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:

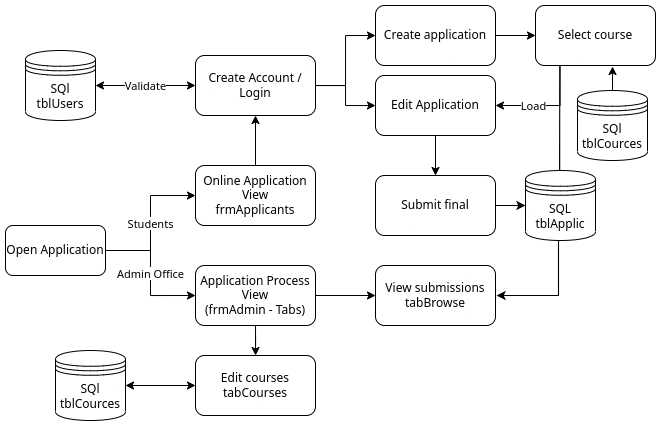
* 2 separate user interfaces, one for the applicants and the other for the institution staff
* An administrator role to manage staff user accounts
* A way for new applicants to create an accounts and log into existing accounts
* Staff members can edit courses offered and dynamically update the scoring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

My solution is create 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 | Access to everything | None |
| Applicant | Temporary account created by applicants | Create/Edit/Submit application | Access other applicants’ submissions and review applications |
| Admin office  (Staff) | Account used to review submitted applications | Manage/Review applications | Create/Edit Applications |

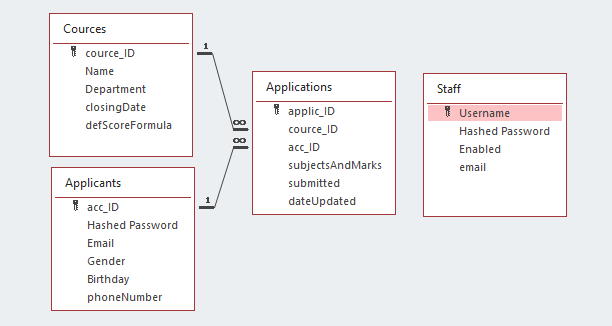
1. **Navigation/Description of Flow Diagram**



1. **Data Structures**

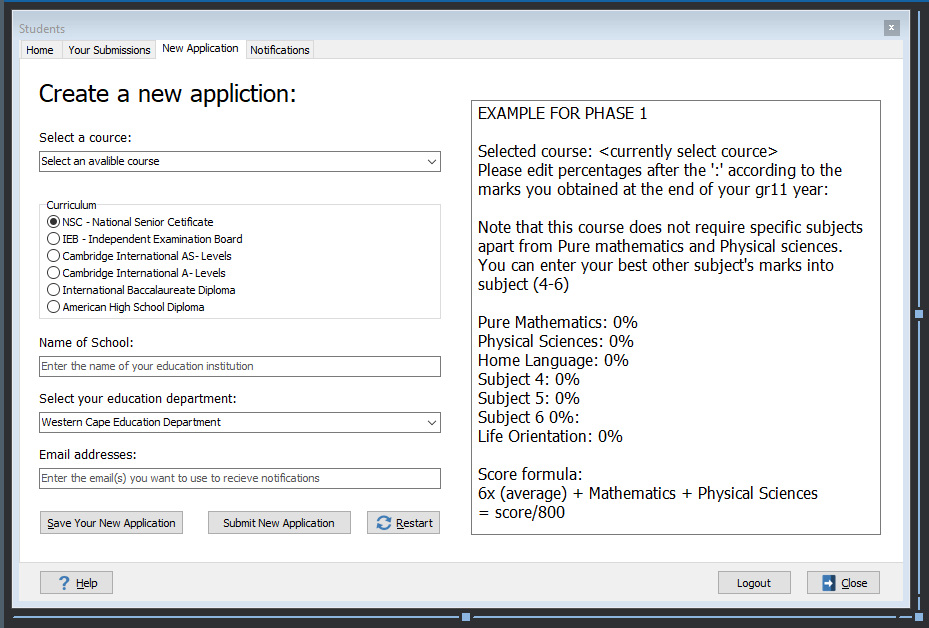
* **Database**:

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



* **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.

1. **GUI design**



1. **Software Design Tool: IPO – ANY TWO (Tab sheets / Forms)**

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | VALIDATION | PROCESSING | OUTPUT |
| -What data will be stored | -What data will be validated | -What will be processed | -What will be displayed |
| -Data type | -Show algorithm | -How will the processing be done | -Format to be displayed |
| -Variety of source: From keyboard / mouse click / computer generated | -Associated error message |  | -Object to be displayed in |

**Input, processing and output IPO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** | |
| Subject and mark achieved | Keyboard | Subject: Enum  Mark: Integer | Percentage without % | Combo box for subject selection and spinedit for Mark ingestion | |
| **Data Validation** | No point in checking combo box as there is pre defined subjects | | | | |
|  | Zero check edtMark (if NOT(spmMark.value > 0))  Error: “Please check that you entered all your marks” | | | | |
| **Processing** | **Action** | | | | **GUI component** |
| Generate Cipher text | WHAT:  Capture plain text and encrypt to Cypher text  HOW:  Read plain text  Do null check if true show Error message 1  **else**  Do range check if false show error message 2 **else** //encrypt  FOR 1 to the length of the Plain text  Letter  a letter from the plain text  Determine the position of the letter in the alphabet  Add each corresponding letter in the mono alphabet to the cipher text | | | | Button |
| **Output** | **Format** | | | | **GUI component** |
| Cipher Text | String | | | | RichEdit |

**Examples of IPO’s:**

**Input, processing and output IPO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** | |
| Plain Text | Keyboard | String | Sentence in English | Edit box | |
| **Data Validation** | Range check  sPlainText[i] IN [‘A’..’Z, ‘a..z’, ‘ ‘]  Error message 1  “Only Alphabet letters and spaces allowed” | | | | |
|  | Null Check  sPlainText = ’’  Error message 2  “Edit cannot be empty, provide a message to encrypt” | | | | |
| **Processing** | **Action** | | | | **GUI component** |
| Generate Cipher text | WHAT:  Capture plain text and encrypt to Cypher text  HOW:  Read plain text  Do null check if true show Error message 1  **else**  Do range check if false show error message 2 **else** //encrypt  FOR 1 to the length of the Plain text  Letter  a letter from the plain text  Determine the position of the letter in the alphabet  Add each corresponding letter in the mono alphabet to the cipher text | | | | Button |
| **Output** | **Format** | | | | **GUI component** |
| Cipher Text | String | | | | RichEdit |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task:** | **Input** | | | | | | **Processing** | | **Output** | | |
| **General** | | | **Validation** | | |
| **Source (GUI Component)** | **Data Type** | **Format** | **What** | **Method** | **Feedback** | **What** | **How** | **What** | **Format** | **Object** |
| User Registration | Keyboard (edtName) | String | Sentence Case | Numbers/  Special Characters | Loop through and check each char. | “Error! Your name cannot contain….” | Store in variable | sName := edtName.Text | Confirmation Message: “Your details have been stored successfully” | Text | Dialog Box |
| Keyboard (edtSurname) | String | Sentence Case | Numbers/  Special Characters | Loop through and check each char. | “Error! Your surname cannot contain….” | Store in variable | sSurname := edtSurname.Text |
| Mouse (dtpDOB) | TDate | Regional Settings  (dd/mm/yyyy) | None  (built-in) | N/A | N/A | Store in variable | sDOB := DateToStr(dtpDOB.Date) |
| Keyboard (edtCell) | String | Digits without any spaces in between e.g.  0789419712 | Has 10 Chars | if(length(edtCell) <10) then…. | “Your cell phone number must have 10 chars” | Store in variable | sCell := edtCell.Text |
| Calculate Total Price | Mouse (spnQty) | integer | Numeric | Range (no more than 5 items) | MaxValue Property | “Error! You cannot have more than 5 items!” | Calculate Price | quantity \* price (constant) = total price | “Your total comes to <price>” | Text  Currency | Price Label |