Sterling Panos
ISYS 540-Final Project

Report Generator for a Legacy System

Executive summary

My father is the Director of the Bachelors of Social Work (BSW) Program at the University of Utah (U of U). His department has a web-based data entry application that gathers large amounts of data from clients all over the world. In particular, clients will be asked to complete questionnaires online regarding the mental health symptoms they are experiencing. Although this product meets specific high security demands, it is narrowly focused on the gathering and transmitting of data securely. It does not meet the complete needs of either my father's department, or the host agencies from which the information is gathered. The program fails their needs in the following ways:

- Responses are not immediately compared to population specific normative data
- No appropriate diagnostic calculations are made
- No reports are generated that can be viewed or printed out immediately by the agency clinicians.

My project solves the above problems and adds additional scope that was not originally stated in my proposal. I implemented a program that automatically obtains data from the data-entry program, and then make the appropriate analyses and calculations needed to generate reports. The program allows researchers to customize the calculations for different psychological assessments, make comparisons with different populations, and then generate agency-specific reports. In particular, my program has the following features that fulfill all of the requirements of this project based on my approved proposal:

- takes in specific responses from the data entry program
- Report generator is dynamic and new tests can be added at any time
- calculate scores for different sub-categories of questions based upon the specific mental-health assessments being used
- compare those score against population-specific normative tables and determine if they fall beyond or within specific cut-off points
- generates narrative report with recommended diagnoses and treatment possibilities based upon the test and combination of obtained scores and cut-off points
- Reports are generated in html format for easy viewing and printing.
- Depending on the test, graphs are generated for the report

Added Scope beyond what was required:

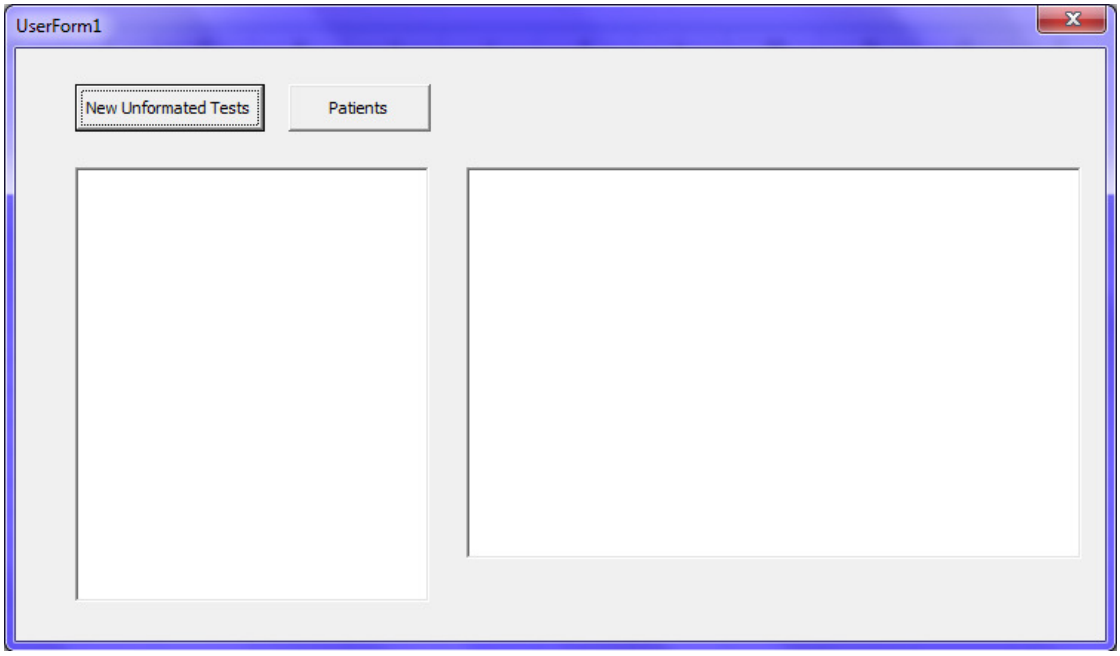
- note creating ability for clinicians to easily add notes to a patient's file
- excel ribbon button to easily start up the program

Implementation documentation

For the following section, I will first provide a screen shot of the element I will be describing and then a description of that element. The textual description of the element will explain why the element

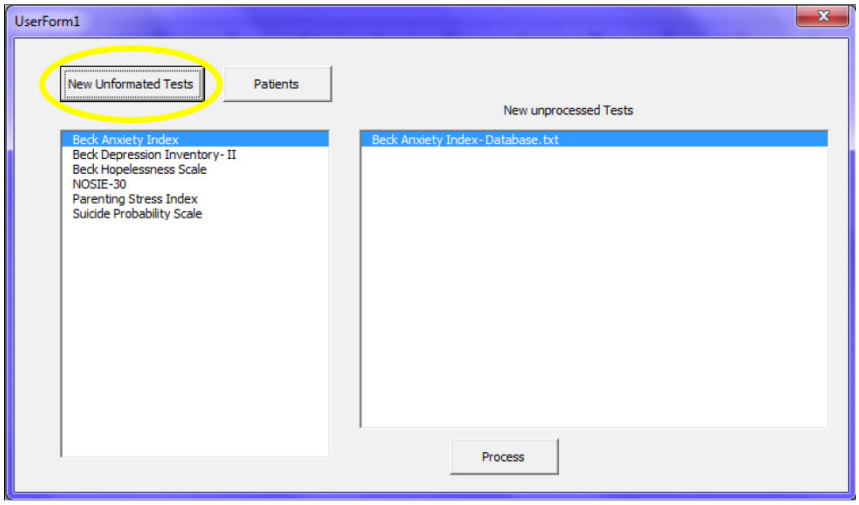
was included in the project, how I created or used the element, and how it is intended to be used. Element 4 is probably the most important and most difficult to program of all the elements. The program should be run in the same folder as the other files that came with it, and I have included dummy data for it to work on. If needed in the folder called “clean”, there is unformatted dummy data.

Element 1: Main Screen



Why the element was included	Creation/Features of element	How it is intended to be used
This is the main screen that the user interacts with, and it allows the user to easily get at all the information and features that he or she is looking for	This screen was created using the form builder in excel. This screen will change depending on what the user has selected and the information displayed is dynamically gathered and displayed.	This main screen is used to find the patient the user is looking for by pressing the “Patients” button (See Element 3), or the user can press the “New Unformatted Tests” button (See Element 2) and view the tests that need to be formatted.

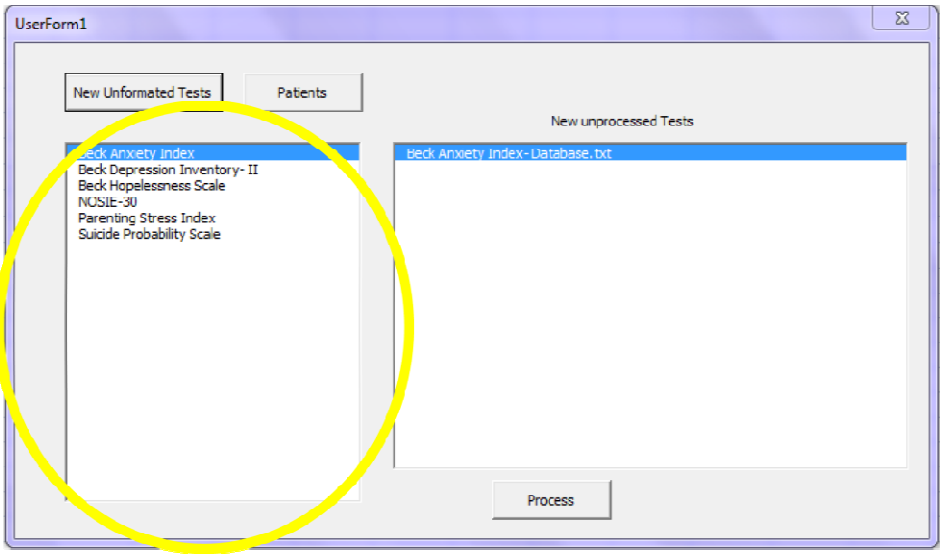
Element 2: “New Unformatted Tests” button



Why the element was included	Creation/Features of element	How it is intended to be used
This element allows the user to switch from displaying patients’ files to displaying of tests that still need to be formatted. This allows the user to see what tests need to be formatted and allows them to choose which ones they want formatted.	When the user clicks the button, a method is called. In this method the following things are done: • The Buttons that should only be used when a patients file is selected are set invisible • The process button is set visible • The list of tests is dynamically generated from a list of folders that are contained in the “EntryPointOutput” Folder • The list selects the first Folder which automatically calls the change event on the “PatientTestListBox“(The List to the Left) (See Element 3)	This Button is meant to be clicked by the user so they can see the various tests and the corresponding files that still need to formatted for that test

This button uses fileSystemObjects to read and display all the different folders using the getsubfolders method. This method returns a list of all the folders, which I used to iterate through and printed off the folder’s name.

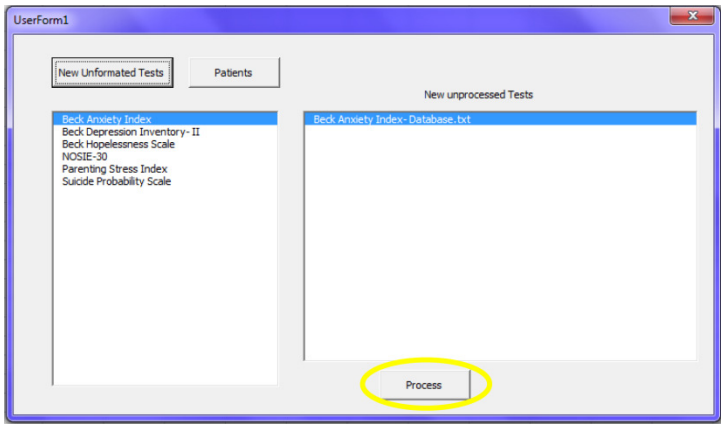
Element 3: “PatientTestListBox”



Why the element was included	Creation/Features of element	How it is intended to be used
This element allows the user to select the specific test files or patient files he or she wants to be displayed. This allows the user to see what files the user can select, so he or she can choose which ones they want formatted or the patients files to be displayed.	When the user clicks on the list, a method is called. In this method the following things are done: • The List to the right is cleared and populated dynamically with the files that still need to be processed or patient files. • The method behind this list, works if either the “patients” or “new unformatted test” button are pressed and displays the appropriate files	The user should select from the list the test he or she wants and then the files for that test or patient will be displayed.

When the list box change event is activated, the method called uses fileSystemObjects to read and display all the different files using the files method. This method returns a list of files in the folder, which I used to iterate through and printed off the file’s name. Next

Element 4: “Process” button



Why the element was included	Creation/Features of element	How it is intended to be used
This element contains the main logic for the program. This button will create the reports for the user, format them correctly, generate the appropriate graphs, and print the reports to html.	When the user clicks the button, a method is called. In this method the following things are done in the following order: 1. The selected file is read in and the file is split in to an array of strings. Because each file contains more than one patient, the file(CSV file) is split to separate the patient's info and responses (using the following code Set ReadFile = fileSystemObj.OpenTextFile(SelectedFile, ForReading) Text = ReadFile.ReadAll ReadFile.Close 'this splits the file between patients ProcessedText = Split(Text, Chr(13) & Chr(10)) 2. The file containing the report formatting for the test is read in and is split up using the split method based on commas 3. Each patient is looped through, using a for loop 4. In the for loop, the patient's personal info is read in 5. If that patient does not already have a folder to put the report into a new folder is created; however, if they already have a folder then the old folder is used 6. A new html file is opened in the patients folder 7. The patients info is printed to the html file 8. A very complicated loop is used to format the test correctly based on the test the patient took and their answers. This loop consist of a do loop to iterate through the patients answers, and multiple for loops to go through and format everything correctly. This is very	This Button is meant to be clicked by the user so the file they have selected will be processed and the patients' files will be updated with the reports that are generated.

	complicated so the next element will be this code. (Element 9) 9. Depending on the test some have graphs that are generated, and these are printed off into the html file(**Caution**: Sometimes Firefox can't display the graphs correctly, so you should use chrome or IE to view the reports) 10. The HTML file is closed 11. The file that needs to be processed is deleted	

Element 5: Complicated Loop used in the process button (see above element)
Because the code was so long, I only included select elements from it to show you its structure. I replaced some code with "//--" and an explanation of what the code did.

```
//-above this is code that loops through the different patients
Do Until y = UBound(ReportFormat)
y = FindNext(x) //--this code finds the next tag so I know how far to loop through the below elements

If ReportFormat(x) = "[Score]" Then
   //--this calculated a score and made a diagnoses based on the patient's score
ElseIf ReportFormat(x) = "[Alt]" Then
   SubScore = 0
   For z = x + 2 To y - 1
   SubScore = SubScore + Val(Answers(Val(ReportFormat(z)) + 6)) //-- this calculates the patient's sub score
   Next z
   //--    this recoreded the data so it could be made into a graph later
ElseIf ReportFormat(x) = "[PERCENTILE]" Then
   For z = x + 1 To y - 1 Step 2
   //--this calculated the subscore's percentile
   Next z
ElseIf ReportFormat(x) = "[Interpretation]" Then
   //--this printed the interpretation for the report
ElseIf ReportFormat(x) = "[TScore]" Then
   //-- this calculated and printed the Tscore for each sub score
ElseIf ReportFormat(x) = "[TScoreTotal]" Then
  //-- this calculated the T score for the total
End If
x = y
Loop
```

Why the element was included	Creation/Features of element	How it is intended to be used
This element does the main logic for formatting the reports and it was made so, new reports with the same tags could be added without the program breaking	This allows the program to loop through different arrays and do calculations on their values. The most complex thing about the loops is that you may have to jump to different parts of the array to do the calculations correctly, and that depending on the tag, do completely different calculations, and each test is completely different. With each "tag", there is a different logical operation.	This is a procedure in the "process" button and is done automatically when the "process" button is clicked.

	For example, the “[Alt]” tag tells the program to add up specific answers, not necessarily in order, and then take that score and print what it means.	

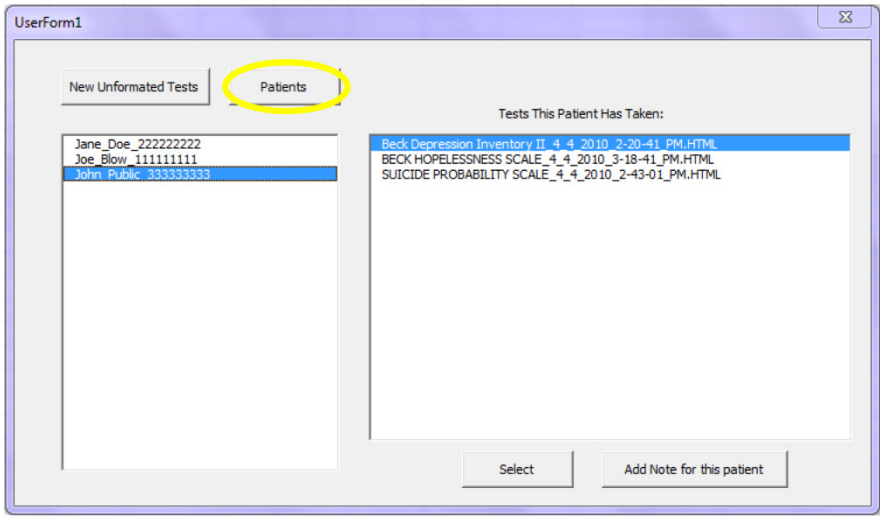
To make this section clearer, I will provide an example using the Parenting Stress Index report format (see TestFormmating>Parenting Stress Index>Questions.txt) as an example.

In this file, there are three different tags, the Alt, Percentile and Interpretation tags. With each tag, a different operation needs to be done. The Alt tag tells the program that the numbers that follow are specific questions that the test should add up and score for that particular sub scoring. For example, the first Alt tag requires the program to add up the patient’s responses for questions 1, 2, 3, 7, 8, 9, and 11 and then report that score as the patient’s Defensive Responding score.

To know how far to loop through the file I created a function called FindNext, which will find where the next tag is and return that value. With that value, I can tell the loop to stop when it reaches that value. While looping through the Alt tag the program will save the sub score on a newly created sheet, so it can make a graph later. Then the program must find its corresponding percentile tag to find the sub score’s percentile. The problem with this tag is that it has a step of two. This means that the for loop needs to iterate through the array two at a time rather than the traditional one. After the program has looped through all the Alt and Percentiles tags, it encounters the Interpretation tag. This tag finds the corresponding diagnoses for its score and prints off the diagnoses with the person’s name being put into the string to make it more personalized.

This example shows how complicated the calculations are for just one report, but there are many different reports that have a drastically different logic structure. For example, for the Suicide Probability Scale report the program must calculate T-scores for the different sub scores, and has its own logic structure.

Element 6: “Patients” button

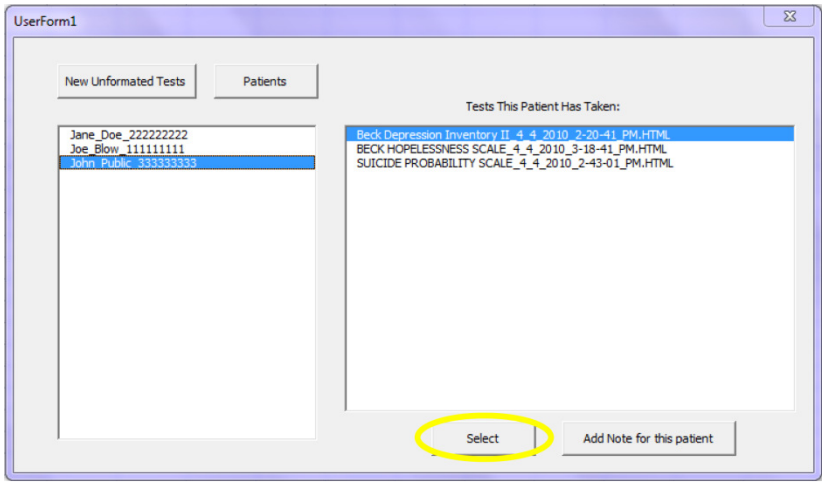


Why the element was included	Creation/Features of element	How it is intended to be used

This element allows the user to switch from displaying patients' files to displaying of tests that still need to be formatted. This allows the user to see what tests need to be formatted and allows them to choose which ones they want formatted.	When the user clicks the button, a method is called. In this method the following things are done: • The Buttons that should only be used with reports that need to be formatted are set invisible • The select button is set visible • The list of patients is dynamically generated from a list of folders that are contained in the "Patients" Folder • The list selects the first Folder which automatically calls the change event on the TestPatientList(The List to the Left) (See Element 3)	This Button is meant to be clicked by the user so they can see the various patients and the corresponding reports that the patient has taken.

This button uses fileSystemObjects to read and display all the different folders using the getsubfolders method. This method returns a list of all the folders, which I used to iterate through and printed off the folder's name

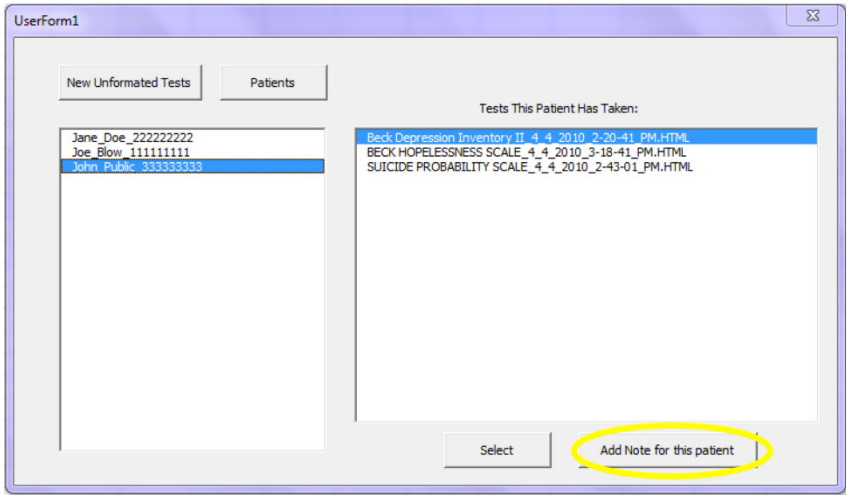
Element 7: "Select" button



Why the element was included	Creation/Features of element	How it is intended to be used
This element allows	When the user clicks the button, a method is called. In	This Button is

the user to display the report that is selected. It reads the file and opens it up in the users default browser.	this method the following things are done: • The file is opened by using the function: Private Declare Function ShellExecute Lib "shell32.dll" _ Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal _ lpOperation As String, ByVal lpFile As String, ByVal _ lpParameters As String, ByVal lpDirectory As String, _ ByVal nShowCmd As Long) As Long	meant to be clicked by the user to view the selected patient's report

Element 8: "Add Note" button



Why the element was included	Creation/Features of element	How it is intended to be used
This element allows the user to display to add a note to the user's folder that is selected. It opens up an instance of notepad for the user to take notes.	When the user clicks the button, a method is called. In this method the following things are done: • The method checks to see if there was a previous note file if there is no note.txt file a new one is created • The method opens up the notes.txt in an instance of notepad using the function in the previous element(7)	This Button is meant to be clicked by the user so they can add a note, or view previous notes.

Element 9: Example of a report

Tests Restults for Test: SUICIDE PROBABILITY SCALE Patient: John Public

PatientID:333333333 BirthDay: 2/2/1960 Gender: M

Test Given On: 4/4/2010 2:43:01 PM Test Administered By: 529902076

Total Score: 91

John Public is a severe risk case

T-Score: 74

Probability Scores:

High presumptive-risk category=80

Intermediate presumptive-risk category=67

Low presumptive-risk category=22

Hopelessness Score: 31

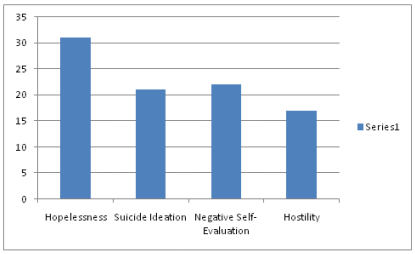
T-Score: 71

Suicide Ideation Score: 21

T-Score: 69

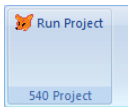
Negative Self-Evaluation Score: 22

T-Score: 73



Why the element was included	Creation/Features of element	How it is intended to be used
This element allows the clinician to quickly view the patients scores and depending on the test and interpretation and graph of the scores	This report is created using element 4 from above. Each report is to be viewed by a clinician for easier diagnostic tools.	This report is to be used by a clinician to evaluate a patient's test responses.

Element 10: Ribbon Button



Why the element was included	Creation/Features of element	How it is intended to be used
This ribbon button opens up the main screen. It was included in the project so it is easier for clinicians to run, and gain access to the program.	This button runs a method that opens up the main program	The user is meant to click on the button so they can gain access to the main program.

The program described above has many features, but many are hard for the user to see, because they are all behind the scenes. Users only has to press a few buttons for them to get the reports they need, but to generate the reports requires a lot of logic that took a while to implement

Discussion of Learning and Conceptual Difficulties Encountered

The project provided me with a couple of difficulties that I solved. First, I had to figure out the logic for the program. This was the hardest thing for me, because there were so many moving parts with this program. The problem is that the file containing the report formatting had many different components and each one had to be handled in a different way. For example, one report may be formatted to display a sub score for multiple sub calculations, and another test may require percentiles or a t score to be calculated from the sub score. Because each report is dramatically different in its formatting, I had to come up with a method to deal with these differences. I came up with a variety of solutions for this problem, but all of them were very complicated and could not handle new tests that may be add in the future. I spent a whole afternoon figuring out my final solution. My solution used multiple loops that would iterate through the patient, the file with the formatting and the patients specific responses. In addition, I created a function that would tell me how far to iterate through the formatting file before the calculations in the file change. This allowed me to create a solution that is robust (i.e. can handle new test) efficient. (**See element 5 above for an example**)

The next problem I faced was open files directly in their default program. I faced many problems with the shell function. I believe these problems are a result of my operating system being Windows 7, because I found many instances of people running both Windows Vista and Windows 7, who had the same problems as I did. To solve this problem, I created a new function with the following code, which worked great:

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long, _
ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory _
As String, ByVal nShowCmd As Long) As Long
```

This code takes in a file path and opens up any file the file path is pointing to in its default browser. With this solution in hand, it was easy for me to add the functionality of open up html files and text files.

This project was a great learning experience for me. I learned how to split csv files, and how to create, delete, and save different files and folders, using the filesystemobject methods. This project has given me a lot of experience and I am glad to have done it, and to have helped my dad with his work.