

|  |
| --- |
|  |
| Single Elimination Tournament  IT PAT PHASE 1 |
| |  |  |  |  | | --- | --- | --- | --- | | Joshua Blom | 11SB | 13288 | Hillcrest High School | |

Table of Contents

[**Task definition:** 2](#_Toc118058877)

[**User story and acceptance test:** 2](#_Toc118058878)

[**UML diagram** 3](#_Toc118058879)

[**Navigation Flow Diagram** 4](#_Toc118058880)

[**List of functions the program will perform** 5](#_Toc118058881)

[**Database Design** 6](#_Toc118058882)

[**Database Design** 7](#_Toc118058883)

[**Data Dictionary** 8](#_Toc118058884)

[**Input, Processing, Output (IPO) and data validation** 9](#_Toc118058885)

[**Human-Computer-Interaction Principles** 12](#_Toc118058886)

[**Help received/declaration** 20](#_Toc118058887)

# **Task definition:**

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. The program is targeted at the event organiser, and as such will need to contain an administrator login with its own specific (and higher) access rights compared to the “standard” user access rights. My program will utilise a single-elimination tournament style 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).

# **User story and acceptance test:**

|  |  |  |
| --- | --- | --- |
|  | General User | Administrator |
| Role | The role of the general user is to log in to the program and view the records in either database. They will also view the podium positions and be able to edit 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 | Viewing data (also contains changing the method of viewing through filtering, sorting, etc). Either input, update or delete their own record in the database. | The administrator will be able to perform all activities that the general user can, as well as 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

Forgot Password

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

Forgot Password

View podium positions

# **List of functions the program will perform**

|  |  |  |
| --- | --- | --- |
| 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/Register with authorisation | Allows general users to log in or register with 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. |

# **Database Design**

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

# **Database Design**

|  |  |  |
| --- | --- | --- |
| 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, 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 records. |
| 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 a text file that contains a changelog.

**Arrays:**

The array of Integers named arrCol will be created. The length of this array will be set to the number of columns in the dbgDisplay (TDBGrid that displays information to the user), the array will then be loaded with the width of the text displayed in each field contained in every column. The array is then sorted and compared to the width of the text displayed in each column name and the highest value for each column is found. Once the biggest width (in pixels) for each column is found, the width of that column will be assigned to the value stored in the array. This then resizes the width of dbgDisplay. This ensures that the user will not have to scroll in the dbgDisplay, and that each column size is not bigger than it needs to be.

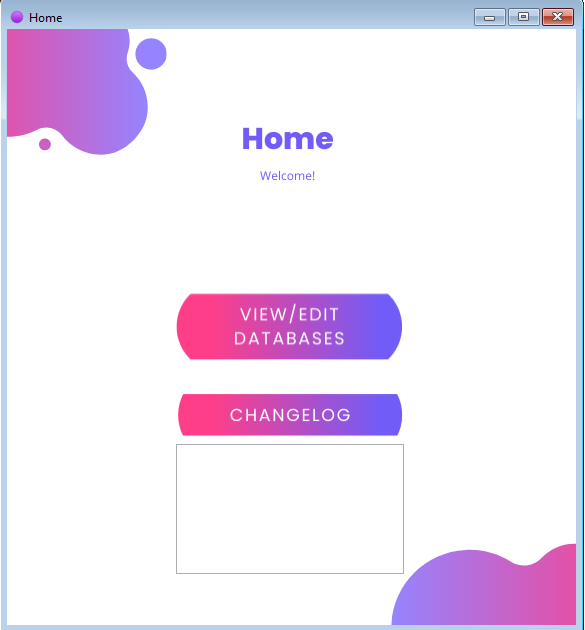
Arrays are also used to find the winner in “games” for each game played, provided that the winner column is empty. Loads player 1 score and player2 score, finds which is bigger and outputs relevant data to winners column (can also be a Draw).

|  |  |  |
| --- | --- | --- |
| **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 |
| isValid | Boolean | Global |
| userEmail | String | Global |
| isAdminSlc | Boolean | Local |
| isAdmin | Boolean | Global |
| selectedDB | TADOTable | Local |
| dbColor | TColor | Local |
| pixSpc (const assignment) | 5 | Local |
| arrCol | Array of Integer | Local |
| selectedDS | TDataSet | Local |
| currentPOS | TBookmark | Local |
| width | Integer | Local |
| colWidth | Integer | Local |
| columnsAMT | Integer | Local |
| iLoop | Integer | Local |
| aLine | String | Local |
| idNum | Integer | Local |
| tf | Text File | Local |

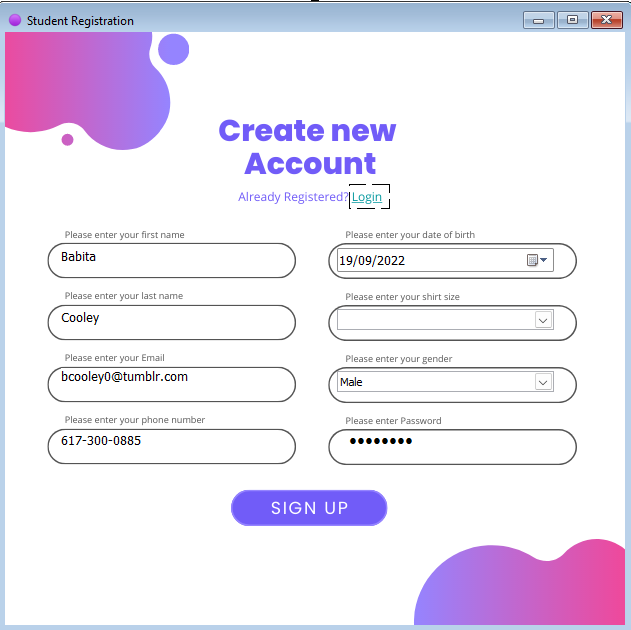
# **Input, Processing, Output (IPO) and data validation**

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Input | Process | Output |
| cboGenders | When the “Sign Up” button is clicked, the selection in the combo box will be captured. | No need for verification due to the component being a TDBLookupComboBox component and being linked to tblGenders and having a selection already present upon activation. | Into the table “players”as either an update or input (by use of ADO). |
| cboShirtSizes | When the “Sign Up” 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 additional verification due to the component being a TDBLookupComboBox component and being linked to tblShirtSizes). | Into the table “players”, as either an update or input (through the use of ADO). |
| edtFirstName | When the “Sign Up” button is clicked, the selection in the TDBEdit 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 the use of ADO). |
| edtLastName | When the “Sign Up” button is clicked, the selection in the TDBEdit 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. | In the database “players” as either an update or input (through the use of ADO). |
| imgSignUp | 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, such as invalid data entries or a duplicate record is detected. | Changing the colour of certain components to red and/or displaying a warning through messageDlg (if irregularity detected). Inputting all data into tblPlayers. |
| edtEmail | When the “Sign Up” button is clicked, the selection in the TDBEdit box will be captured. | The TDBEdit’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 the use of ADO). |
| edtPhone | When the “Sign Up” button is clicked, the selection in the TDBEdit will be captured. | The TDBEdit’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 the use of ADO). |
| dtpBirth | When the “Sign Up” 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 the use of ADO). |
| popupDBGrids.clkEdit | The user right-clicks on a tbgDisplay row and then clicks on the Edit option. | The program will check if the user is an admin, or if their email matches the selected row, to determine if they are authorised to edit the record. | The program will either display an error message or display frmEdit/ frmEditTournament. |
| popupDBGrids.clkInsert | The user right-clicks on a tbgDisplay row and then clicks on the Insert option. | The program will check if the user is an admin, to determine if they are authorised to insert a record. | The program will either display an error message or display frmRegistration/ frmAddTournament. |
| popupDBGrids.clkDelete | The user right-clicks on a tbgDisplay row and then clicks on the Delete option. | The program will check if the user is an admin, or if their email matches the selected row, to determine if they are authorised to delete the record. The program will then ask the user for confirmation of deletion through a messageDlg, if received, it will then delete the appropriate record in the database (by use of ADO code). | The program will then either display an error message, or a message showing a successful deletion. |

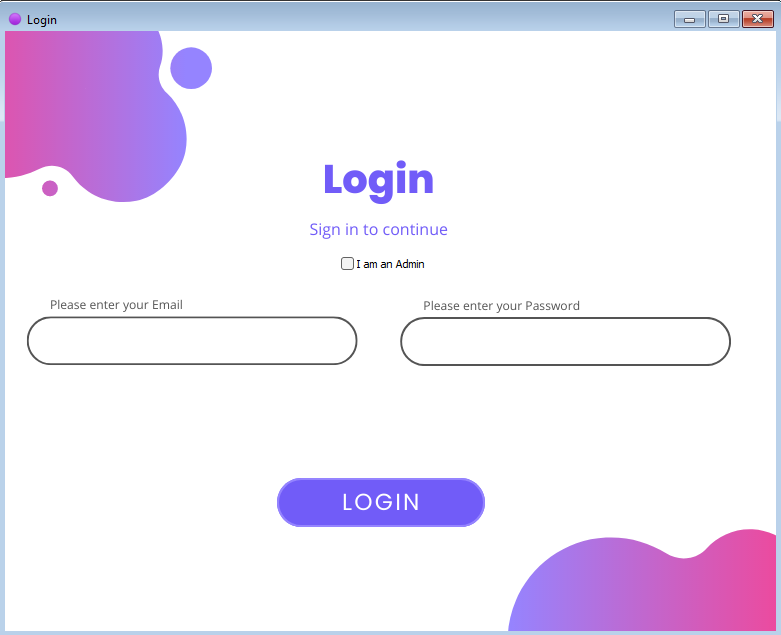
# **Human-Computer-Interaction Principles**



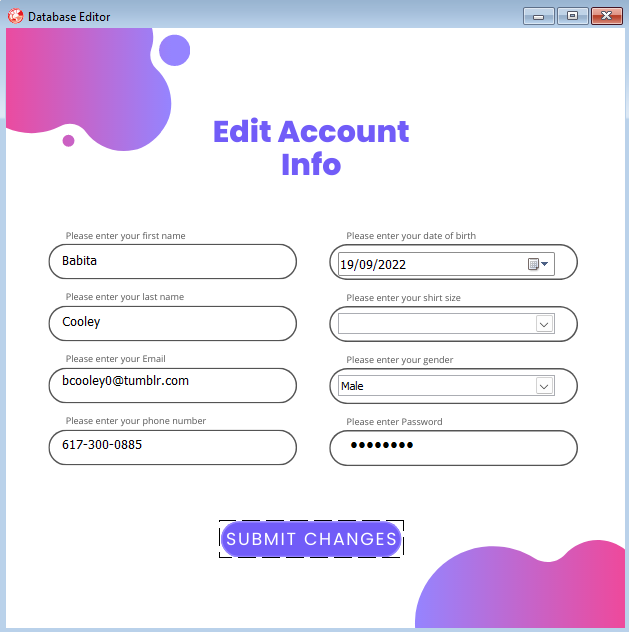
|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, with 2 TImages being used in place of buttons (higher aesthetic value) to view/edit Databases, and view the changelog (this TImage is only visible if logged in as admin). |
| TRichEdit | Is made visible and filled with the data contained within the changelog file. |



|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, also used to detect when the user clicks on either “Login” or Sign Up. |
| TDBEdit | Collectively used to input a record into the table “players”. |
| TDBLookupComboBox | Displays options for input. Collectively used in conjunction with all TDBEdits to input a record into table “players”. |
| TDateTimePicker | Allows the user to select their DoB, used in conjunction with all TDBEdit’s to insert a record into table “players”. |



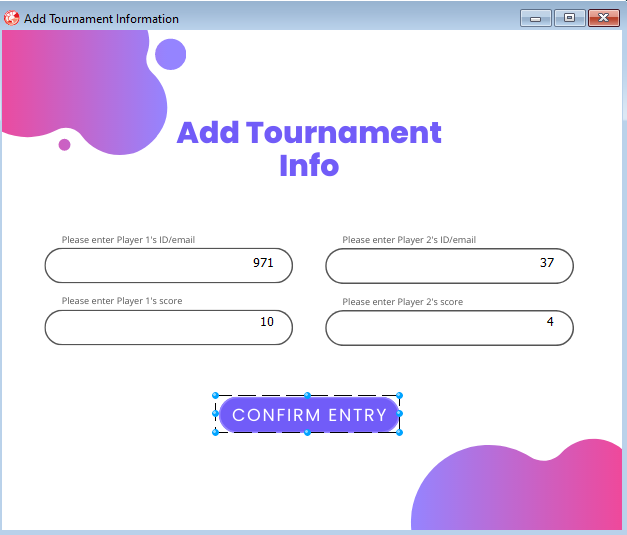
|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, also used to detect when the user clicks on Login. |
| TCheckBox | Allows the user to select that they want to log in as an admin |
| TEdit | Allows the user to input an email and password to be checked. If TCheckBox is selected, only the password TEdit will be enabled. |



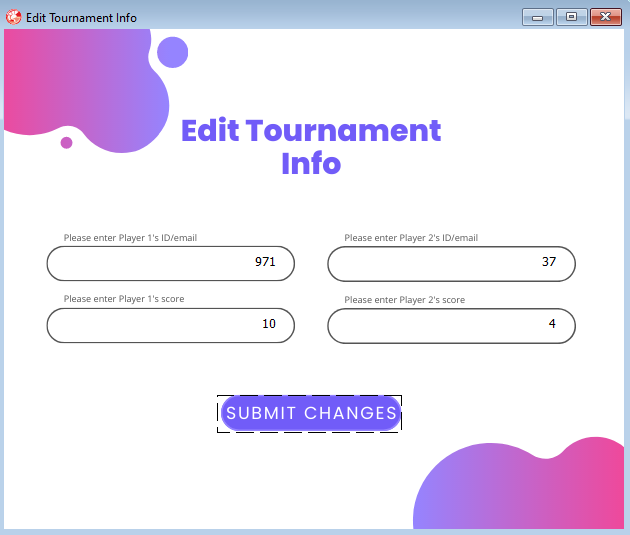
|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, also used to detect when the user clicks on Submit Changes. |
| TDBEdit | Collectively used to edit a record in the table “players”. |
| TDBLookupComboBox | Displays options for editing. Collectively used in conjunction with all TDBEdits to edit a record in table “players”. |
| TDateTimePicker | Allows the user to select their DoB, used in conjunction with all TDBEdit’s to edit a record in table “players”. |



|  |  |
| --- | --- |
| Component | Use |
| TDBGrid | Displays tblPlayers, tblGames or the result of a query. |
| TButton | Allows the user to navigate through the displayed table, refresh the TDBGrid, view the podium positions, generate a report containing fields displayed in the TDBGrid, or load SQL from a file. |
| TImage | Background of GUI. |
| TPopupMenu | Displays a popup with options of Insert, Edit or Delete upon right click of the displayed table in TDBGrid. |
| TRadioGroup | Allows the user to choose which table they want to display. |
| TComboBox | Allows the user to sort, filter or search within either tblPlayers or tblGames. Executes through SQL. |



|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, detects when the user clicks on Confirm Entry. |
| TDBEdit | Collectively used to insert a record into table “games”. |



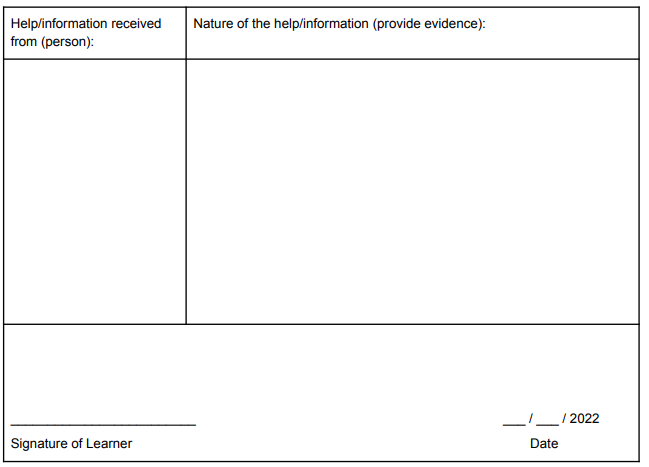
|  |  |
| --- | --- |
| Component | Use |
| TImage | Background of GUI, detects when the user clicks on Submit Changes. |
| TDBEdit | Collectively used to edit a record in table “games”. |

Graphical user interface, application

Description automatically generated

|  |  |
| --- | --- |
| Component | Use |
| TImage | The background of the GUI, detects when the user clicks on Send. |
| TEdit | Used to send a message to the admin (through the changelog), to alert the admin that this user has forgotten their password and requires a reset/assistance. |

# **Help received/declaration**

Graphical user interface, table

Description automatically generated

Assistance with SQL for finding the winner in “games”, specifically with the INNER JOIN section of the SQL present.

Deeksha Tanwar

Base of PAT provided (none of the provided code/GUI was actually used in my final PAT), teaching done in lessons.

Mr Peal

10

30

Joshua Blom

Mr Peal

0508035656083

Joshua Blom

Joshua Blom

30

10