**COM570: Project**

**Dissertation**

**School of Computing & Information Engineering**

**Jason McKee (B00553970)**

**Data Cleansing System**

**Supervisor: Gerry Parr**

**Second Marker: David McSherry**

**Date: 2nd May 2014**

****

Plagiarism Statement

I declare that this is my own work and that any material I have referred to has been accurately and consistently referenced. I have read the University’s policy on plagiarism and understand the definition of plagiarism as given in the Project Handbook. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Acknowledgements

Thanks to anyone who contributed directly or indirectly to the project, in any way.

Contents

[Abstract 1](#_Toc383980825)

[Introduction 2](#_Toc383980826)

[Problem Statement 2](#_Toc383980827)

[Software Methodology Chosen 3](#_Toc383980828)

[Aim of Project 4](#_Toc383980829)

[Objectives 4](#_Toc383980830)

[The Project 5](#_Toc383980831)

[Report Structure 6](#_Toc383980832)

[Analysis 6](#_Toc383980833)

[Design 20](#_Toc383980834)

[Implementation, Testing and Evaluation 33](#_Toc383980835)

[Conclusions 49](#_Toc383980836)

[References 49](#_Toc383980837)

[Appendices 50](#_Toc383980838)

[A1. Analysis Models 50](#_Toc383980839)

[A2. Design Models 50](#_Toc383980840)

[A3. Code 50](#_Toc383980841)

[A4. Test Suite 50](#_Toc383980842)

[A5. Questionnaire Results 50](#_Toc383980843)

# Abstract

The problem is that the NISRA (Northen Ireland Statictics and Research Agency, 2014) receive several datasets throughout the year which contain thousands of addresses on which critical statistical data outputs rely on such as Census Data. It has been noticed that a percentage of those address cannot be validated as correct address. This is due to the addresses being captured through forms completed by people which lead to human error in the addresses. If an address cannot be confirmed it is not then reliable enough to be used in key statistical data. The current process of confirming these addresses is expensive in both time and money.

It was then proposed to create a less expensive application that would carry out the current process more quickly. The application would make use of the current software Microsoft Access and the programming language Visual Basic with Applications (VBA). This is because current applications that NISRA use are built using these utilities and therefore they are recognised by the staff and also reduces training to a minimum. Another request was the ability to carry out the process through Microsoft SQL Server (Microsoft, 2008) as this is where the data for processing is held for cleansing. This was acceptable as MS Access and MS SQL Server can communicate quite effectively. The finished solution is to carry out the majority of the processes within MS SQL Server as Access allows this via pass through queries which allows for quicker process times and also incorporates the current security procedures within NISRA. The purpose of MS Access is to provide a user friendly front end User Interface that the staff can use effectively.

To date an application was created within MS Access to act as a front end user interface for the address validation process which is carried out in the back end within MS SQL Server. Many improvements have been made over the development of the project as will be outlined in this report but the original idea has been followed with the addition of some useful features. The application works as required and this report will outline some possible future versions of this project.

This current version of this project is being used successfully by staff at NISRA.

# Introduction

## Problem Statement

Northern Ireland Statistics and Research Agency (NISRA) produce official statistics and social research on Northern Ireland every year which inform public policy and associated debate such as the Census. Many of these statistics are widely available to the public which means the results need to be as correct and accurate as possible. NISRA receive several datasets throughout the year from different government organisations which contain thousands of addresses located within the whole of Northern Ireland. The datasets which NISRA receive from other government organisations tend to be formatted in a specific way that is relevant to the organisation sending the dataset. It has been noticed that a percentage of those address cannot be validated as correct address. This is due to the addresses being captured through forms completed by people. These forms are normally read into database via Optical Character Recognition (OCR) software or again by humans. Both of these processes have disadvantages such as a person’s handwriting may not be legible by either methods or it may be just certain characters that are not legible. You will also find that some letters can be mistaken for other letters which generate an incorrect address which is quite hard to detect as that particular address may meet the visual criteria of a correct address. If an address cannot be confirmed as correct, it is not then reliable enough to be used in key statistical data and therefore has to be rejected. The current process of confirming these addresses is expensive in both time and money as the application that is currently used requires extensive training by trained technician. The current application is not specifically tailored to be used for just this purpose therefore there are quite a number of additional features within the application that are not used by the staff. As the software is so expensive only one license is affordable to purchase for NISRA and can only be installed on one computer. Also during certain times of the year this software is needed by more than one user therefore a queue for this software forms. Currently the database used to cleanse the address data for the current software can only be updated every quarter year. This means that any addresses that have been added to new datasets received by NISRA may not be included in the current version until the software is updated and therefore those addresses cannot be verified and used in the collection of statistics. The same can be said for buildings that may have been demolished between updates of the software data. The process of the updated data within the current software is that the company which produces the software collect their updated data and run several tests on this along with their processes to verify that the software continues to work before pushing the update out to the customers. This means that the update that NISRA receive for the software is already out of date when they receive the update. Depending on the growth of housing developments this can be a massive issue which will be explained further in the report.

Therefore the underlying problem for NISRA is that the software they currently use is complex, requires extensive training, expensive and out of date in terms of data.

## Software Methodology Chosen

A methodology to guide the project during development needs to be chosen it has therefore been decided that this methodology will be ‘Agile Time Development’. This methodology has been chosen as the developer wanted a methodology that they would be able to have a timeline to follow, this would make the development of this project run smoother as every step would be outlined and no time would be wasted trying to figure out what part of the project needs to be completed next. A large amount of time would be spent on developing the work plan for the development of the project at the beginning of the project, this means that only small amounts of time would be required to update this plan and this plan is called the ‘Project Backlog’. The project backlog will consist of a list of tasks, and for each of these tasks a difficulty, time frame and estimated completion date will be shown. At the start of each week the developer will develop a sprint which involves choosing specific tasks from the Project Backlog that will be tackled that week. The developer chooses tasks that they will be able to complete within the week and that would require working out the how long the tasks will take. Some of the tasks in the Project Backlog depend on the completion of other tasks so therefore can only be included in a sprint when the relied on task has been completed.

The advantages of using this methodology is that having all tasks planned out allows the project to run very smoothly as no time is wasted on planning, also because of the project backlog and some of the tasks depending on others the developer will have developed a prototype for NISRA to test and receive feedback from NISRA. When all the tasks have been assessed and given a length of time an estimated time of completion can be predicted for the developer.

The developer will need to have daily stand ups every day which they will outline what tasks will be carried out during that day. Also in this process the developer will assess how the previous days tasks went and if they need to be carried over to the next day. This process allows the developer to adjust both the tasks ahead and also the weeks sprint so that the developer knows what will succeed and what will not be completed in the week.

## Aim of Project

The solution to this problem is to create an application that can carry out the same process as the current software which is the ability to take addresses in a dataset and test their validity by running them with defined matching algorithms against an already clean address dataset. This software will save NISRA expenses in both software and training as NISRA have access to a pre-cleansed address dataset. The software must achieve at least the same accuracy percentage as the current software as well as having an easy to use interface so that all relevant members of staff can use the software. The software must be able to run on multiple computers so that if it is needed by more than one person at one time it can be done. The software must be able to be updated as and when a new updated version of the pre-cleansed dataset becomes available.

## Objectives

The main objective is to develop an application that will accept a given dataset that contains addresses that need to be validated. The application will run user chosen algorithms to validate the addresses against an up-to-date pre-cleansed dataset which will append a Unique Property Reference Number to each address that the application is able to validate. This process will need to output a percentage of valid addresses that matches or is greater than that of the previous software. Additional objectives include:

* Making use of the currently used data store Microsoft SQL Server to access the datasets of addresses.
* Having an easy-to-use interface that can be operated with little to no training. This includes the processes of the application being almost fully automated.
* Allowing the user to choose which algorithms to run to validate the addresses.
* Being able to update the pre-cleansed dataset with ease for a member of staff that may not be advanced in IT.

There are also personal objectives to be achieved and these are:

* To further my development in application development, programing languages and to understand the processes involved in creating an application.
* Gaining more experience in problem solving and being able to produce relevant documentation to back up to development of a solution as well as implementing the solution for others benefits.
* Building on current knowledge of management solutions and gaining more experience in professional communication and time management.

## The Project

Research has been carried out with in regards to what products are currently available and that can achieve the objectives listed above. The products have been tested and assessed which can be read further on in the report. A detailed analysis of what NISRA are looking for in terms of requirements has been carried out to make sure the application can meet the needs of NISRA. A large amount of the research was carried out on the development of the user interface as the back end data store was already decided on, therefor extensive research was needed on what software and programing languages operate well with the back end data store.

Throughout the project communications with NISRA were ongoing so that all areas of the project were covered and that any complications that may have arose could be dealt with and a proposed solution could be explained and then implemented.

****The project has been developed and has achieved the entire primary requirements that NISRA sought after at the beginning of the project. Currently the application has achieved validation rates matching that of the previous software. The project has also saved NISRA a substantial amount of money in both training and purchasing of the software.  
NISRA are very pleased to have this software replace their previous software and also are open to future development of this software to add further features.

Figure 6 Agile Time Management Model

## Report Structure

The remainder of this report is structured as followed:

***Analysis***

This part of the report documents the stages in finding a suitable solution to the outlined problem. It will include all research into existing solutions and an understanding to the proposed solution and why the individual components were chosen.

***Design***

In this part of the report the user interface, software architecture, data definitions, algorithms and other high-level descriptions of the system are explained in detail to give an understanding of the system in full.

***Implementation, testing and evaluation***

This part of the report describes the steps taken to implement to proposed solution as well as all the extensive testing and evaluation of the software making sure all relevant requirements are met.

***Conclusions***

In this part of the report a summary of the work carried out during the project can be found along with what problems and challenges were encountered along the way and if any of the initial requirements were amended. At the end of this chapter discussions about further development can be found.

References and appendices can be found at the end of the report and can be used alongside this report to gain a further insight to the problem and the solution. The appendices contain all relevant figures and diagrams mentioned throughout the report and should be viewed when mentioned to grasp the context in which the report explains.

# Analysis

“NISRA is an Agency of the Department of Finance and Personnel. The Chief Executive is Dr Norman Caven. NISRA is the principal source of official statistics and social research on Northern Ireland. These statistics and research inform public policy and associated debate in the wider society.” Their vision is “To be a centre of excellence in all areas of our business, dedicated to continuous improvement in the quality of our work.” And their mission “is to provide a high quality and cost effective registration, statistics and research service.” (Northern Ireland Statistics and Research Agency, 2014)

The developer of this project carried out a year’s work experience within NISRA meaning they have first-hand experience of the current system within NISRA. This includes the current software that NISRA have and the current hardware that NISRA are using. This experience is very beneficial to the developer as not only is there communication between NISRA and the developer, the developer has access to test data that can be used to verify the completed solutions effectiveness. As the developer has knowledge of the current hardware specifications of the current computers, this can be taken into account when researching current solutions to the problem. With the developer having communication with NISRA it can also be found that the technical ability of the staff can be assessed and this can be taken into account when developing the software. Also as the developer can contact the staff, this means that in terms of the user interface the developer can request that a questionnaire be developed and completed by the staff as to what they would find easiest to use in terms of interface.

The problem is that NISRA require an inexpensive, easy-to-use application to validate addresses from datasets which they receive throughout the year containing a substantial amount of addresses from location all over Northern Ireland. Addresses are a major part of generating statistics produced by NISRA as the address is used to specify a location to all persons in the datasets. Many of NISRA’s statistics can be found on their website which the link can be found in the reference section of this report.

The addresses need to be validated to prove that the address linked to a specific person on record is indeed a correct address as said above, location can be a key attribute to many statistics and if the address cannot be validated then it cannot be used as part of any statistics. Most of the data that NISRA receive has been captured from forms completed by the general public who could mistakenly input the data incorrectly resulting in human error. This could be caused by a misspelt street, town or city name or a misheard house number or postcode. When this occurs, an address which is actually located in one part of a town could be placed in the complete opposite of the town and therefore become an invalid address.

During the developer’s time spent at NISRA it was learnt that some of the tables that NISRA receive have been created by administrative staff inputting the data into tables by reading them from forms. This process involves the staff taking a form that has been completed and entering all the information on the forms into the table. The amount of forms that the staff may view and enter the data for each day could reach up to 200 forms each and each form may contain up to 20 fields (Fig. 1) (see Fig. 2 and 3 for correct and incorrect versions respectively of the same address). When this is added to the amount of datasets that NISRA receive the chance of an address being incorrect when entered into the tables is high and this chance will rise if the numbers of forms or fields rise. Therefore that is why the addresses in the database need to be validated.

It has to be noted for many statistics that NISRA produce the address does not have to be 100% correct in terms of the street address, as long as the address can be validated for that specific postal area. The process in which NISRA validate their addresses is by appending a Unique Property Reference Number (UPRN) to the address records. When querying for statistics if a UPRN is found alongside an address it is deemed as a correct address, as a UPRN is uniquely specific to every letterbox, not just every building. This means that flats and apartments are able to be validated as well.

It is then proposed that NISRA require an application that is inexpensive and can accept any given dataset that NISRA receive and validate all of the contained addresses so that the correct addresses are identified and the incorrect addresses are made known. This application must be easy to use by any non-advanced user and also can be used on any of NISRA’s computers at any given time. The benefits of this application is that it will allow more than one NISRA staff to use this application at the one time which will eliminate the waiting time for NISRA staff and the current application. Expenses will be saved through training with the use of this application as currently only a set number of staff can use the current application, but with the development of the proposed solution, the new application will be easy-to-use so that any member of staff will be able to operate it, reducing training costs.

There is an existing standalone application named Experian Quick Address (QAS) Batch (Experian, 2014) which can run on any of the NISRA staff computers but can only be installed on one of the computers due the license only allowing one computer installation. This application can carry out the validation of addresses in the manner that NISRA require as well as using the required pre-cleansed dataset. The cost of this application is £2000 for a 12 month one computer licence; this includes the pre-cleansed POINTER dataset which would be used to validate all of the addresses, as well as other extras which are outlined in Figure 4 in an email received from Experian when requesting a quote for the price of the software that NISRA receive. Although the application is quite expensive on a yearly basis for only one user, the application does have some very interesting features such as boasting the ability to cleanse any given address dataset by updating address fields to the correct street names, counties etc. as well as moving data in the datasets into the correct fields (Fig 5). The application even boasts to even inform the users if a person in the record row has moved or is deceased. Some of these features require additional data which is available for purchase although one of the datasets, named “suppression” has a cost per record to carry out the suppression function which can be seen in the mentioned email. As well as a standalone application QAS Batch is also available as an Application Programming Interface (API) which can be embedded into a current system.

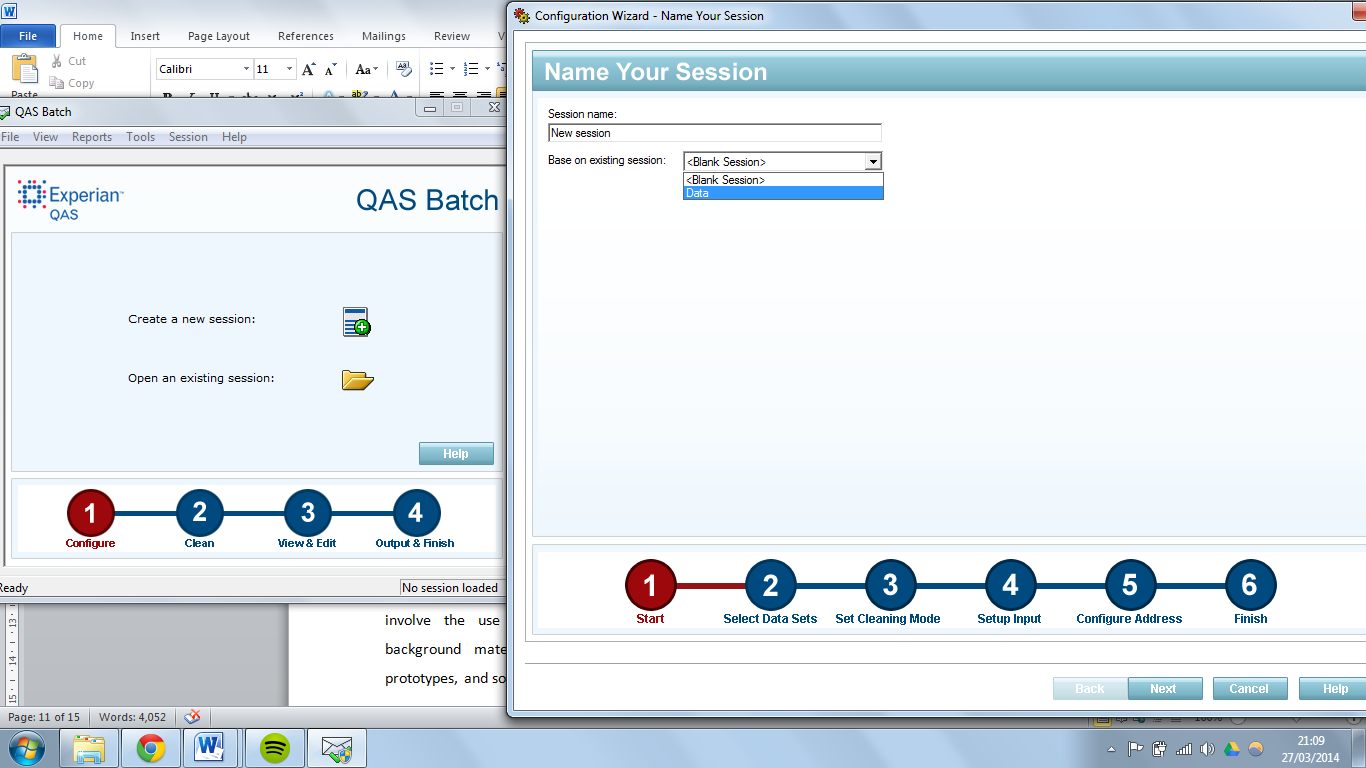
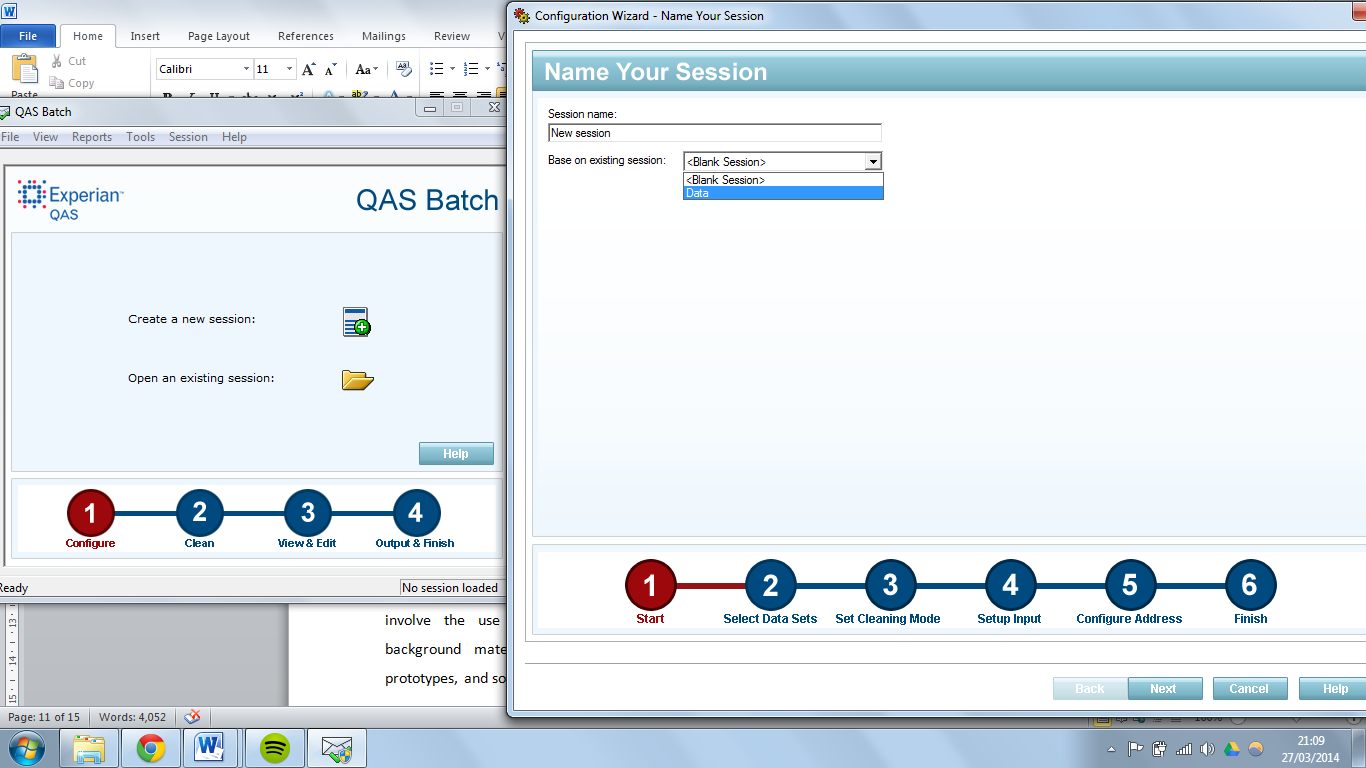
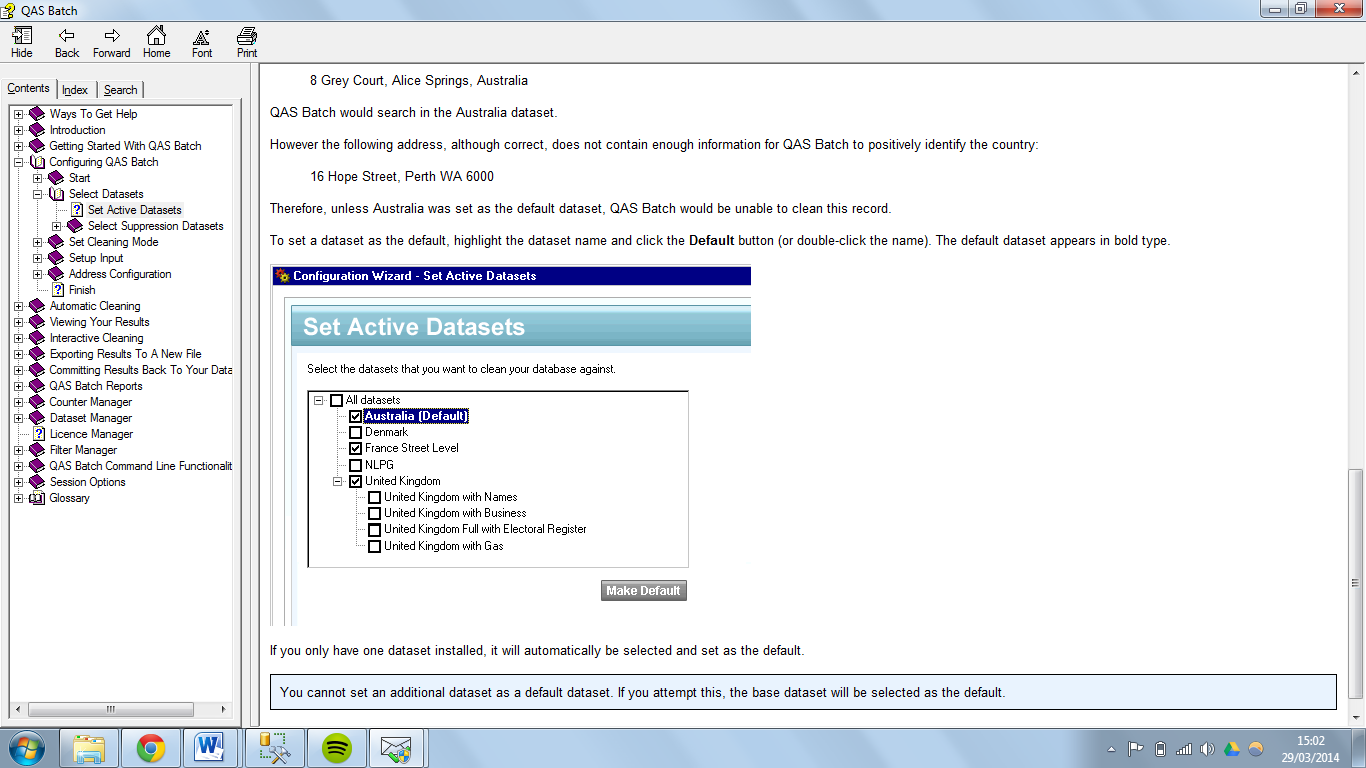
The process at which QAS carries out the validation can be seen below and the significant features that the application includes will be outlined.

Figure 8 when you create a new session.

We can identify some of the features that QAS Batch offers from the above screen shots. In figure 7 you can see that you have two options to choose from, ‘Create a new session’ or ‘Open an existing session’. The feature of opening an existing sessions is a very useful feature if you have different settings for different processes. This feature enables you to quickly load up the correct settings for the correct process saving time configuring each session. You will also notice in figure 8 that when you choose to create a new session you have to option to base the new session on a current session. This is helpful if you have to carry out a new process that needs to be highly configured but it is similar to that of a previous session you can simply select the session you wish to base the new session off. Although these features are useful, they are not useful to NISRA as they only have one process that has to be configured and it is a basic configuration therefore this feature is not needed. In both figures 7 and 8 it can be seen at the bottom of the screenshots that there are numbers to specify what task the user is on and how many tasks are left to be completed before the process is completed. This is a useful feature that any user will find useful as a user can see clearly what tasks have been completed, what the current stage of the process is on and what tasks are left to be completed before the process is complete. This feature is not specific to QAS Batch as many applications implement a feature like this one.

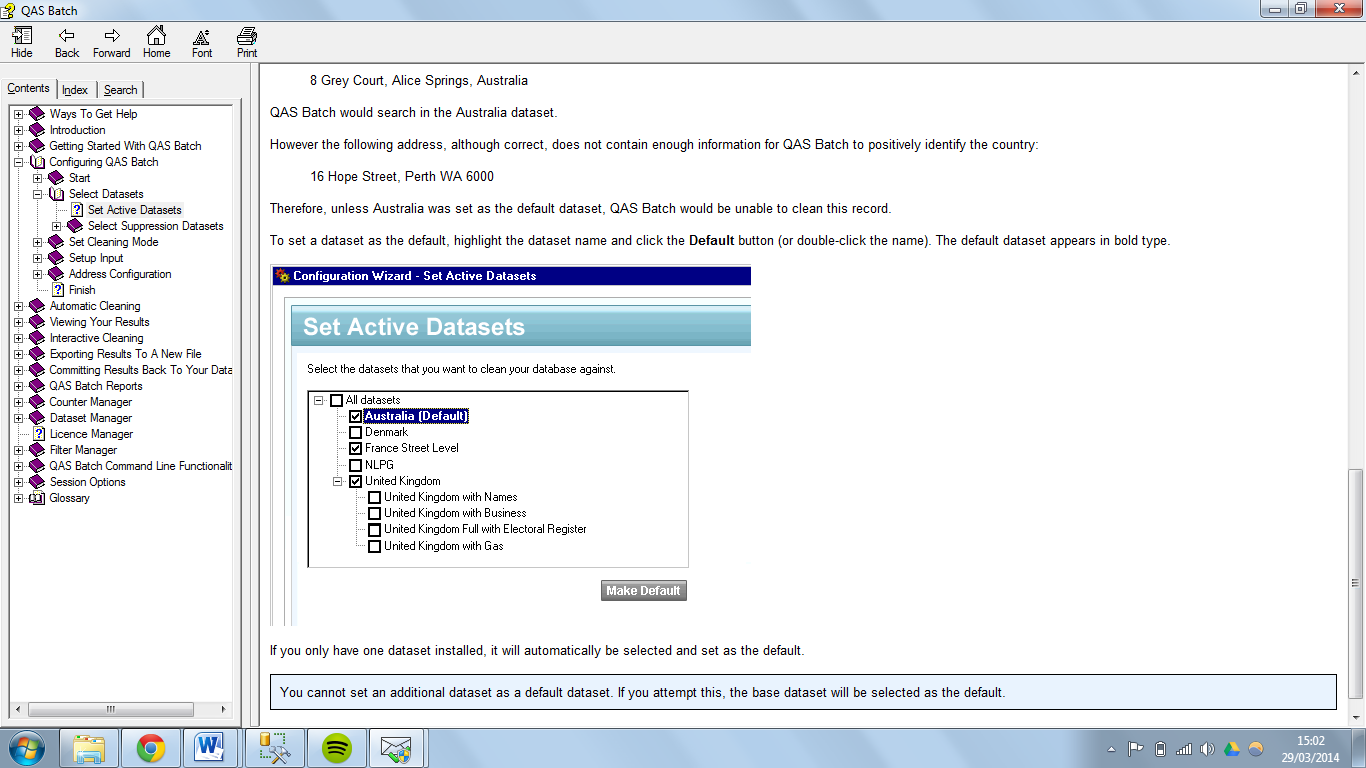
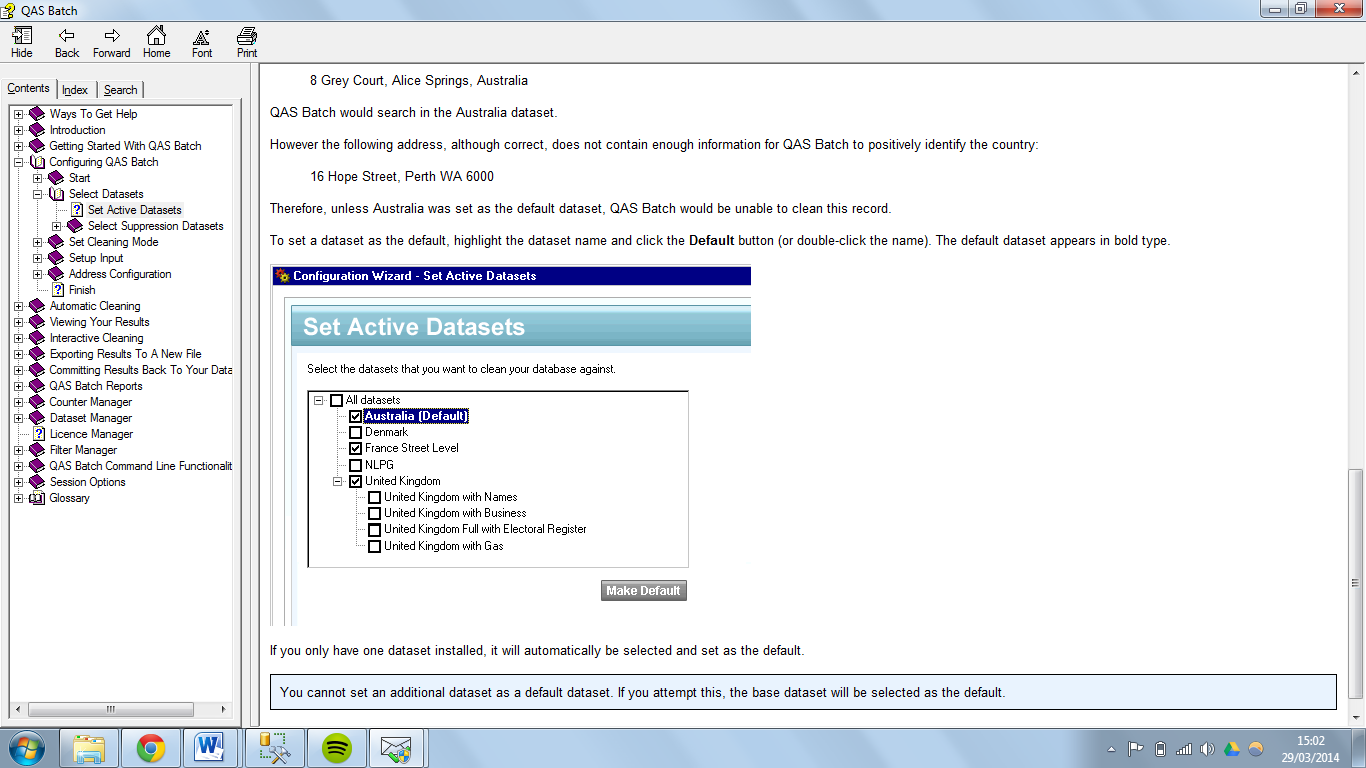
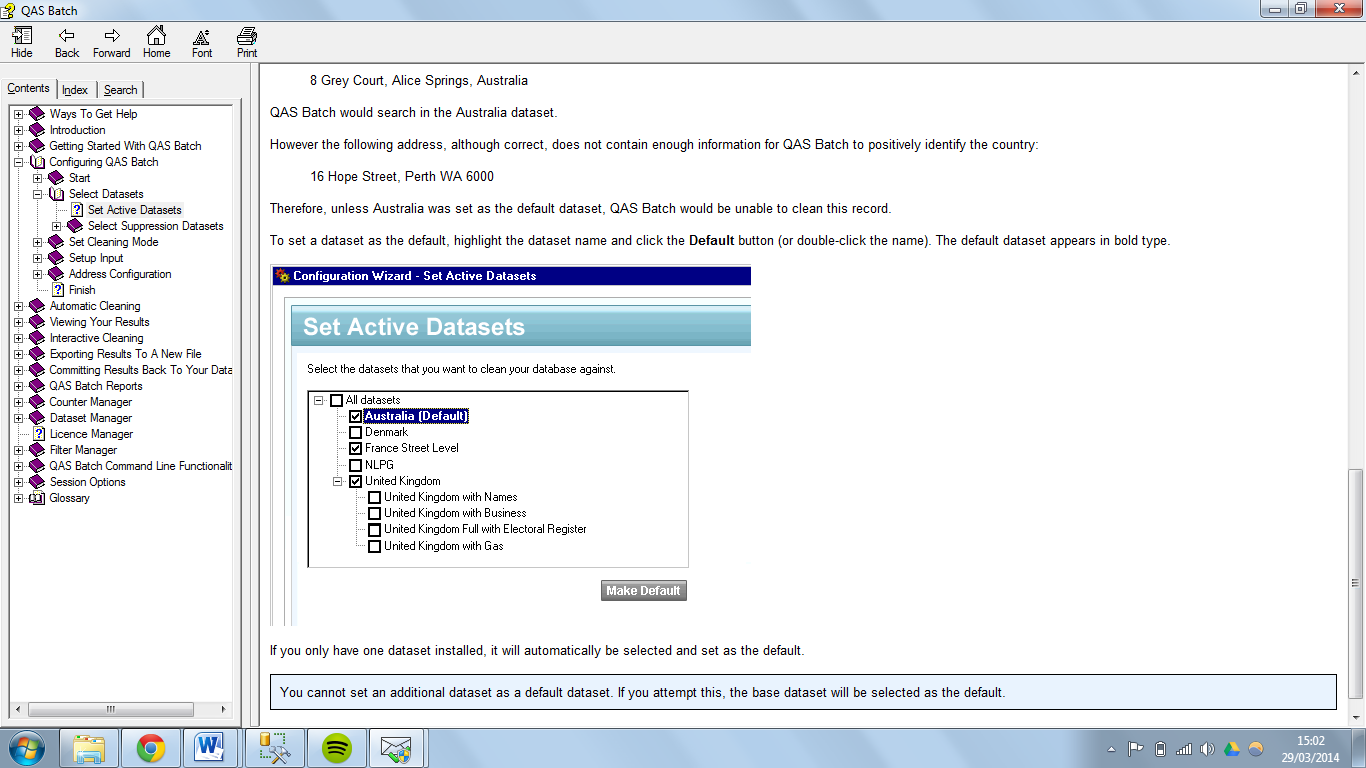
Figure 7 showing the opening screen of QAS Batch

Figure 9 Selecting the active datasets to use to cleanse the user dataset.



Australia

United Kingdom with Pointer (Default)



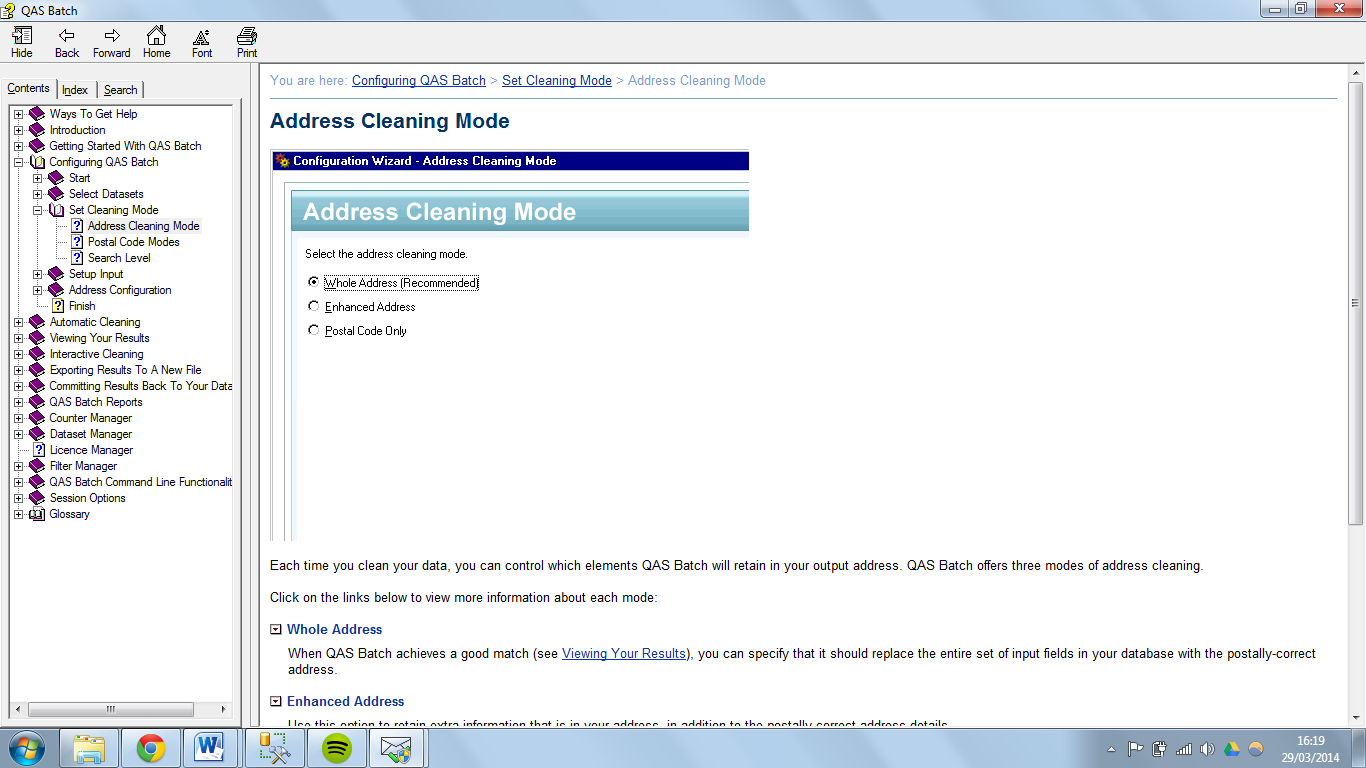
From figure 9 you can see you are given the option to choose what datasets you would like to use to clean the user dataset. Only the datasets that you have loaded onto your computer and that have the valid licenses will be shown. You can choose more than one dataset depending on the data you are cleansing but as NISRA will only be cleansing address data only ‘United Kingdom with Pointer’ needs to be checked. The ability to choose a default dataset is very efficient feature that if a user has several datasets that they use to cleanse their own datasets they can choose the most used dataset and that dataset will become the default dataset to use when carrying out the cleansing process. This feature is not a requirement for NISRA as they only use the one dataset to cleanse their data which is the Pointer dataset.

Figure 10 Choosing the address cleaning mode.

Every time you clean a user dataset, you can control which of the attributes QAS Batch will keep in the output address and which attributes QAS Batch will drop. As you can see from figure 10 there are 3 options;

‘Whole Address’ – When QAS Batch identifies a good match (explained later) you can specify that it should replace all the input fields in the user dataset with the postally-correct address in the chosen QAS Batch dataset.

‘Enhanced Address’ – This option retains extra information that is in the address from the user dataset, as well as the postally-correct address details. E.g. if a house was named “Home Farm” by the owner (and this information may be present in the user address data), the name of the house may not be part of the postally-correct address and would be lost if you selected the ‘Whole Address’ option. It has to be noted that enhanced address matching will not always retain all additional information. For example, when different partial matches are found for a certain address then any attributes that are not shared across the matches will be lost, even under ‘Enhanced Address’ matching.

‘Postal Code Only’ – with using this option you can specify QAS Batch to only update any incorrect postcodes in the user datasets.

For NISRA the only option that is ever used is the ‘Whole Address’ option. Although the ‘Enhanced Address’ option saves any extra information about the address, NISRA only require the postal address as this is what will be used to match against the Pointer dataset.

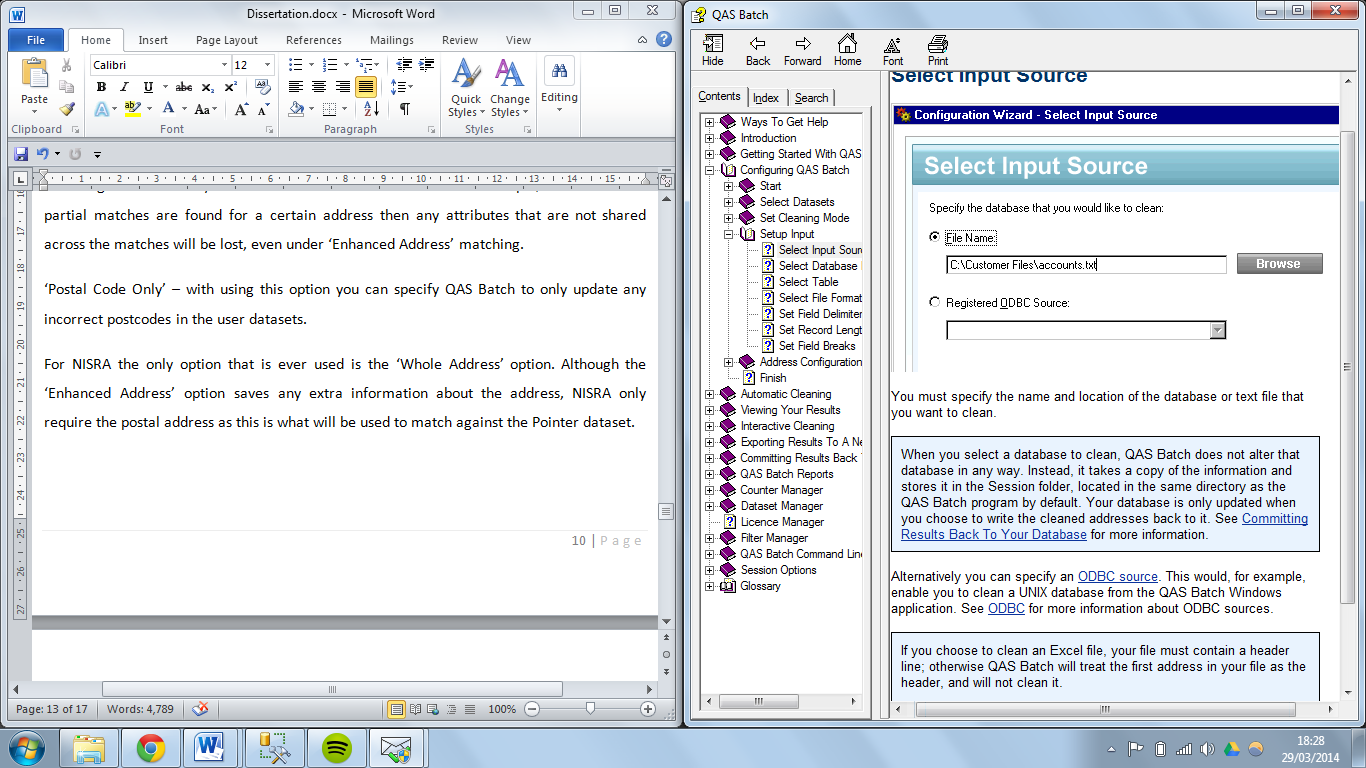
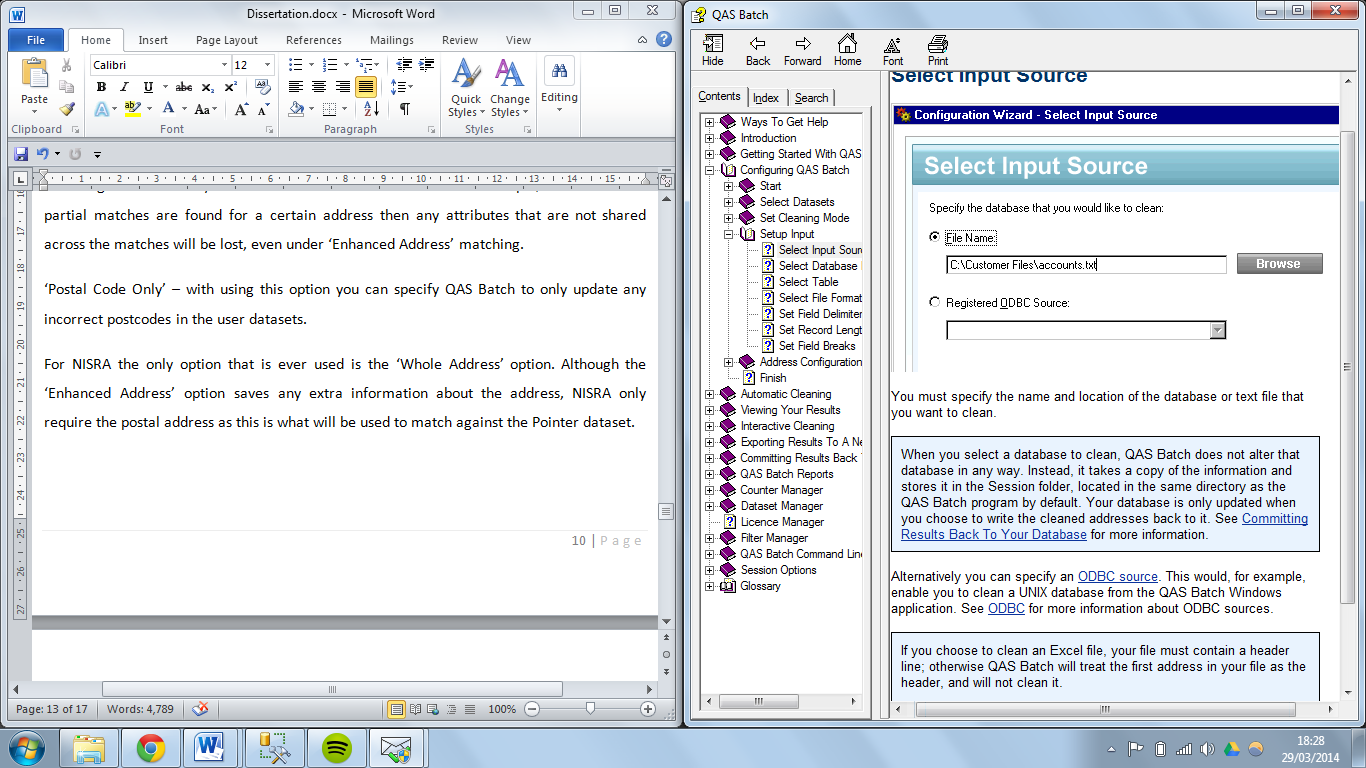


Figure 11 Specifying the location of the database to be cleaned.

When choosing an input source you have two choices as shown in figure 11, you can locate the file if it is stored locally on your computer by specifying the name and location of the file, later in the wizard you have the option of choosing what format the file is in. Alternatively you can choose the option to use an ODBC (Open DataBase Connectivity) source which enables you to link a table within the connected database to be cleaned. It has to be noted that the database you select to be cleaned, whether it is a file or within an ODBC connection, QAS Batch does not alter that database in any way. Instead it takes a copy of the database and stores it in a session folder. Your database is only updated when you choose to write the cleaned addresses back to it.

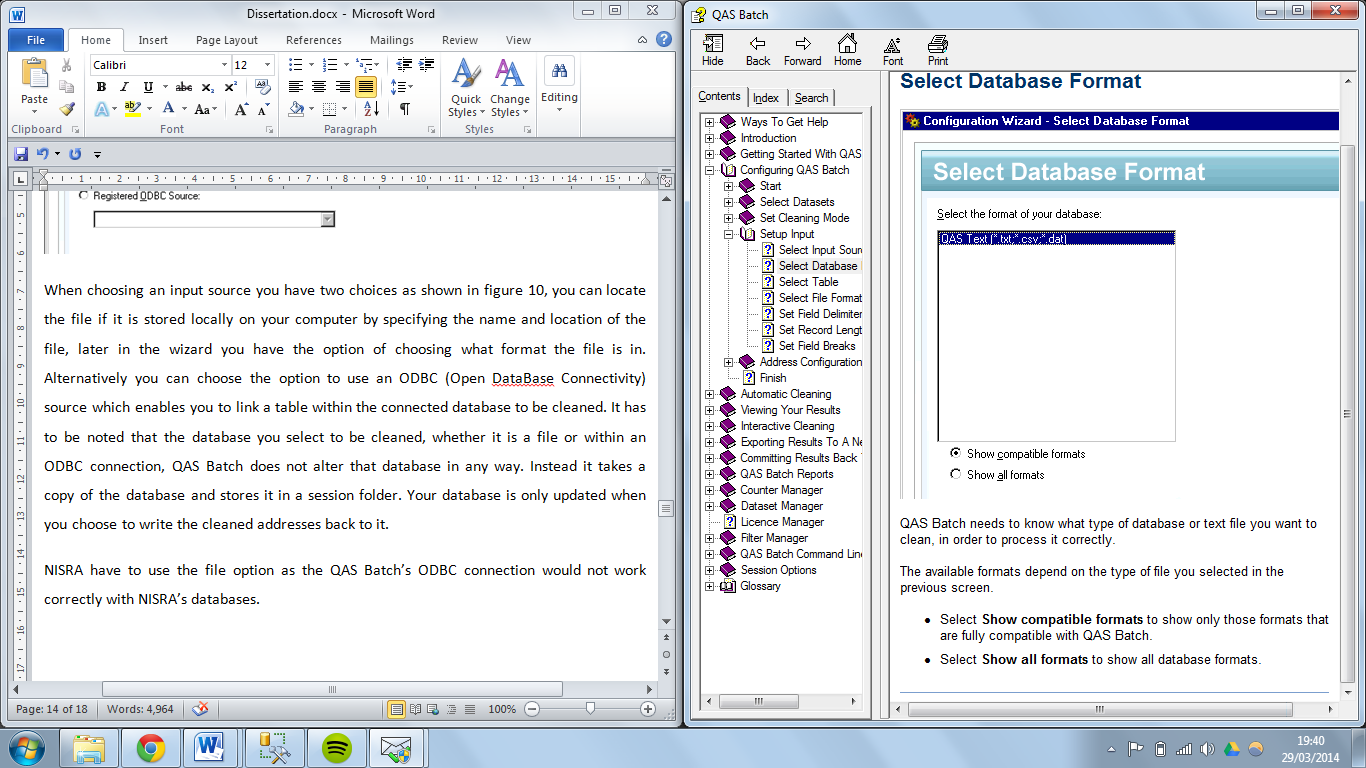
NISRA have to use the file option as the QAS Batch’s ODBC connection would not work correctly with NISRA’s databases.

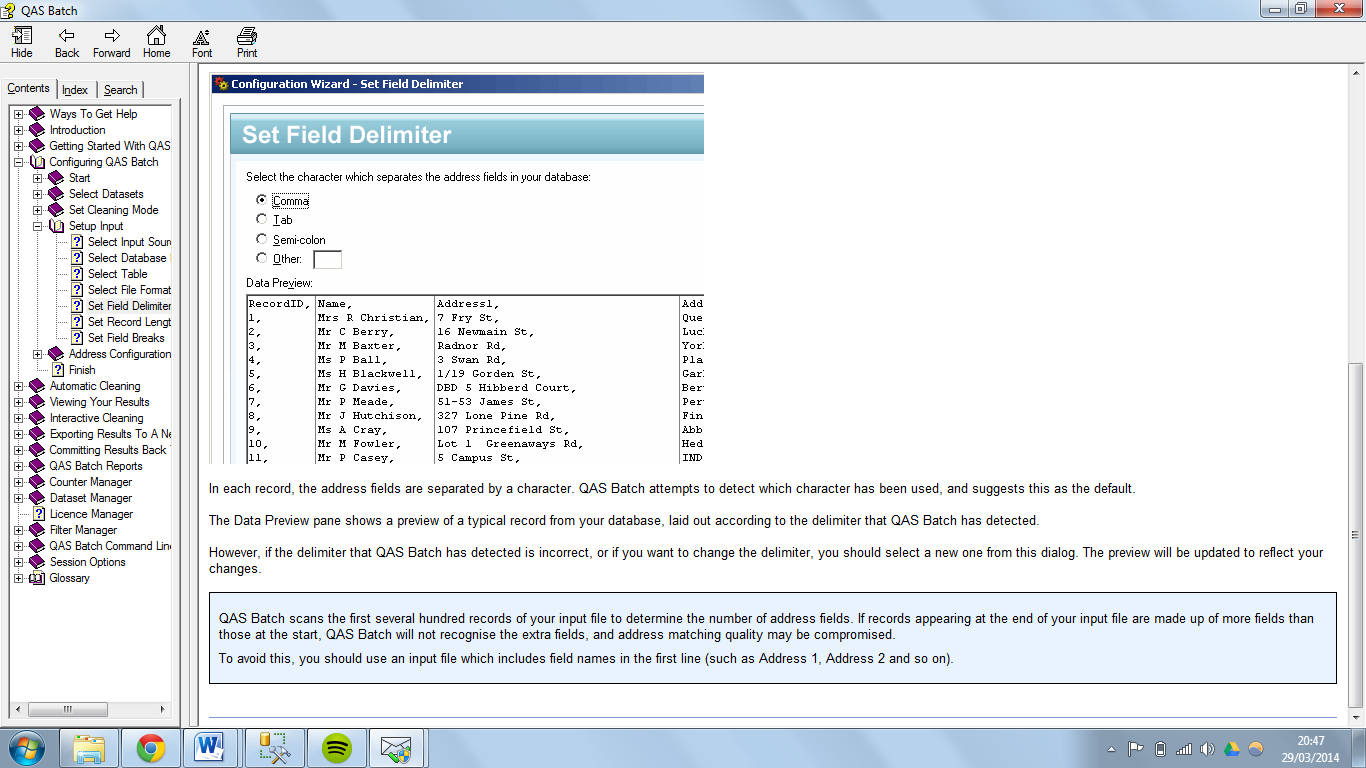
Figure 12 Selecting the format of the database file.

As mentioned above you have to choose the format of the file that you have selected (figure 12). The options that are displayed are relevant to the file that you upload. QAS will try to determine the file type before this screen is shown.

If the database file that you chose contains more than one table you will have to specify which table that you want to cleanse. QAS Batch will display a preview of the tables attributes and a preview of the content in each attribute in the database to allow for easy identification of the table you wish to select.

Figure 13 configuring the format of the text file.

(Note that the data shown is dummy data and not of any connection with NISRA)

When presented with the screen shown in figure 13 you are able to choose the format of the table from your file you selected is in. The options are ‘Delimited’ which means the file has fields that are separate by a specific character which, if chosen, will be declared later in the process. The other option is ‘Fixed width’, this option means the file has fields that are the same throughout the file, and in other words the information is aligned in columns. Notice that the check box allows you to specify if the first row of the file contains field names. Also the data preview box allows you to see how the above choices will affect your data.

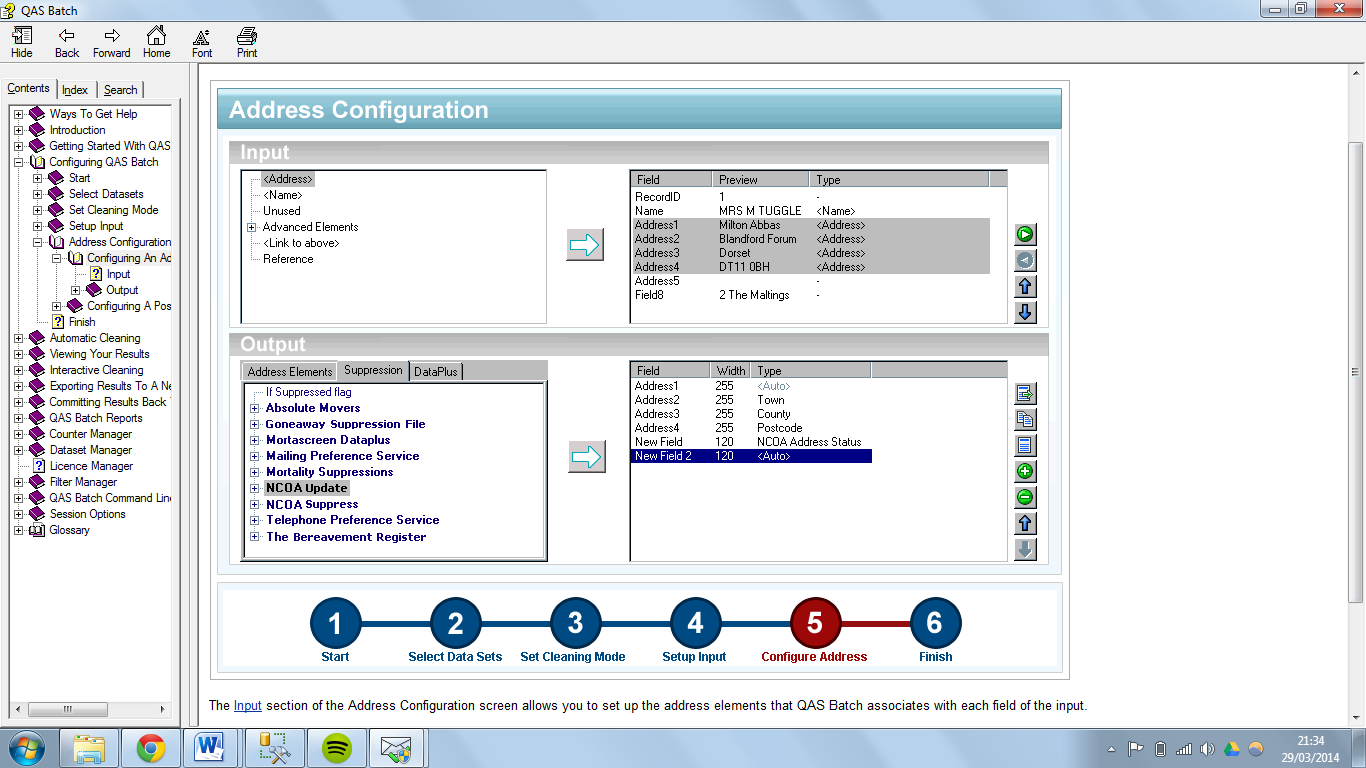
This step of the configuration, shown in figure 14, allows you to specify the character that will separate the fields in the table of the database. As you can see there are a few default options that would be most used and the ability to define your own character.

Figure 15 configuring the input and output address elements.

(Note not actual data. Only for explanation purposes.)

Figure 14 selecting the character that separates the fields in the database.

This part of the process, as shown in figure 15, is a very important part of the process as the more accurate you are with configuring both the input and output elements the more accurate the cleansing process will be. On the left hand side of the input configuration you have a list of all the elements available to select such as the basics of ‘<Name>’ and ‘<Address>’. There are also some more specific selections such as ‘Unused’ which means QAS Batch will ignore this element as it contains irrelevant data, ‘Reference’ is used when elements contain data that you do not want QAS Batch to search on but wish this data to be visible both during and after the cleaning process. Within the ‘Advanced Elements’ node in the tree you will find specific address types such as a Street Line or PO Box, these elements will make cleansing more accurate when matching.

In the output section of figure 15 you will see the same layout, on the left hand side you will find a list of all the elements that can be selected for output. Normally you will select the elements that were configured in the input section but you can add more elements that were not in the input section. E.g. if in the input section you configured the elements ‘<Address>, <Street> and <Postcode>’ but for statistics you need the county of address you could add the ‘<County>’ element. This would add this information into the output of the address even though you did not have this in the input data. It does this by using the information in the chosen cleaning datasets.

This feature is a very smart feature if used correctly and when used correctly can save the user a substantial amount of cleaning time. Also the way in which the input elements are configure impacts how well the data will be cleaned, so the better the configuration the better the cleaning. The use of this feature requires extensive knowledge of the data that you are cleaning and the attributes that the data includes. The way in which the data to be input is presented has a massive factor in the configuration as if an address in one row has county in address line 3 whereas other rows may have county in address field 4. This can cause difficulties when configuring the inputs as you may not be able to select county for both inputs. As NISRA receive several datasets which are all formatted differently this process does take some time to configure each time it has to be carried out.

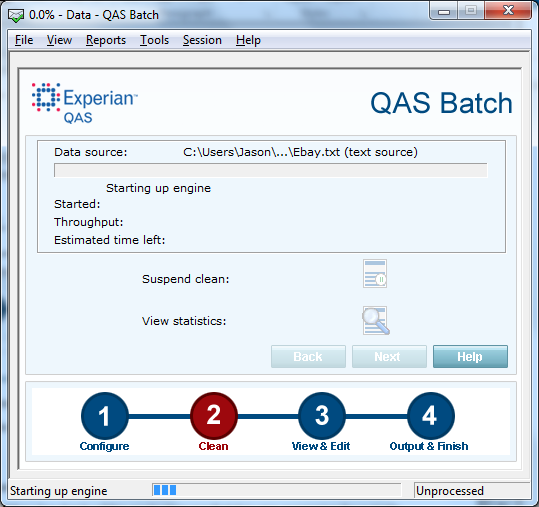
As you can see from figure 16 the progress screen is very basic but has all the essential information required such as the time started and the time that is remaining. The progress bar enables the user to see how much of the cleaning process is remaining and you will also notice the percentage of the work completed at the top of the window. All of these features are essential for a process such as this as it does take some time to complete therefore when the user has an estimate of how long is remaining they can rest at ease and plan around this.

Figure 16 progress screen after the configuration is complete the cleaning has started.

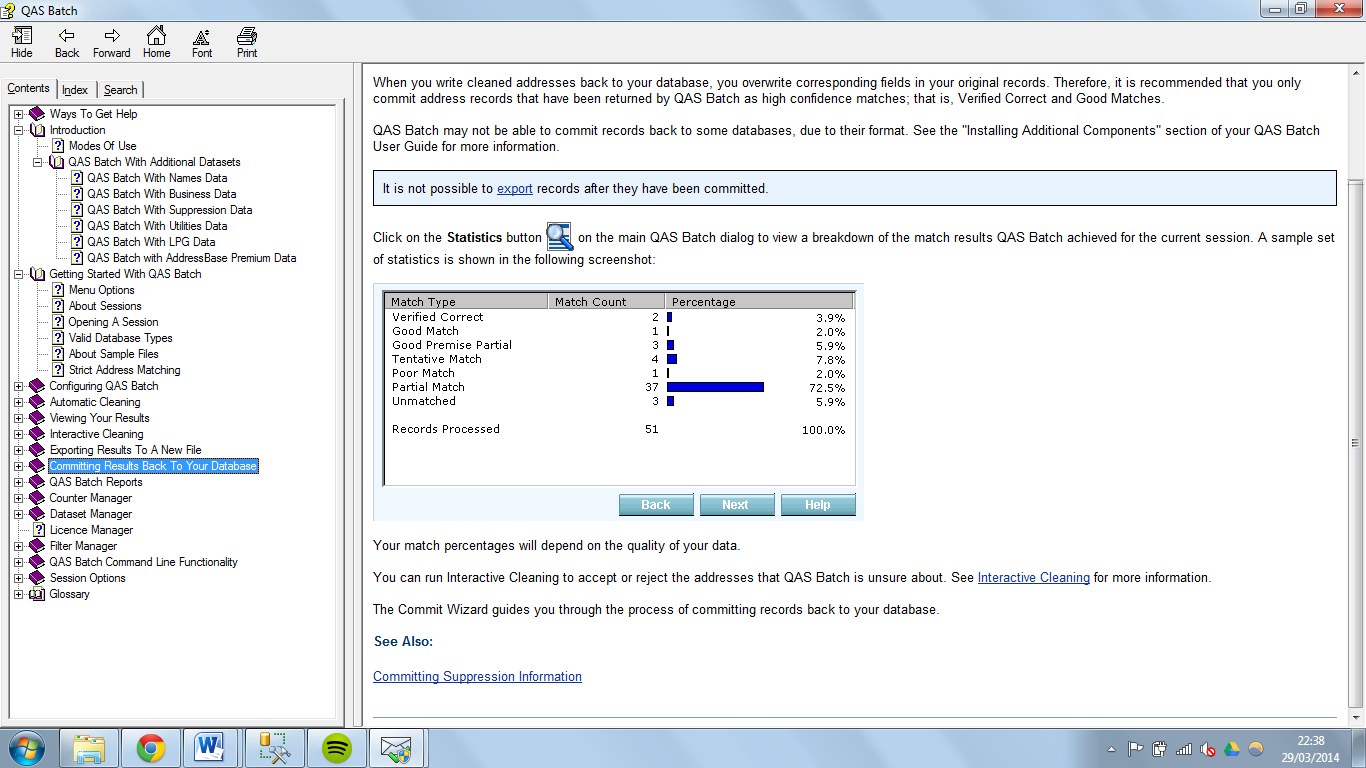


Figure 17 viewing the statistics of the cleaning process

This screen shown in figure 17 is the statistics screen for the cleaning process, and this can be seen both during the process and after the process. The screen shown is after the matching process and you can see the detail in which the cleaning process achieved. These results reassure the user of the data that they may be re-importing into their database that it has in fact been cleaned and how much of the data has been cleaned. You will notice that only 3.9% of the records were verified as correct, this means that all of those records were correct before the cleaning process started and that they did not have to be changed during the process. All of the other records were either changes or no match was found. This feature is essential for the user to view as they can be assured of what process has been carried out on their data. Remember that the data that was cleaned is a copy of the original data meaning that if the user is not satisfied they can go back and change some of the elements. The output statistics may change depending on what elements are selected during the input configuration as this will have an impact on cleaning.

As NISRA’s datasets which they receive vary in format and layout this process can take some time to configure for each dataset which is very time consuming for staff. Configuring the input elements can cause problems with the output statistic as the data can keep continually changes every set they receive.

When the user is satisfied with the output statistics they can continue to either store the cleaned data as a text file to be imported to the users database at a later point in time or the user can directly output the cleaned data via ODBC directly to the table that it originated from. This process is the reversed process of selecting the input file.

All in all this application does carry out the process that NISRA needs in regards to appending the UPRN to the datasets and can also carry out data cleaning if NISRA were to require this at a later time. Unfortunately as mentioned above, the pricing in regards to the purchasing of the application and the licencing of multiple users as well as the training of the staff to use the application is extensive. As you can see from the walkthrough of the process there are a lot of features in the application and training would definitely be required to some if not all of the staff so that they could efficiently continue to use this software. It has to be noted that currently there is only one person in NISRA that can use the current version of the software but they have not been trained in doing so therefore the efficiency of this is affected. Also if the version of the software changes more training may be required which again will cost money.

**Feedback from Questionnaires**

NISRA staff was presented with questionnaires (found in the appendices section of this report) in regards to the user interface of the new application and the idea of this questionnaire was to outline what the staff would find simplest to navigate. It was found that the majority of the staff wanted a simplistic design that allowed them to press a few buttons and the process would be complete. When asked about current user interfaces that they might use within work or even at home it was clear that the staff wanted a well instructed interface so that if new staff were to be employed they would be able to easily operate the application due to the on screen instructions. A number of people made reference to the easiness of an application installation and how this was a well instructed process. When asked to list applications with user interface that they preferred the most common answer among the employees was Microsoft Access. This would be because many of NISRA’s applications involve the use of this application as a front end to NISRA’s databases. Microsoft Access will be considered as a user interface in the design section of this report.

**The Solution**

The developer plans to design a “Data Cleansing” application that will allow the user to select a ‘dirty’ database with addresses to be cleaned. This application will use the existing address fields in the dataset to match those addresses to the addresses in a pre-cleaned dataset, when a match is found; the UPRN of the matched address from the pre-cleaned dataset will be appended to the ‘dirty’ matching address.

The matching process for the existing application is not available to view for licensing reasons so therefore it is proposed to create a new matching system which will require the use of what the developer is calling ‘MatchKeys’. These matchkeys will be created using an ‘Unique Postcode Identifier’ (UPCID) which will be made up of different parts of the address (this will be discussed and explained in the design section of the report). These UPCID’s will be created for all records in both the user dataset and the pre-cleansed dataset and it will be on these fields that the matching process will be carried out.

The user interface of the application will be simplistic and easy to use for the staff at NISRA. The user interface must be developed in a way that if NISRA employ new staff they are able to operate the application without training.

The application will be based off the current application that NISRA are using and where possible features will be ported.

The data handling application that will be used will be Microsoft SQL Server as this is where the current processing of the addresses is carried out and many of the front end software developments are able to communicate with this. As NISRA contain millions of records it would not be feasible, both for time and money to change the data store.

**Requirements**

Through researching the current problem and through feedback from questionnaire as well as time spent at NISRA the developer and NISRA came up with a list of functional and non-functional requirements. Diagrams were used to help understand the requirements.

* A user should be prompted to log into the SQL server database with their SQL user name and password when the application opens. If the user does not enter both of these or if they are incorrect the application should prompt the user again.

User is prompted to enter SQL username and password

SQL server: Validate Username and password

Incorrect

Correct

Application opens

`

* A user should be able to select a table name in the SQL database from an automatically populated list.
* A user should be able to click a minimum of one button which will carry out the cleansing process with minimum user intervention
* Throughout the entire cleansing process the user should be informed if a process has already been completed on a chosen table.
* A user should be able to view a report for the matching process, outlining how many UPRNs were added to the table and what percentage of the records that covers.
* A user should be shown a form after the match process is complete displaying all the records from the table used in the match process that were assigned a UPRN along with the records in the pre-cleaned table that have the same UPRN.

User selects table from dropdown list

User clicks button to carry out match process

SQL Server: matches UPCIDs and adds UPRNs

Error message displayed

Failed

Success message displayed and comparison form opens

Completed

* The application should have an easy to use graphical user interface with clear navigation options that will allow a user of any level to use.
* A user should be able to use this application without any prior knowledge to SQL server.
* If more than one button is required for a certain process then these buttons should be found together as to not confuse the user.

# Design

**Software Chosen**

In this section of the report the software that was chosen will be explained.

*Backend Database*

Microsoft SQL Server will be used to handle both the storing of data and also most of the processing of the data. The reason this software was chosen was because NISRA already are using this software for many of their other data processing tasks, this means that no extra expense will occur for this part of the project. The developer found that MS SQL Server can communicate with an extensive list of front end software applications which made the choice of this application easier as it does not limit the software that can be chosen for the front end system. The developer has previous knowledge of this application and therefore will know how to optimise functionality within the project, optimising this project will be essential as it will be very process heavy due to the amount of records being processed.

Other database applications were considered for this project such as MySQL (Oracle, 2014) but were not decided on in the end. This application is another brilliant database system that is very popular and one of the most used database applications available but the reason the developer did not use this application was it would not be efficient for the users of the application. NISRA receive datasets in text file format which could be simply read into this application, the data would have to be exported to a text file when the cleaning was complete and then it would have to be imported into MS SQL Server as that is where all NISRA’s data and statistic processing is carried out. Therefore it was proposed that due to efficiency MySQL would not be used due to this and not due to overall rating. When NISRA receive the datasets they can simply import the data into MS SQL Server and all the processing, both for data cleansing and for statistics can be carried out here. Another advantage of MS SQL Server is that NISRA perform backups of all data stored in MS SQL Server so any cleansed data will get backup, therefore if MySQL was chosen, a separate backup system would have to be created to be compliant with NISRA’s data act.

*Frontend Application*

When choosing the application for the frontend many options were considered such as; Java, C++, Microsoft Access, C# and also the possibility of a frontend web application using PHP and XML. All of the above mentioned were researched fully with consideration to the development options available in regards to a user interface as well as the integration of MS SQL Server. All of these methods had the ability to connect and communicate with MS SQL Server either via API or via ODBC, although when researched many of the methods had several disadvantages. Some of these disadvantages included the lack of skill and knowledge that the developer had in regards to the mentioned languages. The amount of time and effort that the developer would have to spend learning the language would not be feasible if there is already an application that the developer can use that they already know how to operate. For developing in the languages mentioned above the developer would have been able to create a user interface using the languages but a substantial amount of time would have been wasted in researching and developing both the MS SQL Server connection as well as the processes to be carried out.

It was then proposed to develop the front end application within MS Access as not only does the developer have great experience in using this software but certain NISRA staff also have knowledge of this software. This is a great advantage these members of staff will be able to troubleshoot problems should problems arise or also be able to identify what the problem is and pass that information on to the developer to be fixed. This application was also chosen as not only can you connect to a back end database to carry out processes, you can also carry out processes within this application itself, this will be useful when outputting information back to the user in terms of estimated time of completion etc. By using this application you can further develop the front end of the project without having to compile code meaning that small changes to the UI can be made without large down times. The ability to edit these features can be set so that it is only available to certain users with certain rights on the system. This also leads to advanced security within this application, when using MS Access the developer can create the application so that users have to log in with their correct information before any user can start any of the processes, this means that any user that should not have access to the system cannot log in and view data they are not cleared to view.

*Software Version Control*

From the developers previous application development experience it was decided that version control should be implemented. As NISRA take backups of their data system frequently the backend database does not need to be backed up and therefore the only version control that will be needed will be for the frontend application. Extensive research has been carried out to find the best software version control for the frontend application. As it has been decided that MS Access will be used to develop the frontend user interface it has been discovered that there are no official applications that suit the need of version control with MS Access. Many of the suitable applications found cost money to implement and do not provide trials as they are third party applications therefore they could not be tested to see if they work with MS Access. The developer was able to acquire open source code that is available to freely use which requires the user to run a single line of code (ExportAllSource) from inside the Visual Basic with Applications (VBA) window, and that code will export all the code from inside the MS Access application, which includes Form Code, Macros and queries as well as tables, to a specified location on the current computer. The developer then implemented the use of the software Git to upload and store any increments of the software on a web server. Git allows users, with the installed framework, to run command line commands to upload files from a specified folder to a web server with 3 commands, ‘git Add -A’, ‘git commit’ and ‘git push’.

* ‘git Add -A’ – there are several instances of ‘git Add’ that can be used depending on what files you wish to upload. I.e. if you have a project that is large in size and only want to upload several files that you have changed then you could specify this. As any time this project will be modified all the code will have changed this is the code needed to upload the new software.
* ‘git commit’ – this command simply prepares the files that have been added for the upload to the server. From this command you have to write a heading for the upload, this is to identify one upload from another. You also have the choice to add more text to describe the files you are uploading if they need more explanation.
* ‘git push’ – this command will start uploading the files to the git depository on the web server and notify the uploader when this is complete.

It has to be noted that each upload does not overwrite previous uploads, rather new uploads are stored alongside previous uploads. This is very useful in case a modification is made and uploaded without discovering that there has been a problem, the user can then simply roll back a commit so that a previous upload is now the main version.

This process will be very useful for the future development of this project as the developer has access to all the code of the project from anywhere they wish and can be downloaded onto any computer, it will also allow the developer to upload new versions of the software so as NISRA can download them and simply import the new code.

It has been proposed to upload code of any changes made after that change has been made but one a two week basis an upload of the whole project will be made and this will be a major version release.

**Pre-cleaned dataset**

It has already been decided that the pre-cleansed dataset to be used will be the Pointer dataset from LPS (Land & Property Services, n.d.).

‘Pointer is the address database for Northern Ireland and is maintained by Land & Property Services (LPS), with input from Local Councils and Royal Mail (RM). This is now the common standard address for every property in Northern Ireland. It is important to note that Pointer is a dataset for addressable buildings in Northern Ireland. Each building has a UNIQUE\_BUILDING\_ID, which uniquely identifies a Primary Addressable Object (PAO). A PAO is defined as the ‘physical footprint’, i.e. the building shell. Each property has a Unique Property Reference Number (UPRN). The UPRN represents the Secondary Addressable Object (SAO) e.g. a residence or business within a building.

Pointer has been allocated a set of UPRNs from the national hub, which are allocated to all addresses within the dataset. This will ensure consistency of UPRNs across Northern Ireland and Great Britain.’ (Ordnance Survey of Northern Ireland, n.d.)

The reasons for choosing this dataset are;

* It is the leading address dataset for Northern Ireland,
* It has full Northern Ireland coverage,
* It is maintained by Local Authority staff,
* It is the only NI spatial address database,
* It contains information on multiple occupancy, building use and organisation type, and
* It has full incorporation of Townland names

This dataset, as mentioned above is sold with QAS Batch as an additional dataset but you can also buy this separately, although NISRA are contracted to receive this dataset already therefore it is efficient for the developer to use this dataset. It is updated frequently and NISRA can acquire a copy of this dataset on a regular basis without having to wait for this to be released.

**Matching methodology**

This part of the project will take the longest to develop as it is the main body of all processes in the project. As stated above in the report the existing application carries out this process and this is the process that needs to be replicated but the method QAS Batch use cannot be seen due to licensing. The idea behind the developer’s methodology relates back to the problem of the dirty addresses being recorded as shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Address Data | | | | | | |
| Address1 | Address2 | Address3 | Address4 | County | Postcode |  |
| 65 | Castle Street |  |  | Armahg | BT12 3AB |  |
| Pointer Data | | | | | | |
| Address1 | Address2 | Address3 | Address4 | County | Postcode | UPRN |
| 65 | Castle Road |  | Tandragee | Armagh | BT12 3AB | 12345 |

The above example data shows how simple mistakes can give an incorrect address. It can be seen that the ‘Address2’ field in both datasets are different, this could be down to the fact that a person misheard the information when they were keying in the data. It can also be noticed that in the user data the ‘County’ field has a spelling mistake in the word, this again could be down to careless typing, none the less the address has been made incorrect. The numbering of the address suggest that there are a substantial amount of houses in that street therefore the postcodes could not be identical so again the address is incorrect.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Address Data | | | | | | | | | | | | | |
| Address1 | Address2 | Address3 | Address4 | County | Postcode | UPCID\_N\_WP | | | UPCID\_N\_C\_NP | | … |  | |
| 65 | Castle Street |  |  | Armahg | BT12 3AB | 65BT123AB | | | 65ArmahgB123 | |  |  | |
| Pointer Data | | | | | | | | | | | | | |
| Address1 | Address2 | Address3 | Address4 | County | Postcode | | UPCID\_N\_WP | UPCID\_N\_C\_NP | | … | | | UPRN |
| 65 | Castle Road |  | Tandragee | Armagh | BT12 3AB | | 65BT123AB | 65Armagh123 | |  | | | 12345 |

A method of creating matchkeys has been created by the developer to be able to tackle these misspelt and misinterpreted addresses. The basis of these matchkeys are down to different combinations of the addresses split up and concatenated together to form a unique postcode identification (upcid). These upcids would be created for all records in both the user dataset and the Pointer dataset via an automated process in the backend of the application. An example of a upcid can be seen below using the same sample data from above.

UPCID\_N\_WP = UPCID made up from Building Number (N) and the Whole Postcode (WP)

UPCID\_N\_C\_NP = UPCID made up from Building Number (N), the County (C) and the Number from the Postcode (NP)

The two UPCIDs listed above are only two of 14 UPCIDs used, the full 14 can be seen below:

|  |  |
| --- | --- |
| *MatchKeys* | *Description* |
| UPCID\_N\_WP | Door number of the address **(N)** + whole postcode **(WP)** |
| UPCID\_N\_NP | Door number **(N)** + only numbers from postcode **(NP)** |
| UPCID\_WS\_WP | Whole street address including numbers **(WS)** + whole postcode **(WP)** |
| UPCID\_WS\_NP | Whole street address including numbers **(WS)** + numbers from postcode **(NP)** |
| UPCID\_CS\_WP | All characters from street address **(CS)** + whole postcode **(WP)** |
| UPCID\_CS\_NP | All characters from street address **(CS)** + numbers from postcode **(NP)** |
| UPCID\_N\_NVD\_WP | Door numbers **(N)** + street address with no vowels or doubles **(NVD)** + whole postcode **(WP)** |
| UPCID\_N\_NVD\_NP | Door numbers **(N)** + street address with no vowels or doubles **(NVD)** + numbers from postcode **(NP)** |
| UPCID\_NVD\_WP | Street address with no vowels or doubles **(NVD)** + whole postcode **(WP)** |
| UPCID\_NVD\_NP | Street address with no vowels or doubles **(NVD)** + numbers from postcode **(NP)** |
| UPCID\_N\_FC\_WP | Door numbers **(N)** + first character of street address **(FC)** + whole postcode **(WP)** |
| UPCID\_N\_FC\_NP | Door numbers **(N)** + first character of street address **(FC)** + numbers from postcode **(NP)** |
| UPCID\_N\_3C\_WP | Door numbers **(N)** + first 3 characters of street address **(3C)** + whole postcode **(WP)** |
| UPCID\_N\_3C\_NP | Door numbers **(N)** + first 3 characters of street address **(3C)** + numbers from postcode **(NP)** |

The need for several UPCIDs as matchkeys is to allow for all the possible mistakes that can occur. As seen above in the example, it is possible that one mistake can make a UPCID unmatchable to the Pointer dataset whereas a different UPCID which was constructed in a different manner may return a match. The structure of these UPCIDs are based around making sure that all possible mistakes are covered as well as gaining matches for all possible types of houses.

The code to populate these UPCIDs can be found in appendix ??????, certain functions are used to collect the relevant data for these UPCIDs and the code for these functions can also be found in said appendix.

The matching process for matching the UPCIDs to retrieve the UPRNs on a specific table is quite complex and the code for this can be found in appendix ???????? however, below is pseudo code to outline this process. This is also assuming that the application has created the needed columns in each table and populated them with the UPCID data. This code is only for matching on one UPCID and it should be noted that the code is replicated when matching on more than one UPCID.

select user defined UPCID, pointer UPCID, pointer UPRN

into temporary table

from user defined table

inner join Pointer

on user defined UPCID = Pointer UPCID

This first part of the matching query is selecting the relevant information needed into a temporary table from both the user defined table and the Pointer table. It is selecting all the user defined UPCIDs from the user defined table as well as all the same Pointer UPCIDs and the Pointer UPRNs. The reason a temporary table is used is because the data within the table is not needed on a permanent basis and will change for each match process.

select user defined UPCID, count all records

into temporary table

from previous temporary table

group by user defined UPCID having count of all records >= 1

This second part of the match query is selecting all of the user defined UPCIDs from the above temporary table and grouping them together and only selecting the records where the total number of identical UPCIDs is equal to or greater than one (this is avoid blank records from the first temporary table).

update user defined table

set user defined UPRN = UPRN from second temporary table

from user defined table

inner join second temporary table

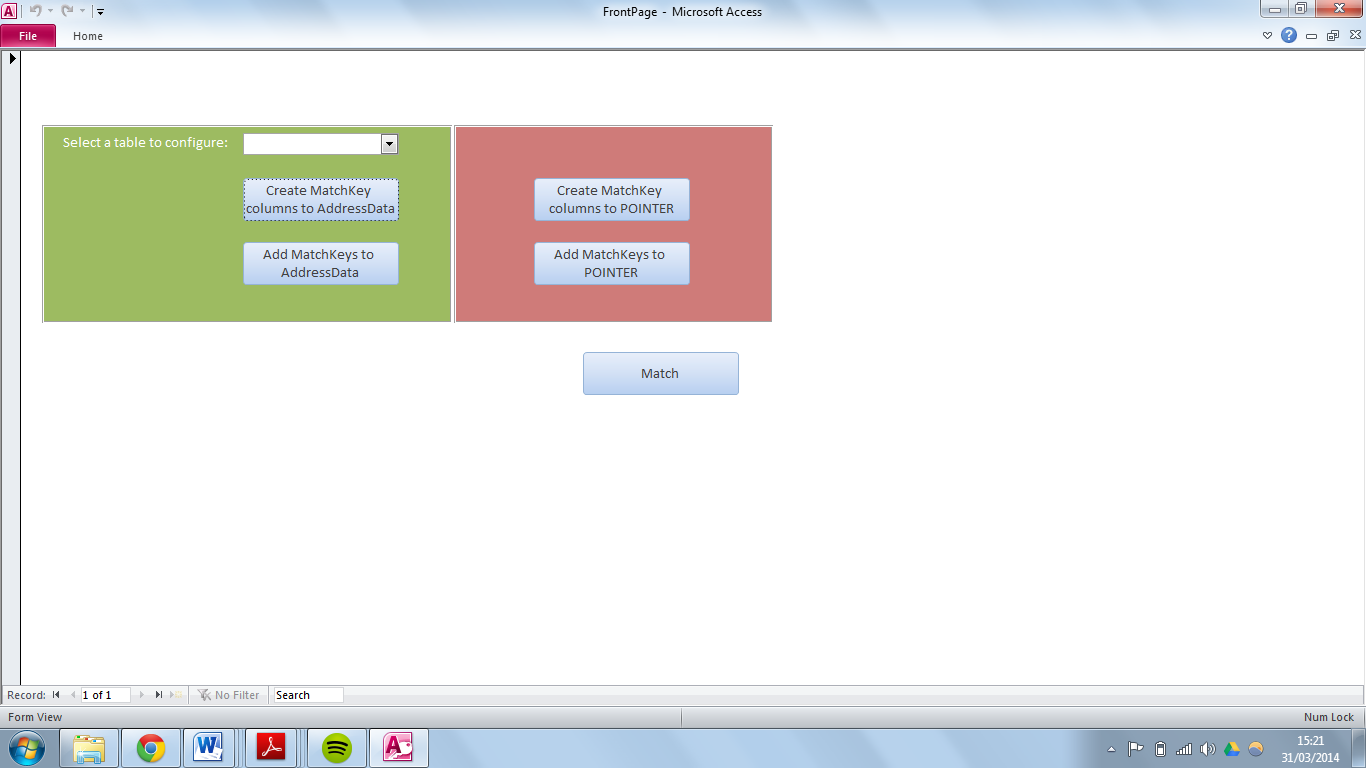
on user defined UPCID = second temporary table Pointer UPCID

inner join first temporary table

on user defined UPCID = first temporary table user defined UPCID

where user defined UPRN is empty and length of UPCID is > 1

drop both temporary tables

The last part of the query updates the user defined UPRN from the user defined table as long as the user defined UPCID is contained in both, the second temporary table and the first temporary table and the user defined UPRN is empty and the UPCID length is larger than one. The UPRN has to be empty when updating so that no other UPCID match overwrites the UPRN a previous UPCID match received and the length of the UPCID has to be greater than one so that no blank addresses are matched on.

**Initial Prototype**

Figure 18 showing the initial prototype of the project

Figure 18 shows the initial prototype of the project that was created. This prototype was mainly to test functionality of the back end queries to allow the developer to tweak the matching process so that the highest match rates can be achieved.

With consideration for the questionnaire results from staff at NISRA a new user interface was designed taking into account the colours that the NISRA logo contains.

When the new interface was developed all functions of the prototype were kept and developed on further and no function was discarded.

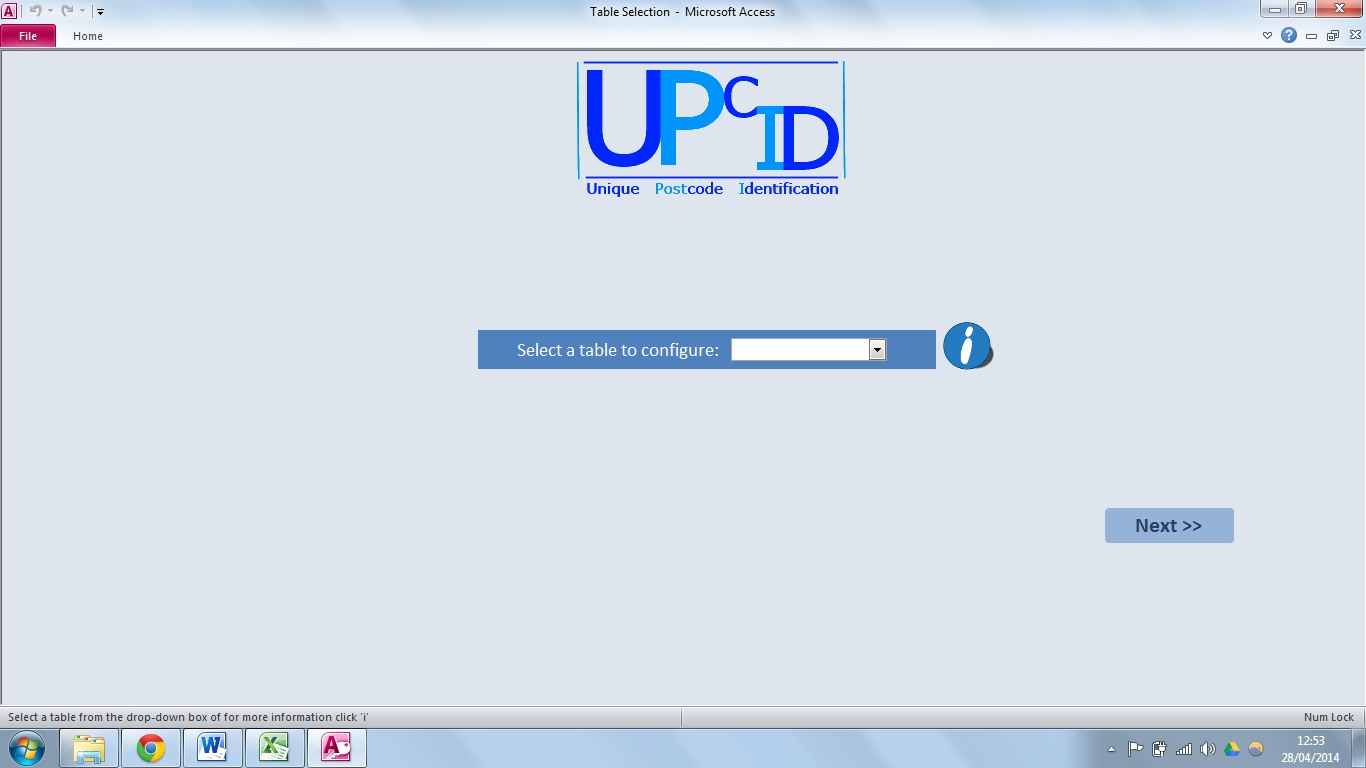
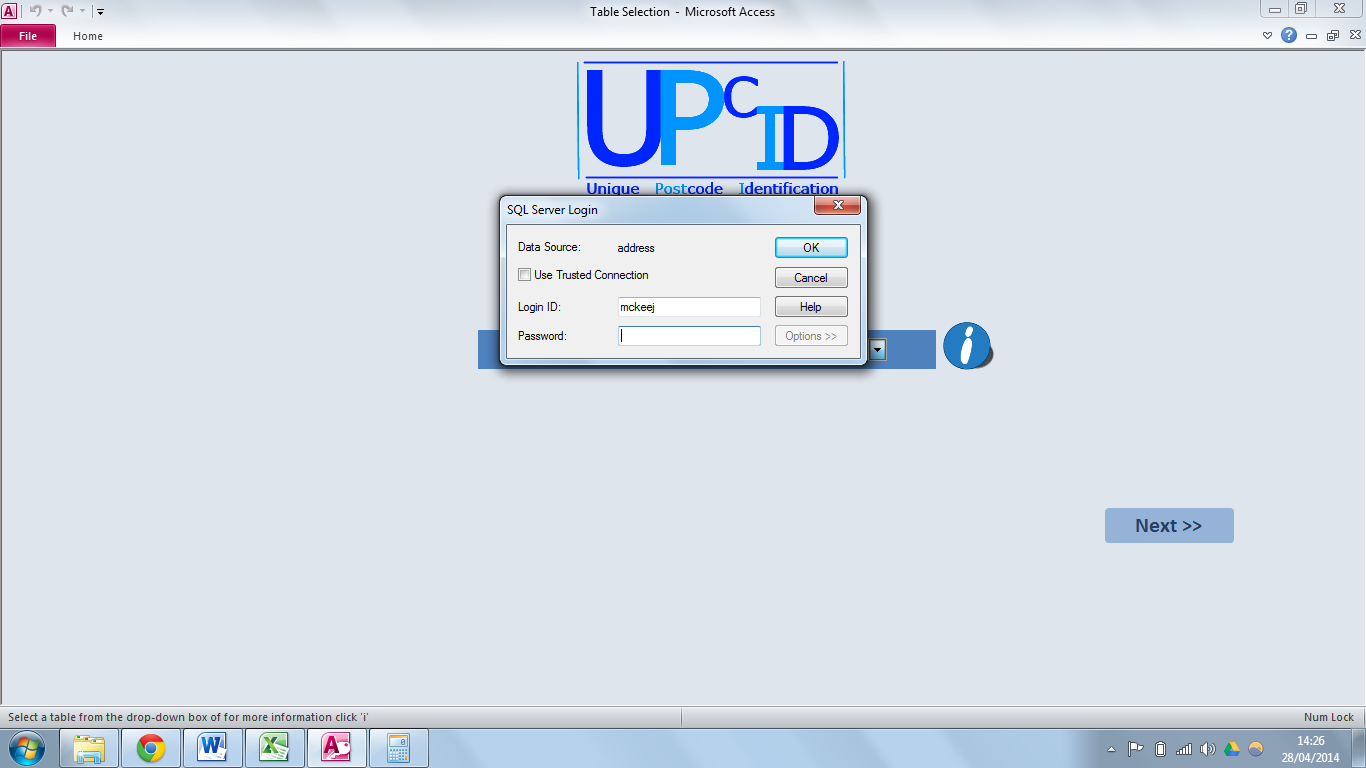


Figure 20 showing the login prompt.

Figure 19 showing the start screen.

Figure 19 show the start screen that users will see when the load up the application. It can be seen that NISRA’s staff feedback has been taken into consideration and the interface is simplistic and the colour scheme matches that of NISRA’s logo. When you try to select a table from the drop down box for the first time when opening the application you are prompted to log into your SQL account (as seen in figure 20) for security purposes. Without logging in the user cannot carry out any of the processes in the application.

Figure 22 showing a close up of the help text that is displayed

Figure 21 showing the more information window

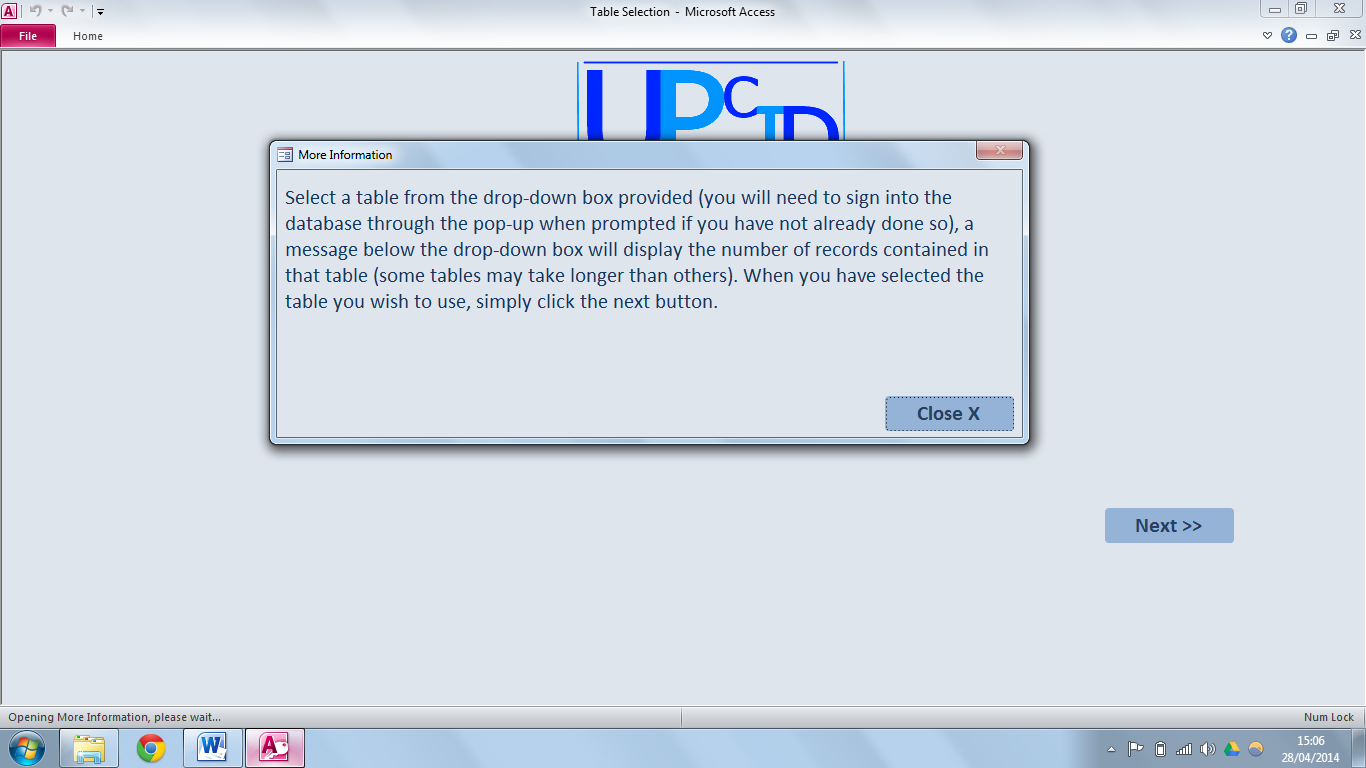
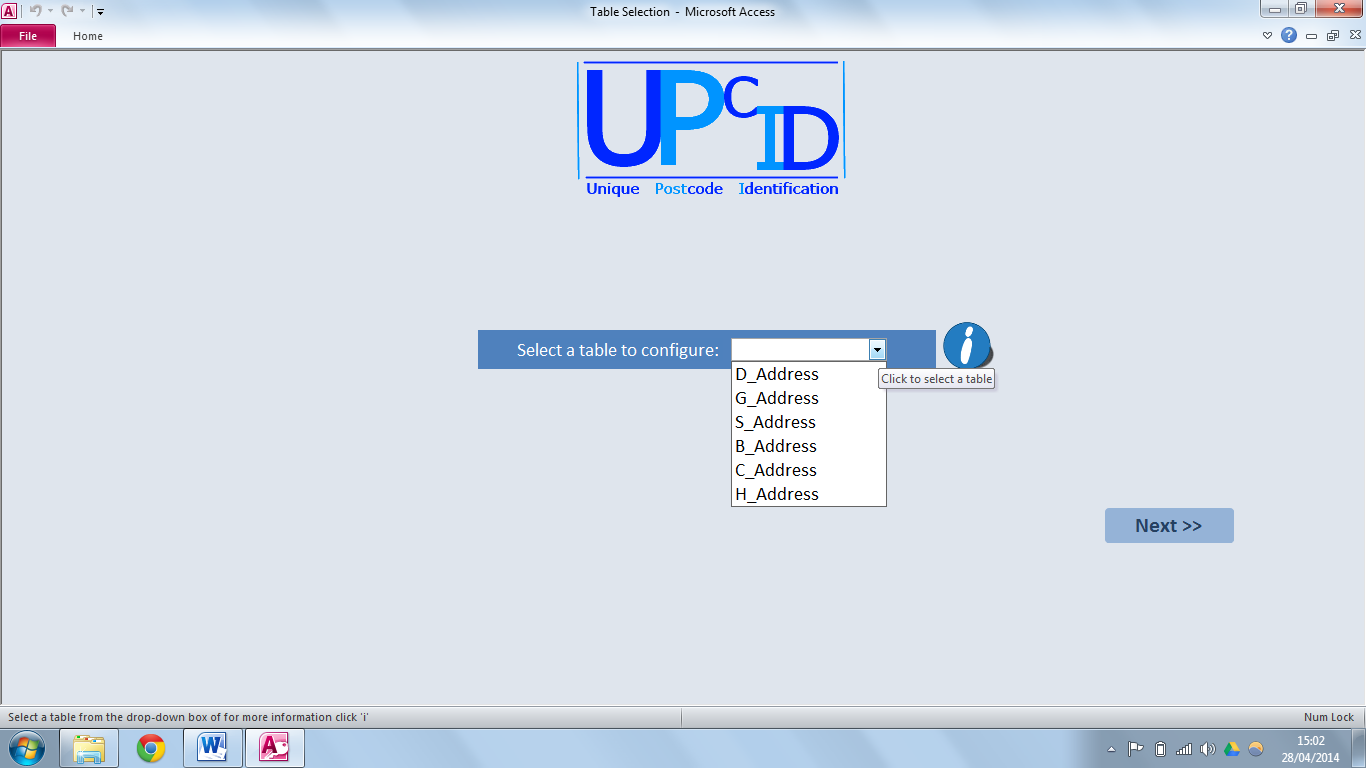


Figure 20 showing the populated drop down

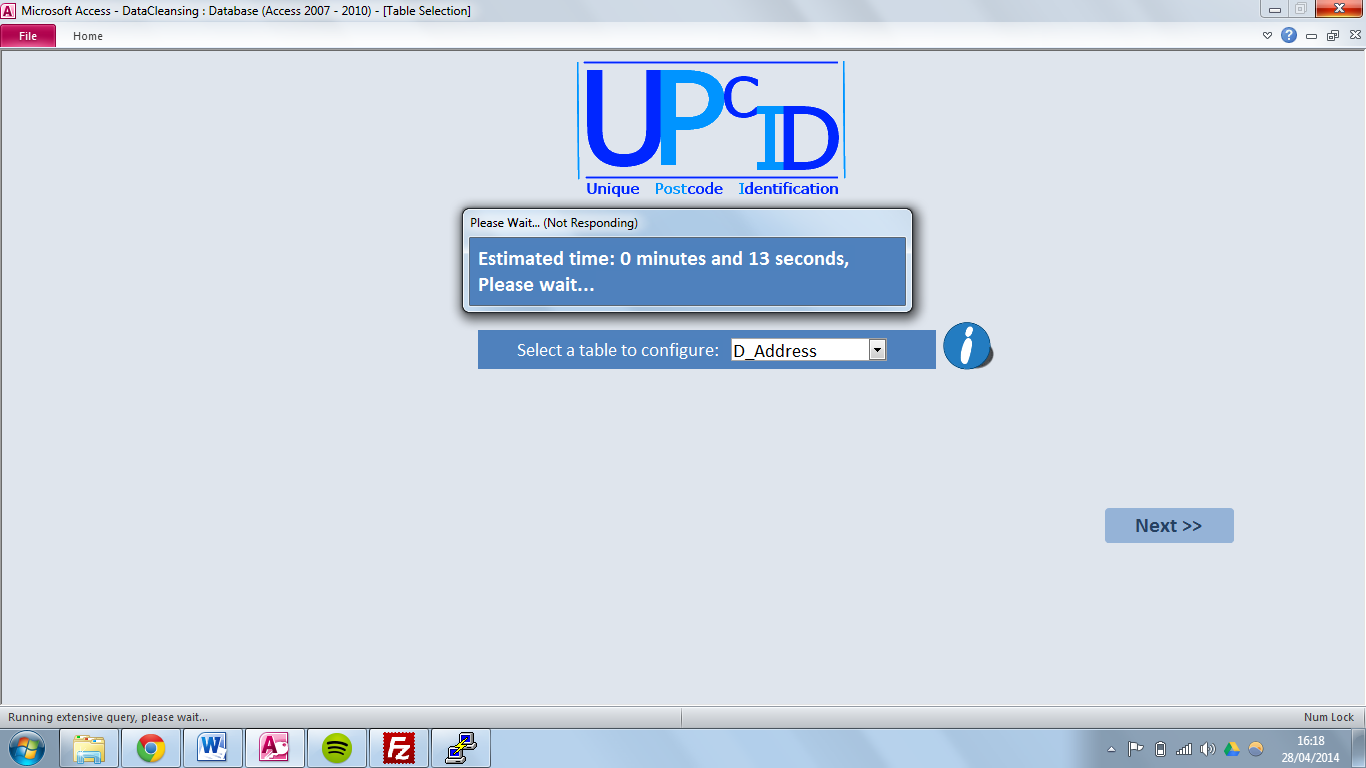
 Feedback from the questionnaires showed that NISRA staff wanted clear and concise instructions as to how to navigate around the application, therefore as many help features were implemented as possible. Figure 21 shows that an information window appears when the ‘i’ button is pressed giving the users instructions as to how to use the form that is presented. Figure 22 shows short instructions as to what the user has to currently do and when the user carries out that function the next instruction is posted. These features will be frequent across the entire application.

Figure 23 showing Please Wait form

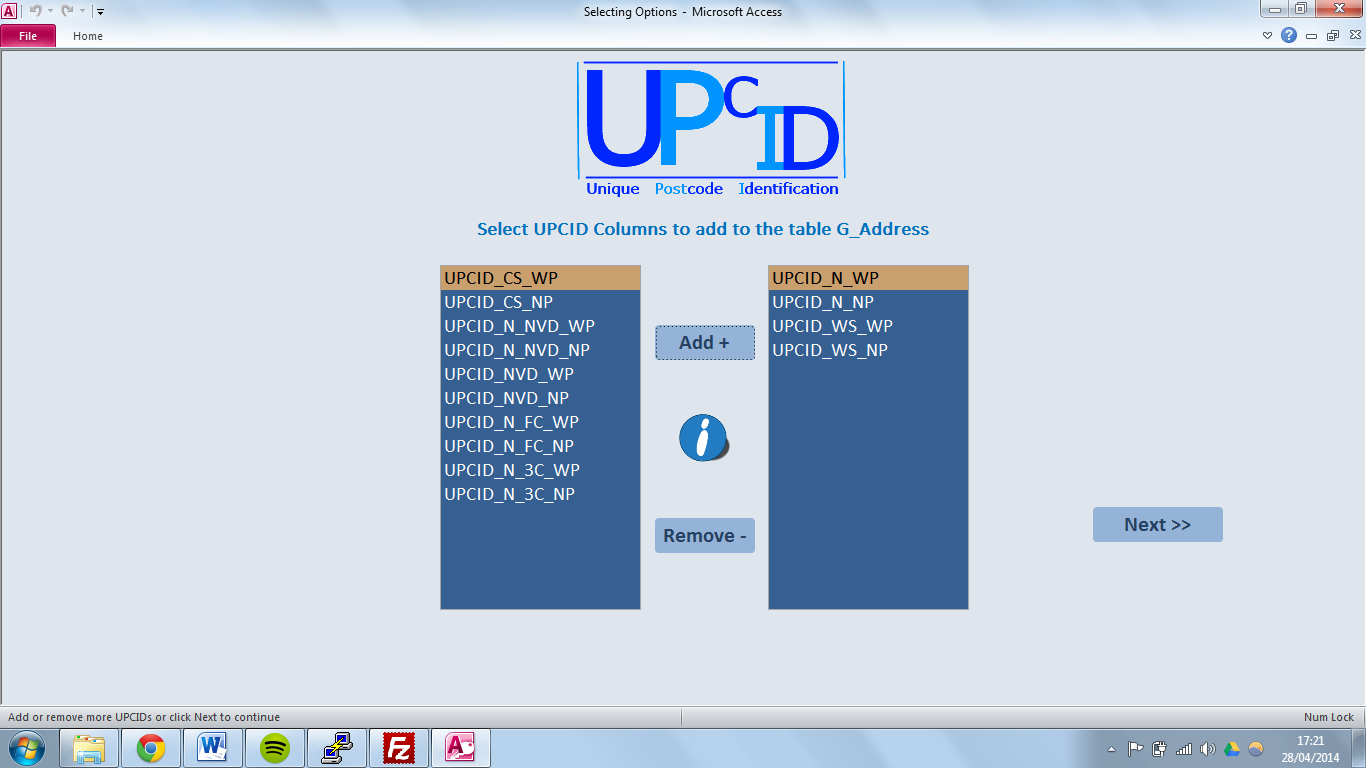
The form shown in figure 23 is a very useful feature implemented into the application. This form will display an estimated time that the current process will take to complete. The estimation is the time that the process took to run the previous time it was executed. This information is stored in a table and updated every time the process runs to give the most up-to-date estimation. This is to allow for growth in the user defined tables which would cause the processes to take longer to run. The reason that the window shows “(Not Responding)” is because a query is running which make the application unusable until the query is complete. The code for this feature can be seen in appendix ??????????

Figure 24 showing the selection screen

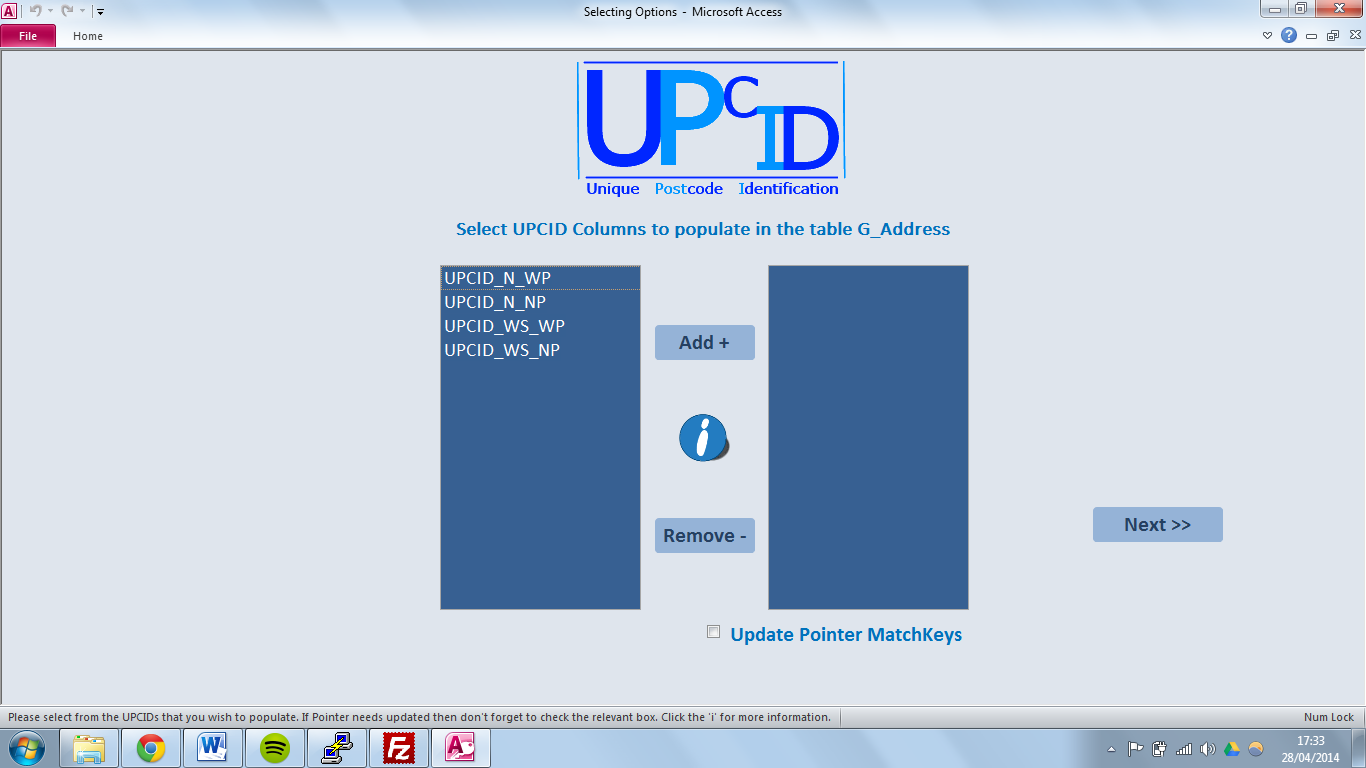
Figure 24 shows the form that is used for the next stages of the process is updated to suit the particular part of the process. The label at the top of the form gets updated to alert the user as to what they have to do as well as the more information form which is displayed when the ‘i’ button is clicked. Figure 24 shows the UPCID column adding process where the user selects from the 14 UPCIDs in the left hand list box which UPCIDs they wish to add. They do this by selecting the relevant UPCID from the left hand box and clicking the ‘add’ button, the UPCID will be removed from the list box on the left and added to the list box on the right. The process is reversed for removing the UPCIDs from the list box on the left. In this part of the process when the application is loaded and the relevant table is chosen the list boxes are populated depending on the current status of the table chosen. If there are already UPCID columns in the selected table then they will automatically be displayed in the right hand list box and not in the left. This also effects how the remove button is used. If an UPCID from the right hand list box is removed and that column was in the table, when the next button is clicked to move onto the next process the removed UPCID will be fully removed from the selected table. The two list boxes in the centre of the form also get updated depending on the stage of the process you are on. The UPCIDs selected in figure 24 will be the only UPCIDs available to choose from in the population process as shown in figure 25 as they are the only UPCID columns in the selected table.

Figure 25 showing the selection screen during a different process

There is a check box at this stage of the process that you can tick if you want to update the UPCIDs in the Pointer table. This check box is only available to tick if the UPCIDs in the Pointer table have not been updated in the past 30 days. This is to allow for the updated Pointer file to be read into the database on a monthly basis and then it can be updated and there unneeded to be updated before then.

**OUTPUTS TO BE ADDED!!!!**

*Database architecture*

This application is connected to an outside database application named SQL server via Open DataBase Connectivity (ODBC). This connection allows queries to run through SQL server and return the output to MS Access which saves processing time. The ODBC timeout length had to be omitted from the queries as it is difficult to be sure of the length a query will take and therefore the connection may timeout causing the application to crash.

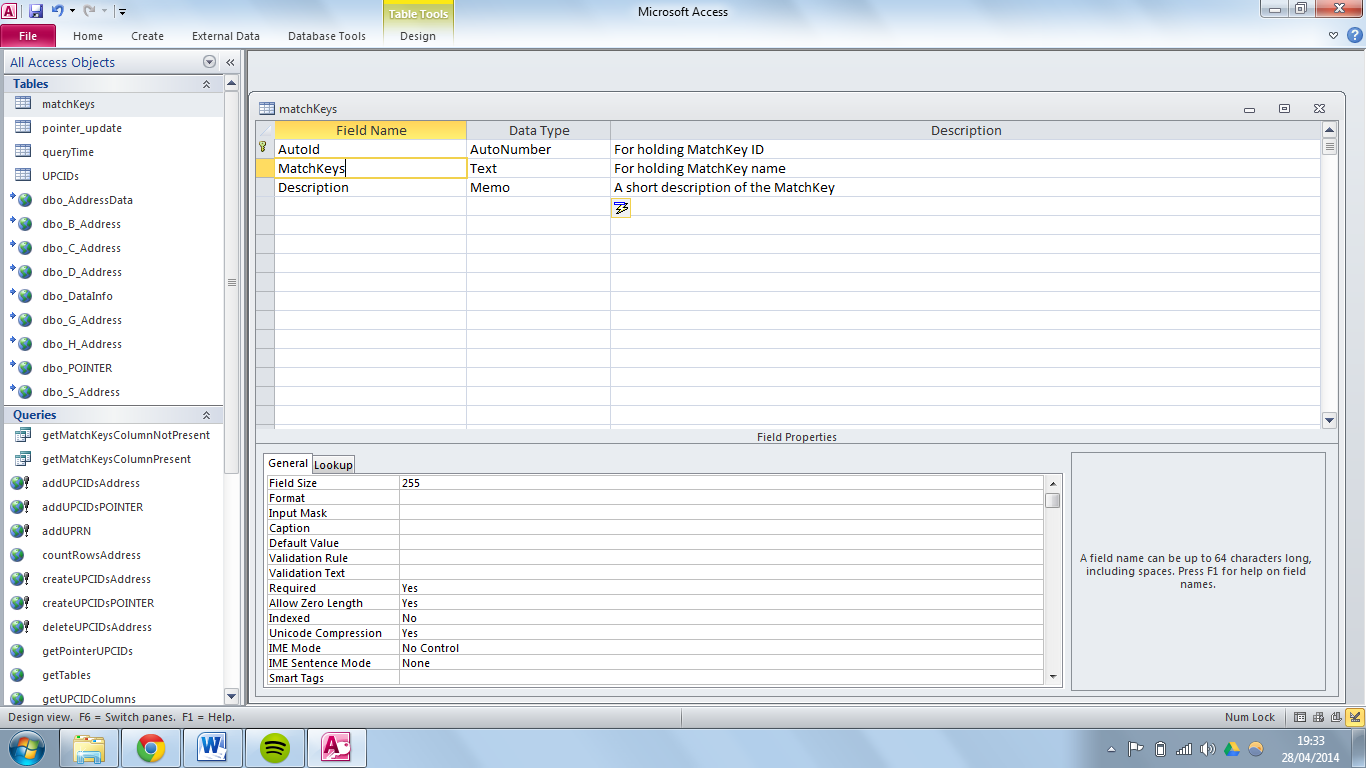
For some of the functions within MS Access other tables needed to be created and the architecture for these is shown below.

Figure 27 showing the architecture of the pointer\_update table

Figure 26 showing the architecture of the matchKeys table

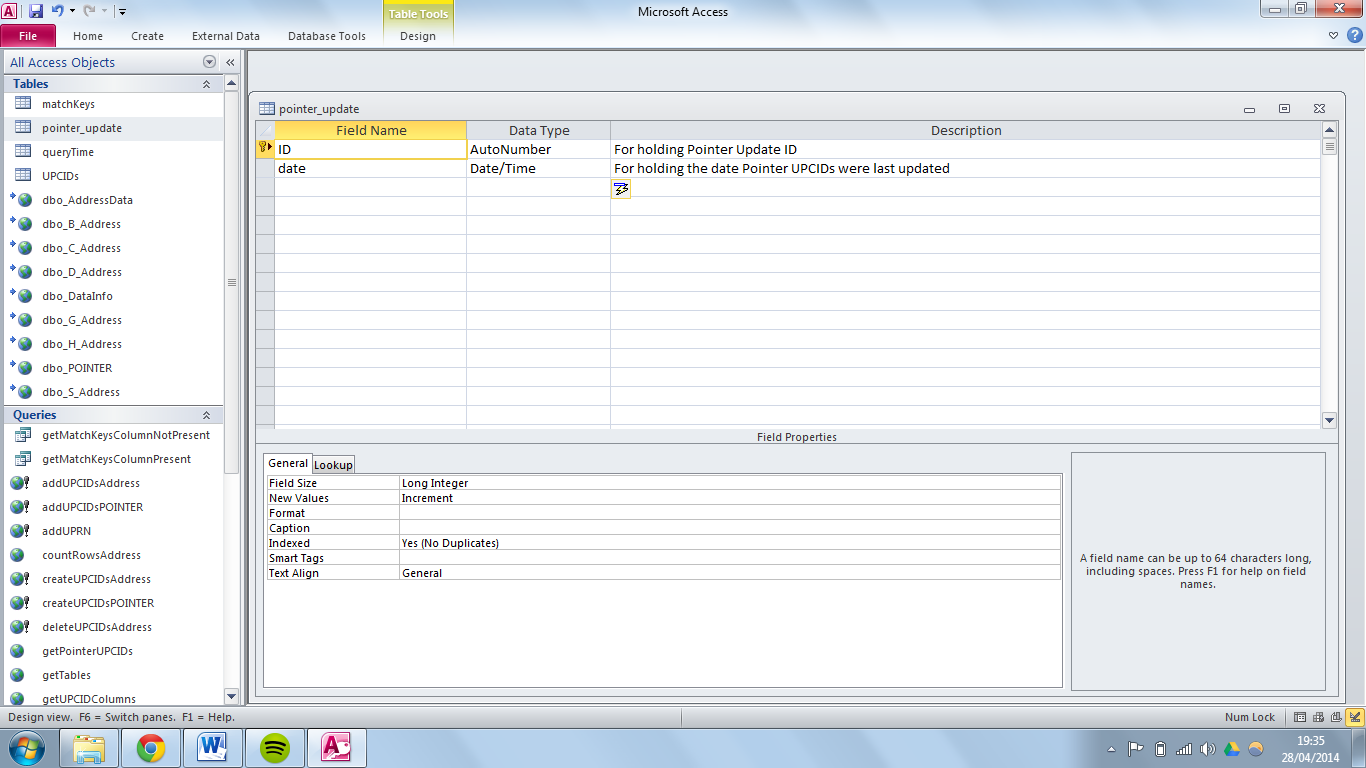
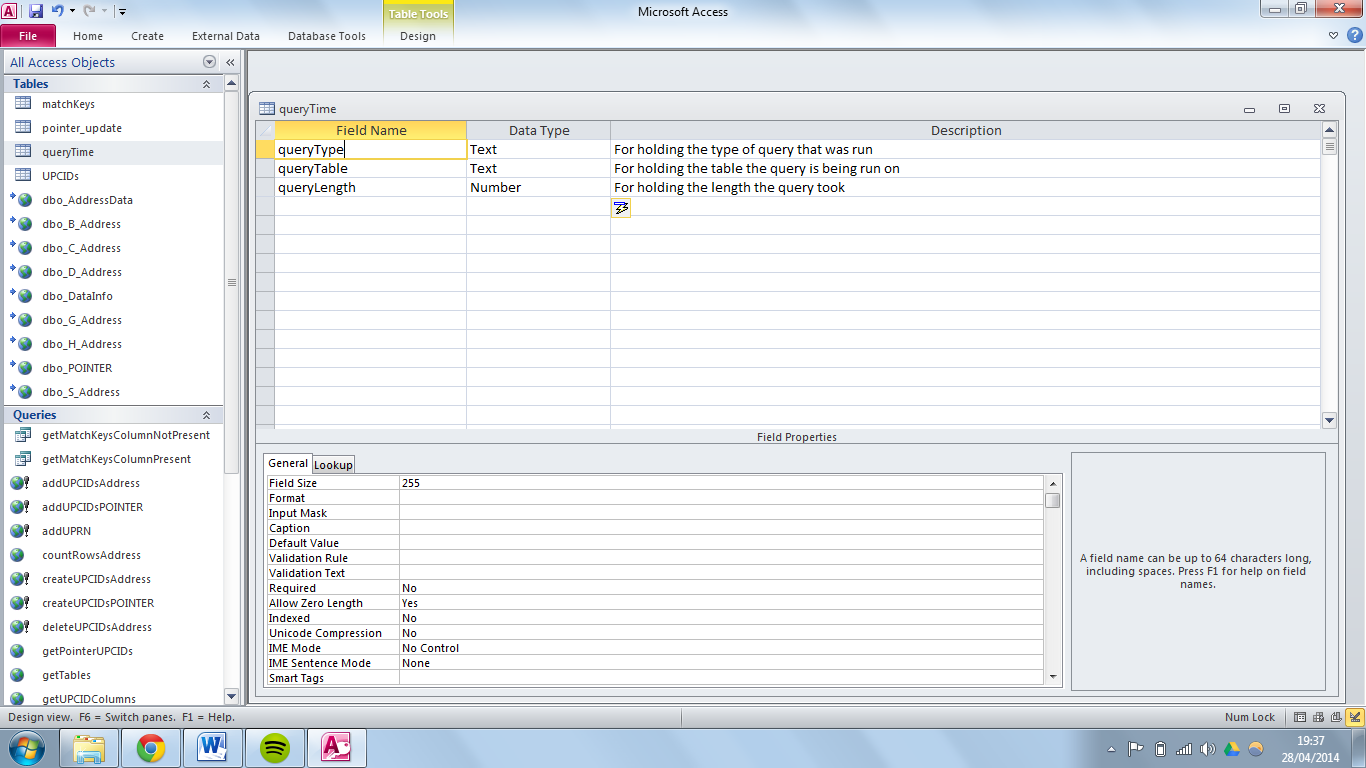
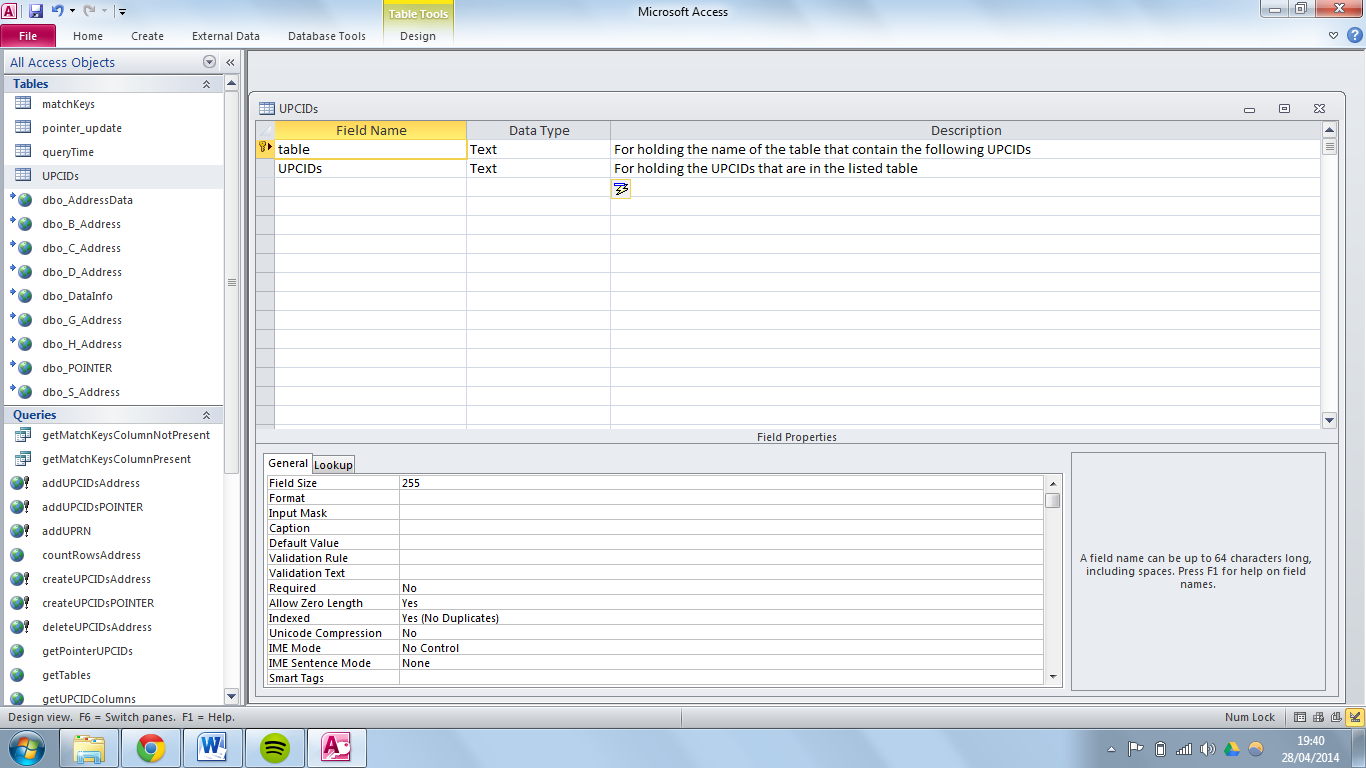


Figure 29 showing the architecture of the UPCIDs table

Figure 28 showing the architecture of the queryTime table

As the application involves the use of tables within itself as well as external tables, figure 30 was made to help understand the communication between the application and tables, queries and functions.

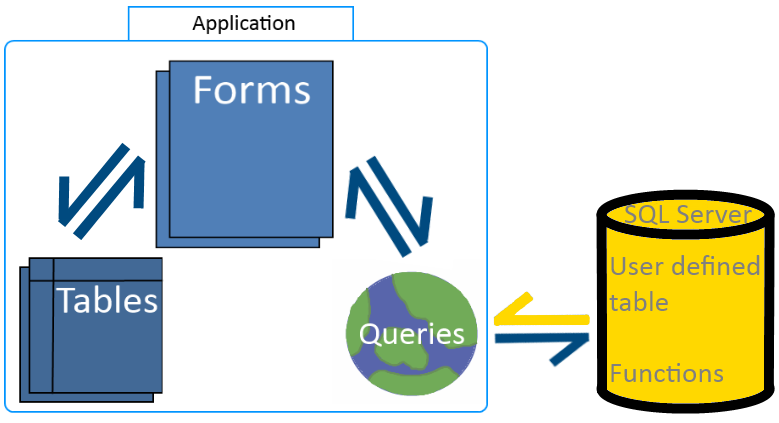


Figure 30 showing the communication between the application and its components.

# Implementation, Testing and Evaluation

In this chapter the developer will discuss how they implemented the functionality of the application in accordance to the design specification, outlining any adjustments that had to be made during the development and some examples of code will be provided to back this up.

The approach taken to developing this application was using a loosely based agile methodology. This allowed for a basic working prototype developed and ready to be demonstrated to NISRA staff. Then the prototype could be built upon to finally arrive at a fully working application that NISRA are happy with.

After the using the prototype and allowing NISRA to test the prototype it was clear that some changes needed to be implemented and further features needed to be added.

The prototype carried out the basic processes required such as adding the UPCID columns to the user defined table and the pointer table. The code for this is shown below.

*User defined table:*

ALTER TABLE G\_Address

ADD UPCID\_CS\_WP varchar(128), UPCID\_CS\_NP varchar(128), UPCID\_N\_NVD\_WP varchar(128), UPCID\_N\_NVD\_NP varchar(128), UPCID\_NVD\_WP varchar(128), UPCID\_NVD\_NP varchar(128), UPCID\_N\_FC\_WP varchar(128), UPCID\_N\_FC\_NP varchar(128), UPCID\_N\_3C\_WP varchar(128), UPCID\_N\_3C\_NP varchar(128)

*Pointer table:*

ALTER TABLE POINTER

add UPCID\_N\_WP varchar(128), UPCID\_N\_NP varchar(128), UPCID\_WS\_WP varchar(128), UPCID\_WS\_NP varchar(128), UPCID\_CS\_WP varchar(128), UPCID\_CS\_NP varchar(128), UPCID\_N\_NVD\_WP varchar(128), UPCID\_N\_NVD\_NP varchar(128), UPCID\_NVD\_WP varchar(128), UPCID\_NVD\_NP varchar(128), UPCID\_N\_FC\_WP varchar(128), UPCID\_N\_FC\_NP varchar(128), UPCID\_N\_3C\_WP varchar(128), UPCID\_N\_3C\_NP varchar(128)

The prototype also enabled the user to populate the UPCIDs for both the user defined tables and also the Pointer table as shown below:

*User defined table:*

Update G\_Address set upcid\_n\_wp = REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers') + ISNULL(POSTCODE, ''), ' ', '')Update G\_Address set upcid\_n\_np = REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')+ISNULL(POSTCODE, ''), 'numbers'), ' ', '')

*Pointer table:*

UPDATE POINTER

SET upcid\_n\_wp = REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + ISNULL(POSTCODE, ''), ' ', ''),

upcid\_n\_np = REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE,'') + ISNULL(POSTCODE, ''), 'numbers'), ' ', ''),

UPCID\_WS\_WP = dbo.unwanted(REPLACE(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '') + ISNULL(POSTCODE, ''), ' ', '')),

UPCID\_WS\_NP = dbo.unwanted(REPLACE(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '') + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', '')),

upcid\_cs\_wp = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters') + ISNULL(POSTCODE, ''), ' ', '')),

upcid\_cs\_np = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters') + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', '')),

UPCID\_N\_NVD\_WP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + dbo.nvd(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '')) + ISNULL(POSTCODE, ''), ' ', '')),

UPCID\_N\_NVD\_NP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + dbo.nvd(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '')) + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', '')),

UPCID\_NVD\_WP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(dbo.nvd(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '')), 'letters') + ISNULL(POSTCODE, ''), ' ', '')),

UPCID\_NVD\_NP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(dbo.nvd(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, '')), 'letters') + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', '')),

UPCID\_N\_FC\_WP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + LEFT(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters'), 1) + ISNULL(POSTCODE, ''), ' ', '')),

UPCID\_N\_FC\_NP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + LEFT(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters'), 1) + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', '')),

UPCID\_N\_3C\_WP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + LEFT(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters'), 3) + ISNULL(POSTCODE, ''), ' ', '')),

UPCID\_N\_3C\_NP = dbo.unwanted(REPLACE(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'numbers') + LEFT(dbo.fn\_extract\_chars(ISNULL(SUB\_BUILDING\_NAME, '') + ISNULL(BUILDING\_NAME, '') + ISNULL(BUILDING\_NUMBER, '') + ISNULL(PRIMARY\_THORFARE, ''), 'letters'), 3) + dbo.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))

Within the populating statements you will notice the use of several functions. These functions were created to extract specific parts of the addresses to be used for the MatchKeys and the code for these can be seen below:

*Extract characters (dbo.fn\_extract\_chars)*

ALTER FUNCTION [dbo].[fn\_extract\_chars](@x VARCHAR(128), @y CHAR(7))

RETURNS VARCHAR(128)

AS

BEGIN

DECLARE @chars VARCHAR(128)

DECLARE @pos INT

DECLARE @action VARCHAR(32)

SET @pos = 0

SET @chars = ''

IF @y = 'numbers' SET @action = '[0-9]'

ELSE IF @y = 'letters' SET @action = '[a-zA-Z]'

WHILE @pos < (DATALENGTH(@x) + 1)

BEGIN

IF PATINDEX(@action,SUBSTRING(@x, @pos, 1)) > 0

BEGIN

SET @chars = @chars + (SELECT substring(@x, @pos, 1))

END

SET @pos = @pos + 1

END

RETURN(@chars)

END

This function is called using the keyword dbo.fn\_extract\_chars (x, y) and passing two variables x and y. Variable x is the string that the characters will be extracted from and variable y is the keyword that will determine whether the characters being extracted are ‘letters’ or ‘numbers’. The function works by first determining whether the characters to be extracted are letters or numbers, then while the length of the string is greater than 1 the function takes each character of the string in turn and analyses it to find if it is a letter or number. Depending on what variable y is the character will be appended to a string which at the end of the function will be returned.

*No vowels or doubles (dbo.NVD)*

ALTER function [dbo].[nvd]( @Word varchar(100) )

returns varchar(100)

as

begin

declare @WordLength int

declare @ReturnValue varchar(100)

declare @icounter int

declare @pointer int

set @wordlength = len (@word)

SET @icounter = 1

set @pointer = -1

while ( @icounter <= @WordLength )

BEGIN

if not ( SUBSTRING(@Word,@icounter,1)='a' or

SUBSTRING(@Word,@icounter,1)='e'or

SUBSTRING(@Word,@icounter,1)='i'or

SUBSTRING(@Word,@icounter,1)='o'or

SUBSTRING(@Word,@icounter,1)='u')

Begin

set @pointer = @icounter

set @returnvalue = SUBSTRING(@Word,@icounter,1)

break

end

set @icounter = @icounter + 1

END

if (@pointer > 0 )

BEGIN

WHILE ( @pointer <= @WordLength )

BEGIN

if not ( SUBSTRING(@Word,@pointer,1)='a' or

SUBSTRING(@Word,@pointer,1)='e'or

SUBSTRING(@Word,@pointer,1)='i'or

SUBSTRING(@Word,@pointer,1)='o'or

SUBSTRING(@Word,@pointer,1)='u')

Begin

IF SUBSTRING(@Word,@pointer,1) <> RIGHT (@RETURNVALUE , 1) SET @RETURNVALUE = @RETURNVALUE + SUBSTRING(@Word,@pointer,1)

END

set @pointer = @pointer + 1

END

END

ELSE

BEGIN

SET @RETURNVALUE = NULL

END

RETURN (@returnvalue)

end

This function works almost the same as the above function in that it is called using the keyword db.nvd() and passing a string as a variable. The variable is then measured for length and using this length each character of the string is examined to check whether it is a vowel and also whether it is identical to the next character, if either of these or both of these conditions are true then the character is removed from the string. The string is then returned at the end of the function.

*Unwanted (dbo.unwanted)*

ALTER FUNCTION [dbo].[unwanted] (@theString VARCHAR(100))

RETURNS VARCHAR(100)

AS

BEGIN

declare @unwantedStrings table (item varchar(50))

INSERT INTO @unwantedStrings(item)

SELECT 'flat' UNION ALL

SELECT 'unit' UNION ALL

SELECT 'site' UNION ALL

SELECT 'apartment' UNION ALL

SELECT 'flt' UNION ALL

SELECT 'apt' UNION ALL

SELECT '.' UNION ALL

SELECT '-' UNION ALL

SELECT '/' UNION ALL

SELECT ','

SELECT @theString = REPLACE(@theString, item, '') FROM @unwantedStrings

RETURN(@theString)

END

This function again uses the keyword dbo.unwanted and a variable is passed, only this function creates a table full of unwanted strings and checks the received variable against all values in the table to see if it matches any of the table values, if there is a match then the variable is removed from the string. This functions purpose is to remove the common words of ‘flat’ and ‘apartment’ etc. from address strings as this interferes with certain UPCIDs when retrieving characters of the address. The string is returned at the end of the function if there are any remaining characters.

The match process is the only other function that was found in the prototype and that code can be found in the appendices.

The feedback from the prototype when demonstrated was very helpful and critical in terms of the user interface and the need of more features. When asked about the user interface the NISRA staff suggested that a simplistic approach be taken so that users do not feel bombarded and over loaded with features.

When taking the feedback into consideration a more user friendly interface was developed that allowed the user to work through the process in a step by step process only having one step on the screen at the one time. When the user opens the application they are prompted to choose a table that they wish to match on then they click the next button. On the next screen the user is prompted to choose which UPCIDs they wish to create columns for in the chosen table. The user also now has all 14 UPCIDs to choose from and the UPCIDs they choose will be the only UPCIDs that are available to choose from in the next process, this reduces confusion to the user in terms of UPCIDs. Also by using the same form for 3 steps the process keeps uniformity to the user and allows them to be comfortable with the process as they do not have several form layouts to remember how to navigate.

More feedback that was taken from the demonstration of the prototype was that ability to choose which UPCIDs the user could add to the chosen table. This feature was wanted so that the most useful UPCIDs could be chosen for a quick match on a small table and more UPCIDs could be chosen for a table that was substantially large. Therefor in an updated version of the application this feature was added via the use of list boxes. The user is shown two list boxes, one that contains a list of all the UPCIDs that can be added to a chosen table, from here the user can add the wanted UPCIDs to the other list box so that they can be added to the table. The code for the add and remove buttons are as follows:

*Add*

For i = 0 To List0.listCount - 1

If List0.Selected(i) Then

List2.AddItem List0.Column(0)

List0.RemoveItem List0.Column(0)

End If

Next i

*Remove*

For i = 0 To List2.listCount - 1

If List2.Selected(i) Then

List0.AddItem List2.Column(0)

List2.RemoveItem List2.Column(0)

End If

Next i

The above pieces of code simply remove one selected UPCID from the list box and add it to the other list box and vice versa.

The list boxes are pre-populated every time the form opens depending on the process at hand. If the user is adding the UPCID columns to the table then the right hand list box will be pre-populated with the UPCIDs that are currently in the table.

*SELECT COLUMN\_NAME FROM [INFORMATION\_SCHEMA].COLUMNS WHERE COLUMN\_NAME LIKE 'upcid%' AND TABLE\_NAME = '" & strTable & "'*

The above query is used to pre-populate the right hand list box, it is selecting the columns that are in the selected table where the column names contain the string ‘upcid’. The variable strTable is passed from when the user chooses a table on the first form.

When it comes to the second list box form used to populate the UPCID columns that were added, the list box on the left hand side is pre-populated with only the UPCIDs that currently have columns in the selected table. The query used to do this is the same as the query above only this time it is used on the other left list box so that users can select the UPCIDs that they wish to populate. The idea behind this part of the process is so that the user can have several UPCID columns in the selected table and not have to use them straight away as the match process may only require one UPCID.

A function that was implemented from an external source (dark11984, 2013) is called ChangePTStatement. This code enables a pass through query text to be change through visual basic code meaning that if a certain process is carried out such as a button pressed, code within the relevant pass through statement can change. A pass through statement is a query within MS access that does not run in MS Access but instead runs in MS SQL Server and the output is received in MS Access. The code for this feature is shown below.

Public Sub ChangePTStatement(p\_QueryName As String, p\_sql As String)

'for changing pass-through's in this db

'dark11984 Access Forums

Dim qdef As DAO.QueryDef

Set qdef = CurrentDb.QueryDefs(p\_QueryName)

qdef.sql = p\_sql

qdef.Close

Set qdef = Nothing

End Sub

This function works by passing two variable strings through the keyword ChangePTStatement(). One variable holds the pass through query name to be changed and the other variable holds the query string that the pass through needs to be changed to.

The above feature works well in conjunction with another feature that is used to generate the queries for creating the UPCID column list, the list for populating the UPCIDs and also the list for matching the UPCIDs. For demonstration purposes the population of the UPCID columns will be used as an example of this feature. Firstly the list of UPCIDs that are to be used need to be identified and this is done by inserting the values of the right hand list box into an array.

Dim caseStates() As String

caseStates = Split(Replace(List2.RowSource, ";", ", "), ", ")

These same values also need to be collected in a string to be used in the query update statement.

Dim forSQL As String

forSQL = Replace(List2.RowSource, ";", ", ")

A for loop is then used to loop through all the UPCIDs held in the array and within the for loop are a list of case statements where if the UPCID in the array matches the UPCID in the case statement then the relevant query data is added to a string.

For i = LBound(caseStates) To UBound(caseStates)

Select Case caseStates(i)

Case "UPCID\_N\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_wp = REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers') + ISNULL(POSTCODE, ''), ' ', '')"

Case "UPCID\_N\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_np = REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')+ISNULL(POSTCODE, ''), 'numbers'), ' ', '')"

Case "UPCID\_WS\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_ws\_wp = dbo.unwanted(REPLACE(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')+ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_WS\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_ws\_np = dbo.unwanted(REPLACE(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')+DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

Case "UPCID\_CS\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_cs\_wp = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters') + ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_CS\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_cs\_np = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters')+DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

Case "UPCID\_N\_NVD\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_nvd\_wp = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+DBO.nvd(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')) +ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_N\_NVD\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_nvd\_np = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+DBO.nvd(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')) +DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

Case "UPCID\_NVD\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_nvd\_wp = dbo.unwanted(REPLACE(DBO.nvd(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')) +ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_NVD\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_nvd\_np = dbo.unwanted(REPLACE(DBO.nvd(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, '')) +POINTER.DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

Case "UPCID\_N\_FC\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_fc\_wp = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+LEFT(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters'), 1)+ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_N\_FC\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_fc\_np = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+LEFT(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters'), 1)+DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

Case "UPCID\_N\_3C\_WP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_3c\_wp = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+LEFT(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters'), 3)+ISNULL(POSTCODE, ''), ' ', ''))"

Case "UPCID\_N\_3C\_NP"

sqlstr = sqlstr & sqlDefault & " upcid\_n\_3c\_np = dbo.unwanted(REPLACE(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'numbers')+LEFT(DBO.fn\_extract\_chars(ISNULL(Address1, '')+ISNULL(Address2, '')+ISNULL(Address3, '')+ISNULL(Address4, ''), 'letters'), 3)+DBO.fn\_extract\_chars(ISNULL(POSTCODE, ''), 'numbers'), ' ', ''))"

End Select

Next i

When the for loop ends the string with all the relevant query data is used within the ChangePTStatement function to create the query needed to carry out populating the selected UPCIDs in the table.

This process is the same for the adding UPCIDs to the tables as well as the matching process.

Other feedback from NISRA staff was in regards to colour scheme and one member of staff was keen on using the colours of the NISRA logo within the application to give the application a homely feel. This feedback was taken into consideration and the colours were implemented into the application to give it a professional and fresh feel.

**Testing**

Testing is a major part of any application development to ensure that all possible bug and errors are patched and solved within the system and this part of the chapter is committed to making sure that this application is fully functional and free from error.

The two types of testing that will be covered are unit testing and acceptance testing. Unit testing will be testing every component of the application and also testing every possibility of every feature including the use of all combinations of UPCIDs. Acceptance testing is were several users, some who are familiar with the application and some who are not, to test the application without many instructions to test and see if the application is easily used.

A summary of the unit testing can be seen in figure 31 and full details can be found in appendix ?????????

*User Acceptance Testing*

A small number of colleagues as well as member of the NISRA staff were tasked to use this application with only the instructions that are provided within the application. They were asked to complete the process on whichever table they chose and to add as many UPCIDs as they wished.

The reason that not all of the test members are from NISRA is to allow for users that are completely new to the environment and have not seen the application at all before and therefore have no previous knowledge of the process or data.

When asked for feedback on the use of the application all the feedback said that the application was extremely easy to use thanks to the instructions. The non-NISRA staff said that there was confusion over what the UPCIDs were made up of as this is not told in the application and they had no previous knowledge of the software or process whereas NISRA did not mention this as they are familiar with the process.

**Evaluation**

At the beginning of the project objectives were created which were to be accomplished by the end of the project and these were also used to create a list of requirements between the developer and NISRA. All of the requirements were to be met as well as having a fully working implemented system for NISRA to use.

The application that was to be developed was to enable NISRA users to carry out a process in which UPRNs were to be appended to addresses located in a user defined table. This processes aim was to at least match that of the current application that NISRA use if not improve on that. The application needed to be easy to use with a simplistic interface that users could navigate and understand.

Although not all of the features were implemented due to time frames, all of the main features were implemented to a working manner which will allow the application to be used by NISRA until further developments are made. NISRA were happy with the features that are currently in operation in the application and are excited to be using this application.

The overall interface of the application is pleasing to all users both NISRA staff, the developer and also to the testers who sampled the application. The interface is simplistic and easy to use for users of all abilities.

The developer was extremely happy with the final product but felt that further development could be made to better the application and help it reach its full potential. The process that the application carries out is a much needed process and therefore there is the potential of many more features that could be added.

In terms of the user requirements below is a short summary of how the final product sizes against the requirements.

|  |  |  |
| --- | --- | --- |
| Requirement | Met/Did not meet | Notes |
| User prompted to log into the SQL server when application opens. If user does not enter both or they are incorrect the application should prompt again. | Did not meet – see notes | User is prompted to log into SQL Server when application opens but if declined user is not prompted again and user cannot continue using the application |
| A user should be able to select a table name in the SQL database from an automatically populated list. | Met |  |
| A user should be able to click a minimum of one button which will carry out the cleansing process with minimum user intervention | Did not meet – see notes | This requirement was modified after the development of the prototype to allow more customisation of the process although the process is still short. |
| Throughout the entire cleansing process the user should be informed if a process has already been completed on a chosen table. | Met |  |
| A user should be able to view a report for the matching process, outlining how many UPRNs were added to the table and what percentage of the records that covers. | Met |  |
| A user should be shown a form after the match process is complete displaying all the records from the table used in the match process that were assigned a UPRN along with the records in the pre-cleaned table that have the same UPRN. | Met |  |
| The application should have an easy to use graphical user interface with clear navigation options that will allow a user of any level to use. | Met |  |
| A user should be able to use this application without any prior knowledge to SQL server. | Met |  |
| If more than one button is required for a certain process then these buttons should be found together as to not confuse the user. | Met |  |

# Conclusions

In the final chapter you should summarise your project work overall and assess it critically. This should indicate what lessons you have learned and so clarify what you might do differently if faced with the same situation again. In particular, you should identify and discuss how the project plan evolved as the project progressed.

The limited time available for implementation means that you are likely to have ideas for further work. These should also be included in the conclusions.

The Conclusions chapter, like the Introduction, should be freestanding, allowing the reader to understand what the project has achieved without studying other chapters.

Length: ~4 pages

# References

dark11984, 2013. *Pass Through Criteria.* [Online]   
Available at: http://www.access-programmers.co.uk/forums/showthread.php?p=1225601  
[Accessed 20 11 2013].

Experian QAS, 2013. *Data cleansing solutions for improved data accuracy.* [Online]   
Available at: http://www.qas.co.uk/solutions/data-quality-software/data-cleansing/batch.htm  
[Accessed 22 10 2013].

Experian, 2014. *Data Cleansing Solutions.* [Online]   
Available at: http://www.qas.co.uk/solutions/data-quality-software/data-cleansing/batch.htm  
[Accessed 26 03 2014].

Land & Property Services, n.d. *Address Data and Gazetteers.* [Online]   
Available at: http://maps.osni.gov.uk/cmspages/moreinfo\_address\_data.aspx  
[Accessed 22 10 2013].

Microsoft, 2008. *SQL Server.* [Online]   
Available at: http://www.microsoft.com/en-us/server-cloud/products/sql-server/  
[Accessed 31 03 2014].

Northen Ireland Statictics and Research Agency, 2014. [Online]   
Available at: http://www.nisra.gov.uk/  
[Accessed 25 03 2014].

Northern Ireland Statistics and Research Agency, 2014. *About NISRA.* [Online]   
Available at: http://www.nisra.gov.uk/aboutus/index.html  
[Accessed 26 03 2014].

Oracle, 2014. *MySQL :: The World's Most Popular Open Source Database.* [Online]   
Available at: http://www.mysql.com/  
[Accessed 31 03 2014].

Ordnance Survey of Northern Ireland, n.d. *Address Data and Gazetteers.* [Online]   
Available at: https://maps.osni.gov.uk/cmspages/moreinfo\_address\_data.aspx  
[Accessed 31 03 2014].

# Appendices

## A1. Analysis Models

e.g. SSM models

## A2. Design Models

e.g. database schema

## A3. Code

Code developed through the project

## A4. Test Suite

Full set of tests applied to the software

## A5. Questionnaire Results

Results of questionnaires used to evaluate the software or identify requirements