

**Succeed with speed**

# Reikwydte

**Onderwerp:**

Vir die Graad 12 Inligtingstegnologie Praktiese Assesseringstaak (PAT) vir 2024, gaan ek 'n sagtewareprogram ontwikkel wat fokus op “**Die Eiendomsmark**”. Die sagteware sal 'n gebruiker lei om die beste plek op aarde te identifiseer om te woon, gebaseer op sy unieke antwoorde op 'n reeks vrae. Die taak behels die skep van 'n sagteware-ontwikkelingsprojek wat met hierdie tema verband hou.

**Doel Probleem:**

Vir Suid-Afrikaanse individue wat vir die eerste keer 'n eiendom wil koop, is daar tans geen gebruikersvriendelike en effektiewe metode om te bepaal watter stad die beste sou wees om 'n eiendom in te koop nie. Daar is 'n dringende behoefte aan 'n hulpmiddel wat hierdie groot besluit kan vergemaklik en toekomstige generasies tyd kan bespaar.

**Moontlike Oplossing:**

My sagtewareprogram sal die gebruiker 'n reeks vrae stel en dan 'n spesifieke stad voorstel wat die beste by die gebruiker se behoeftes en voorkeure sou pas.

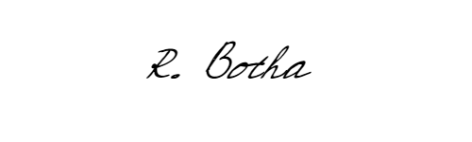
**Reikwydte:**

Die program sal nie alle plekke op aarde oorweeg nie. Dit sal fokus op bekende gebiede en gebiede waaroor inligting maklik verkrygbaar is, om 'n akkurate besluit te kan neem.

# Gebruikersvereistes

|  |  |  |
| --- | --- | --- |
|  | ***Gebruiker 1: ‘General user’ – Algemene gebruiker*** | ***Gebruiker 2: ‘Admin’ –Administrateur*** |
| Wie is die gebruiker? | ‘n Klient wie wil weet wat die mees gepaste stad is om sy huis te koop. | Die eienaaar of ‘n aangestelde persoon/werknemer wie met die data in die databasis werk. |
| Rol: | Die Algemene Gebruiker gebruik die sagteware om vrae te beantwoord en voorstelle te ontvang oor waar hulle moet woon. | Die Admin Gebruiker is verantwoordelik vir die bestuur van die sagteware. Dit sluit take soos die opdatering, invoeg en verwyder van die databasis se data en die hantering van enige tegniese kwessies in. |
| Aktiwiteit: | Die Algemene Gebruiker kan vrae beantwoord, en die voorstelle wat deur die sagteware verskaf word, bekyk. | Die Admin Gebruiker kan data byvoeg, opdateer, of verwyder van die databasis. Hulle kan ook gebruikersrekeninge bestuur, wat take insluit soos die herstel van wagwoorde, die verbanning van gebruikers, en die toekenning of intrekking van toestemmings. Admin gebruikers is vir die instandhouding van die sagteware, wat take kan insluit soos die opdatering van die sagteware en die opsporing van enige tegniese kwessies verantwoordelik. |
| Beperkings: | Die Algemene Gebruiker kan slegs interaksie hê met die data wat aan hulle voorgelê word deur die koppelvlak van die sagteware. Hulle kan nie die data of instellings van die app verander, behalwe vir hul eie gebruikers profiele nie. Hulle kan ook nie toegang kry tot ander gebruikers se data of enige administratiewe funksies van die app nie.  Opdatering van ‘n gebruiker profiel kan gesien word as die resultaat van blyplek verander indien die toets oorgeskryf word. | Die Admin Gebruiker het volle toegang tot die toepassing en sy data, so hulle hoofbeperking is om te verseker dat hulle hierdie mag verantwoordelik gebruik. Hulle moet gebruikersdata en privaatheid beskerm, en verseker dat die app funksioneel en toeganklik bly vir sy gebruikers. |

# Datawoordeboek

*I prefer to use English for the next section of my PAT to maintain consistency with the English language used in Delphi and Access.*

*\_\_\_\_\_\_\_\_\_\_\_*

CTO & Founder

## Database

*First column PK for primary key fields and FK for foreign key fields, field size column describes the* ***maximum size*** *(characters, unless otherwise specified) and data type may consist of types of Text/ Integer/ Boolean/ Currency/ Real*

*No empty fields or nul-values are allowed to be stored except in the ‘Cities’-field in the tblProfiles table.*

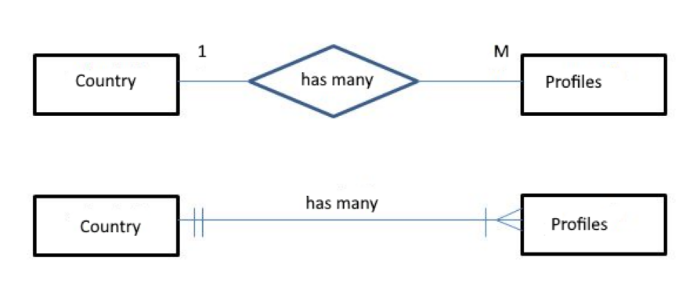
***Table name: tblProfiles***

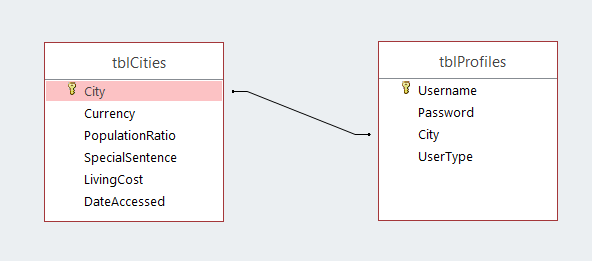
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Field name | Data type | Format | Field Size | Descriptions of the relevant and well-chosen fields |
| PK | Username | Short Text | N/A | 10 | This field will store the usernames of all of the users . |
|  | Password | Short Text | N/A | 4 | This field will store the passwords of all of the users. This field, together with the usernames field, determines whether or not the user have done the test already so the previous result can be displayed if there is one. |
| FK | City | Short Text | N/A | 10 | This field stores the latest city the user has received. |
|  | UserType | Boolean | True/False | Predefined field size (1 byte) | This field will determine whether the user is a client or an admin. This determines what page (tab sheet) should be shown next (statistics or questions for the client) |

***Table name: tblCities***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Field name | Data type | Format | Field Size | Descriptions of the relevant and well-chosen fields |
| PK | City | Short Text | N/A | 10 | This field stores all of the possible cities the user may have for a result. |
|  | Currency | Short Text | N/A | 10 | This field stores the currency used in the specific city to give the user an idea of the economic status of the city. |
|  | Population | Number | N/A | Long Integer | This field stores the population ratio – number of people per square metre. |
|  | SpecialSentence | Short Text | N/A | 50 | This field stores a sentence describing |
|  | LivingCost | Currency | N/A | Predefined field size (8 bytes) | This field stores the estimate cost of living per month. (The minimum amount of money needed to live comforatbly in the city) |
|  | DateAccessed | Date/Time | Short Date | Predefined field size (8 bytes) | This field stores the exact date the city was added to the database, because a lot of errors can be given as an output and the date accessed (printed in a log file) makes it easier for the software engineer and CTO to debug the code or return to a previous working version on the software. |

Relationships:

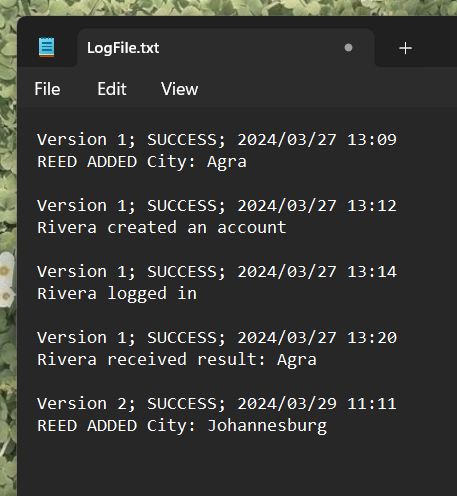
The primary key in tblCitiesies (the ‘City’ field) is a foreign key in tblProfiles. This is a one-to-many relationship. Each city in the tblCities table can have multiple related records (profiles) in the tblProfiles table, but each profile in the tblProfiles table can only be related to one city in the tblCities table.



City

City

## Textfiles



Textfile:  
LogFile.txt  
Format: Version #; yyyy/mm/dd HH:MM

This Textfile is going to be used to save all of the changes made to the data in the database. The version is updated each time data in the data base is updated, there can either be ‘SUCCESS’ or ‘FAIL’ printed, depending on whether or not the software calls an error message. The exact date and time is given to make debugging easier or to return to previous versions of the software.

The message beneath the above explained line gives important information to describe what aspect has been changed.

The text will automatically be generated by the software. Everybody will have access to this file because it is stored on a local machine and does not contain sensitive information. If there is an error while the data is being read or written, the data will simply be deleted and the user will receive an error message saying that the data does not meet specific requirements.





This Textfile is going to be used to store the first city the software recommends the user to live in. This text file is going to be used to compare the last city the user received to the first. A text file is used to keep the log file accurate – this data should not be stored in the log file, because of that, I use this textfile. This is data that could be beneficial to the user, but not the people (software engineers, administrators etc.) viewing the log file.

The text will automatically be generated by the software. Everybody will have access to this file because it is stored on a local machine and does not contain sensitive information. If there is an error while the data is being read or written, the data will simply be deleted and the user will receive an error message saying that the data does not meet specific requirements.



## Classes and objects

This class will keep the details and update the details of the user. This class will be used to make the main code more readable and to not repeat any lines.

|  |
| --- |
| **Class Name: TUser** |
| ***Attributes*** |
| * fUsername : string; * fPassword : string; * fCity : string; // The city the user received after taking the test, else it is a space ‘ ’ * fUserType : boolean; // The type of user... admin or general |
| ***Methods*** |
| * Constructor Create (sUsername, sPassword, sCity : string; bUserType: boolean) ; //The //constructor initialises values and gives variables their first values. (Constructor) * Function GetUsername: string; // Returns the username (Accessor method) * Function GetPassword: string; // Returns the password (Accessor method) * Function GetCity: string; // Returns the city (result) (Accessor method) * Function GetUserType: boolean; // Returns the user type (Accessor method) * Procedure UpdateCity(iIndex: integer); //Updates the city depending on which index of an // array of different cities is the highest number. Every answer to a question increases // a certain index of the array of cities. (Mutator method) |

*‘-’ sign is for private declarations while ‘+’ sign is for public.*



## Arrays

These parallel arrays help end makes it easy to calculate in which city the user should live in.

Example of the arrays:

*Array 1: arrTally*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Index:*** | 1 | 2 | 3 | 4 | 5 |
| ***Value:*** | 0 | 3 | 8 | 2 | 1 |

*Array 2: arrCities*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Index:*** | 1 | 2 | 3 | 4 | 5 |
| ***Value:*** | Agra | Johannesburg | Seoul | Sedgefield | Dubai |

**How does these arrays work?**

Each time the user answers a question, a specific index number of arrTally gets updated by being increased with one. The highest number of arrTally Is then calculated and the corresponding index number is stored. The same index number of arrCities is then taken and the corresponding city becomes the city that the software outputs to the user in which he should live in.

**Example:**

The user answers multiple choice questions displayed by the software. Each time the user chooses option C, index number 3 of arrTally is increased with one. After the user answers all of the questions, index 3 (C) is the most popular answer. Index 3 in arrCities is ‘Seoul’. Thus, the user should live in Seoul and the output is ‘Seoul’.

# Navigasie

**Algemene Gebruiker**

2)

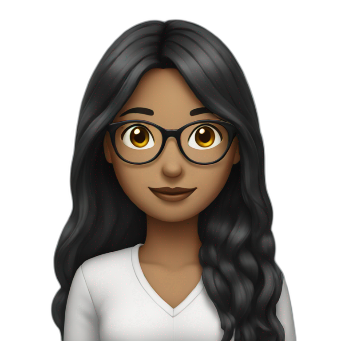
Resultaat word gestoor

Index se getal in array van plekke word opgedateer afhangende van ‘n spesifieke antwoord op ‘n spesifieke vraag.

1)

Kyk watter index van die array is die hoogste en gee dan die resultaat aan die gebruiker (die hoogste getal)

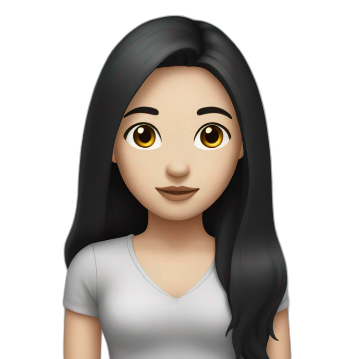
As wagwoord en gebruikersnaam korrek is en by mekaar pas.



**Admin**

As wagwoord en gebruikersnaam korrek is en by mekaar pas.

Data word verander



# Grafiese Gebruikers Koppelvlak Ontwerp

Only one form will be used for my entire PAT and the name of the form will be ‘frmPAT’. I will use a page control component and different tabsheets to make the design of my program professional.

|  |
| --- |
| Tabsheet 1: tbsSignIn |
|  |
| Doel van program en gebruiker en hoekom die ontwerp van die gebruikerskoppelvlak daarby pas:  Die gebruiker wil hierdie tabsheet gebruik om in te teken. Die tabsheet het ‘n eenvoudige en moderne uitleg om daardie spesifieke taak te verrig. Daar is ‘n konstante ontwerp patroon deur die hele program met die groen forms en wit logo te gerbuik. Die font van die teks bly ook konstant en is maklik om te lees. Dit neem nie te veel tyd van die gebruiker op nie. |
| Gepaste komponente gebruik:  Ek het die komponente gekies op grond van gemak vir die gebruiker en spoed waarmee hulle kan inteken. Die editbokse is gekies omdat dit maklik in ‘strings’ ingelees kan word en getoets kan word of die gebruikersnaam en wagwoord bymekaar pas. Dit is maklik en vinnig vir die gebruiker om alle data in die editbokse te tik en so in te teken. Die ‘sign up’ knoppie en help-knoppie is ‘panels’ omdat die kleur van ‘panels’ verander kan word vir ‘n beter gebruikerservaring omdat dit mooier kleurvol lyk. |
| Gebruiksgemak, logiese vloei:  Die cursor sal van die ‘Name’ edit boks outomaties oorbeweeg na die ‘password’ edit boks indien die veld se inligting korrek ingetik word byvoorbeeld die naam is net letters. Die gerbuiker kan gemaklik na die volgende skerm toe gaan (indien die wagwoord en gebruikersnaam bymekaar pas) deur die sign up panel te klik. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys wat verduidelik hoe die gebruiker moet inteken en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Duidelik gemerkte navigasie:  Alle navigasie komponente is duidelik sigbaar deur my gebruik van mooi kontrasterende kleure en het net een doel wat duidelik bekend is aan die gebruiker (dit is om in te teken). Daar is ‘n hulp knoppie (panel) wat verduidelik op watter inligting ingevoer moet word om die gewenste resultate te kry en verduidelik op watter knoppies (panels) geklik moet word. |
| Vriendelike dialoog en hulp (dus ook nuttige foutboodskappe):  Indien iemand op die ‘HELP’ paneel klik, sal ‘n vriendelike en nuttige boodskap verskyn wat hulp aan die gebruiker bied. Dit sal verduidelik dat die gebruikers hulle name en wagwoordemoet insleutel.  Indien toevoer van ‘n verkeerde formaat soos letters waar nommer moet wees of daar op ‘n plek geen toevoer gegee word nie, sal ‘n foutboodskap verskyn wat presies verduidelik wat die gebruiker ingetik het en wat die gerbuiker eintlik moet intik. |

|  |
| --- |
| Tabsheet 2: tbsQuestions |
| **Back**  NEXT |
| Doel van program en gebruiker en hoekom die ontwerp van die gebruikerskoppelvlak daarby pas:  Die gebruiker wil hierdie tabsheet gebruik om vrae te beantoowrd om ‘n resultaat te kry. Die tabsheet het ‘n eenvoudige en modern uitleg om daardie spesifieke taak te verrig. Daar is ‘n konstante ontwerp patroon deur die hele program met die groen forms en logo. Die font van die teks bly ook konstant en is maklik om te lees. Dit neem nie te veel tyd van die gebruiker op nie. |
| Gepaste komponente gebruik:  Panele is as knoppies gebruik om ‘n mooier uitleg te voorsien. ‘n Vraag word in ‘n label gegee sodat die gebruiker elke keer die vraag kan beantwoord en die vraag maklik verander kan word. ‘n Label is ook gebruik as die ‘Help’ knoppie sodat dit nie te veel plek opneem nie en dit pas by ander sagteware se help-knoppie uitlegte. |
| Gebruiksgemak, logiese vloei:  Die gerbuiker kan gemaklik na die volgende skerm toe gaan deur net vrae te beantwoord en elke keer ‘Next’ te druk. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Duidelik gemerkte navigasie:  Alle navigasie komponente is duidelik sigbaar en het net een doel wat duidelik bekend is aan die gebruiker. Daar is ‘n hulp knoppie (panel) wat verduidelik op watter inligting ingevoer moet word om die gewenste resultate te kry en verduidelik op watter ander knoppies (panels) geklik moet word. |
| Vriendelike dialoog en hulp (dus ook nuttige foutboodskappe):  Indien iemand op die ‘HELP’ paneel klik, sal ‘n boodskap verskyn wat hulp aan die gebruiker bied. Dit sal verduidelik dat die gebruikers die vyf vrae moet beantwoord om die resultate te kry. |
| Standaard GUI ontwerpbeginsels:  Ek het die komponente gekies op grond van gemak vir die gebruiker en spoed wat hulle die vrae sal kan beantwoord. |

|  |
| --- |
| Tabsheet 3: tbsResult |
| **Back** |
| Doel van program en gebruiker en hoekom die ontwerp van die gebruikerskoppelvlak daarby pas:  Die gebruiker wil hierdie tabsheet gebruik om net sy resultaat te sien en dan weer dalk antwoorde op vrae te verander. Die tabsheet het ‘n eenvoudige en modern uitleg om daardie spesifieke taak te verrig. Daar is ‘n konstante ontwerp patroon deur die hele program met die groen tabsheets en logo. Die font van die teks bly ook konstant en is maklik om te lees. Dit neem nie te veel tyd van die gebruiker op nie. |
| Gepaste komponente gebruik:  ‘n Mooi foto word deur ‘n image component gewys an die stad wat verkry is. Inligitng soos die naam en ‘n paar interressante feite oor die stad word dan ook gegee. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Gebruiksgemak, logiese vloei:  As die gebruiker op ‘Back’ kliek, kan hy antwoorde op vrae verander en ‘n nuwe resultaat verkry. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Duidelik gemerkte navigasie:  Alle navigasie komponente is duidelik sigbaar en het net een doel wat duidelik bekend is aan die gebruiker. Daar is ‘n hulp knoppie (panel) wat verduidelik op watter inligting ingevoer moet word om die gewenste resultate te kry en verduidelik op watter ander knoppies (panels) geklik moet word. |
| Vriendelike dialoog en hulp (dus ook nuttige foutboodskappe):  Indien iemand op die ‘HELP’ paneel klik, sal ‘n boodskap verskyn wat hulp aan die gebruiker bied. Dit sal verduidelik dat die gebruikers terug kan gaan en vrae se antwoorde kan beantwoord of die kruisie regs bo kan druk om af te sluit. |

|  |
| --- |
| Tabsheet 4: tbsStats |
| **Back**  *Help*    View TOP city    **DATA HIER** |
| Doel van program en gebruiker en hoekom die ontwerp van die gebruikerskoppelvlak daarby pas:  Admin wil hierdie tabsheet gebruik toegang te verkry tot data, in ‘n databasis gestoor, of dit te verander. Die tabsheet het ‘n eenvoudige en modern uitleg om daardie spesifieke taak te verrig. Daar is ‘n konstante ontwerp patroon deur die hele program met die groen tabsheets en logo. Die font van die teks bly ook konstant en is maklik om te lees. Dit neem nie te veel tyd van die gebruiker op nie. |
| Gepaste komponente gebruik:  Panels word as knoppies gebruik om verskeie take met die data uit te voer. Edit bokse wys die data en bied ‘n plek waar die gebruiker die data kan verander. ‘n DBGrid word gebruik om alle data te wys. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Gebruiksgemak, logiese vloei:  Alle aksies word deeglik beskryf by teks of op die knoppie self of op n label langs aan die knoppie wat my program logies goed laat sin maak. Indien die gebruiker op die hulp-boks kliek, word ‘n hulp boodskap gewys en daar is ‘n kruisie wat gedruk kan word om hom te laat af vlieg. Daar is baie min dele waar die gebruiker toevoer hoef te gee, wat my program gerieflik maak. |
| Duidelik gemerkte navigasie:  Alle navigasie komponente is duidelik sigbaar en het net een doel wat duidelik bekend is aan die gebruiker. Daar is ‘n hulp knoppie (panel) wat verduidelik op watter inligting ingevoer moet word om die gewenste resultate te kry en verduidelik op watter ander knoppies (panels) geklik moet word. |
| Vriendelike dialoog en hulp (dus ook nuttige foutboodskappe):  Indien iemand op die ‘HELP’ paneel klik, sal ‘n boodskap verskyn wat hulp aan die gebruiker bied. Dit sal verduidelik dat die gebruikers op enige panel kan kliek om hulle gewenste aksies uit te voer. |

# Sagteware Gereedskap

## Screen 1: [tbsSignIn](#tbsSignIn)

***INPUT***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation - after the Sign in button is pressed |
| sUsername | Keyboard | String | General Text | edtUsername | Check if the username consist of letters only. **(Is username a string?)**  *MTError: “Username may only consist of letters.”*  Check if the user entered a username **(Presence check)**  *MTError: “You have to enter a username.”*  Check if the username is less than or equal to 10 characters. **(Length Check)**  *MTError: “Username has to consist of 10 or less characters.”* |
| sPassword | Keyboard | String | General Text *(May only be 4 numbers)*  *xxxx* | edtPassword | Check if the password consist of numbers only.  *MTError: “Password may only consist of numbers.”*  Check if the password is equal to 4 characters.  *MTError: “Username has to consist of 4 characters.”* |
| *sExistingUsername (When I need to use it in Delphi, but field name is Username in Access)* | Database | String in Delphi  (Short Text in Access) | General Text | N/A | **In cases where the access data is being verified:**  Check if the username consist of letters only.  *MTError: “Username may only consist of letters.”*  Check if the username is less than or equal to 10 characters.  *MTError: “Username has to consist of 10 or less characters.”* |
| *sExistingPassword (When I need to use it in Delphi, but field name is Password in Access)* | Database | String in Delphi  (Short Text in Access) | General Text *(May only be 4 numbers)*  *xxxx* | N/A | Check if the password consist of numbers only.  *MTError: “Password may only consist of numbers.”*  Check if the password is equal to 4 characters.  *MTError: “Username has to consist of 4 characters.”* |

***PROCESSING***

|  |  |
| --- | --- |
| What processing needs to be done | How processing will be done |
| Verify that the password matches the user profile. | N/A |
| Ensure that the username only consist of letters. | N/A |
| Add new users to the database. | N/A |

***OUTPUT***

|  |
| --- |
| GUI Component |
| Tabsheet – if the username and password fits one another, the tabsheet displayed will change to tbsQuestions. |

## Screen 2: [tbsStats](#tbsStats)

***INPUT***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation |
| iPeople  (When I need to use it in Delphi, but PopulationRatio field in Access) | Keyboard | Integer in Delphi  (Number in Access) | Numbers  *xxx* | sedPopulationRatio  *(Spin edit will only be visible when pnlCities is clicked to show the data of the cities)* | Check if user entered *an integer* **(type check)** between 1 and 999, both included. **(range check)**  *MTError: “Please enter an integer number between 1 and 999, both included”* |
| bUserType  (When I need to use it in Delphi, but UserType field in Access) | Mouse click | Boolean in Delphi  (Yes/No is Access) | True/False | rgpUserType | **Check if user selected a value or not.**  *MTError: “UserType balue not found.”*  Check that the value in the database is a boolean value to ensure that one of my software engineer colleagues did not compromise our data integrity.  *MTError: “Oh no! UserType is not a boolean value.”* |
| *dDateAccessed (When I need to use it in Delphi, but field name is DateAccessed in Access)* | Keyboard | Date in Delphi  (Date/Time in Access) | Date  *dd/mm/yyyy* | edtDateAccessed | Check if the characters entered is in the format of a date.  *MTError: “Please enter a date in the format: dd/mm/yyyy”* |
| *sExistingUsername (When I need to use it in Delphi, but field name is Username in Access)* | Database | String in Delphi  (Short Text in Access) | General Text | edtUsernameStats | Check if the username consist of letters only.  *MTError: “Username may only consist of letters.”*  Check if the username is less than or equal to 10 characters.  *MTError: “Username has to consist of 10 or less characters.”* |
| *sExistingPassword (When I need to use it in Delphi, but field name is Password in Access)* | Database | String in Delphi  (Short Text in Access) | General Text *(May only be 4 numbers)*  *xxxx* | edtPasswordStats | Check if the username consist of letters only.  *MTError: “Username may only consist of letters.”*  Check if the username is less than or equal to 10 characters.  *MTError: “Username has to consist of 10 or less characters.”* |

***PROCESSING***

|  |  |
| --- | --- |
| What processing needs to be done | How processing will be done |
| Delete a user from database and update the array to remove the deleted user. | if iClicked mod 2 = 0 then //iClicked is increased every time pnlCities or pnlProfiles is clicked to view the  // corresponding data. (Starts by showing profiles)  begin  bProfiles := false;  end;  with dmPAT do  begin  if bProfiles = true then // Profiles are being shown  begin  tblProfiles.Delete;  // Update the array again...  dbgInfo.DataSource := nil;  dbgInfo.DataSource := dmPAT.dsrProfiles;  i := 0;  tblProfiles.First;  while not tblProfiles.Eof and (i <= 300) do  begin  inc(i);  arrUsernames[i] := tblProfiles['Username'];  tblProfiles.Next;  end;  end;  end; |

|  |  |
| --- | --- |
| Insert a city into the database. | // After validating and ensuring the correctness of the fields, this step is executed.  tblCities.insert;  sCity := edtCity.text;  sCurrency := edtCurrency.text;  iPopulationRatio := strtoint(edtPopulationRatio.text); // validation also done here, I am not adding it, because it  // is irrelevant in this *specific* algorithm  sSpecialSentence := edtSpecialSentence.text;  rLivingCost := strtofloat(edtLivingCost.text);  dDate := stringtodate(edtDateAccessed.text);  tblCities.Edit  tblCities['City'] := sCity;  tblCities['Currency'] := sCurrency;  tblCities['PopulationRation'] := iPopulationRatio;  tblCities['SpecialSentence'] :=sSpecialSentence;  tblCities['LivingCost'] := rLivingCost;  tblCities['DateAccessed'] :=dDate;  dmPAT.tblItems.Post; |
| Edit a city already in the database. | tblCities.Edit;  sCity := edtCity.text;  sCurrency := edtCurrency.text;  iPopulationRatio := strtoint(edtPopulationRatio.text); // validation also done here, I am not adding it, because it  // is irrelevant in this ***specific***algorithm  sSpecialSentence := edtSpecialSentence.text;  rLivingCost := strtofloat(edtLivingCost.text);  dDate := stringtodate(edtDateAccessed.text);  tblCities.Edit  tblCities['City'] := sCity;  tblCities['Currency'] := sCurrency;  tblCities['PopulationRation'] := iPopulationRatio;  tblCities['SpecialSentence'] :=sSpecialSentence;  tblCities['LivingCost'] := rLivingCost;  tblCities['DateAccessed'] :=dDate;  tblItems.post |
| Get the city received by the most amount of users as a result. | // SQL Query:  **SELECT** Top 1 City **FROM** tblProfiles **GROUP BY** City **ORDER BY** count(Username) DESC; |

***OUTPUT***

|  |  |  |
| --- | --- | --- |
| Data | Format | GUI Component |
| Username,  Password,  Account type | <Username field in text> or <General text>  <Password field in text > or <General text>  <Account type field in text > or <boolean> | dbGrid |
| Username,  Password,  Account type | <Username field in text > or <General text>  <Password field in text > or <General text>  <Account type field in text > or <boolean> | Rich Edit |
| City,  currency,  population ratio,  special sentence,  cost of living,  date accessed | <General text>  <General text>  <Number (Integer)>  <General text>  Currency,  dd/mm/yyyy (date/time) | dbGrid |

## Screen 3: [tbsQuestions](#tbsQuestions)

***PROCESSING***

|  |  |
| --- | --- |
| What processing needs to be done | How processing will be done |
| Calculate in what city the user should live in. |  |

## Screen 4: tbsResult

***OUTPUT***

|  |  |  |
| --- | --- | --- |
| Data | Format | GUI Component |
| City Name | <General text> | Label |