***TASK 1A:***

The task provided is for the programmer to select a tournament style and implement it into a Delphi program, with a GUI made up of at least two forms or screens (allowing for navigation), utilising a database with at least two tables, and making proper use of arrays and a text file. This program will need to sort through all members/players in the database using the chosen tournament style and find the winner within the number of rounds specified by the chosen tournament style. The program is targeted at the organiser of the event, and as such will need to contain an administrator login with its own specific (and higher) access rights when compared to the “standard” user access rights. My program will utilise a single-elimination tournament style in order to instantaneously organise all results entered in the database and provide the user with the podium positions (1st, 2nd, 3rd) as output. In accordance with the chosen tournament style, it will only have one “round” wherein the podium positions will be chosen. The program will also ensure that the user has a seamless GUI navigation experience, with a fully functional and editable database (this will be accomplished through the use of SQL to update, delete, and allow new entries).

***TASK 1B:***

|  |  |  |
| --- | --- | --- |
|  | General User | Administrator |
| Role | The role of the general user is to log in to the program and view or alter their own record in the database. | The role of the administrator is to log in to the program and view or alter any record in the database, as well as be able to see a changelog of all actions performed by General Users. |
| Activity | Either input, update or delete their own record in the database. | The administrator will be able to monitor changes to each database entry, oversee the General User’s ease of access and ensure all records follow the rules set out for data entry into the database by the organisation in control, through either inputting, updating or deleting these records. |
| Value | The program will provide a seamless, easy-to-use and effective method to enter, edit and compare scores. | The program will provide monitoring capabilities and the ability to alter records made by general users |

**UML diagram**

Administrator

General User

Register Form

Login

Manage all records

Home

Access to changelog

View podium positions

Manage own record

Insert any record

Update any records

Delete any records

Insert own record

Update own record records

Delete own records

**Navigation Flow Diagram**

Manage own record

Insert own record

Update own record records

Delete own records

View podium positions

END

Login

START

If Admin = true

If Admin = false

Home

Manage all records

Insert any record

Update any records

Delete any records

Access to changelog

Register Form

|  |  |  |
| --- | --- | --- |
| Function | General User | Administrator |
| Inserting a record | The program will allow the general user to insert their own record after the appropriate button press and data entry, granted the user does not already have a record in the database. | The program will allow the administrator to insert a record for/of any user after the appropriate button press and data entry. |
| Updating a record | The program will allow the general user to update their own record after the appropriate button press, and data entry. | The program will allow the administrator to update a record for/of any user after the appropriate button press, and data entry. |
| Deleting a record | The program will allow the general user to delete their own record after the appropriate button press, and confirmation of deletion by the user. | The program will allow the administrator to delete a record for/of any user after the appropriate button press, and confirmation of the deletion. |
| Displaying the podium positions (1st, 2nd, 3rd) | The program will organise the data in the database and find the podium positions, which it will then display. | The program will organise the data in the database and find the podium positions, which it will then display. |
| Login with authorisation | Allows general users to log in to the program and will grant them base access rights to the database (insert, update and delete their own record) | Allows administrators to log in to the program and will grant them higher access rights to the database (insert, update and delete any record) |
| Changelog | The program will log each action that affects the database that is performed by the general user. | The program will display a log of the general user’s actions that affect the database to the administrator. |

tblPlayers

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Size** |
| ID | AutoNumber | Long Integer |
| last\_name | Short Text | 25 |
| first\_name | Short Text | 25 |
| birth | Date/Time | - |
| phone | Short Text | 15 |
| email | Short Text | 255 |
| password | Short Text | 24 |

tblGames

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Size** |
| ID | AutoNumber | Long Integer |
| player1\_id | Number | Double |
| player2\_id | Number | Double |
| player1\_score | Number | Double |
| player2\_score | Number | Double |
| winner | Short Text | 1 |

tblShirtSize

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Size** |
| ID | AutoNumber | Long Integer |
| Size | Short Text | 5 |

tblGender

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Size** |
| ID | AutoNumber | Long Integer |
| type | Short Text | 7 |

Graphical user interface

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Item | Description | Purpose |
| Datasource.First; | Navigates to the first entry in the database. | Commonly used in conjunction with Datasource.Eof to loop through the entire database. |
| Datasource.Last; | Navigates to the last entry in the database. | Used to find the latest ID in the system and then be increased by 1 and added to a variable to ensure that there are no duplicate AutoNumber values in the database. |
| Datasource.Next; | Navigates 1 entry downwards in the database. | Commonly used within the loop of Datasource.Eof and Datasource.First to move the program to the next entry in the database and extract the necessary data. |
| Datasource.Prior; | Navigates 1 entry upwards in the database. | Used to help the general user/administrator navigate through the database. |
| Datasource[‘example’] := input; | Input describes the data that will be inputted into the database field on the currently selected row. This data will replace the previous data present in that field and essentially “Update” the field to the new data. | Used to replace outdated data present in a field. Note: Administrators will be able to use this functionality for any record, while general users will be limited to their own record. |
| Datasource[‘example’] | Selects the data stored in that field, on the currently selected row. | Data can be displayed or manipulated through this functionality. |
| Datasource.Delete; | Deletes the currently selected record. | Used to delete records of players who no longer participate in the event, or delete records that violate the rules of the host organisation. |
| Datasource.Locate(‘example’, searchfor,[]) | Used to search for a variable/entry within a specific database field. Returns a Boolean. | Used to validate if a user already exists within a database. |

***DATA DICTIONARY***

**Text files:**

Text files allow for quick and simple data storage, as there are no specific fields to input into, and no field requirements. Due to this ease of use, text files can be used to generate reports by any user, these reports will contain field types of the user’s choice. The admin will also have access to an encrypted text file that contains a changelog.

**Arrays:**

The array will be loaded with the scores of all winners, by use of a while loop. The array will then be sorted using a fast and efficient sorting method. The reordered array will then have the podium positions in the first 3 positions of the array, which will then be displayed to the user when requested.

|  |  |  |
| --- | --- | --- |
| **Variables** | **Data Types** | **Local/Global** |
| con | TADOConnection | Global |
| tblPlayers | TADOTable | Global |
| dbsMembers | TDataSource | Global |
| tblGames | TADOTable | Global |
| dbsGames | TDataSource | Global |
| qry | TADOQuery | Global |
| dbsSQL | TDataSource | Global |
| sSQL | String | Global |
| iLoop | Integer | Local |
| aLine | String | Local |
| idNum | Integer | Local |
| tf | Text File | Local |

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Input | Process | Output |
| cboGenders | When the “Validate” button is clicked, the selection in the combo box will be captured. | The combobox’s ItemIndex will be checked to ensure that an item is selected and will be highlighted red if not (no need for verification due to the component being a TDBLookupComboBox component and being linked to tblGenders). | Into the database “Genders”, as either an update or input (by use of SQL). |
| cboShirtSizes | When the “Validate” button is clicked, the selection in the combo box will be captured. | The combobox’s ItemIndex will be checked to ensure that an item is selected and will be highlighted red if not (no need for verification due to the component being a TDBLookupComboBox component and being linked to tblShirtSizes). | Into the database “shirtSizes”, as either an update or input (through use of SQL). |
| edtFirstName | When the “Validate” button is clicked, the selection in the combo box will be captured. | The TEdit’s content will be validated (by checking the string is not null, and that the first character is an uppercase letter) to ensure that GIGO does not occur. | Into the database “players” as either an update or input (through use of SQL). |
| edtLastName | When the “Validate” button is clicked, the selection in the combo box will be captured. | The TEdit’s content will be validated (by checking the string is not null, and that the first character is an uppercase letter) to ensure that GIGO does not occur. | Into the database “players” as either an update or input (through use of SQL). |
| btnOK | When clicked, all data will be gathered from uRegistration. | A check will run to ensure that the validate button has been pressed, if not, it shall be highlighted red and the user will be alerted. | A record in table “players” will be updated/ inputted. |
| btnValidate | When clicked, all data on form uRegistration will be validated. | Performs validation checks upon all necessary data entries and notifies the user if there is an irregularity detected (usually by changing the component colour) | Changing the colour of certain components to red and/or displays a warning through messageDlg. |
| edtEmail | When the “Validate” button is clicked, the selection in the TEdit box will be captured. | The TEdit’s content will be validated (by checking the string is not null, and that an @ exists within the string) to ensure that GIGO does not occur. | Into the database “players” as either an update or input (through use of SQL). |
| edtPhone | When the “Validate” button is clicked, the selection in the TEdit will be captured. | The TEdit’s content will be validated (by checking the string is not null, and that the string is formatted correctly) to ensure that GIGO does not occur. | Into the database “players” as either an update or input (through use of SQL). |
| dtpBirth | When the “Validate” button is clicked, the selection in the TDateTimePicker will be captured. | The TDateTimePicker content will be validated (by checking that the date is ) to ensure that GIGO does not occur. | Into the database “players” as either an update or input (through use of SQL). |
| Form Activate | The program starts. | The program will clear the redOutput and add all records from the relevant databases. | The redOutput will be cleared. |
| btnUpdate | User clicks the button. | The program will change the label of btnOK and load uRegistration. | Into the database “players” as either an update or input (through use of SQL). |
| btnInsert | User clicks the button. | The program will change the label of btnOK and load uRegistration. | Into the database “players” as either an update or input (through use of SQL). |