sCode; Rewrite(FSCore); WriteLn(FScore,sCode); CloseFile(FSCore);
Generate the item code	tblScore['ItemCode'] := Uppercase(Copy(sItem,1,3)) +sCode;

Sort tblScore after a new item is added	`tblScore.Sort := 'ItemCode ASC';`
Add user's item information to database tblScore	`tblScore.Insert;` `tblScore['ItemCode'] := Uppercase (Copy(sItem,1,3))+ sCode;` `tblScore['Username'] := sUsername;` `tblScore['ItemName'] := sItemName;` `tblScore['PartTime'] := iApproxT;` `tblScore['LastRoundPart'] := 0;` `tblScore['Score'] := 0;` `tblScore.Post;  lbledtSurname.Clear;` `dtpDOB.Date := Date;` `lbedtRegPW.Clear;` *`Take to Viewing tab sheet`* *`Change label captions`*

- **Output**

Data	Format	GUI Component
Message when item entered successfully.	Text displayed in message: You've entered your item successfully! Check the View tab above for more information.	MessageDlg of type Information with OK button GUI changes automatically to tbsView.

View item tab sheet

GUI can be found here.

- **Variables and components**

  Panel:
  pnlTitleView – This panel is on all of the forms/panels/tab sheets.

  Labels:
  lblView and lblViewInfo – These are displayed below the panel above. It gives the user instructions on what to do.

  lblUsername, lblCode, lblViewName, lblScore, lblLastPart, lblStillPart, lblOwed, lblComment – These labels show what the user can see in the labels next to them.

  lblFillUsername, lblFillCode, lblFillViewName, lblFillScore, lblFillLastPart, lblFillStillPart, lblFillOwed, lblFillComment – These labels show the user's entry's information that has been extracted from the tblScore table in the database.

  Rich edit:
  redComments – The user's latest comments on their item will be displayed in the rich edit.

- **Input and Validation**

Input variable	Source	Data type	Format	GUI Component	Validation
None. Label captions added without variables.	Database tblScore ['Username']	String	General text with a length of no more than 42 chars	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['ItemCode']	String	General text with length of 5 chars	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['ItemName']	String	General text with a length of no more than 50 chars	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['Score']	Integer	Integer	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['LastRound Part']	Integer	Integer	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['Participatin g']	Boolea n	True/False Yes/No	Database tblScore	(No validation as data is retrieved from database)
None. Label captions added without variables.	Database tblScore ['Comments']	String	General text with no length limit (Long text)	Database tblScore	(No validation as data is retrieved from database)

- **Processing**

What processing will be done?	How will processing be done?
Determine the value of rOwed	`rOwed := dmTournamentData.tblScore ['LastRoundPart']*250 - dmTournamentData.tblUsers ['Paid'];`
Determine what caption to display if not participated yet	`if dmTournamentData.tblScore['Score'] = 0 then begin lblFillScore.Caption := 'No score yet'; end else lblFillScore.Caption := IntToStr(dmTournamentData.tblScore ['Score'])+'%'; if dmTournamentData.tblScore ['LastRoundPart'] = 0 then lblFillLastPart.Caption := 'Not participated yet' else lblFillLastPart.Caption := dmTournamentData.tblScore ['LastRoundPart']; if dmTournamentData.tblScore ['Participating'] = False then lblFillStillPart.Caption := 'No' else lblFillStillPart.Caption := 'Yes (View latest schedule to find out when)';`
Can the user still enter their item?	`AssignFile(fEnter,'Enter.txt'); reset(fEnter); readln(fEnter,sEnter); if sEnter = 'No' then begin tbsView.Enabled := False; tbsEnter.Enabled := False; lblView.Caption := 'Too late!'; lblViewInfo.Caption := 'You cannot enter. The tournament has already started!'; lblView.Width := 766; lblViewInfo.Width := 766; end;`

- **Output**

Data	Format	GUI Component
Username	General text with no more than 42 chars.	lblFillUsername
Item code	General text with length of 5 chars	lblFillItemCode
Item name	General text with no more than 50 chars	lblFillItemName
Item score	Number displayed as percentage. If Score = 0, then caption will be: No score yet	lblFillItemScore
Last round participated in	Number If LastRoundPart = 0, then caption will be: Not participated yet'	lblFillLastPart
Still Participating?	Boolean value displayed as Yes/No If Participating = Yes, caption will be: Yes (View latest schedule to find out when)	lblFillStillPart
Amount owed	Real value formatted as currency	lblFillOwed
Comments	Long text in rich edit.	redComments

Schedule creator tab sheet

GUI can be found here.

- **Variables and Components**

Panel:

pnlTitleCreate – This panel is on all of the forms/panels/tab sheets.

Labels:

lblCreate and lblCreateInfo – These are displayed below the panel above. It gives the user instructions on what to do.

lblItem, lblLunch, lblTime – These labels show what the user should input into the respective combobox, checkbox and date time pickers.

lblOne – This smaller label informs the user that they can only enter one item.

Combo box:

cmbItem – The organiser uses this combo box to select the item they want to create a schedule for. Items are loaded from the Items.txt text file and sorted using arrays.

Checkbox:

cbxTea, cbxLunch, cbxCoffee – The user can use these to select if the schedule should include teatime, lunchtime, and later coffeetime.

Date-time picker:

dtpDate, dtpTime – Is used to select on what date, and at what time the round should start

Richedit:

redCreated – Used to show the output of the created schedule

Panel button:

pnlbtnEnter – This panel that serves as a button executes the code that runs when the user attempts to create a schedule. Variables in the OnClick procedure include:

- **String:** sItem, sTICode, sItemCode, sDash, sUsername, sParticipant
- **TextFile:** FSchedule
- **Integer:** k, iRound, iDash, iBut, iPrint
- **Boolean:** bLunchAdded, bTeaAdded, bCoffeAdded
- **TTime:** tStartTime, tNowTime
- **Array:** arrUsername (String)

pnlbtnYes, pnlbtnNo – These panels that serves as buttons allows the organiser to prevent participants from signing up. This ensures that no one can sign up after the tournament has started. Variables in the OnClick procedures include:

- **String:** sEnter
- **Text File:** FEnter

• Input and Validation

Input variable	Source	Data type	Format	GUI Component	Validation
sItem	Mouse/ Keyboard and Combo box	String	General text from one of the Combo box's items	cmbItem	If the user did not select an item **Error:** You need to select an item in the combobox to continue.
Processing: iUsernameC	Global variable	Integer	The number of users in arrUserna me	Global variable	If the variable = 0 then show error: **Error:** No participants for this item. Create a schedule for another item

• Processing

What processing will be done?	How will processing be done?
Add users to arrUsername and determine current round	`while NOT dmTournamentData.tblScore.EOF do begin` `  sTICode := Copy(dmTournamentData.tblScore ['ItemCode'],1,3);` `  if (dmTournamentData.tblScore ['Participating'] = True) AND (sTICode = sItemCode) then` `  begin` `  inc(iUsernameC);`

	`//All the usernames in the array will be those with the item selected, that are still participating...` `arrUsername[iUsernameC] := dmTournamentData.tblScore['Username'];` `iRound := iRound + dmTournamentData.tblScore['LastRoundPart'];` `end;//if` `dmTournamentData.tblScore.Next;` `end;//while` `if iUsernameC <> 0 then` `iRound := Round(iRound/iUsernameC)+1` `else` `begin` `iBut := MessageDlg('No participants for this item. Create a schedule for another item',mtError,[mbAbort],0);` `Exit;` `end;`
Determine if it is an appropriate time to add tea/lunch/coffee to schedule	`if (bTeaAdded = False) AND(HourOF(tNowTime) = 11) AND (cbxTea.Checked) then //Tea time` `begin` `bTeaAdded := True;` `redCreated.Lines.Add(FormatDateTime ('HH:NN',tNowTime)+#9'---------- Tea ----------'+#9+'20 min');` `tNowTime := IncMinute(tNowTime,20);` `end;`
Determine participant name and surname	`if (dmTournamentData.tblScore['Participating'] = True) AND (sTICode = sItemCode) then` `begin` `with dmTournamentData do` `begin` `tblUsers.First;` `sUsername := tblScore['Username'];` `while not tblUsers.EOF do` `begin` `if tblUsers['Username'] = sUsername then` `sParticipant := tblUsers ['First_Name']+' '+tblUsers ['Last_Name'];` `tblUsers.Next;` `end;//while`
Determine if users can still enter	`AssignFile(FEnter,'Enter.txt');` `reset(FEnter);` `readln(FEnter, sEnter);` `if sEnter = 'Yes' then` `begin` `lblYesNo.Caption := 'Yes';` `pnlbtnYes.Enabled := False;` `pnlbtnNo.Enabled := True;`

	`` end else if sEnter = 'No' then begin lblYesNo.Caption := 'No'; pnlbtnYes.Enabled := True; pnlbtnNo.Enabled := False; end; ```

- **Output**

Data	Format	GUI Component
Participant's information	Time + ItemCode + Participant name and surname + Approximate duration + ItemName	redCreated
Created schedule	Text directly from redCreated	Text file/Printer

Score participants tab sheet

GUI can be found here.

- **Variables and Components**

Panel:
pnlTitleScore – This panel is on all of the forms/panels/tab sheets.

Labels:
lblScore and lblInfoScore, lblScoreR, lblCOmmentR, lblPaidT – These are displayed below the panel above. It gives the user instructions on what to do/enter

lblItemCode, lblRound – These labels show what the user should input into the respective edit and spin edit.

lblParticipant, lblItemName, lblStart, lblEnd – These labels show what the labels next to them represent/display

lblOne – This smaller label informs the user that they can only enter one item.

Spin edit:
spnRound, spnScore – The organiser uses this to select the round the participant is currently participating in, and enter their score

Edit Box:
edtItemCode, edtPaid – The organiser uses this to enter the item code of the participant and how much they paid

DBGrid:
dbScore – Is used to see all the participants. Allows the organisers to change information

Panel button:
pnlbtnNext – When this panel that serves as a button is clicked, it runs code that automatically determines the next valid item code. Variables in the OnClick procedure include:

- **String:** sItem, sItemcode
- **Integer:** iBut, iRound
- **Boolean:** bFound

pnlbtnStart – This panel that serves as a button executes code that ensures the validity of the user's input. It must be clicked to load the user's information into the components. Variables in the OnClick procedure include:

- **String:** sItemcode
- **Integer:** iBut, iRound
- **Boolean:** bFound, bValid

pnlbtnSumbit – This panel that serves as a button executes code that post the current participant's scores into the database. Variables in the OnClick procedure include:

- **String:** sCommetns, sItemCode, sUsername, sParticipant
- **Integer:** iBut, iRound, iScore
- **Boolean:** bFound
- **TextFile:** FScore
- **Extended (Real):** rPaid

- **Input and Validation**

pnlbtnNext

Input variable	Source	Data type	Format	GUI Component	Validation
sItemCode	Keyboard in edit box	String	5 char uppercase code that represents the participant code	edtItemCode	If the user did not enter an item code **Error:** You need to enter an item code to continue.
iRound	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the round	spnRound	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 20

pnlbtnStart

Input variable	Source	Data type	Format	GUI Component	Validation
sItemCode	Keyboard in edit box	String	5 char uppercase code that represents the participant code	edtItemCode	If the user did not enter an item code **Error:** 'No item code entered'
iRound	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the round	spnRound	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 20

pnlbtnSubmit

Input variable	Source	Data type	Format	GUI Component	Validation
sItemCode	Keyboard in edit box	String	5 char uppercase code that represents the participant code	edtItemCode	If the user did not enter an item code **Error:** 'No item code entered'
iRound	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the round	spnRound	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 20
iScore	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the score	spnScore	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 100
rPaid	Keyboard in edit box	Real/ Currency	Currency	edtPaid	If the string in the edit box cannot be converted to real, show error: **Error:** 'Amount paid is in the wrong format. Do not add a "R", and ensure the decimal separator is correct.'

- **Processing**

pnlbtnNext

What processing will be done?	How will processing be done?
Determine the next valid item code	`tblScore.First;` `tblScore.Sort := 'ItemCode ASC';` `bFound := False;` `while not (tblScore.EOF) and (bFound = false) do` `begin` `if (Copy(tblScore['ItemCode'],1,3) = Copy(sItemcode,1,3)) AND ((tblScore['LastRoundPart']+1) = iRound)` `AND ((tblScore['Participating'])) AND (StrToInt(Copy(tblScore['ItemCode'],4,2)) > StrToInt(Copy(sItemcode,4,2))) then` `begin` `bFound := True;` `sItemCode := tblScore['ItemCode'];` `end else`

	tblScore.Next; end;//while

pnlbtnStart

What processing will be done?	How will processing be done?
Test if the currently entered item code if valid	while not (tblScore.EOF) and (bFound = false) AND (bValid = True) do begin if tblScore['ItemCode'] = sItemcode then begin bFound := True; end else tblScore.Next; end;//while if bFound = False then begin iBut := MessageDlg('Could not locate item code.',mtError,[mbAbort],0); Exit; end; if (tblScore['Participating'] = False) AND (bFound = True) then bValid := False; if ((tblScore['LastRoundPart']+1) <> iRound) AND (bFound = True) then begin bValid := False; end;

pnlbtnSubmit

What processing will be done?	How will processing be done?
Determine how much the user has paid in total	tblUsers['Paid'] := tblUsers['Paid'] +rPaid;

- **Output**

pnlbtnNext

Data	Format	GUI Component
The next valid item code	5 uppercase chars that represents a valid participant's item	edtItemCode

pnlbtnStart

This output is retrieved with the LoadItem procedure.

Data	Format	GUI Component
Participant	Name + Surname found in database	edtFillParticipant
Start Time	Short time (HH:NN)	edtFillStart
End Time	Short time (HH:NN)	edtFillEnd
Item name	Item name found in database	edtFillItemName

pnlbtnSubmit

Data	Format	GUI Component
Score	Score found in spnScore (Integer)	tblScore – Database
Comments	String found in redComments.Text	tblScore – Database
Last Round Participated	Integer	tblScore – Database

Retrieve results tab sheet

GUI can be found here.

- **Variables and Components**

  Panel:

  pnlResultTitle – This panel is on all of the forms/panels/tab sheets.

  Labels:

  lblResultTitle and lblResultInfo – These are displayed below the panel above. It gives the user instructions on what to do/enter

  lblResultItem, lblResultRound – These labels show what the user should select and enter into the combo box and spin edit next to them.

  Combo box:

  cmbResultItem – The organiser uses this combo box to select the item they want to create a result sheet for. Items are loaded from the Items.txt text file and sorted using arrays.

  Spin edit:

  spnReslutRound – The organiser uses this to select the round they want to retrieve results for.

  Richedit:

  redResults – Used to show the output of the retrieved results.

  Panel button:

  pnlbtnResults – This panel that serves as a button executes code that generates and retrieves results. Variables in the OnClick procedure include:

  - **String:** sItem, sTICode, sUsername, sParticipant, sItemCode
  - **Integer:** iRound, iPos, iBut, iPrint
  - **TextFile:** FResults;

- **Input and Validation**

Input variable	Source	Data type	Format	GUI Component	Validation
sItem	Mouse/ Keyboard and Combo box	String	General text from one of the Combo box's items	cmbResultItem	If the user did not select an item **Error:** You need to select an item in the combobox
iRound	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the round	spnResultRound	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 20

- **Processing**

What processing will be done?	How will processing be done?
Determine participant name and surname	`if (dmTournamentData.tblScore['Participating'] = True) AND (sTICode = sItemCode) then begin with dmTournamentData do begin tblUsers.First; sUsername := tblScore['Username']; while not tblUsers.EOF do begin if tblUsers['Username'] = sUsername then sParticipant := tblUsers ['First_Name']+' '+tblUsers ['Last_Name']; tblUsers.Next; end;//while`
Sort the database tblScore according to score, from highest to lowest	`dmTournamentData.tblScore.Sort := 'Score DESC'; dmTournamentData.tblScore.First;`

- **Output**

Data	Format	GUI Component
Participant's information	Position + ItemCode + Participant name and surname + Score in percent + ItemName	redResults
Created schedule	Text directly from redResults	Text file/Printer

Eliminate participants tab sheet

GUI can be found here.

- ## Components and variables

  Panel:

  pnlTitleElim – This panel is on all of the forms/panels/tab sheets.

  Labels:

  lblEliminate, lblElimInfo, lblComplete and lblCompleteInfo – These are displayed below the panel above. It gives the user instructions on what to do/enter

  lblElimItem, lblElimRound, lblElimCriteria, lblDeleteItem – These labels show what the user should select and enter into the combo box and spin edits next to them.

  Combo box:

  cmbElimItem and cmbDeleteItem – The organiser uses this combo box to select the item they want to eliminate or delete.

  Spin edit:

  spnElimRound – The organiser uses this to select the round they want to retrieve results for.

  spnTop – The organiser uses this to select the number of participants going through to the next round.

  Panel button:

  pnlbtnEliminate – This panel that serves as a button executes code that eliminates users. Variables in the OnClick procedure include:

  - **String:** sItem, sTemp, sThrough
  - **Integer:** iRound, iTop, iSize, k, l, iTemp, iBut
  - **Arrays:** arrScore (Integer), arrUser (String)

  pnlbtnDelete – This panel that serves as a button executes code that delete all participants that take part in the selected event. Variables in the OnClick procedure include:

  - **String:** sItem
  - **Integer:** iCount, iBut
  - **TextFile:** FScore

- ## Input and Validation

pnlbtnEliminate

Input variable	Source	Data type	Format	GUI Component	Validation
sItem	Mouse/ Keyboard and Combo box	String	General text from one of the Combo box's items	cmbElimItem	If the user did not select an item **Error:** You need to select an item in the combobox
iRound	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the round	spnElimRound	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 20

iTop	Keyboard/ Mouse with spin edit	Integer	Small Integer that represents the amount going through	spnTop	This does not get validated. However, the spin edit values are limited: **Min:** 1 **Max:** 1000
arrScore	tblScore (Database)	Integer	Represents the user's score	Database	None
arrUser	tblScore (Database)	String	The user's username	Database	None
iSize (Processing)		Integer	How many participant	None	If it is 0 then show error **Error:** No participants found. Ensure the round you entered is correct for the program to correctly identify participants.

pnlbtnDelete

Input variable	Source	Data type	Format	GUI Component	Validation
sItem	Mouse/ Keyboard and Combo box	String	General text from one of the Combo box's items	cmbDeleteItem	If the user did not select an item **Error:** Please select an item in the combobox

- **Processing**

pnlbtnEliminate

What processing will be done?	How will processing be done?
Sort arrays descending according to scores Selection sorting	`for k := 1 to iSize - 1 do` `  for l := k+1 to iSize do` `  begin` `  if arrScore[k] < arrScore [l] then` `  begin` `  iTemp := arrScore[k];` `  arrScore[k] := arrScore[l];` `  arrScore[l] := iTemp;` `  sTemp := arrUser[k];` `  arrUser[k] := arrUser[l];` `  arrUser[l] := sTemp;` `  end;` `  end;`

Determine the top participants in correct order	with dmTournamentData do begin tblScore.First; for k := 1 to iTop do begin if tblScore.Locate('Username',arrUser[k],[]) then begin tblScore.Edit; tblScore['Participating'] := True; tblScore.Post; sThrough := tblScore['Username'] + ', '+sThrough; end;//if end;//for end;//with

pnlbtnDelete

What processing will be done?	How will processing be done?
Count how many items will be deleted	with dmTournamentData do begin tblScore.First; tblScore.Sort := 'ItemCode ASC'; while not tblScore.EOF do begin if (uppercase(Copy(sItem,1,3)) = copy(tblScore['ItemCode'],1,3)) then begin inc(iCount); end;//if tblScore.Next; end;//while
Delete records	while not tblScore.EOF do begin if (uppercase(Copy(sItem,1,3)) = copy(tblScore['ItemCode'],1,3)) then begin tblScore.Delete; end else//if tblScore.Next; end;//while
Reset the item's text file to ensure new items restart at 01	AssignFile(FScore,'Items\'+sItem+'.txt'); if FileExists('Items\'+sItem+'.txt') then begin Rewrite(FScore); WriteLn(FScore,'00'); Closefile(FScore); end;

- **Output**

pnlbtnEliminate

Data	Format	GUI Component
The participants going through to the next round	The following users are through to the next round: + sThrough +The rest have been eliminated.	MessageDlg (Information)

pnlbtnDelete

Data	Format	GUI Component
The number of records deleted	iCount+ records deleted	MessageDlg (Information)

Use case diagrams

Please find the two top level use case tables under Task 1B.

Extra information

Databases

Find extracts of databases here.

Credits

- **Mr Long** has helped me with my PAT significantly. A full playlist of his videos can be found here.
- **Delphi basics:** https://www.delphibasics.co.uk/ is a must-use resource for any IT learner. It clearly explains all the properties and variables of functions, and mostly everything in Delphi! Quite handy to help you learn new things. I used Delphi Basics to learn how to use the MessageDlg built-in sub-routine
- **Encrypt:** https://edn.embarcadero.com/article/28325 was used for the Encrypt function.

Delphi plugins and external tools

No third-party plugins or Delphi related external tools were used for the program at the time of writing.

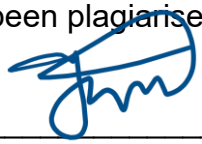
Declaration of authenticity

PRACTICAL ASSESSMENT TASK (PAT)

Learner Name: Johan Nel
Grade 11
Year: 2022
ID Number: 0511235121080

Teacher: A. Ferreira

I hereby declare that the contents of this assessment task are my own original word (except where there is clear acknowledgement and appropriate reference to the work of others) and have not been plagiarised, copied from someone else or previously submitted for assessment by anyone.



_____ 16 / 10 / 2022

Signature of learner Date

Program documentation

This program includes detailed documentation which can be found here:

Application documentation.doc

It is recommended that you read through this whole document to ensure you can use the program to its full potential without issues. Most of this document's information has also been loaded into the Help buttons.

On top of that, a detailed video demonstrating the program has been created. Find it here:

Program Demonstraiton vide

Youtube: https://youtu.be/EJMGtWttWfw