Jamie Cartwright Marks

**Grade 12 IT PAT 2022 Phase 1**

**Scenario and scope:**

The topic of this IT PAT is online learning, and the creation of an application which aids in the online learning process. I decided after careful consideration that the option involving the creation of a piece of software that is used in the process of scheduling online or, more specifically, tutoring lessons was the most practical and fitting task for me, when compared to the other possible takes on online learning mentioned in the PAT outline. I chose this specific option because I know first-hand how the market for external learning and tutoring is expanding and is a useful supplementary resource for many people including a considerable amount of my peers and friends, thus creating this program would help people organise and use the newly popularised online learning process via tutoring using my software solution.

The task or purpose of this project involves creating a piece of software in, the Delphi RAD IDE, that acts as a platform for both students and teachers to interact and schedule online learning/tutoring sessions, in an easy user-friendly manner thus aiding both learners and teachers in the online learning scenario and solving the issue of scheduling discrepancies and confusion involving online lessons. It also involves creating a well thought out, well-designed and user-friendly interface to access the scheduling software.

The solution to this task for me will be to create a lightweight, easy to use piece of software for both teachers and students to utilise in scheduling and managing online learning and tutoring sessions. The software assumes that communication about session dates and times is done outside of the software (between student and teacher) and the software is used just for the scheduling of these sessions. This system would work via the use of unique, user determined usernames and passwords which are stored securely in the database along with the associated necessary information about each student and teacher. The teacher will be the main entity in the system, once logged in to their account, the teacher will schedule a meeting using the software, along with a plethora of other features like creating reminders and queries regarding their schedule all within the software. This meeting will then be visible to the student once they log in to the software, thereafter, the student can perform actions like sorting through their upcoming sessions, changing log in details, accessing extra online material and leave a note regarding a certain session if they, for example, are going to be late or cannot attend the session which the teacher will be able to see once they log in to the software. The information entered by both teachers and students.

The scope of my project will focus mostly on the main sub-topic of the PAT which is a scheduling software for online lessons and tutoring to aid in the new trend of online learning, therefore the software will be well thought out and easy to use from all aspects and focused on the main purpose of scheduling lessons. The program will also involve a well-designed and correctly utilised database via Microsoft access. The database will be secure and have detailed but necessary information that is used within the software to help increase both the security and ease of access for all entities involved in the use of the software.

**User requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Teacher** | **Learner** | **Database administrator** |
| **Role:** | * *To view scheduled lessons and schedule lessons with learners* | * *To view scheduled lessons and edit information about those lessons* | * *To delete, insert and edit records in the associated database to this program* * *To ensure the efficiency of the database and to perform maintenance on the database* |
| **Activity:** | * *View future and past lessons* * *Edit teacher log in details* * *Sort and manipulate lesson data* * *Change database information about lessons* | * *To write notes before sessions regarding certain sessions* * *Edit log in details* * *View past and future scheduled lessons* * *To access the software’s educational resources* | * *Delete records in the database* * *Edit records in the database* * *Insert records in the database* * *Change the fundamental structure of the database* * *Add queries and filters* * *Perform routine maintenance on the database* * *Add or remove records and associated data* * *View the database without censorship* |
| **Limitations:** | * *Cannot change any of the source code or front-end of the program* * *Cannot control learner accounts or activity* * *Cannot edit the database in the program* * *Limited data access* |  | * *Cannot change any of the source code or front-end of the program* * *Cannot monitor the activity of the program* * *Cannot access tblAdmin table* * *Limited to only controlling the database of the program and associated data* |

**Description of flow diagram:**

**ENCRYPTION**:

Plain text message inputted via typing

Splash screen launched

Home page launched

Log in as previous user or new user

Text file name chosen and saved

Encrypted text outputted

Key word inputted

**DECRYPTION:**

Encrypted text message inputted, or text file loaded

Splash screen launched

Log in as previous user or new user

Home page launched

Text file name chosen and saved

Plain text outputted

Key word inputted,or loaded from database

**DATABASE ADMINISTRATION:**

Make changes and view Encryptions and Users table, as well as run queries

Log in as database administrator with registered passcode and username

Splash screen launched

Home page launched

Enter SQL section via button

Text file name chosen and saved

Run desired sql query

**DATA STRUCTURES:**

**-DATABASE (UsersDB):**

tblUsers:

Graphical user interface, application, table

Description automatically generated with medium confidence

tblEncryptions:

Table

Description automatically generated

tblAdmin:

Application, table

Description automatically generated

-**TEXT FILES:**

Text files will be used to store the encrypted messages after the keyword cypher has been used within the program. After the user has chosen an appropriate message as well as keyword, the encrypted message will be stored in a text file with a user-chosen name. Text files may also be used to read encrypted text for decryption into plain text. If the encrypted message is already saved to a text file within the program’s folder, the user simply has to input the name of the text file when prompted and the context of this text file will be displayed and copied to the user’s keyboard for ease of use, this decrypted text can then be decrypted with the aid of the correct keyword which will be inputted when the user wishes to decrypt the text. The textfile will be controlled by the “myfile”, of type textfile, variable.

-**ARRAYS:**

Arrays will be used in my program mostly in the database administrator form of the program. The arrays named arrUsername, arrPassword, arrNames, arrEncrypted, arrDecrypted and more. The most important arrays are the arrUsername and arrPassword arrays. These will be used to store all the usernames and all the passwords of the administrators which is stored in the tblAdmin table of the database. These arrays will then be used to verify that an administrator that is trying to log in to the database administration form of the program is both an existing administrator and also that the correct password (case sensitive) matches the correct username. Both of these arrays will be of type string and will have as many values in the array as there are admins in the tblAdmin table in the database.

**GUI**

*Splash screen:*

Background pattern, website

Description automatically generated

*A screenshot of a game

Description automatically generated with medium confidenceMain screen:*

Closes the program

Opens a link to a website explaining what a keyword cipher is

Opens the log-in page

Changes the colour scheme of the program

*Graphical user interface, text, application

Description automatically generatedLog-in page:*

Used to open the database administrator login page

Button used to input information for both returning and new users

For people who have used the software before

Graphical user interface, text, application, email

Description automatically generated*Encrypt page:*

‘Hints’ explaining the button function show up when hovering your curser over any button

Text controls

Offers an option to either view or edit account information

Provides detailed explanations of all the ins and outs of the program

Navigation

*Admin form:*

Graphical user interface, text, application

Description automatically generated

**DATA INPUT:**

|  |  |  |  |
| --- | --- | --- | --- |
| Source of Input | Data type | Format | Component used |
| Keyboard | String | A sentence or word | Input Box |
| Text file | String | A decrypted sentence or word | Rich edit |
| Keyboard | String | A word | Input Box |
| Database | String | A decrypted message used previously in the program | Rich edit |

**INPUT VALIDATION:**

|  |  |  |
| --- | --- | --- |
| **Data type** | **Input** | **Error message** |
| String | Keyword (a keyword, meaning one word, must be entered and not a sentence and the InputBox field cannot be null/empty) | **‘**The keyword is more than one word’  Or  ‘You have not entered a keyword’ |
| TextFile | Word(s)(the text file that is read from must contain words) and the referenced text file to read from must exist | ‘This file doesn’t exist’  Or  ‘This file is empty’ |
| Integer | 0, 1 or 2 (the radio group must have an index of 0, 1 or 2 otherwise a gender has not been chosen) | ‘You have not chosen a gender’ |

**Data Processing**

What Processing will be done:

* Encryption
* Decryption
* Creating a key with the keyword inputted by the user
* Removing non-letter characters from the inputted message
* Adding username, gender and date to the encrypted/decrypted message if everything is valid and storing message to a text file
* Reading from a text file

**How the processing will be done:**

Algorithms:

1. Algorithm 1 (Encryption):

***The purpose of this algorithm*:** To retrieve the index of each letter in the message within the alphabet and extract the letter from the key with the same index, then creating an encrypted message with each letter.

***Algorithm:***

*sValid is the alphabet*

*sKey is the key made from the inputted keyword*

var

M, iIndex : integer;

sInput, sOutput , sValid: String;

for M := 1 to length(sInput) do

begin

iIndex := pos(sInput[m], sValid);

sOutput := sOutput + sKey[iIndex];

end;

1. Algorithm 2 (Decryption):

**The purpose of this algorithm:** To retrieve the index of each letter in the message within the key and extract the letter from the alphabet with the same index, then creating the original decrypted message with each letter.

**Algorithm:**

*sValid is the alphabet*

*sKey is the key made from the inputted keyword*

var

M, iIndex : integer;

sInput, sOutput , sValid: String;

for M := 1 to length(sInput) do

begin

iIndex := pos(sInput[m], sKey);

sOutput := sOutput + sValid[iIndex];

end;

1. Algorithm 3 (Creating the key):

**The purpose of this algorithm:** To create a key used to encrypt and decrypt the message with the keyword entered by the user and the alphabet.

**Algorithm:**

*sKey is the key made from the inputted keyword*

*sKeyword is the user inputted keyword*

var

L, K: integer;

sKey, sKeyword: String;

sKey := ‘abcdefghijklmnopqrsstuvwxyz’;

for L := 1 to length(sKey) do

begin

for K := 1 to length(sKeyword) do

begin

if sKey[L] = sKeyword[K] then

begin

iKeyIndex := pos(sKeyword[K], sKey);

Delete(sKey, iKeyIndex, 1);

end;

end;

end;

sKey := sKeyword + sKey;

1. Algorithm 4 (updating database according to log-in details):

**The purpose of this algorithm:** To first check if the user has already used the software (ie. Has a usernumber) and therefore needs a new session logged in the tblEncyrptions table in the databse or if they are new to the program, in which case a new profile in tblUsers should be created for the person.

**Algorithm:**

*sName is the inputed first name of the user into frmLogin*

*sSurname is the inputted surname of the user into frmLogin*

*sInput is the inputted message from the user*

*sUserNumber is the inputted user number if the user has already used the program previously and has a record in tblUsers*

var

I, iEncryptLast, iUserCurrent: Integer;

begin

sName := edtName.Text;

sSurname := edtSurname.Text;

/// //////////////////////database insertion/////////////////////////////

with DataModuleUsers do

begin

tblUsers.sort := 'UserNumber ASC';

tblUsers.last;

if (sUserNumber <> '') AND (strtoint(sUserNumber) <= tblUsers['usernumber']

) then **// *if returning user checks if usernumber is valid and then adds record to encyption table***

begin

tblEncryptions.sort := 'EncryptionNumber ASC';

tblEncryptions.last;

iEncryptLast := tblEncryptions['EncryptionNumber'];

tblEncryptions.Append;

tblEncryptions['Usernumber'] := strtoint(sUserNumber);

tblEncryptions['UseDate'] := today;

tblEncryptions.Post;

ShowMessage('Welcome back! Your'' user number is ' + sUserNumber

+

' remember this for when you use Jamtech(C.) Encryptor in the future!’)

frmLogin.hide;

frmEncrypt.show;

exit;

end

end;

// **for first time customers (if the usernumber is invalid or no usernumber is entered then a new profile will be automatically created for the user)**

with DataModuleUsers do

begin

tblUsers.sort := 'UserNumber ASC';

tblUsers.last;

tblUsers.Append;

tblUsers['Surname'] := sSurname;

tblUsers['Firstname'] := sName;

tblUsers['Gender'] := rgGender.Items[rgGender.ItemIndex][1];

tblUsers.Post;

ShowMessage('Welcome ' + sName + ' ' + sSurname + ' ' +

'enjoy encrypting your messages!');

ShowMessage('Your'''' user number is ' + inttostr(tblUsers['UserNumber']) +

' remember this for when you use Jamtech(C.) Encryptor in the future!');

frmLogin.hide;

frmEncrypt.show;

end;

1. Algorithm 4 5 Removing non-letter characters from the inputted message):

**The purpose of this algorithm:** To remove all characters in the user inputted message that are not letters of the English alphabet.

**Algorithm:**

*sNew is the message without any non-letter characters*

*sInput is the inputted message from the user*

var

C: Integer;

I: Char;

sNew, sInput: String;

for C := 1 to length(sInput) do

begin

for I := 'a' to 'z' do

begin

if sInput[c] = I then

sNew := sNew + I

else if sInput[c] = ' ' then

sNew := sNew + '';

end;

end;

**Data output**

|  |  |  |
| --- | --- | --- |
| **What** | **Format** | **Component** |
| Encrypted message | String | Richedit |
| Decrypted message | String | Richedit |
| Decrypted message with name, gender and date | String, DateTime | Richedit, TextFile |
| Encrypted message with name, gender and date | String | Richedit, TextFile |
| Information about sessions, users and encrypted text | String, integer | dbGrid |