Grade 12 IT PAT Phase 1 - 2022

**Benrico Krog**

1. **Scenario & Scope**

A Tertiary institution decided to replace their previous in-house developed online student application manager and hired me to develop it according to their needs. The client stressed that this application must be secure by design as it would be handling a lot of important personal data and open to the public. This application needs to have the following features:

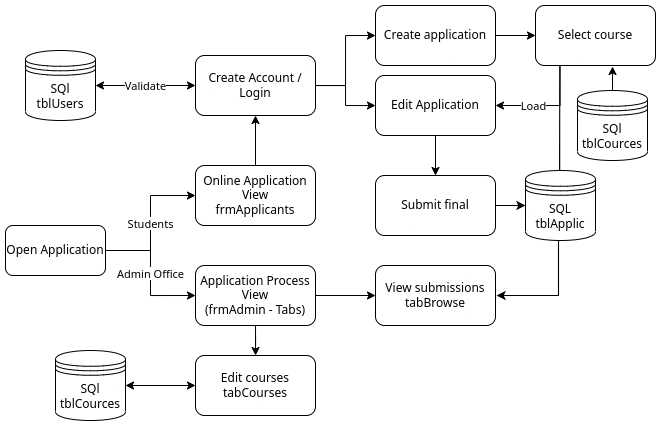
* 2 separate user interfaces, one for the applicants and the other for the institution staff
* An administrator role to manage staff user accounts
* A way for new applicants to create an accounts and log into existing accounts
* Staff members can edit courses offered and dynamically update the scoring algorithm
* Applicants can select a course and fill in the application and continue to edit it before the closing date

My solution is creating a single Delphi application using a database for application data storage. The applicant user interface will be very simple to use so that new users don’t need assistance. The staff user account will be able to dynamically add/edit/remove courses and update the algorithm parameters used for the selection process.

1. **User Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Users** | **Role** | **Activities** | **Limitations** |
| Admin | Administrator | View applications, logs and manager users | Same view as staff, thus create/edit applications |
| Applicant  (Students) | Temporary account created by applicants | Create/Edit/Submit application | Access other applicants’ submissions, review applications, manage users and view logs |
| Admin office  (Staff) | Account used to review submitted applications | Manage/Review applications | Create/Edit Applications, manage users and view logs |

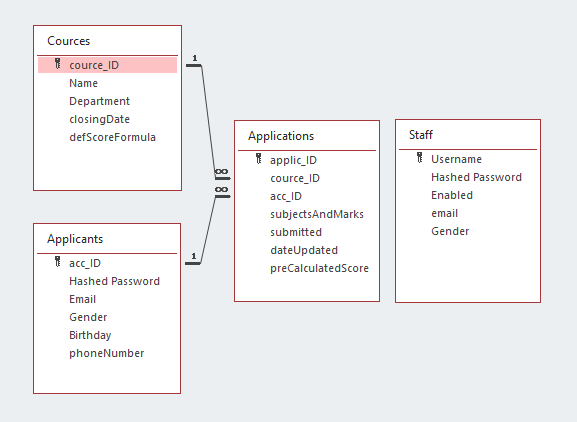
1. **Navigation/Description of Flow Diagram**



1. **Data Structures**

* **Database**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tables** | **Primary Key (Text)** | **Foreign Key(s)** | **Fields** |
| tblApplicants | acc\_ID | N/A | Name(Text), HashedPassword(Text), email(Text), phoneNumber(Text), Gender(Char), Birthday(Date) |
| tblStaff | Username | N/A | Name(Text), HashedPassword(Text), email(Text), enabled(Boolean), Gender(Char) |
| tblCources | cource\_ID | N/A | Name(Text), Department(Text), [definition of score formula](Text), closingDate(Date) |
| tblApplications | applic\_ID | cource\_ID,  acc\_ID | [list of marks](Text), preCalculatedScore(Double), submitted(Boolean), dateUpdated(Date) |



**Text Files**: Can be used to upload list of subjects and marks by the applicant

and to export an application for printing by a staff user. A text file will also be

used to store the application event log for the admin to review.

**Enums**: Will be used to define the state of the application and some of the

dynamic forms, makes code easier to read and understand.

**Records:** Advanced 2D array which allows different data types, unique to

Delphi/pascal. Will be used to define data structures such as a user or

application and thus transport data between objects/forms more efficiently.

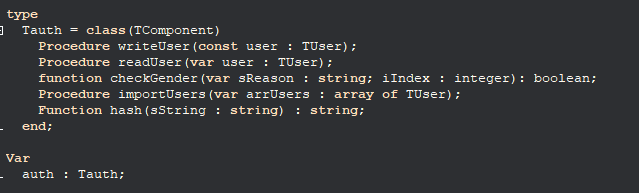
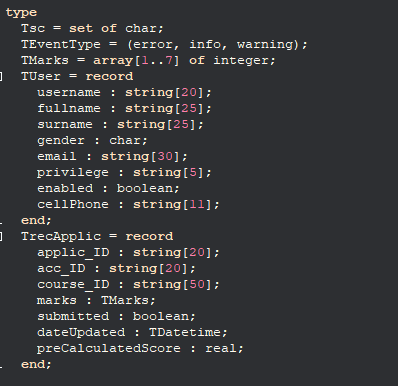
**Arrays**: Arrays will be used for the parallel mathematical calculations done

by the scoring algorithm used to process the marks from the applications

dynamically.

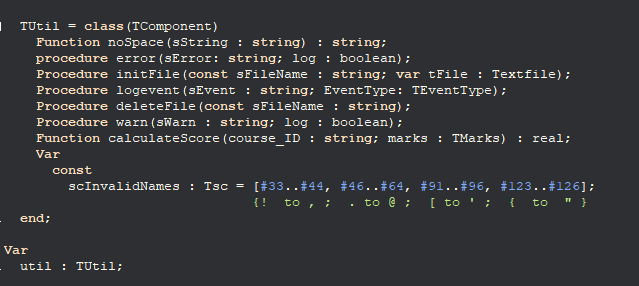
**Class and record definitions:**

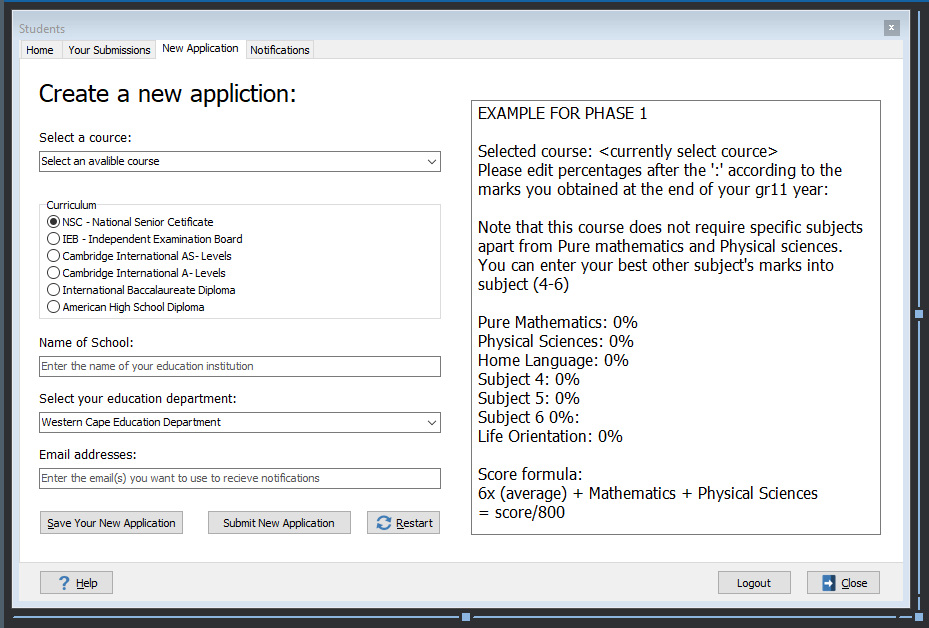
auth\_u.pas



util\_u.pas

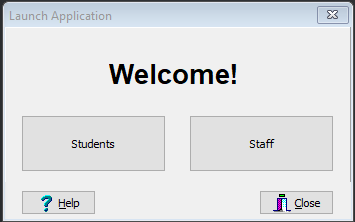
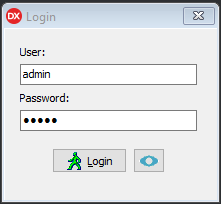
util\_u.pas



1. **GUI design – (more last page)**
2. **Software Design Tool: Input, processing and output (IPO)**

**New Application/Edit application:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI components** |
| Marks achieved | Keyboard | Subject: String  Mark: Integer | User edit Percentages (text preloaded) in richedit | richedit |
| **Data Validation** | - Check if all values have changed, otherwise prompt error message informing the user to make sure to input all their marks.  - Check if values(marks) are in range 0 – 120 (some curriculums you can achieve above 100%) and display message if not | | | |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI components** |
| Application data  - Course  - School name  - Education Department  - Curriculum | Keyboard and mouse | Store everything in a record of strings, enums and integer | - Select from predefined options in combo-boxes  - Select an option from Radio Group  - Text entered into the edit | - Combobox  - RadioGroup  - Edit |
| **Data Validation** | - Check that an option is selected in combo-boxes  - Check that only valid characters are used in the email and school name edits, display which characters are accepted otherwise in a prompt.  - Check that the edit fields are filled in, display error message to user that they should please fill it in before proceeding | | | |
| **Processing** | **Action** | | | **GUI component trigger** |
| Process application data entered by user into a predefined record | 1. Fetch data from components after data validation took place without errors and copy to predefined record   - copy from edt.text; copy string from cbx.Items[cbx.ItemIndex]; get integer from rgp.ItemIndex  - Get acc\_ID from global current user record variable updated when user logged in.  - Get current date with ‘Date’  - If submit button was clicked, set submitted to true, otherwise false   1. Generate a unique application ID for primary key in database   - recApplic.applic\_ID := Copy(currentUser.username, 1, 3) + IntToStr(RandomRange(1000,10000));   1. Run calculation on marks with a Function receiving algorithm definition and marks and returns the score   algorithm def example: [(a+b+c+d+e+f)\*6 + a +b] – \*6 (multiplication) is after sum, algorithm executes calculation from left to right   1. Copy score from function into record   recApplic.score := util.calculateScore(recApplic.marks);   1. Clear all the inputs and reset view to Home where user can see their new application   - Home page will show a summary of all applications by the current user.  6. When the user wants to edit an application by selecting the index with a spin edit and clicking the edit  button, the data must be loaded from tblApplications into the Application record. From there the data can  be loaded into the GUI components.  7. After the data was altered, it needs to be validated again and this whole process gets run through again  from start to finish (without generation of application id nr. 2) and record gets updated instead of inserted  into the table. | | | Submit or Save Button |
| **Output** | **Format** | | | **GUI component** |
| Database | Push Record to a record in tblApplications via a database update helper function | | |  |
| DlgMessage | If database update helper function returned with no error, display a success message to the user  - Otherwise inform the user of an error and log the error | | | DlgMessage |
| Summary and Score(real) | After successful addition of new application, display richedit with tabs the course, notification email, submission status and calculated selection score achieved for that course. Formatted using FormatFloat(score, ‘0.00’) + ‘/800’; | | | Richedit |

** **