

Richard Huang

Project Design Document:

North Arrow





North Arrow

XiaoRui Huang

Exam Number 181012020244

Center Number: 1012

# Table of Content

[Table of Content 2](#_Toc523779866)

[Summary 4](#_Toc523779867)

[GUI Design 5](#_Toc523779868)

[Login Frame 6](#_Toc523779869)

[Sign Up Panel 7](#_Toc523779870)

[Main Frame 9](#_Toc523779871)

[Main Frame Menu 9](#_Toc523779872)

[Student Information Tab 11](#_Toc523779873)

[Personalized– Programmes Tab 13](#_Toc523779874)

[Academic Progression Tab 16](#_Toc523779875)

[Object diagram 17](#_Toc523779876)

[Student 17](#_Toc523779877)

[Field 17](#_Toc523779878)

[Method 18](#_Toc523779879)

[YearMark 20](#_Toc523779880)

[Field 20](#_Toc523779881)

[Method 21](#_Toc523779882)

[TermMark 22](#_Toc523779883)

[Field 22](#_Toc523779884)

[Method 22](#_Toc523779885)

[DBManager 23](#_Toc523779886)

[Field 23](#_Toc523779887)

[Method(donno whats gonna be in here so just gonna leave it for now) 24](#_Toc523779888)

[Database Design 26](#_Toc523779889)

[tblStudent 27](#_Toc523779890)

[tblYearMark 27](#_Toc523779891)

[tblTermMark 28](#_Toc523779892)

[tblProgramme 29](#_Toc523779893)

[Text Field Design 29](#_Toc523779894)

[Flow Charting 31](#_Toc523779895)

[Login Frame 31](#_Toc523779896)

[Sign Up Frame 32](#_Toc523779897)

[Main Frame Menu 33](#_Toc523779898)

[Student Information Tab 34](#_Toc523779899)

[Personalized Programmes Tab 35](#_Toc523779900)

[Academic Progress Tab 36](#_Toc523779901)

[Help Design 37](#_Toc523779902)

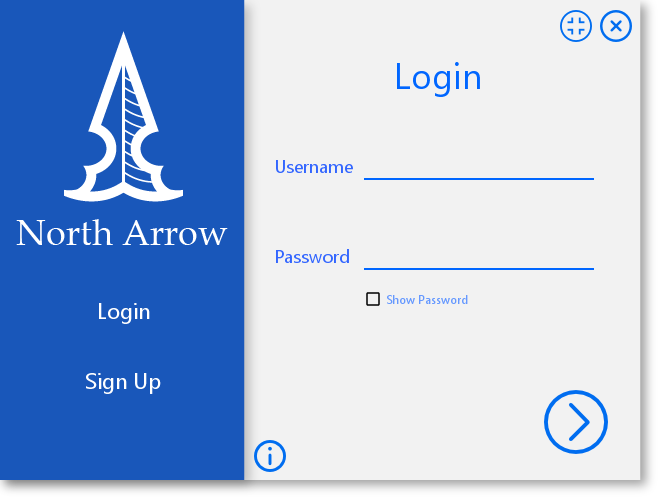
# Summary

The Program at use is a University programme advisor created by Richard Huang. The program is designed to be implemented at any NSC/IEB High School for student who (South African citizen or PR holder only, as there are different criteria for anyone otherwise) struggle to decide their university and programme (A Bachelor’s Degree) choices in South Africa (currently only supports UCT, Wits, UP, Stellenbosch). The program ensures that Students can view their possible choices with detail and ease. The program comprises an easy to interface (using Google’s material design) and features specially designed to help students to decide their future study plans and aims. The program records user’s academic records and personal traits to provide viable programmes for the student. This eliminates the ass-biting work of calculating various APS score, filtering out not applicable programmes, pinpoints academic weakness, while providing a graphical academic progress tracking interface in aid for students to get admit into their desired programmes (more useful for lower grade users). It simplifies the complex tertiary education admission system, categorising, summarises and provided a simple, elegant way to making a choice.

# GUI Design

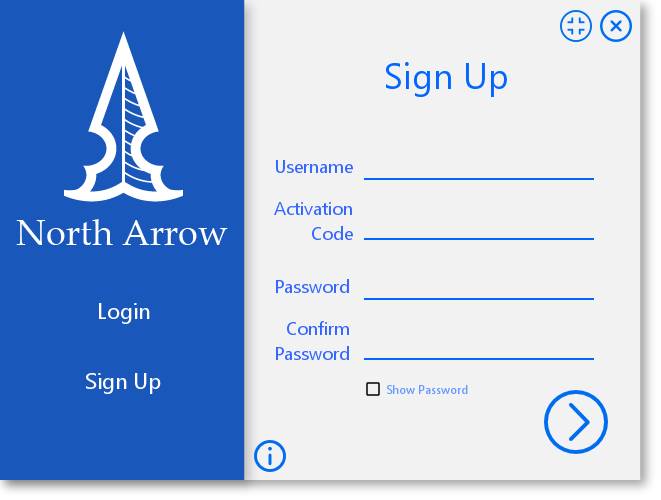
* Most icon in this programme enables hover feature, meaning that it will change icon to a word instead of just a picture. e.g. the “i” icon, when the mouse it on top of it will display “Help”
* “Cancel” button is visible in all frame of this Programme- This ends the Programme
* “minimize” button is visible in all frame of this Programme- This iconifies the Programme
* (Just trying to show off) A shadow effect is applied to all frame to create a material design look and feel

## Login Frame



|  |  |
| --- | --- |
| **Description** | This is the first program that appears in the program. The frame allows users to login or register to be able to access further features |
| **Security Group** | Anyone is able to access the frame by running the program. However only people with valid login credentials will be able to get past this frame. This restricts who is allowed to use the program. |
| **Data** | Username text field is used to get the username from the user.  Password text field is used to get the password from the user. Password in the text file is **decrypted** and checked against user input.  The two pieces of data are used when logging in and checked against the Password.txt file that is accessed by the scanner  It acts as an initial display and platform for further frames |
| **Actions** | * “Sign Up” button clicked- This takes the user to the register page * “Next” button clicked(button with arrow to the right)- This verifies login inputs in the text field and either proceeds to Main frame or returns an error * “Login” button in the login frame will do nothing * “Help” button clicked- This provides instructions to the user on how to deal with this frame. * “Show Password” toggle allows convenience for user when an error is made when entering their password, while keeping the security feature that all character entered is in bullet point form |

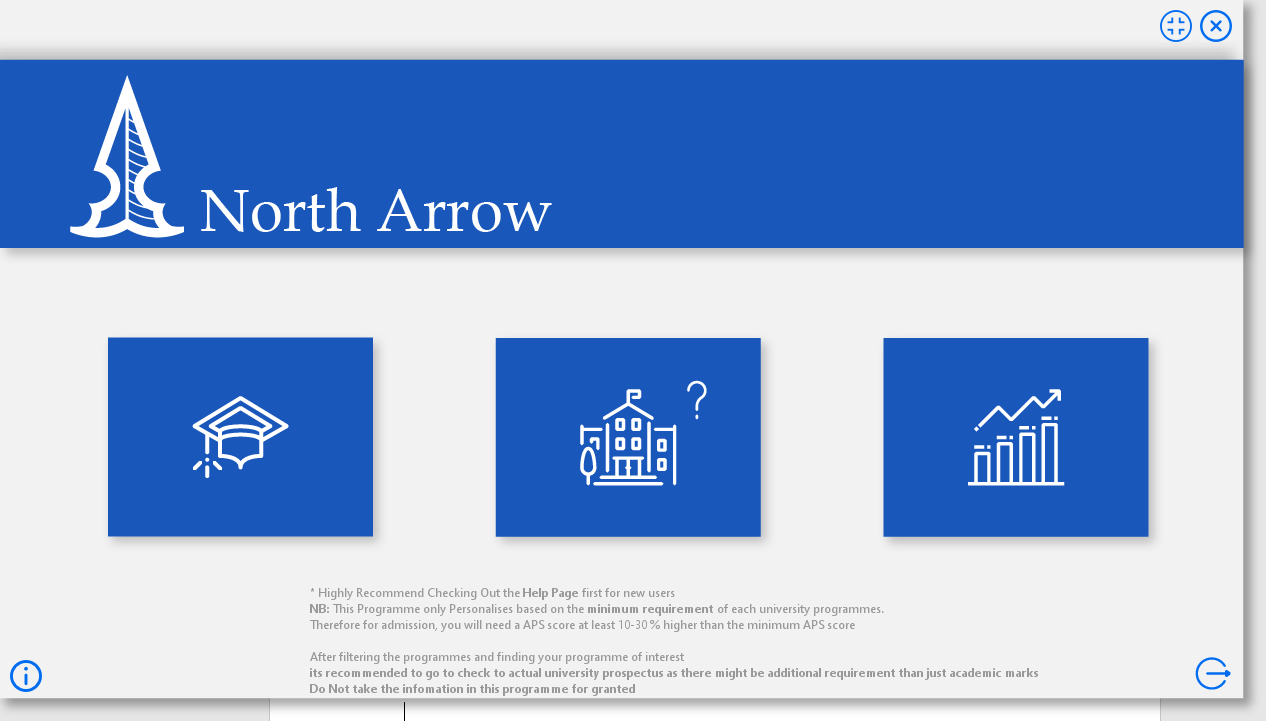
## Sign Up Panel



|  |  |
| --- | --- |
| **Description** | This page allows users to register and to be able to proceed to further features in the program. The page is accessed when “Sign Up” is clicked on the side menu of the frame. Data is taken from the text fields, **encrypted** and then saved in a text file. Precautions (**confirm password field**) are taken to make sure data is entered and that passwords match. |
| **Security Group** | Anyone is able to access the frame by running the program, proceeding from the login frame. However, only users with the right activation code will be successfully registered. |
| **Data** | The first text box is used to get a username from the user. Username is checked with existing usernames to make sure the username is **unique**. This is saved to the text file and in the database (primary key) if the registration is valid.  The second textbox is used to verify the activation code which is checked against the specific code of the program that allows users to register. If not matching, an error will appear.  The third and fourth text boxes are used to get a password from the user. These two values must match to be able to proceed. The password is processed and **encrypted** within the program. This is saved to the text file if the registration is valid. |
| **Actions** | * “Next” button is clicked(button with arrow to the right)- This will add the new username and password to the text file, provided that all the specifications are met. * “Sign Up” button in the sign up frame will no nothing * “Login” button is clicked- This will return the user to the login frame * “Help” button clicked- This provides instructions to the user on how to deal with this frame. * “Show Password” toggle allows convenience for user when an error is made when entering their password, while keeping the security feature that all character entered is in bullet point form |

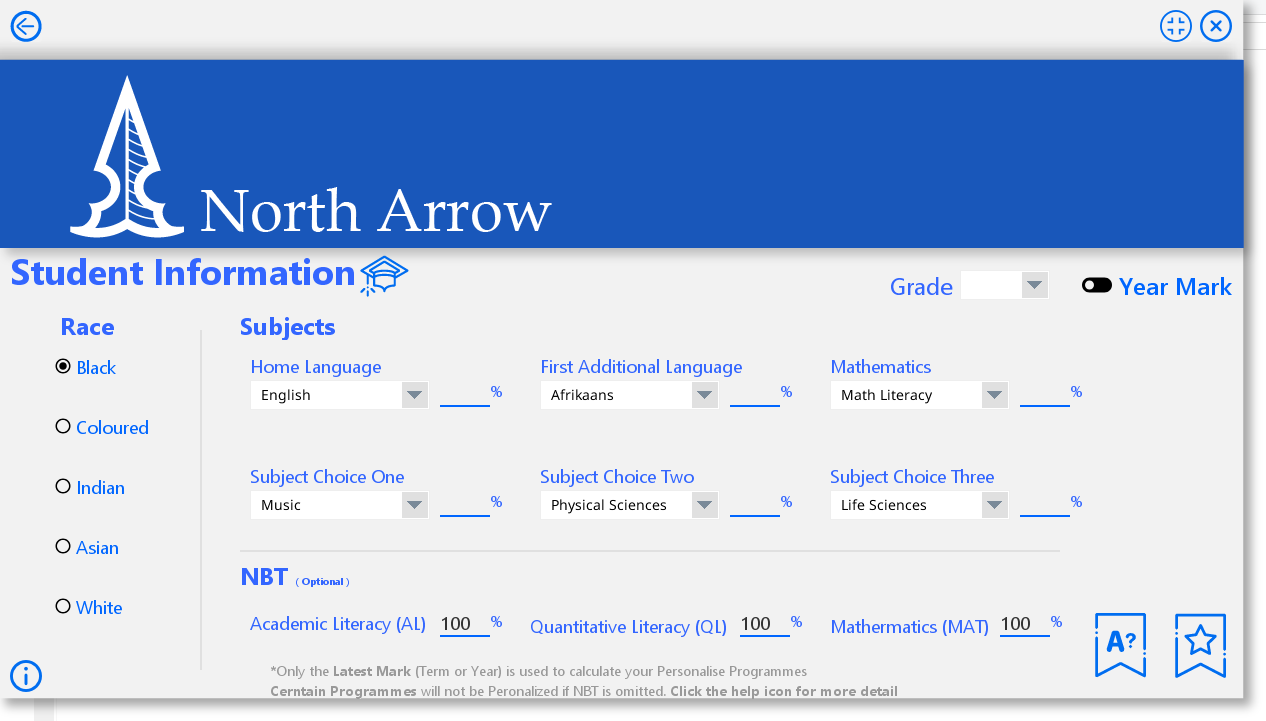
## Main Frame

### Main Frame Menu



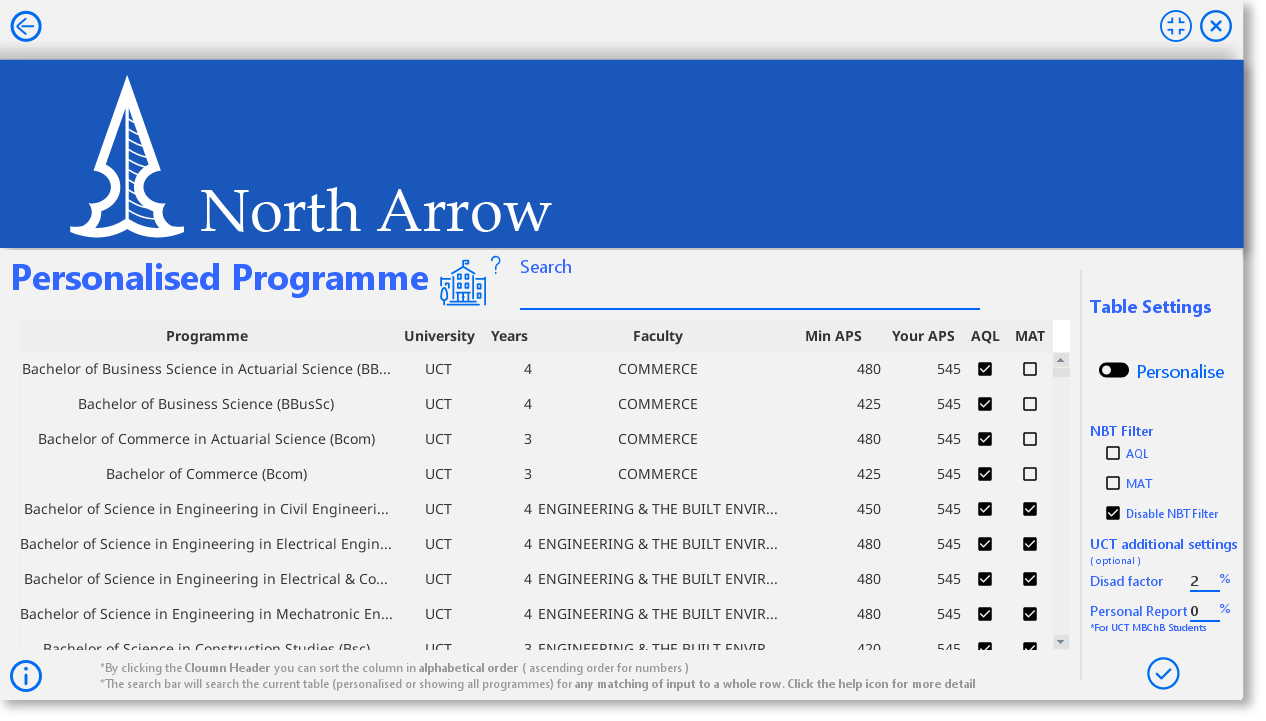
|  |  |
| --- | --- |
| **Description** | This tab allows users to input all necessary data to provide access for more features. The page is accessed when login is successful clicking “Login” on the login frame. |
| **Security Group** | Only people with valid login credentials will be able to get to this tab. There are no security features for or within this particular tab. |
| **Data** | Username (a primary key in the data base) is pushed through to the main frame from the login page. This is a key data in specifying where other data goes and will be used by all other tabs.  Data validation technics are fully implemented. E.g. restricted access to only “student information” Tab is applied on new users until sufficient data is inputted through the Student Information tab to unlock the relevant features.  Relevant error message will be displayed when insufficient or incorrect data is inputted |
| **Actions** | * “Student Information” button is clicked- This will take user to student information tab * “Personalized Programme” button is clicked- This will validate accessibility and take user to the Personalized Programme tab * “Academic Progress” button is clicked- This will validate accessibility and take user to the Academic Progress tab * “Log Out” button is clicked- this will pop out a confirm box, and on confirmation will log the user off and take the user back to the login frame * “Help” button clicked- This provides instructions to the user on how to deal with this tab. |

### Student Information Tab



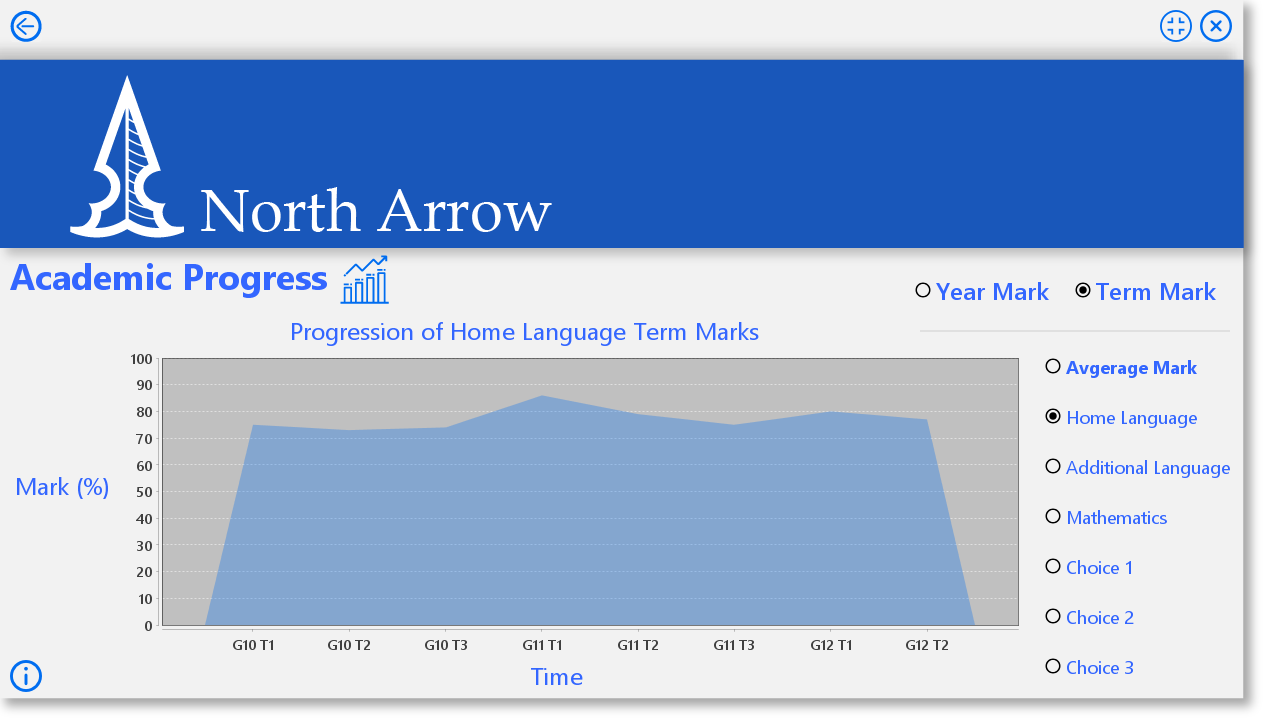
|  |  |
| --- | --- |
| **Description** | This tab allows users to input all necessary data to provide access for more features. The page is accessed when “student information” is clicked in the main frame menu. Data is taken from the text fields, combo boxes, radio buttons and saved in a Database. |
| **Security Group** | Only people with valid login credentials will be able to get to this tab. There are no security features for or within this particular tab. |
| **Data** | Data validation technics are fully implemented. Empty data, data out of range and data in incorrect types are all covered.  Relevant error message will be displayed when insufficient or incorrect data is inputted  The “Race” radio button, each has a button group, hence only one option can be selected at a time.  The six subject-combo box or drop-down box allows the user to select their relevant NSC designated subjects for future function references.  The six text field next to the combo boxes are used to get the students relevant subject marks. The program will only accept integers from 0 to 100, if not an error message will be displayed when “save” is clicked.  The three text field under NBT label is optional allowing users to get more accurate info from other features. It can be left blank, but if enter a range limit of 0 to 100 is applied  “Grade” and “Term” combo box are used to determine what type of data and where the data is to be entered. “Term” combo box will only be visible one the toggle button is toggled to “Term Mark”  This cannot be empty; data validation is applied.  All above user input will be recorded and stored in their relative tables in data base. |
| **Actions** | * “Save” button is clicked- This will perform the majority of data validation, if passes, execute code to store all user input to a data base * “Check your marks” button is clicked. This will find the correct user’s mark at a specific period, chosen by the user, in the data base and display it onto the text fields, in case a user is not sure if data have been entered or just want to quickly check their marks from the past. If no data if found error message will be displayed. * “Back” button- this will take you back to the main frame menu * “Help” button clicked- This provides instructions to the user on how to deal with this tab. |

### Personalized– Programmes Tab



|  |  |
| --- | --- |
| **Description** | This tab allows users to view their APS score for relative Universities, possible Programme available, alongside the minimum APS required for a direct estimation of their chances of admission (or all the Programmes by the toggle settings) and if the Programme requires NBT’s AQL and MAT. Settings can be applied according to user’s choice. Relevant calculation is performed base on previous user input and displayed on the text field and table. |
| **Security Group** | Only people with valid login credentials will be able to get to this tab. There are no security features for or within this particular tab. |
| **Data** | No data send and stored in data base in this tab, only validated and selected for calculation purposes. Except user input in the settings to change their APS scores for UCT (this data is still not stored).  Programme information is displayed in a Table, each row for one Programme. The table contains the Programme name (the full name-took me a whole day to type them in), the University that provides the Programme, number of year of study, the faculty it belongs in that university, the minimum APS needed to be considered for admission, your APS score relevant to the university or Programme and if it needs just AQL or Both NBT test.  The Search bar will accept any input and attempt to find and matching sequence or a whole row. If found it will display the data. This feature is dynamic and will update and filter with each keystroke. E.g. if you enter 4-to find all Programmes that takes 4 years to complete it will not just do so, but also find all Programmes with a min APS of 400 or something similar.  Search only applies on the current table meaning that if you search on top of a filtered table if will only find the Programme in the Filtered Table and not all Programmes |
| **Actions** | * Check boxes “AQL”, ”MAT”, “Disable NBT Filter” gives user the option to view more precise data of whether a Programme require NBTs or not. * “Personalize/All Degree” toggle allows an option to view Programmes that is currently not available to the user cause of subject choices or low marks * “Back” button- this will take you back to the main frame menu * “Disad(vantage) Factor” is a UCT specific feature to boost underprivileged students based on their race and past school attended. This is determined by UCT, and this feature is just a self-estimation. 0-10%(20% for MBChB degree) optional   By default 8% for Black students-16% for MBChB  6% for Coloured students-12% for MBChB  4% for Indian students-8% for MBChB  2% for Asian students-4% for MBChB  (consider your school as well, less premium means higher %)   * “Personal Report” is a UCT MBChB only feature, a mark from 0-100. Optional(if not filled UCT MBChB will not appear in the personalized table) * “Apply” button- This is activate all the above settings and refresh the table accordingly * “Help” button clicked- This provides instructions to the user on how to deal with this tab. |

### Academic Progression Tab



|  |  |
| --- | --- |
| **Description** | This tab allows users track their academic progress graphically. This feature function as a warning visually when academic achievement is plummeting and as an encouragement to do better when academic achievements is on a constant rise, it also helps to establish aims in academics. |
| **Security Group** | Only people with valid login credentials will be able to get to this tab. There are no security features for or within this particular tab. |
| **Data** | No data send and stored in data base in this tab, only validated and selected for calculation purposes. Except user input in the settings to change their APS scores for UCT (this data is still not stored).  Relevant data is selected from the data base and processed into an array. Then displayed as a graph. The graph is refreshed with every single toggle button change. |
| **Actions** | * “Year/Term mark” switch will determine if the data extracted is a term or year mark, it also refreshes the graph accordingly * Subject specific toggle- displays a specific subjects progression. Every action on a toggle will execute code to extract data from the data base and calculate to display relevant graph within the JFreeChart immediately. Therefore, the information within this tab is always up to date with the latest user input.   Hovering on the graph for a few second will display a precise value at that point. Click and drag right an area will zoom into the graph and a click and drag left will reverse that. Left click will give you more options to play with the graph.   * “Back” button- this will take you back to the main frame menu * “Help” button clicked- This provides instructions to the user on how to deal with this tab. |

# Object diagram

## Student

This is used to create individual objects based on the data obtained from the database for easy handling, all other object is summarized here.

### Field

|  |  |
| --- | --- |
| Field | Description |
| -username: String | Stores the student’s username |
| -race: String | Stores the student’s race |
| -homel String | Stores the student’s Home Language Subject |
| -addl: String | Stores the student’s First Additional Language Subject |
| -math: String | Stores the student’s Choice of Math-Math or Math Literacy |
| -ch1: String | Stores the student’s Choice 1 Subject |
| -ch2: String | Stores the student’s Choice 2 Subject |
| -ch3: String | Stores the student’s Choice 3 Subject |
| -int: AL | Stores the student’s AL Marks |
| -int: QL | Stores the student’s QL Marks |
| -int: MAT | Stores the student’s MAT Marks |

### Method

|  |  |
| --- | --- |
| Field | Description |
| -Student (String inUsername, String inRace, String inHomel, String inAddl, String inMath, String inCh1, String inCh2, String inCh3, int inAL, int inQL, int inMAT) | Constructs a student object |
| +getUsername(): String | Returns the student’s username |
| +getRace(): String | Returns the student’s race |
| +getHomel(): String | Returns the student’s Home Language Subject |
| +getAddl(): String | Returns the student’s First Additional Language Subject |
| +getMath(): String | Returns the student’s Choice of Math-Math or Math Literacy |
| +getCh1(): String | Returns the student’s Choice 1 Subject |
| +getCh2(): String | Returns the student’s Choice 2 Subject |
| +getCh3(): String | Returns the student’s Choice 3 Subject |
| +getAL(): int | Returns the student’s AL marks |
| +getQL():int | Returns the student’s QL marks |
| +getMAT():int | Returns the student’s MAT marks |
| +getLatestYearMark():YearMark | Returns the latest year mark object |
| +getLatestTermMark():TermMark | Returns the latest term mark object |
| +isYearLater(YearMark ym, TermMark tm):boolean | Decides if the year mark or term mark is the latest |
| +getLatestMark():LatestMark | Returns the latest mark object |
| +identifyMusic ():String | Identify which choice position(“1”, ”2”, ”3”) the Music subject is in. if user does not take Music returns “0” |
| +identifyPhysicalScience():String | Identify which choice position(“1”, ”2”, ”3”) the Physical Science subject is in. if user does not take Physical science returns “0” |
| +identifyLifeScience ():String | Identify which choice position(“1”, ”2”, ”3”) the Life Science subject is in. if user does not take Life Science returns “0” |
| +getStellen():int | Returns the student’s Stellenbosch APS according to their Latest mark |
| +getUCT\_ScienceFPS():int | Returns the student’s UCT APS for Science faculty according to their Latest mark |
| +getUCT\_HealthFPS ():int | Returns the student’s UCT APS for Health faculty according to their Latest mark |
| +getUCT\_MBChB\_FPS(int PR) :int | Returns the student’s UCT APS for a MBChB degree according to their Latest mark |
| +getUCT\_normFPS ():int | Returns the student’s UCT APS for the rest of the faculties according to their Latest mark |
| +sevenPointScale (int mark):int | Accepts a mark return its relative value in the NSC seven point scale. |
| +getNSC():int | Returns the student’s APS according to their Latest mark in the form of NSC seven point scale. Used by UP |
| +getWits ():int | Returns the student’s Wits APS according to their Latest mark |
| +sevenPointScaleWits (int mark):int | Accepts a mark return its relative value in the NSC seven point scale, wits version. |
| +sevenPointScaleWitsEM (int mark):int | Accepts a mark return its relative value in the NSC seven point scale, wits version for English and Math |

## YearMark

This creates an object that mimics factors based on Annual School Reports

### Field

|  |  |
| --- | --- |
| Field | Description |
| +Username: String | Stores the student’s username |
| -grade: int | Stores the student’s Grade |
| -homel: int | Stores the student’s Home Language Subject Mark |
| -addl: int | Stores the student’s First Additional Language Subject Mark |
| -math: int | Stores the student’s Choice of Math-Math or Math Literacy Mark |
| -ch1: int | Stores the student’s Choice 1 Subject Mark |
| -ch2: int | Stores the student’s Choice 2 Subject Mark |
| -ch3: int | Stores the student’s Choice 3 Subject Mark |
| -avg:int | Stores the student’s final Year Mark |

### Method

|  |  |
| --- | --- |
| Field | Description |
| - YearMark(String inUsername, int inGrade, int inHomel, int inAddl, int inMath, int inCh1, int inCh2, int inCh3) | Stores the student’s Grade |
| +getGrade: int | Returns the student’s Grade |
| +getUsername: String | Returns the student’s Race |
| +getHomeLang: int | Returns the student’s Home Language Subject Mark |
| +getFAddLang: int | Returns the student’s First Additional Language Subject Mark |
| +getMath: int | Returns the student’s Choice of Math-Math or Math Literacy Mark |
| +getCh1: int | Returns the student’s Choice 1 Subject Mark |
| +getCh2: int | Returns the student’s Choice 2 Subject Mark |
| +getCh3: int | Returns the student’s Choice 3 Subject Mark |
| + getAvg: int | Returns the student’s final Year Mark |

## TermMark

This creates an object that mimics factors based on Annual School Reports

### Field

|  |  |
| --- | --- |
| Field | Description |
| +Username: String | Stores the student’s username |
| -grade: int | Stores the student’s Grade |
| -term: int | Stores the term |
| -homel: int | Stores the student’s Home Language Subject Mark |
| -addl: int | Stores the student’s First Additional Language Subject Mark |
| -math: int | Stores the student’s Choice of Math-Math or Math Literacy Mark |
| -ch1: int | Stores the student’s Choice 1 Subject Mark |
| -ch2: int | Stores the student’s Choice 2 Subject Mark |
| -ch3: int | Stores the student’s Choice 3 Subject Mark |
| -avg:int | Stores the student’s final term Mark |

### Method

|  |  |
| --- | --- |
| Field | Description |
| - TermMark(String inUsername, int inGrade, int inTerm, int inHomel, int inAddl, int inMath, int inCh1, int inCh2, int inCh3) | Stores the student’s Grade |
| +getGrade: int | Returns the student’s Grade |
| +getUsername: String | Returns the student’s Race |
| +getTerm: int | Returns the term |
| +getHomeLang: int | Returns the student’s Home Language Subject Mark |
| +getFAddLang: int | Returns the student’s First Additional Language Subject Mark |
| +getMath: int | Returns the student’s Choice of Math-Math or Math Literacy Mark |
| +getCh1: int | Returns the student’s Choice 1 Subject Mark |
| +getCh2: int | Returns the student’s Choice 2 Subject Mark |
| +getCh3: int | Returns the student’s Choice 3 Subject Mark |
| + getAvg: int | Returns the student’s final term Mark |

# DBManager

For retrieving, inserting, updating student information, calculating APS scores other info to help create a graph

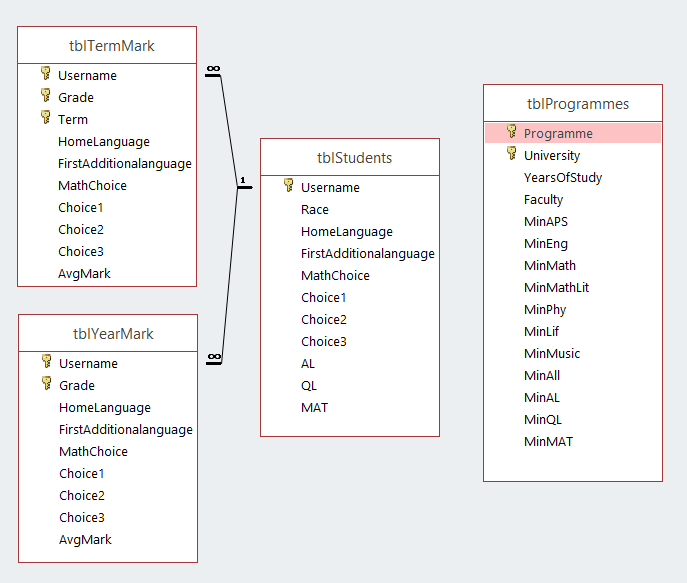
### Field

|  |  |
| --- | --- |
| Field | Description |
| -conn: Connection | Creates a variable that will be used to send establish a connection with the database |
| -filePath:String | Stores the File path to find the data base. |

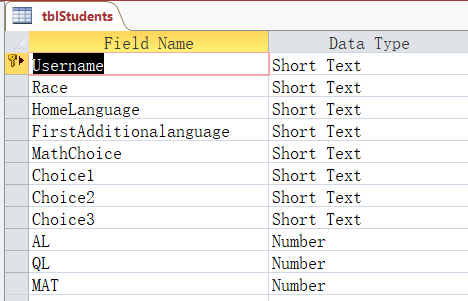
### Method(donno whats gonna be in here so just gonna leave it for now)

|  |  |
| --- | --- |
| Methods | Description |
| +DBManager() | Instantiates DBManager |
| +connect():Statement | A shortcut to link the program to the database. |
| +getAllProgramme():ResultSet | Returns a ResultSet containing all university programme |
| +isNewUser(String inUsername):boolean | Decides if a user is new to this program |
| +isNewYearMark(String inUsername, int inGrade):boolean | Decides if a Year mark is new or already exist |
| +isThereNoYearMark(String inUsername):boolean | Decides if there is a Year mark at all |
| +isNewTermMark (String inUsername, int inGrade, int inTerm):boolean | Decides if a Term mark is new or already exist |
| +isThereNoTermMark(String inUsername):boolean | Decides if there is a Term mark at all |
| +insertNewUser(String inUsername, String inRace, String inHomeLang, String inAddLang, String inMath, String inCh1, String inCh2, String inCh3, int inAL, int inQL, int inMAT):void | Insert a new user into the database |
| +updateStudentInfo(String inUsername, String inRace, String inHomeLang, String inAddLang, String inMath, String inCh1, String inCh2, String inCh3, int inAL, int inQL, int inMAT):void | Update an existing user info onto the database |
| +insertYearMark(String inUsername, int inGrade, int inHomeLang, int inAddLang, int inMath, int inCh1, int inCh2, int inCh3):void | Insert a new Year mark into the database |
| +updateYearMark(String inUsername, int inGrade, int inHomeLang, int inAddLang, int inMath, int inCh1, int inCh2, int inCh3):void | Update an existing Year Mark onto the database |
| +insertTermMark(String inUsername, int inGrade, int inTerm, int inHomeLang, int inAddLang, int inMath, int inCh1, int inCh2, int inCh3) :void | Insert a new Term mark into the database |
| +updateTermMark(String inUsername, int inGrade, int inTerm, int inHomeLang, int inAddLang, int inMath, int inCh1, int inCh2, int inCh3):void | Update an existing Term Mark onto the database |
| +populateStudent(String inUsername): Student | Returns a student object with that Username |
| +populateYearMark(String inUsername, int inGrade):YearMark | Returns a Year Mark object with that username and grade |
| +populateAllYearMark(String inUsername): YearMark[] | Returns a Year Mark object array with all the year mark records of that user |
| +populateTermMark(String inUsername, int inGrade, int inTerm) :TermMark | Returns a Term Mark object with that username, grade and term |
| +populateAllTermMark(String inUsername): TermMark[] | Returns a Term Mark object array with all the term mark records of that user |

# Database Design

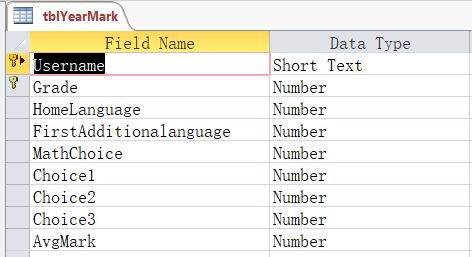


## tblStudent



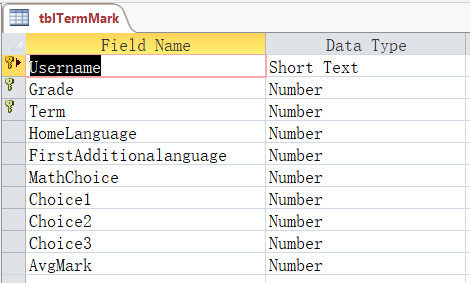
This table contains each student’s subject choices, race, NBT results and its unique username (primary key). This data will be used to create a Student object allowing for easier data manipulation without affecting the database.

## tblYearMark



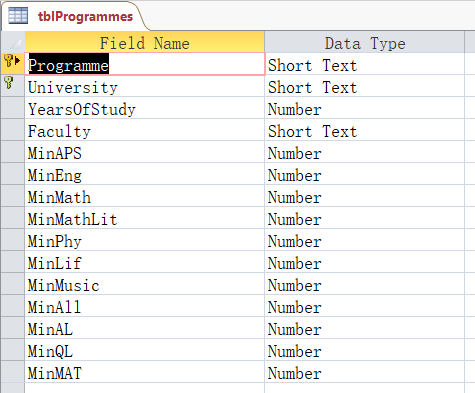
This table contains each student’s Year Mark and is uniquely identified by a composite primary key, Username and grade. This data will be used to create YearMark object allowing for easier data manipulation without affecting the database and APS calculations.

## tblTermMark



This table contains each student’s Term Mark and is uniquely identified by a composite primary key, Username, grade and term. This data will be used to create TermMark object allowing for easier data manipulation without affecting the database and help an academic progress graph.

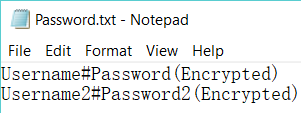
## tblProgramme



This table contains University Programme’s admission information and is uniquely identified by a composite primary key: Programme and University. This data will be used to formulate a personalized table that fits the user’s academic achievements.

(as a better programming practice this table should also be used to create an object, for ease of handling, apology in advance. I used the Resultset type data returned by a SELECT statement, that do however acts similar an object and do not affect the database. I will look out for such problems in the future)

# Text Field Design

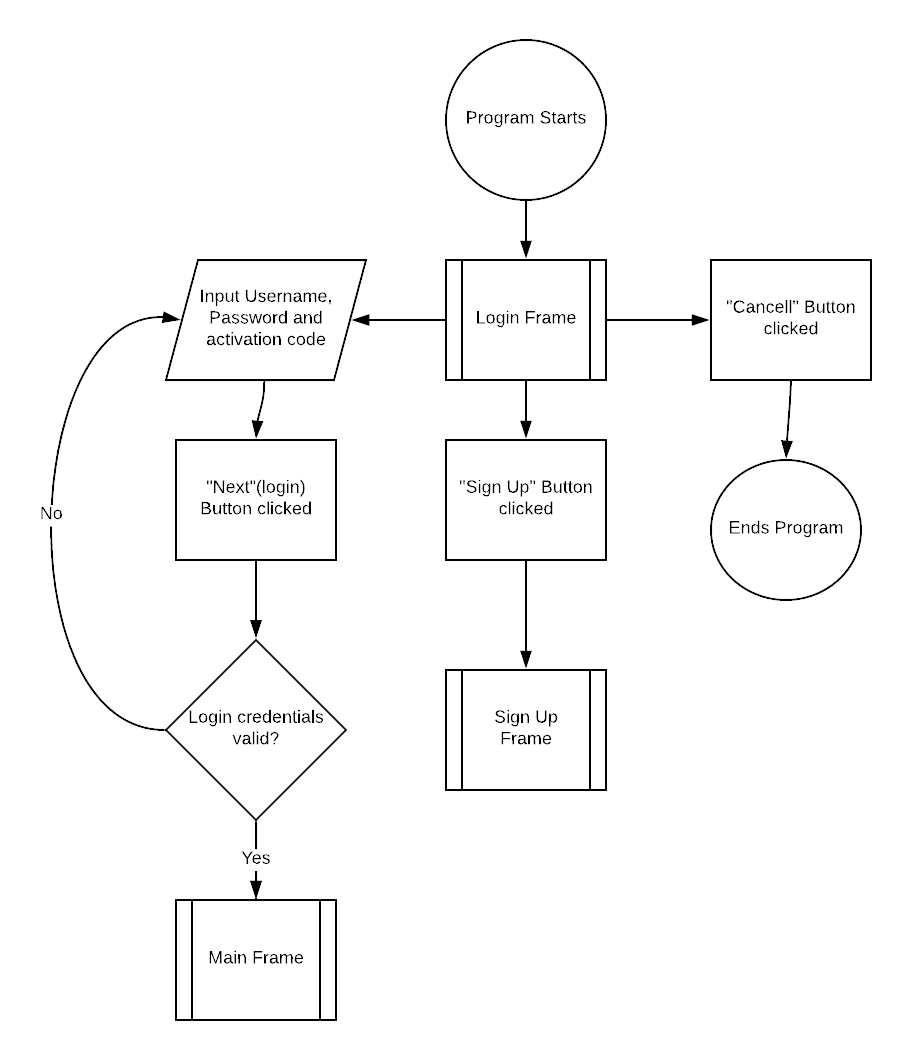
 File Format

 Encryption example

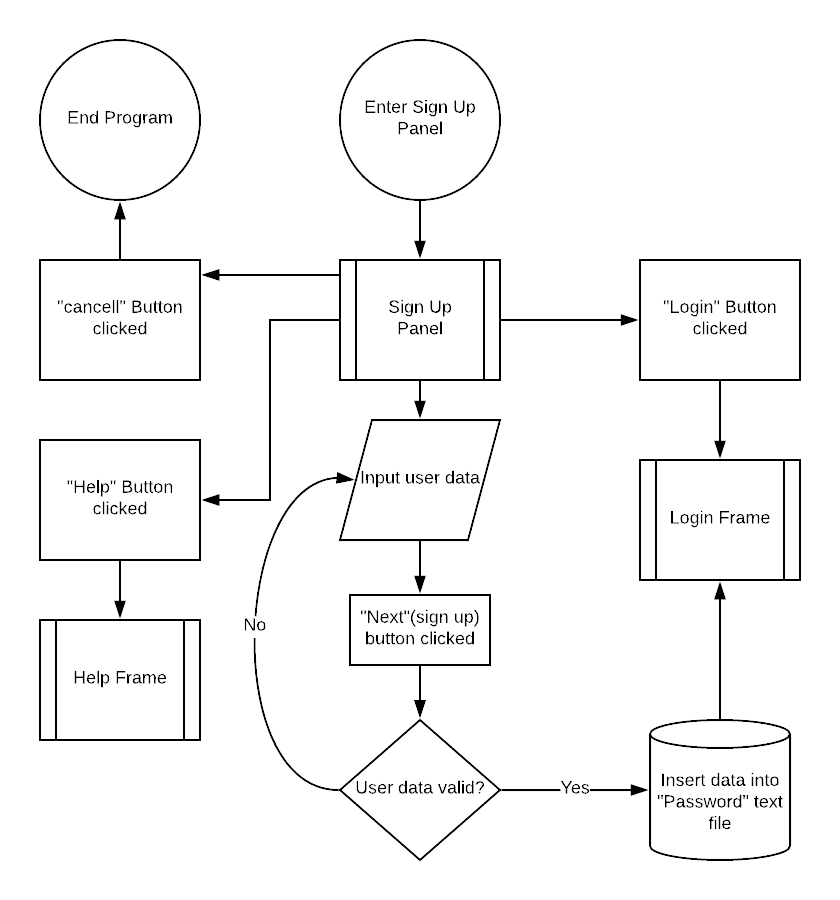
In this case a text file was used for data storage. The text file was used to store registered user’s username and password. A text file was used in this case as the data is very simple and through the use of a scanner, it would be the most simple and efficient option. For security purposes passwords will be encrypted and stored into the text file. When user input their password, the scanner extract the encrypted password and it decrypted within the program then compared for validation.

# Flow Charting

## Login Frame



## Sign Up Frame



## Main Frame Menu

## Student Information Tab

## Personalized Programmes Tab

## Academic Progress Tab

# Help Design

The user has the option on every frame to access the help screen. On each frame, a help button is clearly located at the left bottom of the window and if pressed, will pop up a new window with help information, pertaining to that specific screen. In each case, read the help information which will guide you through the frame and press close or enter to close the help page.

E.g.

