**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.

Special thanks to Northern Ireland Statistics and Research Agency for their continued help throughout the entire development of the project without whom, this project would not have been possible.

Contents

[Abstract 3](#_Toc386787736)

[Introduction 4](#_Toc386787737)

[Problem Statement 4](#_Toc386787738)

[Software Methodology Chosen 5](#_Toc386787739)

[Aim of Project 6](#_Toc386787740)

[Objectives 6](#_Toc386787741)

[The Project 7](#_Toc386787742)

[Report Structure 8](#_Toc386787743)

[Analysis 9](#_Toc386787744)

[Background Information 9](#_Toc386787745)

[Problem Statement 9](#_Toc386787746)

[Existing techniques and solutions 11](#_Toc386787747)

[Feedback from Questionnaires 21](#_Toc386787748)

[The Solution 21](#_Toc386787749)

[Requirements 24](#_Toc386787750)

[Design 26](#_Toc386787751)

[Software Chosen 26](#_Toc386787752)

[Backend Database 26](#_Toc386787753)

[Frontend Application 26](#_Toc386787754)

[Software Version Control 27](#_Toc386787755)

[Pre-cleaned dataset 29](#_Toc386787756)

[Matching methodology 30](#_Toc386787757)

[Initial Prototype 35](#_Toc386787758)

[Database architecture 41](#_Toc386787759)

[Implementation, Testing and Evaluation 42](#_Toc386787760)

[Testing 51](#_Toc386787761)

[User Acceptance Testing 52](#_Toc386787762)

[Evaluation 52](#_Toc386787763)

[Conclusions 55](#_Toc386787764)

[Further Development 57](#_Toc386787765)

[References 58](#_Toc386787766)

[Appendices 60](#_Toc386787767)

[1. Match Rates 60](#_Toc386787768)

[2. Letter of commendation from NISRA 61](#_Toc386787769)

[3. QAS Email 62](#_Toc386787770)

[4. Questionnaire 65](#_Toc386787771)

[5. Project Code 66](#_Toc386787772)

[5. Test Suite 82](#_Toc386787773)

# Abstract

The problem is that NISRA (Northen Ireland Statictics and Research Agency, 2014) require a more user friendly, quicker and less expensive way to validate addresses in which they receive throughout the year. These addresses are used in statistical output data such as the Census and therefore all addresses need to be validated to provide the most accurate results. The process currently used to carry this out is expensive in both time and money.

The objective of this project is to design an application that will operate to the same success validation rate as the currently used software in NISRA, while providing an easy to use interface.

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) as well as Microsoft SQL Server (Microsoft, 2008). 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 the original idea was followed with the addition of some useful features. These are outlined in the report. 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 requires the results 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 addresses. This is due to the addresses being captured through forms completed by people. These forms are normally read into a database via Optical Character Recognition (OCR) software or again by humans. Both of these processes have disadvantages as a person’s handwriting may not be legible by either methods, or certain characters are not legible. Furthermore some letters can be mistaken for other letters which generate an incorrect address which is quite challenging 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 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 technicians. 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 three months. 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 wished to use a methodology which had a timeline to follow. This would ensure that the project would run smoothly as every step would be outlined and no time would be wasted trying to discover what part of the project needs to be completed next. at the beginning, a large amount of time would be spent on developing the work plan for the development of the project. This means that only small amounts of time would be required to update this plan. 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 duration 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 a project backlog is that the planned tasks enable the project to run smoothly as no time is wasted with planning. Furthermore, it enables the developer to create a prototype. This will enable NISRA to test the software and provide feedback. 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 which they will outline what tasks will be carried out during that day. Also in this process the developer will assess how the previous day’s tasks went and decide 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. Figure 1 illustrates this methodology.

****

Figure 1 Agile Time Management Model

## Aim of Project

The solution to this problem is to create an application that can conduct the same process as the current software. This will have the ability to take addresses in a dataset and test their validity by running them together with defined algorithms against an already clean address dataset and appending a Unique Property Reference Number (UPRN) to each record. 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 to ensure that it can be used by multiple users simultaneously. 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 these 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 or 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 current knowledge of 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 conducted on the development of the user interface as the back end data store was already decided on. Extensive research was therefore needed to decoded which software and programming language would operate well with the back end data store.

Throughout the project, communications with NISRA were ongoing. This ensured that all areas of the project were covered and that any complications which arose were dealt with quickly through implementing proposed solutions.

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 greater than that of the previous software (See appendix 1). 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 (see appendix 2).

## Report Structure

The remainder of this report is structured as follows:

***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 during the process 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

## Background Information

“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, but 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.

## Problem Statement

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 records from locations all over Northern Ireland. Addresses are a major part of generating statistics produced by NISRA as the addresses are used to specify locations of all persons in the datasets. Many of NISRA’s statistics can be found on their website which can be read by following the link 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.

Figure 2 shows one of the many ways NISRA use addresses in their statistics which can all be found on their website.

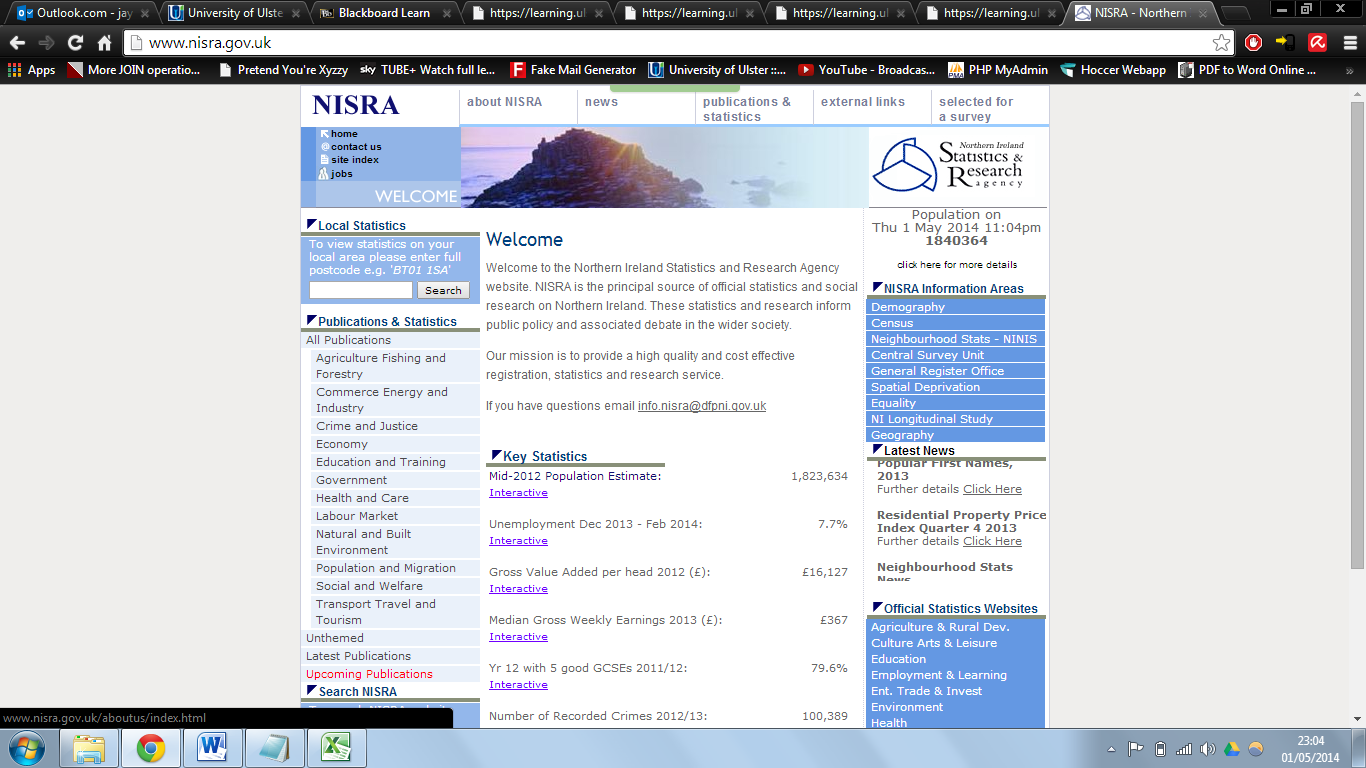
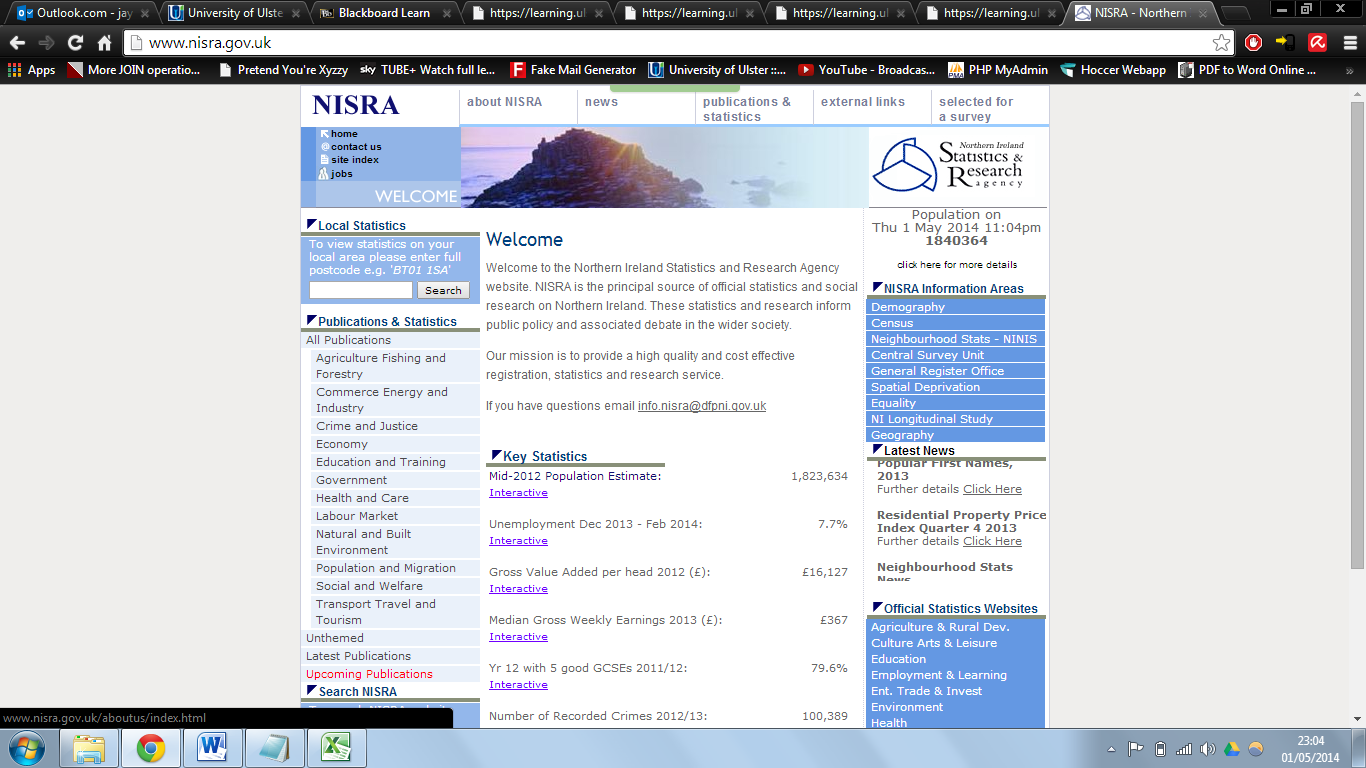


Figure 2 showing how NISRA uses addresses in their statistics

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. An example of a form that can cause human error can be found on the NISRA website (http://www.nisra.gov.uk/archive/census/2011/forms/household.pdf).

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 per staff member and each form may contain up to 20 fields. 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 number of forms or fields rise.

Two studies were conducted by Wahi, et al. (2008) who attempted to reduce errors from the electronic transcript of data collection on paper forms. During these two studies data was inputted and error rates were collected and it was found that ‘The SE (single data entry) error rates can be quite variable, and have been reported to be as low as 10.8 and as high as 124 per 10,000 fields. In one study where two SE datasets were created from the same data, 6.5% of the entered fields did not match in the two datasets. This translates to an error rate of 650 per 10,000 fields. In a study where two professional data managers conducted SE and consistency checks, error rates were lower, at 13 and 15 errors per 10,000 fields, the lower rates being attributed to the addition of consistency checks.’

When this is applied to the vast number of records NISRA process it can be seen that address validation is greatly needed within the agency.

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 can also be used on any of NISRA’s computers at any given time. The benefit of this application is that it will allow more than one NISRA staff member to use this application at the one time which will eliminate the waiting time for NISRA staff with the current application. Other benefits are that 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.

## Existing techniques and solutions

When researching existing techniques and solutions it was evident that the methodology of address validating, cleansing or matching is not publicly documented which is perhaps due to the business opportunities that the methodology creates. As there are available software applications currently on the market that can carry out address validating, cleansing and matching, and each software method is different, it is thought that research has been carried out in regards to these methods but this information is not available to the general public.

There are a number of solutions available on the market such as Address Validation (helpIT Systems, 2014) and PAF Cleansing (Data8, 2014) which do not have any indication of pricing without contacting the company directly. These mentioned software applications have the ability to carry out the process needed and also include extra features that could be beneficial to NISRA although they are not essential. These features include ‘Find new addresses with National Change of Address (NCOALink®) processing’ (Address Validation) and ‘PAF Cleansing service compares your address data against Royal Mail's PAF (Postcode Address File) and we correct any errors that we find in your file’ (PAF Cleansing).

Other existing solutions that have pricing available include PostCoder Batch (Allies, 2014) and Data Cleansing (PostcodeAnywhere, 2013). Currently the pricing for PostCoder Batch is £1575 for use on one computer for 12 months and this does not include a record limit which could possibly incur more charges. Data Cleansing is currently priced at £500 for 91,000 records over a 12 month period which is very limited for the amount of records that NISRA process therefore the actual price for NISRA would be greater. None of the above applications have included pricing information for pre-cleansed datasets which could be added on to the current price, increasing the total cost for each application. As these applications require payment before use it is difficult to assess the reliability of the software for the problem at hand.

There is an existing standalone application named Experian Quick Address (QAS) Batch (Experian, 2014) which NISRA are currently using and 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 appendix 3 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. The application even boasts to 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 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.

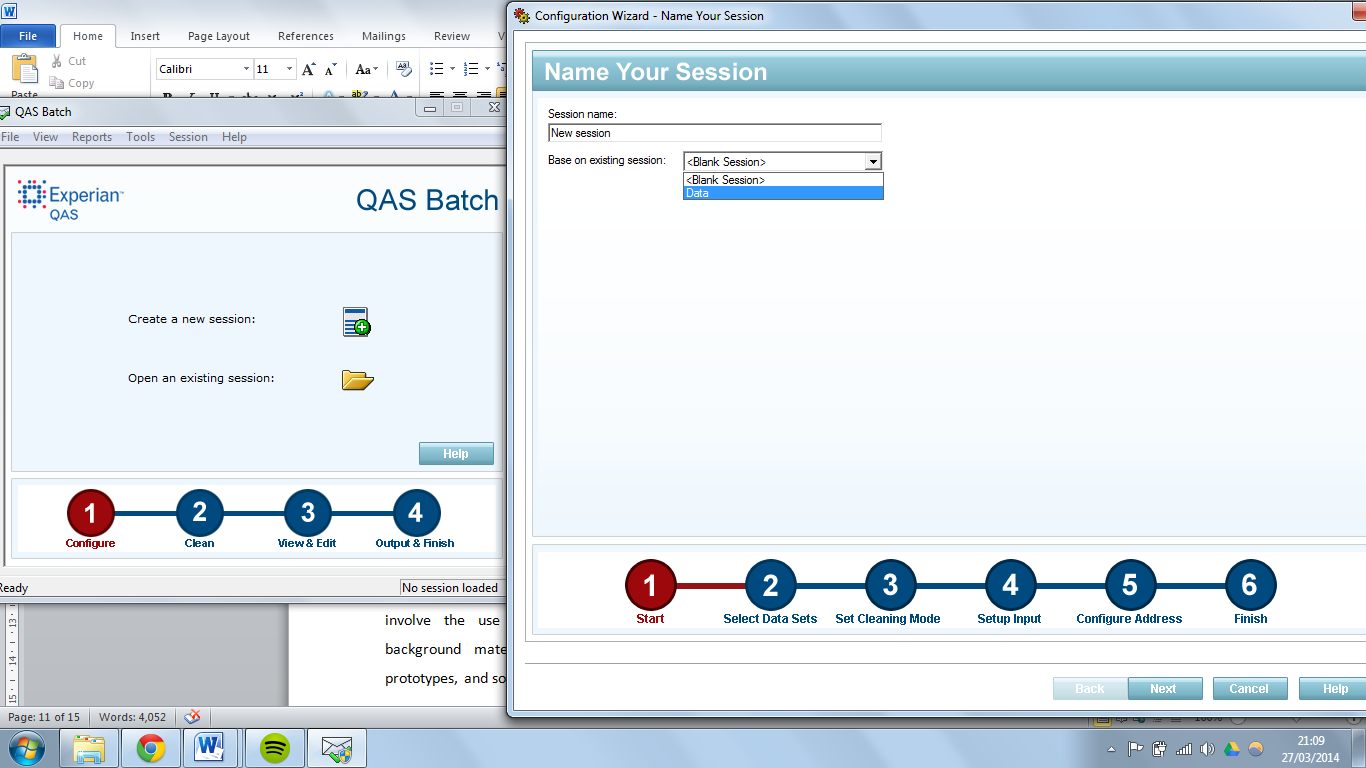


Figure 3 showing the opening screen of QAS Batch

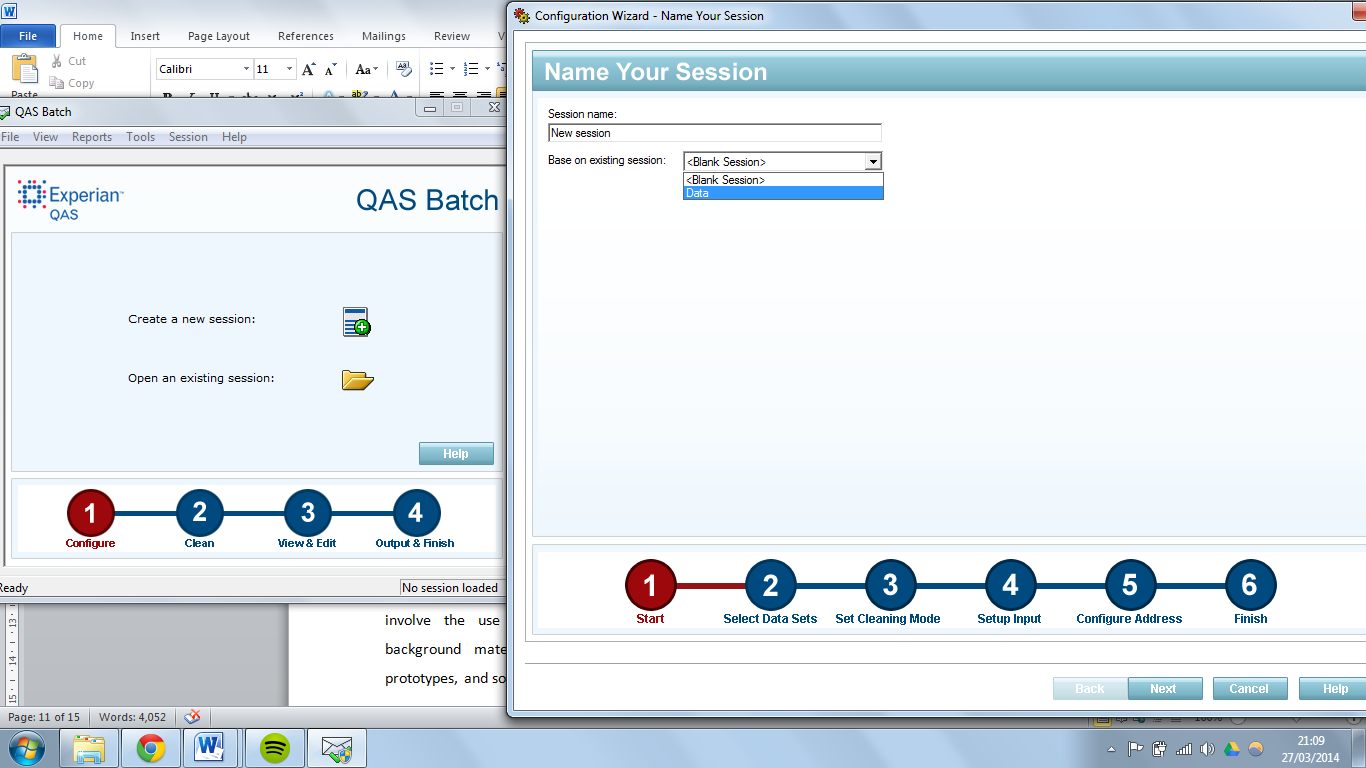
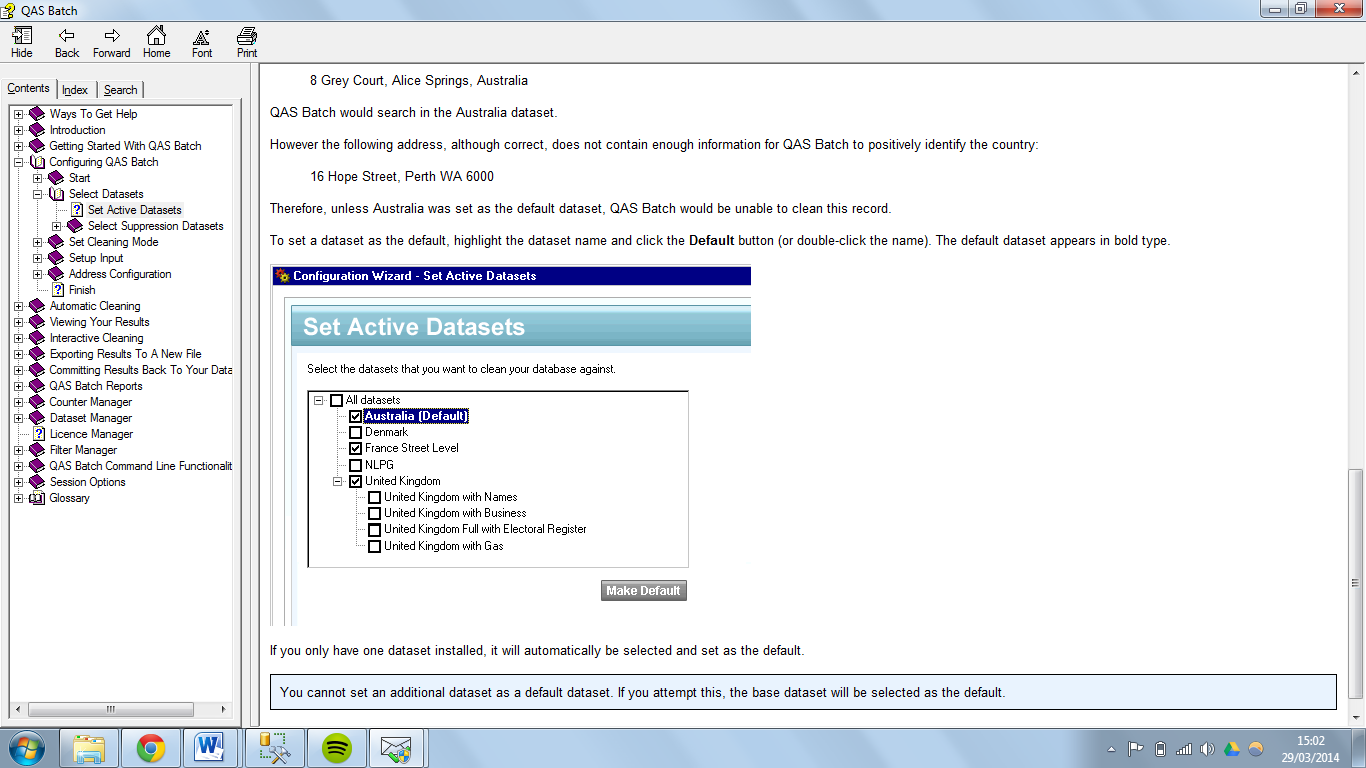
We can identify some of the features that QAS Batch offers from the above screen shots. In figure 3 you can see that you have two options to choose from, ‘Create a new session’ or ‘Open an existing session’. The feature of opening an existing sessions is a very useful feature if you have different settings 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 4 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.

Figure 4 when you create a new session.

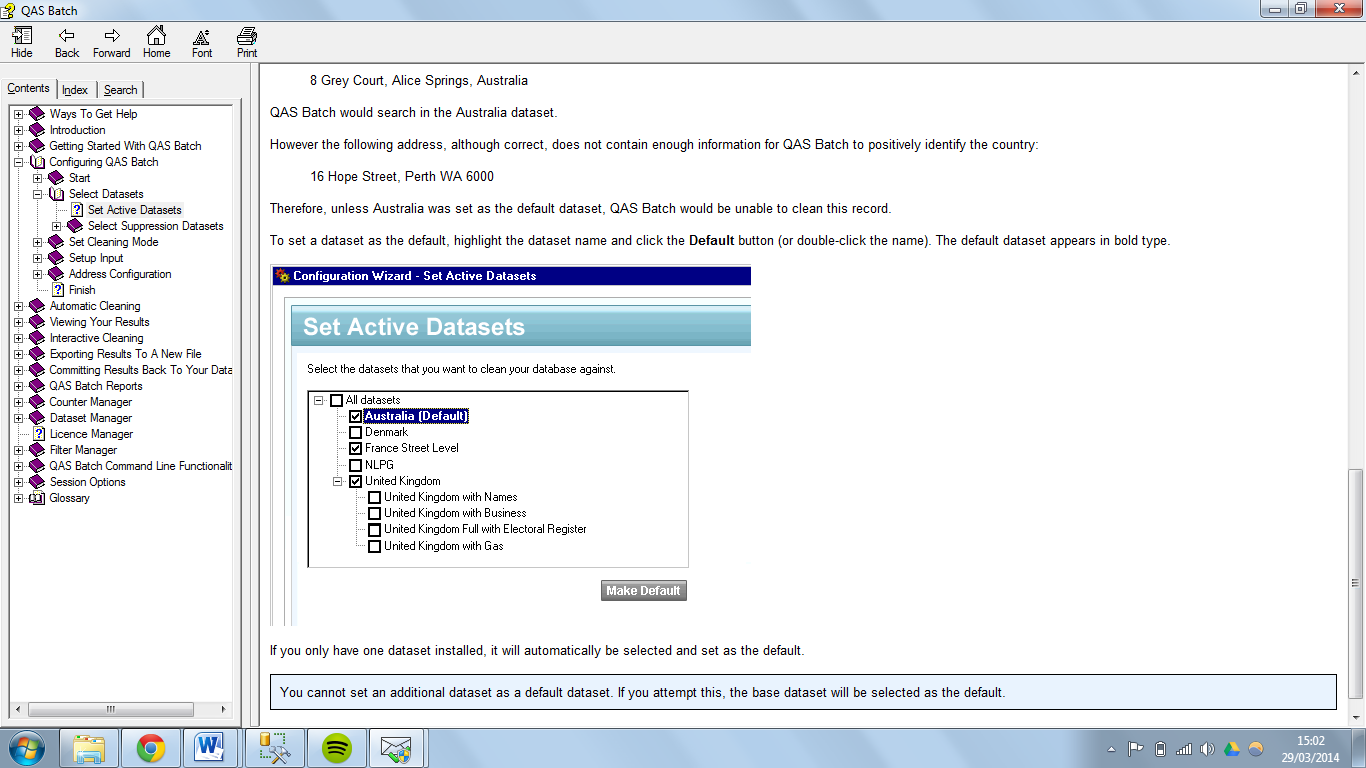
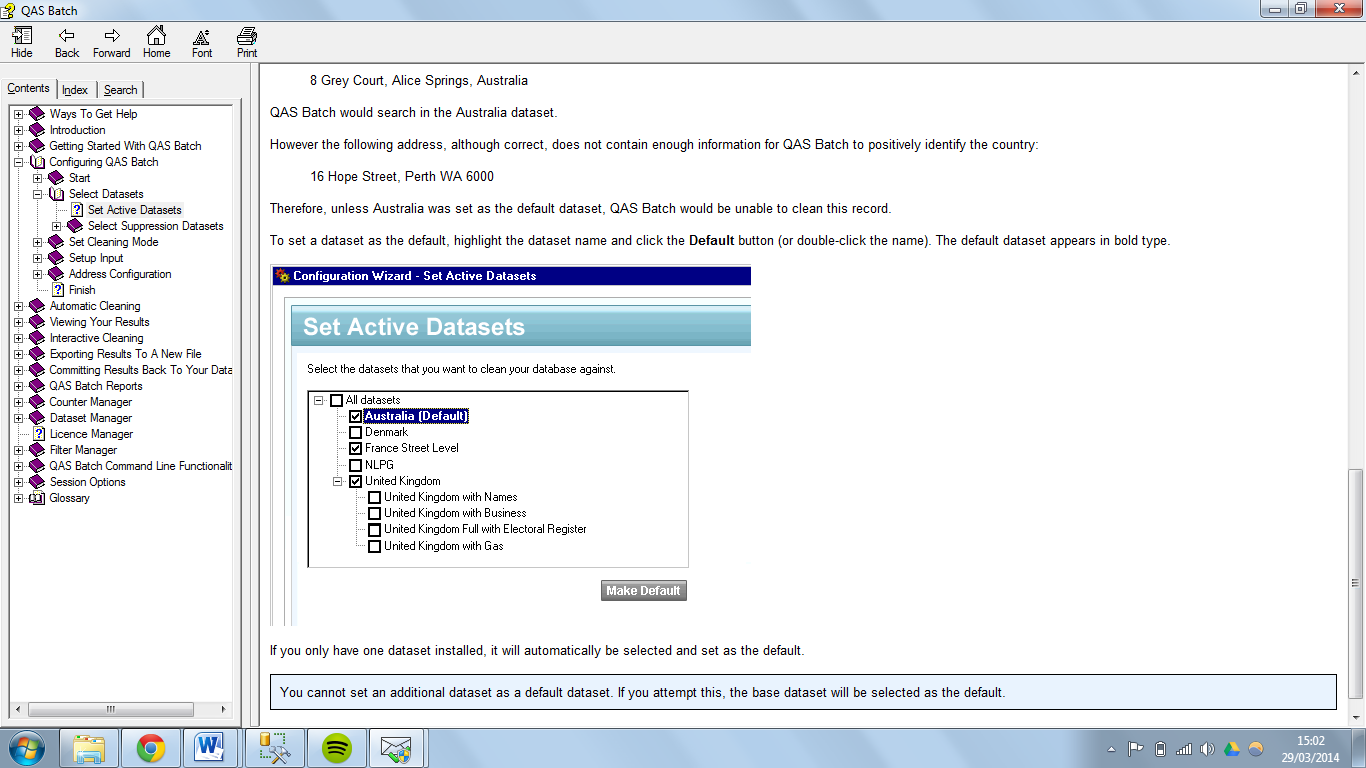
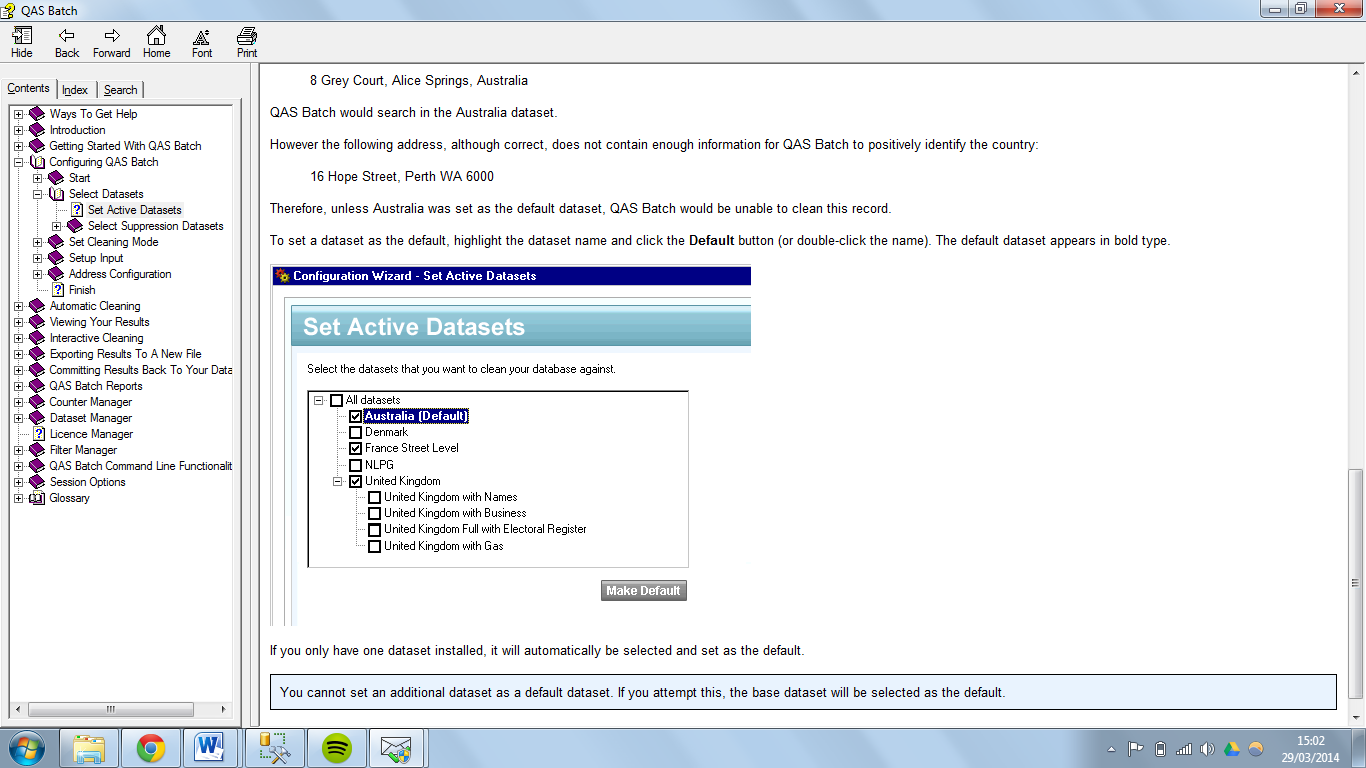
Although these features are useful, they are not useful to NISRA as they only have one process that has to be configured and it is a basic configuration therefore this feature is not needed. In both figures 3 and 4 it can be seen at the bottom of the screenshots that there are numbers to specify what 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 5 Selecting the active datasets to use to cleanse the user dataset.



Australia

United Kingdom with Pointer (Default)



From figure 5 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 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 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.

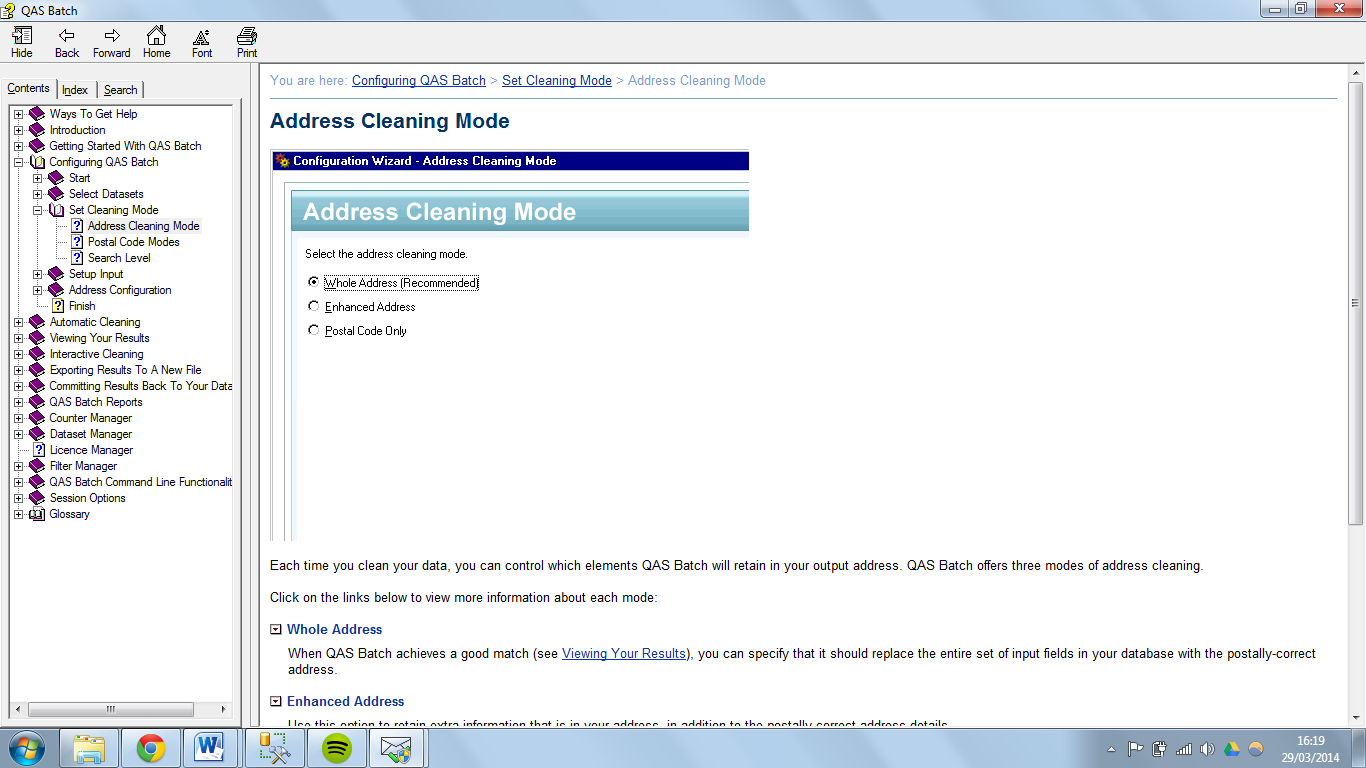


Figure 6 Choosing the address cleaning mode.

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 6 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.

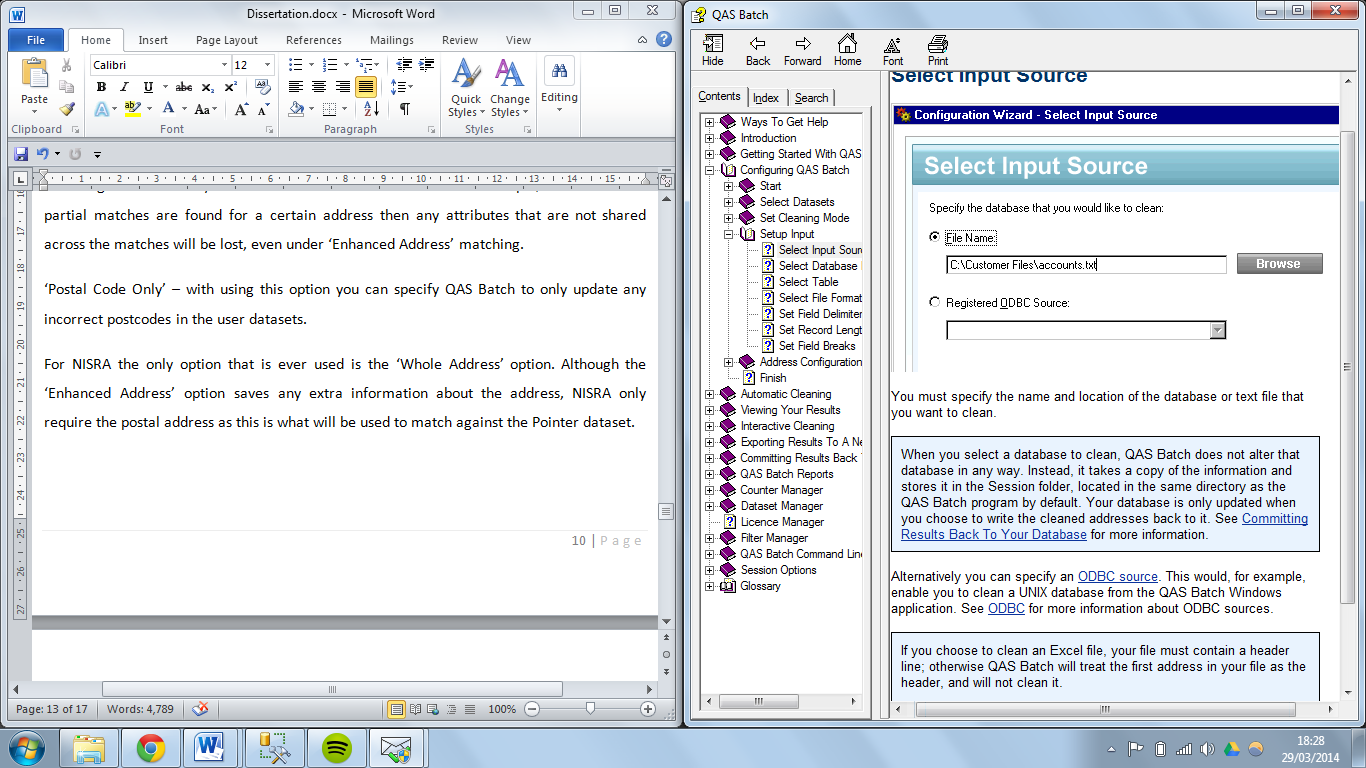
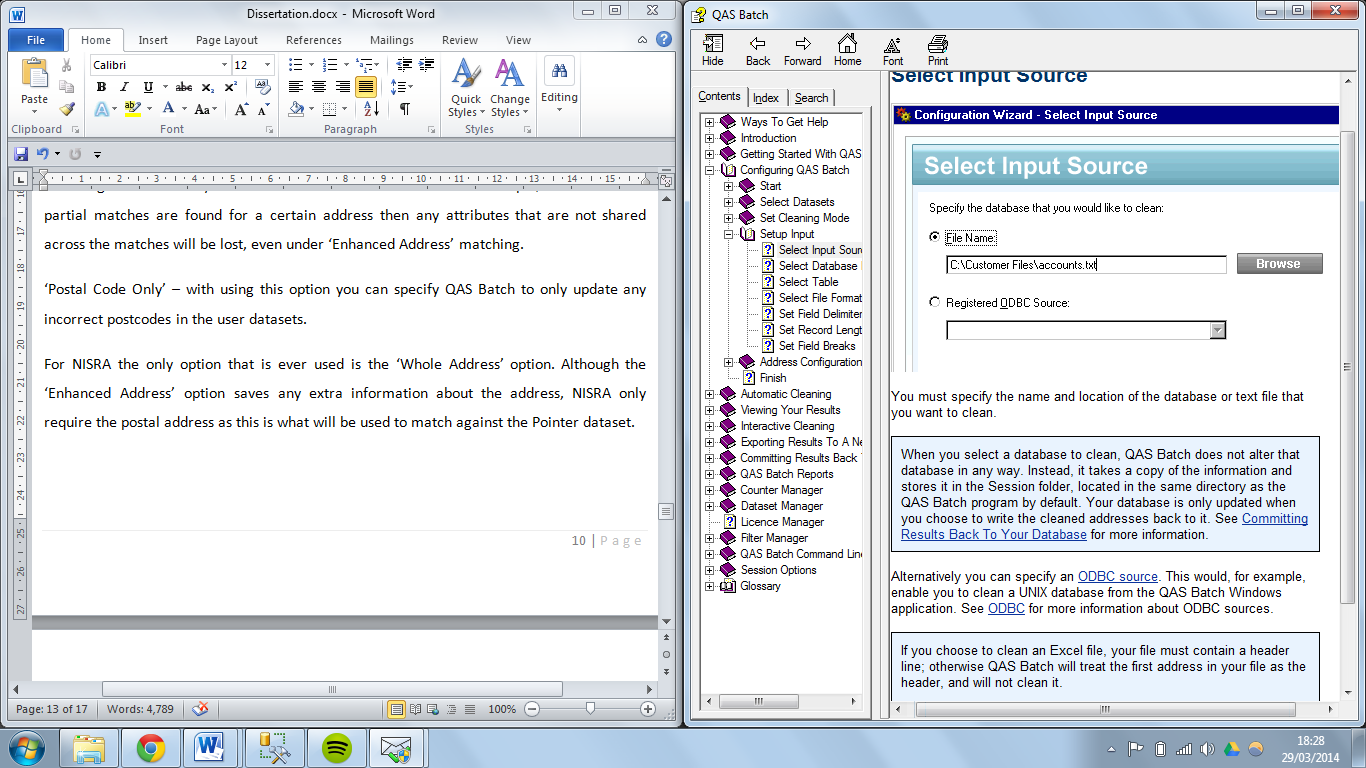


Figure 7 Specifying the location of the database to be cleaned.

When choosing an input source you have two choices as shown in figure 7, 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.

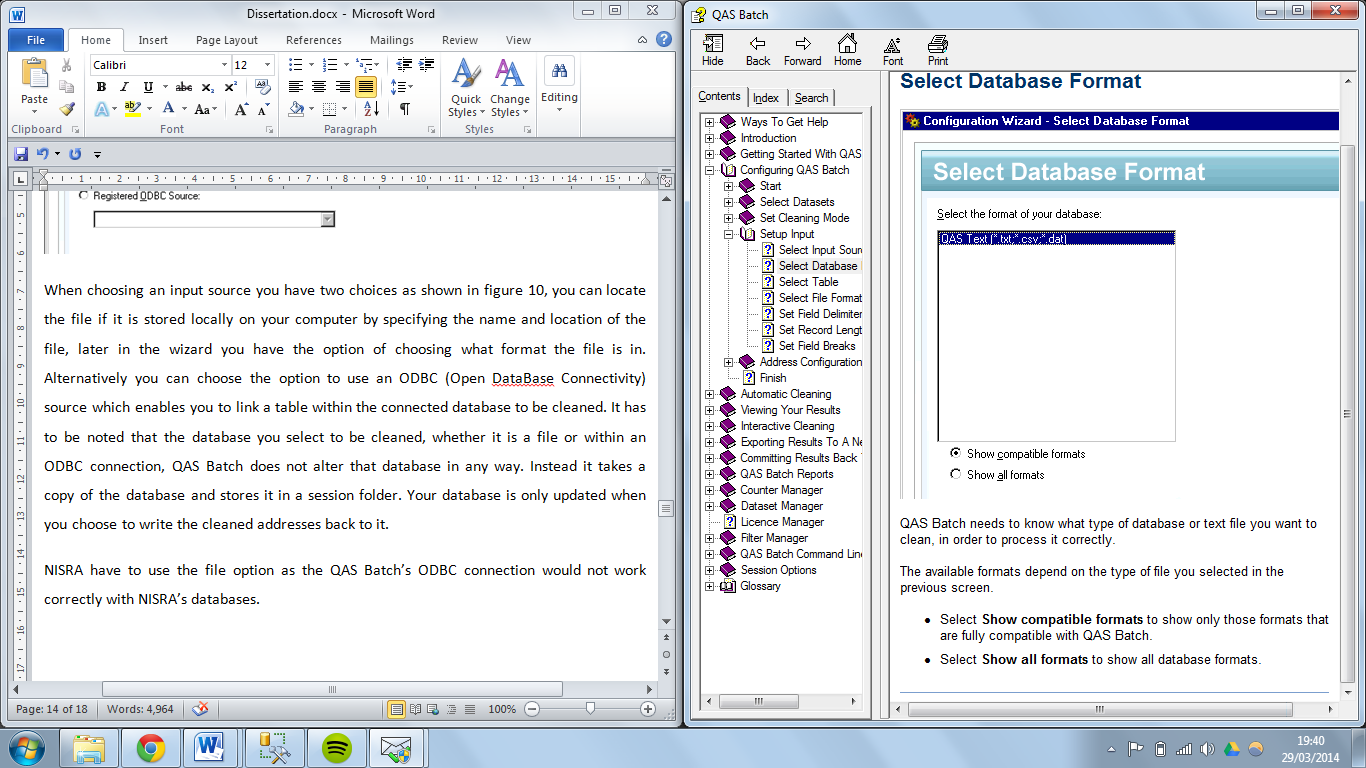
NISRA have to use the file option as the QAS Batch’s ODBC connection would not work correctly with NISRA’s databases.

Figure 8 Selecting the format of the database file.

As mentioned above you have to choose the format of the file that you have selected (figure 8). 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.

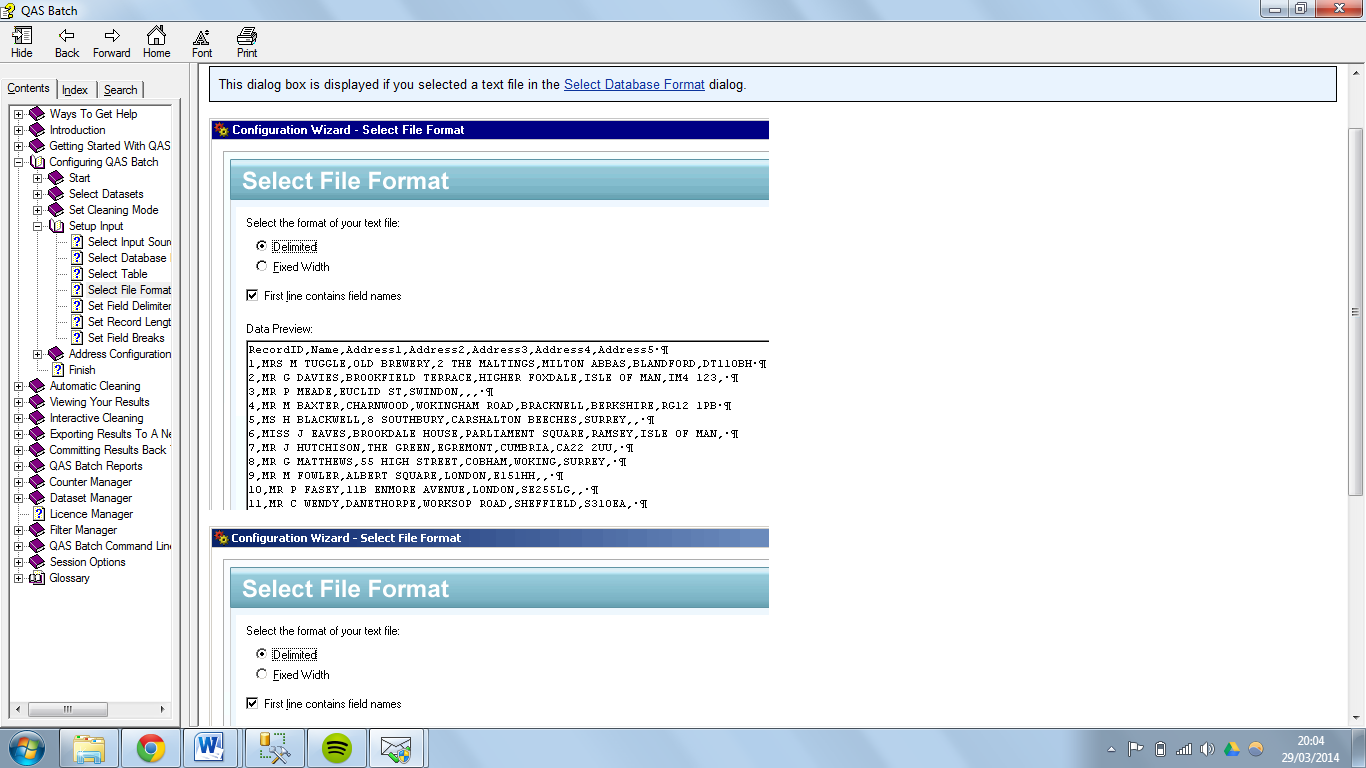
If the database file that you chose contains more than one table you will have to specify which table that you want to cleanse. QAS Batch will display a preview of the tables attributes and a preview of the content in each attribute in the database to allow for easy identification of the table you wish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

Figure 9 configuring the format of the text file.

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

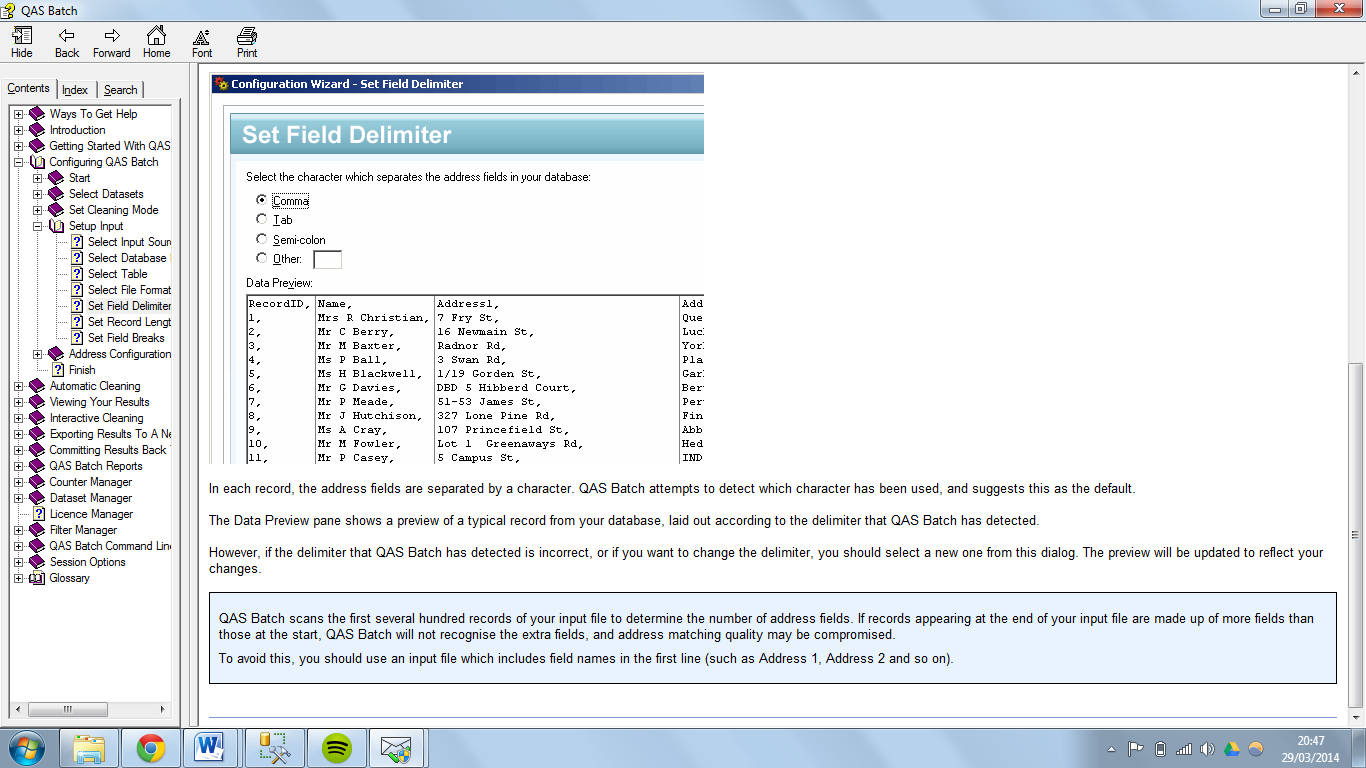
When presented with the screen shown in figure 9 you are able to choose the format of the table from your file you selected is in. The options are ‘Delimited’ which means the file has fields that are separate by a specific character which, if chosen, will be declared later in the process. The other option 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.

Figure 10 selecting the character that separates the fields in the database.

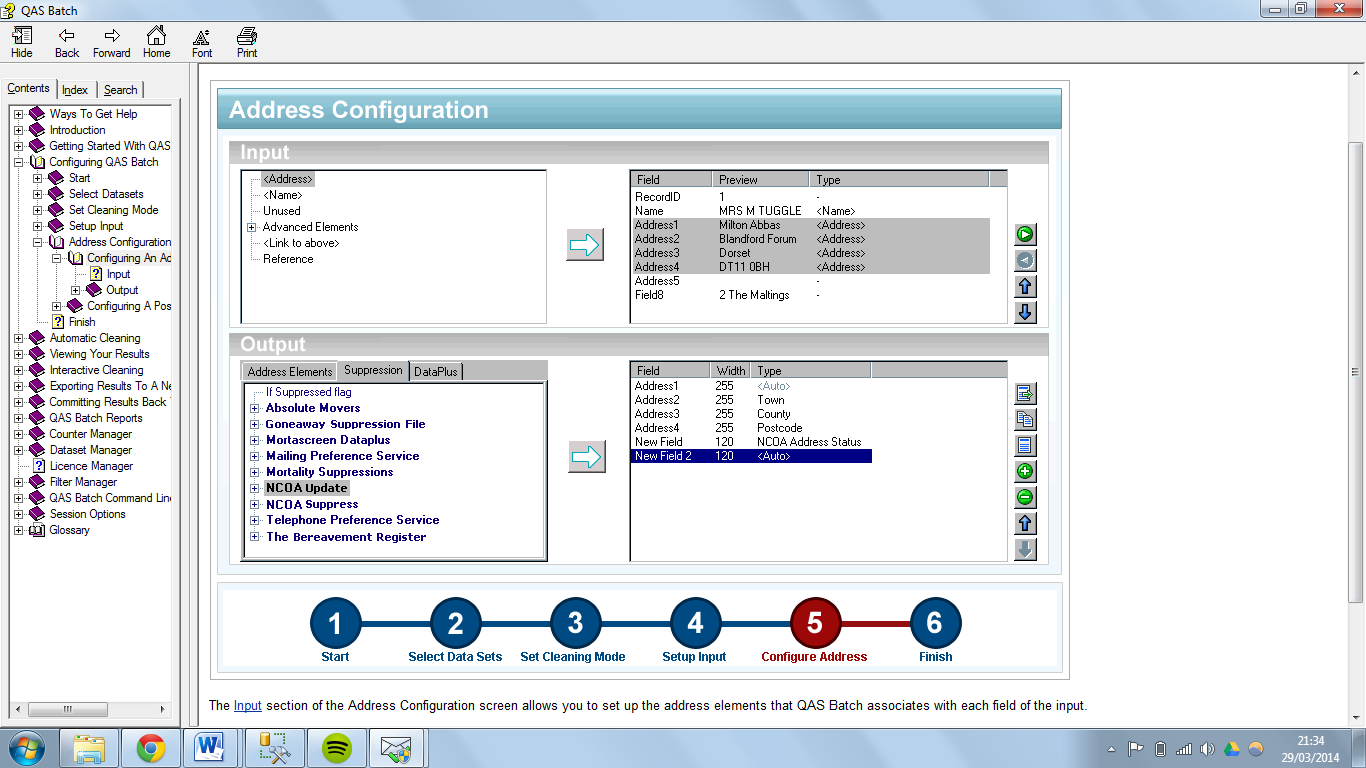
This step of the configuration, shown in figure 10, allows you to specify the character that will separate the fields in the table of the database. As you can see there are a few default options that would be most used and the ability to define your own character.

Figure 11 configuring the input and output address elements.

(Note not actual data. For explanation purposes only.)

This part of the process, as shown in figure 11, is a very important part of the process as depending on how accurate you are with configuring both the input and output elements the accuracy of the cleansing process will improve. 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 11 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.

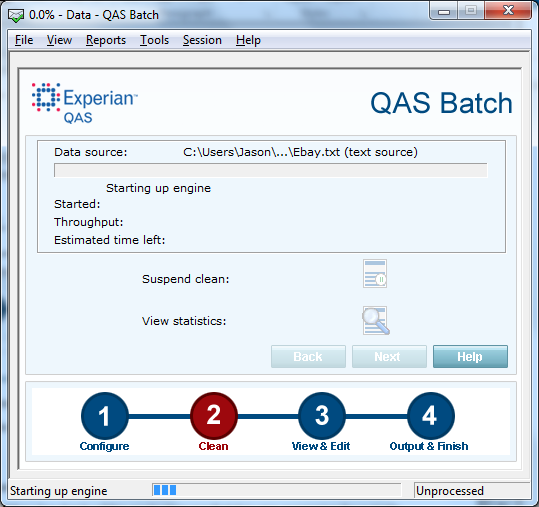
This feature is a very smart feature if used correctly and when used correctly can save the user a substantial amount of cleaning time. Also the way in which the input elements are configure impacts how well the data will be cleaned, so the better the configuration the better the cleaning. The use of 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.

Figure 12 progress screen after the configuration is complete the cleaning has started.

As you can see from figure 12 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.

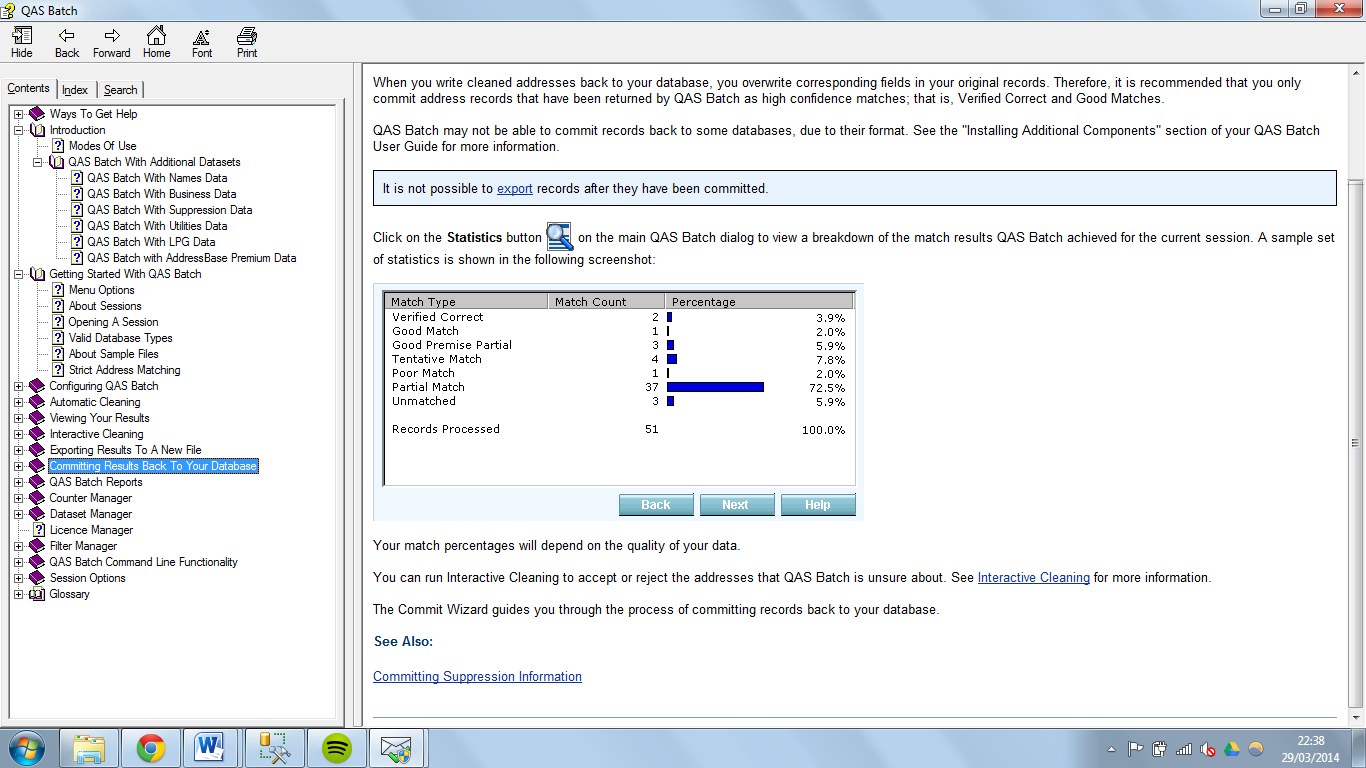


Figure 13 viewing the statistics of the cleaning process

This screen shown in figure 13 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. These features will be noted and will be considered when developing a solution. 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 were presented with questionnaires (sample questionnaire see appendix 4) 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 required them to press as few buttons as possible 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 applications. 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. This UPRN can then be used to alert the dataset users to valid address and records with vacant UPRN fields are invalid addresses.

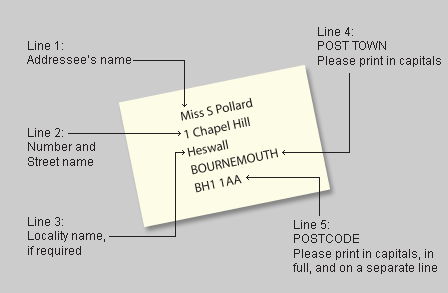
The matching process for the existing application is not available to view for licensing reasons so therefore it is proposed to create a new matching system which will require the use of what the developer is calling ‘MatchKeys’. These matchkeys will be created using an ‘Unique Postcode Identifier’ (UPCID) which will be made up of different parts of the address. 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. Figure 14 shows the depiction of a full and accurate address by Royal Mail (Royal Mail Group Ltd., 2014).

Figure 14 Full and correct address

An Informatica company named AddressDoctor (AddressDoctor, 2014) displays the correct format for a United Kingdom address as shown in figure15.

Figure 15 Correct format for UK address

Format

RECIPIENT  
[FLOOR] [APARTMENT]  
[BUILDING]  
[HOUSE\_NUMBER] STREET\_NAME  
[DEPENDENT\_LOCALITY]  
LOCALITY  
POSTAL\_CODE  
UNITED KINGDOM

From figures 14 and 15 several important features of an address can be learnt. There can be up to 8 components of an address, 3 of which are compulsory, which are house number, street name and postal code. Without these components of an address the address is invalid as it would be near impossible to locate. The postcode is a major component within the address as by using this you can narrow down the location of an address to with several miles, used together with the street name and finally the house number you will eventually locate the addresses building. When structuring a dataset for validating consideration must be taken in regards to making sure a postcode is present as well as assuring the house number and street name are together in one single field. The remainder of the components if available will increase the accuracy of a match as they provide more information on the location of an address.

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

`

Figure 16 User login

* 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.
* If a user does not have a username and password for MS SQL Server they should not be able to use the application.
* 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.

Figure 17 User matching process

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 |

Figure 18 Example of dirty address

The above example (figure 18) data shows how simple mistakes can give an incorrect address, although if these two addresses were given separately the differences would be unseen and there possibly deemed as correct addresses. It can be seen that the ‘Address1’ 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.

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. The main component of the upcids is the address postcode, and this is because the postcode is the part of the address that can narrow down the match results the most. There can be several streets with the same name across all of Northern Ireland as an example shows in figure 19.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address Field 1 | Address Field 2 | Address Field 3 | Address Field 4 | Postcode |
| 9 Bachelors Walk | Portadown | Craigavon | Co. Armagh | BT63 5BQ |
| 9 Bachelors Walk | Lisburn | Belfast | Co. Antrim | BT28 1XJ |

Figure 19 Example of same street name

These addresses have the same house number and street name but different postcodes so therefore to differentiate between identical house numbers on identical streets, the postcode would need to be used. The other components could be used to determine the difference between the two addresses but it is common for those components not to be present in certain datasets.

As the focus of a upcid is the postcode the other components can be used to create specific matchkeys that can be used when matching to allow for mistakes in addresses or missing components.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address Data | | | | | | | | | | | | |
| Address1 | Address2 | Address3 | Address4 | Postcode | UPCID\_N\_WP | | | UPCID\_N\_CS\_NP | | … | UPRN | |
| 65 Castle Street |  |  | Armahg | BT12 3AB | 65BT123AB | | | 65CastleStreet123 | |  |  | |
| Pointer Data | | | | | | | | | | | | |
| 65 Castle Road |  | Tandragee | Armagh | BT12 3AB | | 65BT123AB | 65CastleRoad123 | |  | | | 12345 |

An example of a upcid can be seen below in figure 20 using the same sample data from above.

Figure 20 Example of possible UPCID

Components of all 14 UPCIDs can be seen below in figure 21

|  |  |
| --- | --- |
| *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)** |

Figure 21 Description of UPCIDs

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.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address Data | | | | | | | | | | | |
| Address1 | Address2 | Address3 | Address4 | Postcode | UPCID\_N\_WP | | UPCID\_N\_FC\_NP | … | | |  |
| 65 Castle Street |  |  | Armahg | BT12 3AB | 65BT123AB | | 65C123 |  | | |  |
| Pointer Data | | | | | | | | | | | |
| 65 Castle Road |  |  | Armagh | BT12 3AB | | 65BT123AB | 65C123 | |  | 12345 | | |
| 65 Cornation Street |  |  | Armagh | BT12 3TY | | 65BT123TY | 65C123 | |  |  | | |

It can be said that due to dirty data that this method can create incorrect matchkeys for an address which could pose the threat of an incorrect UPRN. This would be due to the dirty data creating a upcid that matched an upcid in the pointer table which the addresses are completely different. This can be explained by figure 22.

Figure 22 Identical MatchKey, Different Address

Notice that although UPCID\_N\_WP are different, UPCID\_N\_FC\_NP are identical even though the addresses are different. To overcome this problem the matchkeys will only be run one at a time and only on the records that UPRN has a null value. This will mean that no UPRN will be overwritten by the next matchkey. It will also mean that order of the UPCIDs will be important so that the most effective matchkey is ran first.

The code to populate these UPCIDs can be found in appendix 5, 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, 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.

Figure 23 Pseudo code

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 (figure 23) 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.

Figure 24 Pseudo code

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 (figure 24) 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).

Figure 25 Pseudo code

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 (figure 25) 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.

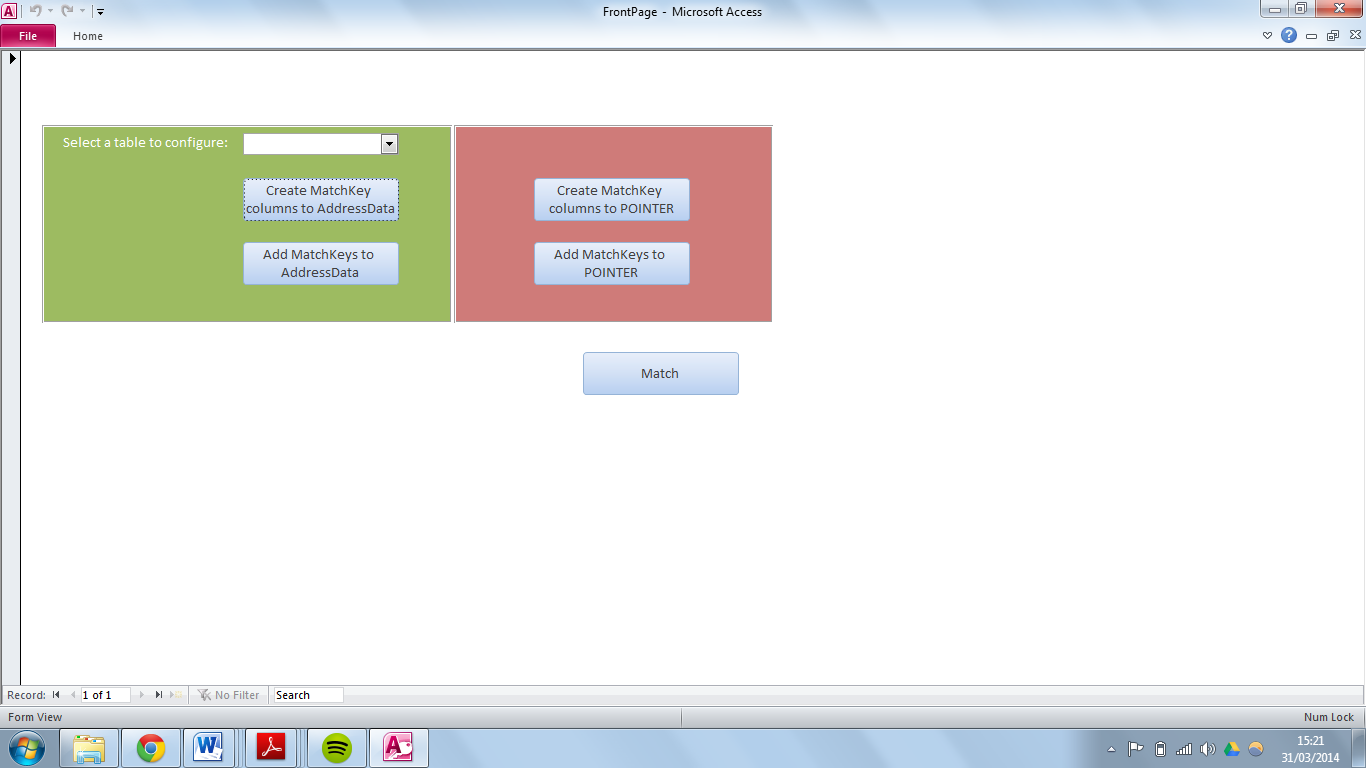
Initial Prototype

Figure 26 showing the initial prototype of the project

Figure 26 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 27 showing the start screen.

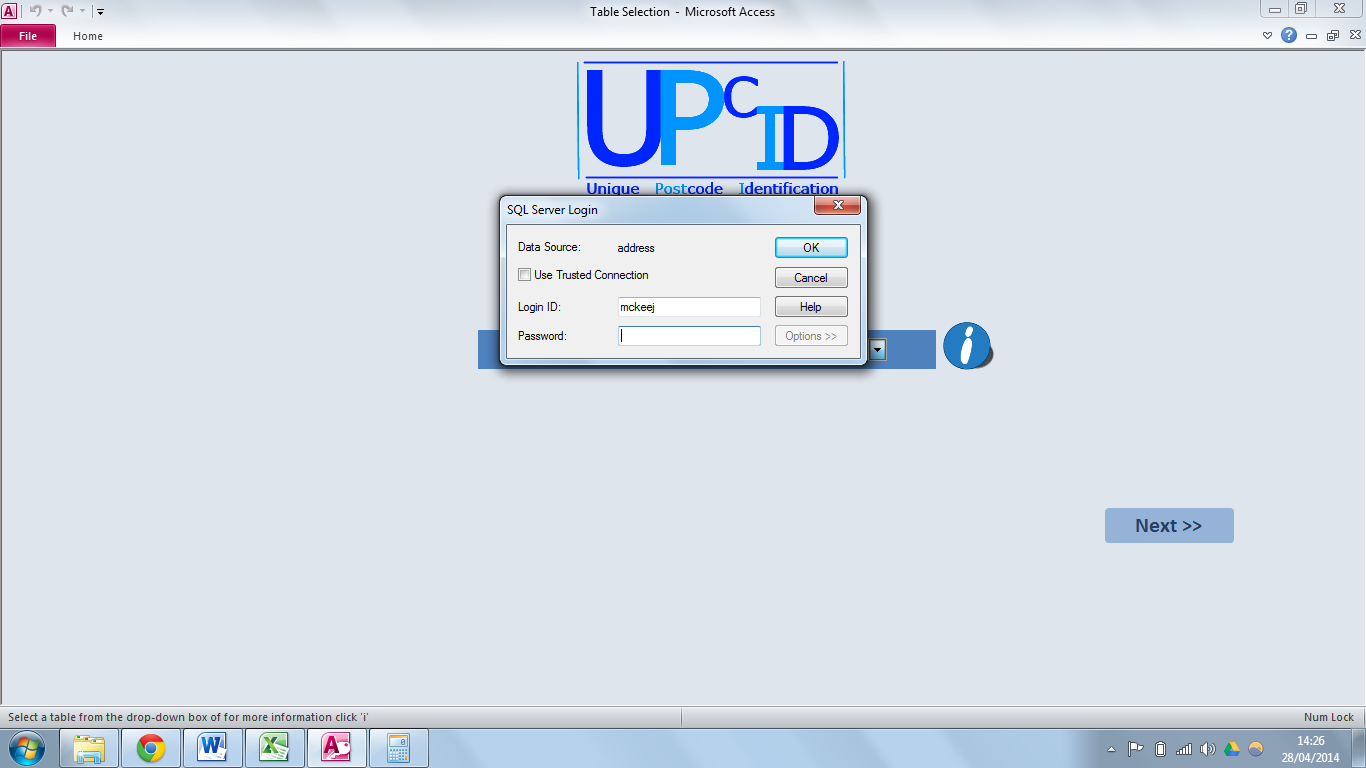


Figure 28 showing the login prompt.

Figure 27 shows 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 28) 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

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 29 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 30 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 29 showing the information window

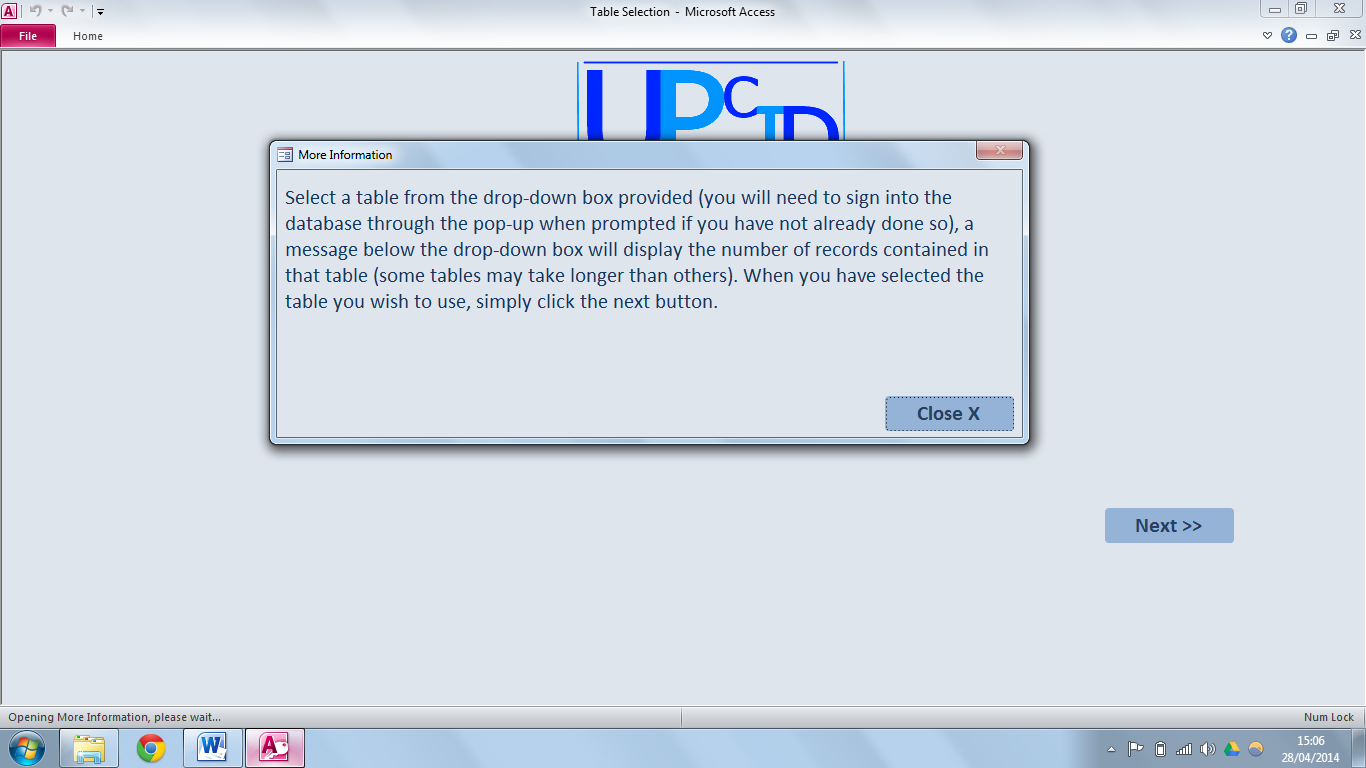
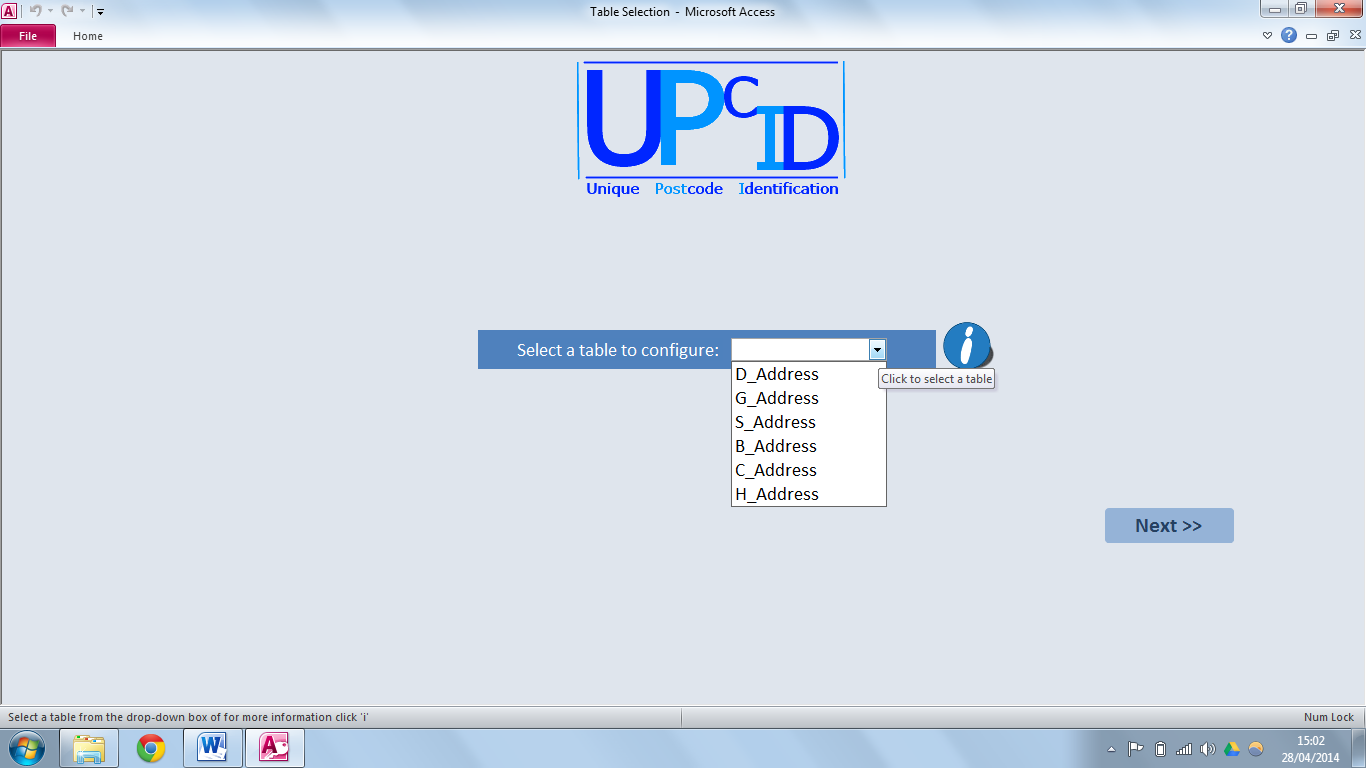


Figure 30 showing instructions for user

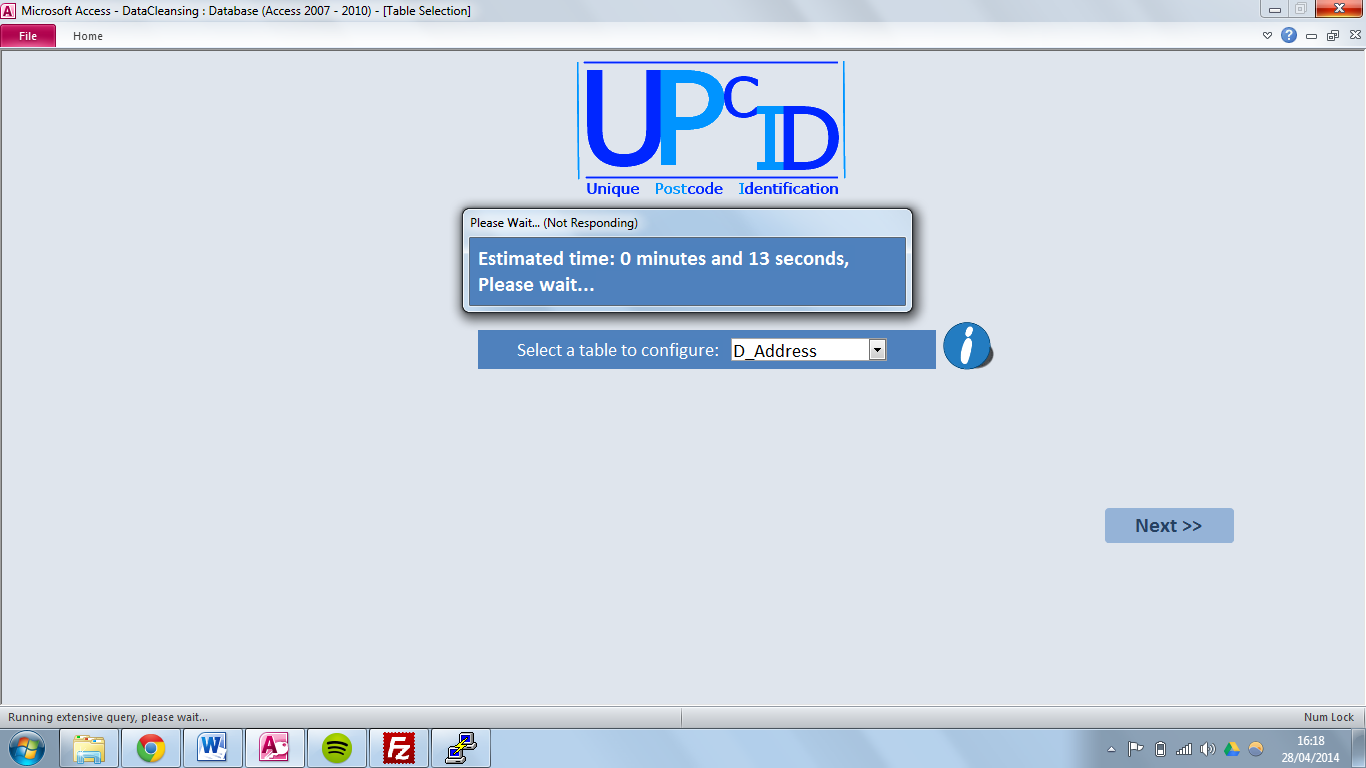
The form shown in figure 31 is a very useful feature implemented into the application. This form will display an estimated time that the current process will take to complete. The estimation is the time that the process took to run the previous time it was executed. This information is stored in a table 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.

Figure 31 showing Please Wait form

Figure 32 showing the selection screen

Figure 32 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 32 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 a 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 32 will be the only UPCIDs available to choose from in the population process as shown in figure 33 as they are the only UPCID columns in the selected table.

Figure 33 showing the selection screen during a different process

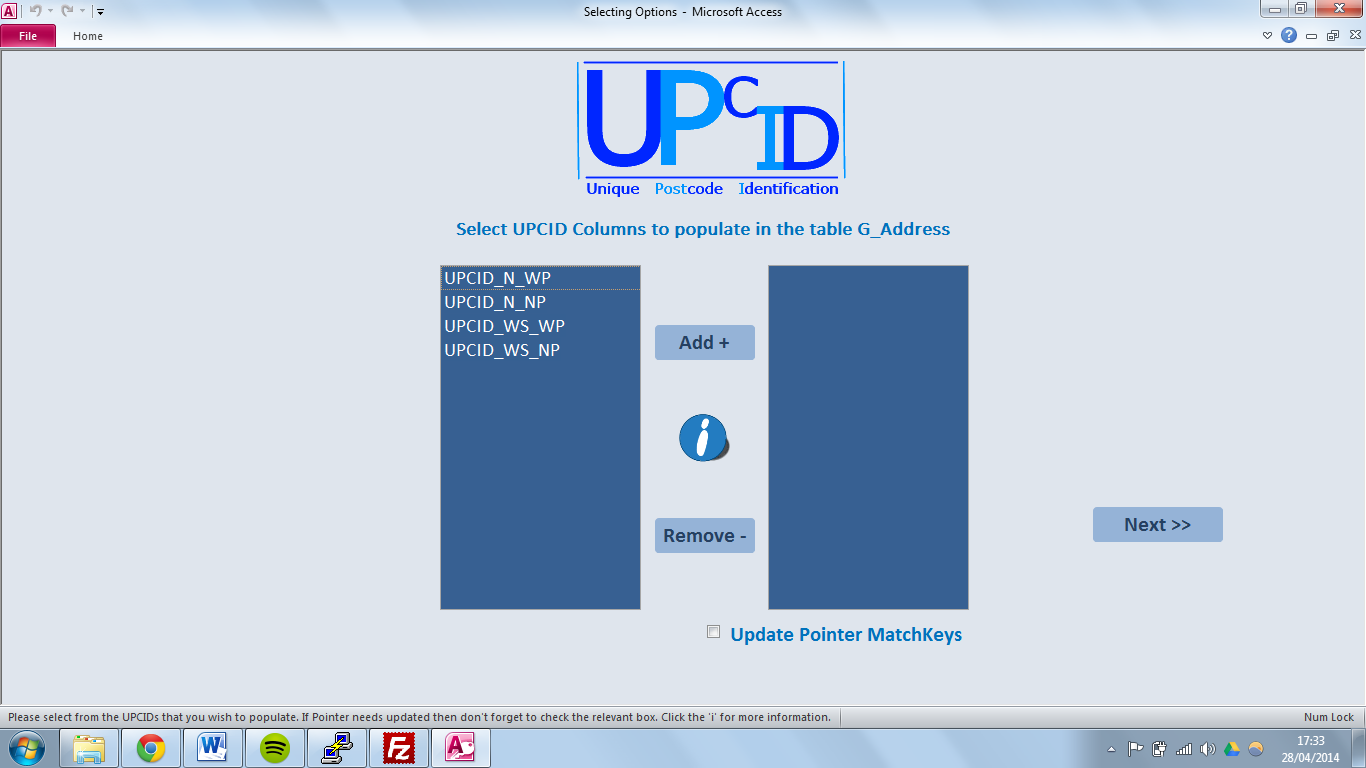
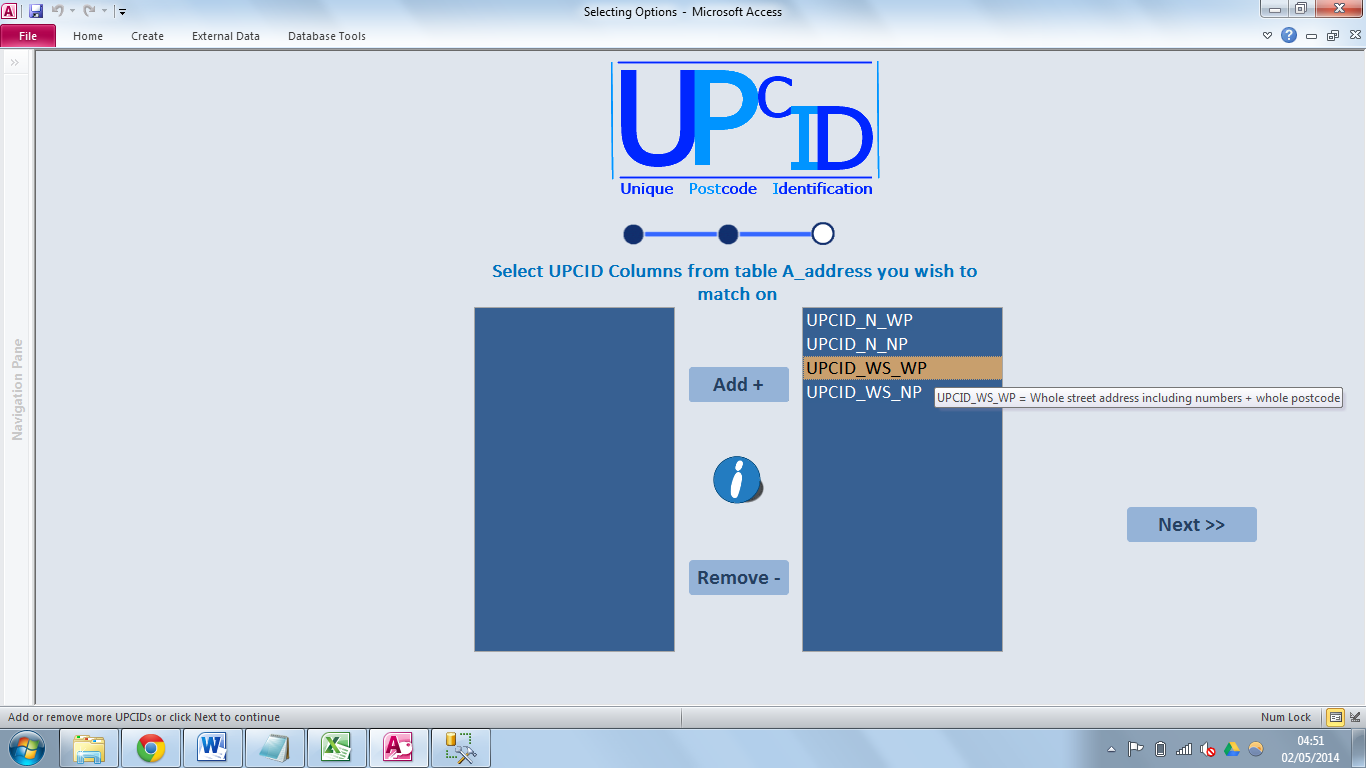
There is a check box at this stage of the process that you can tick if you want to update the UPCIDs in the Pointer table. This check box is only available to tick if the UPCIDs in the Pointer table have not been updated in the past 30 days. This is to allow for the updated Pointer file to be read into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

Figure 34 showing the user output choice screen

Figure 34 shows the use of controlText tip to allow the user to know what address components the UPCID is made of.

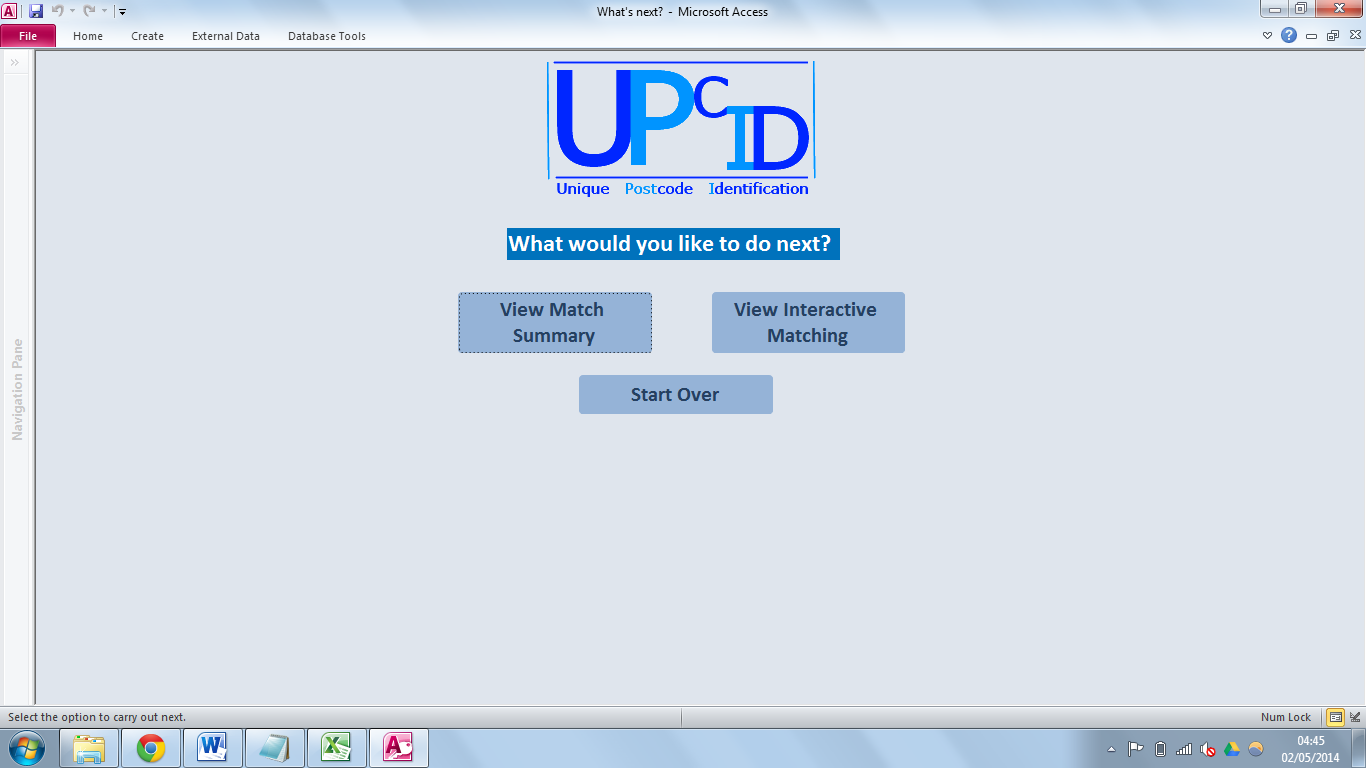


Figure 35 showing the user output choice screen

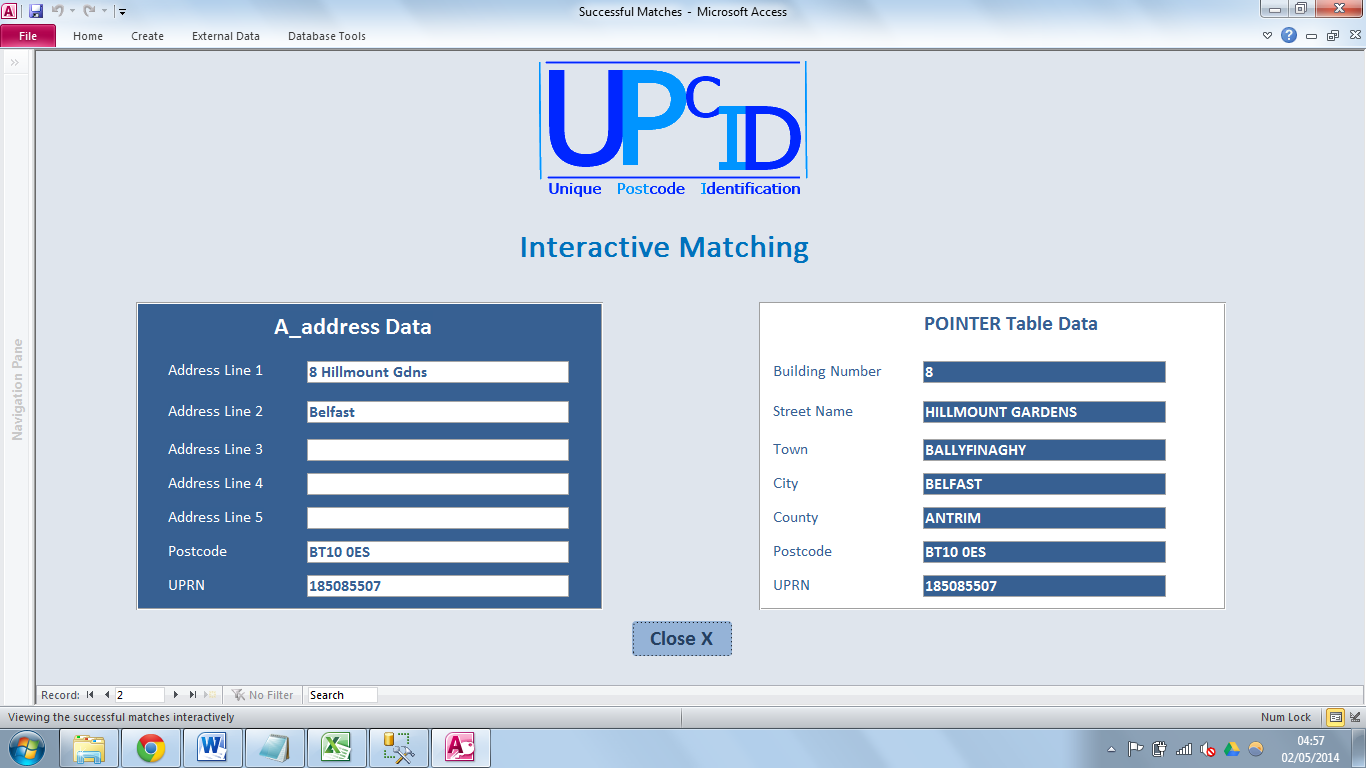
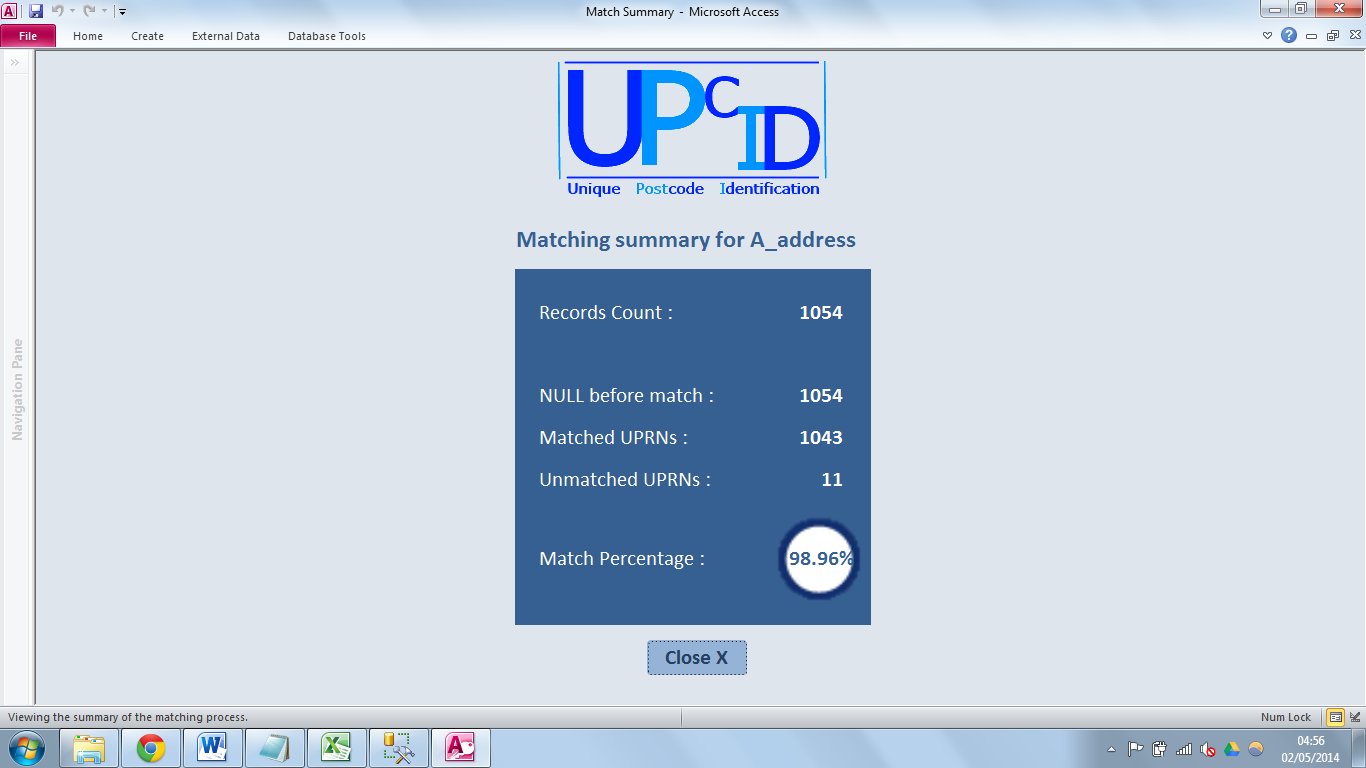
Figure 35 shows the screen the user is presented with when the match process is over. Here they can choose to view the match summary statistic (figure 36) or view the interactive matching screen (figure 37) or start over and return to choosing a table.

Figure 37 showing the interactive matching screen

Figure 36 showing the match summary screen

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.

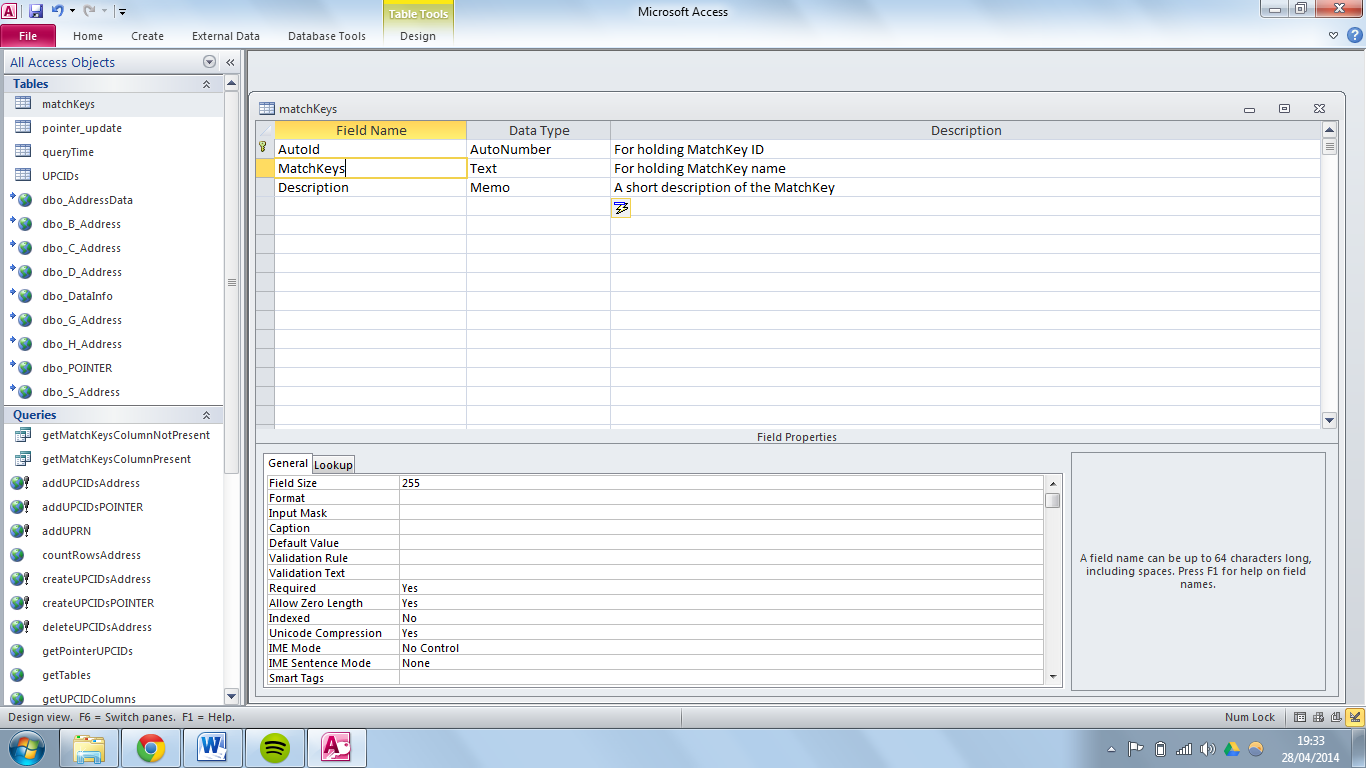
For some of the functions within MS Access other tables needed to be created and the architecture for these is shown below.

Figure 39 showing the architecture of the pointer\_update table

Figure 38 showing the architecture of the matchKeys table

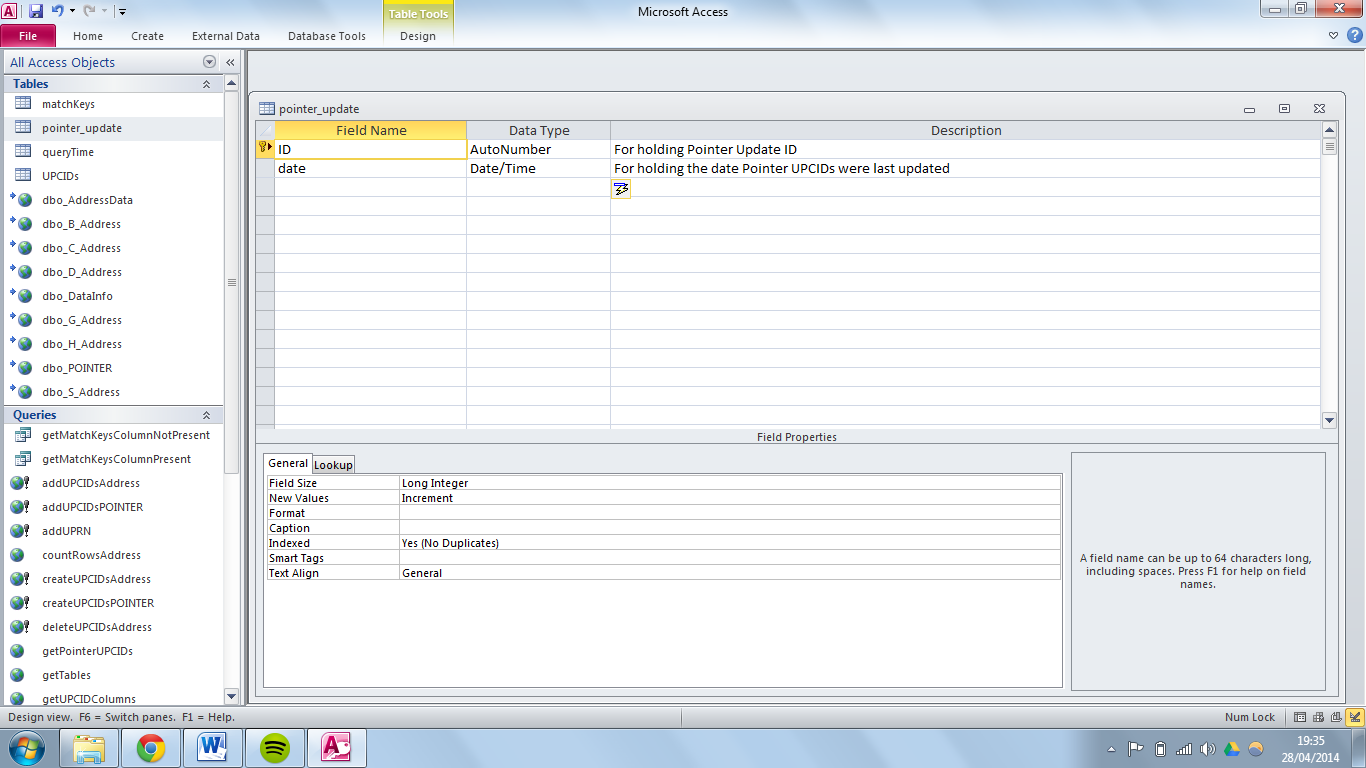
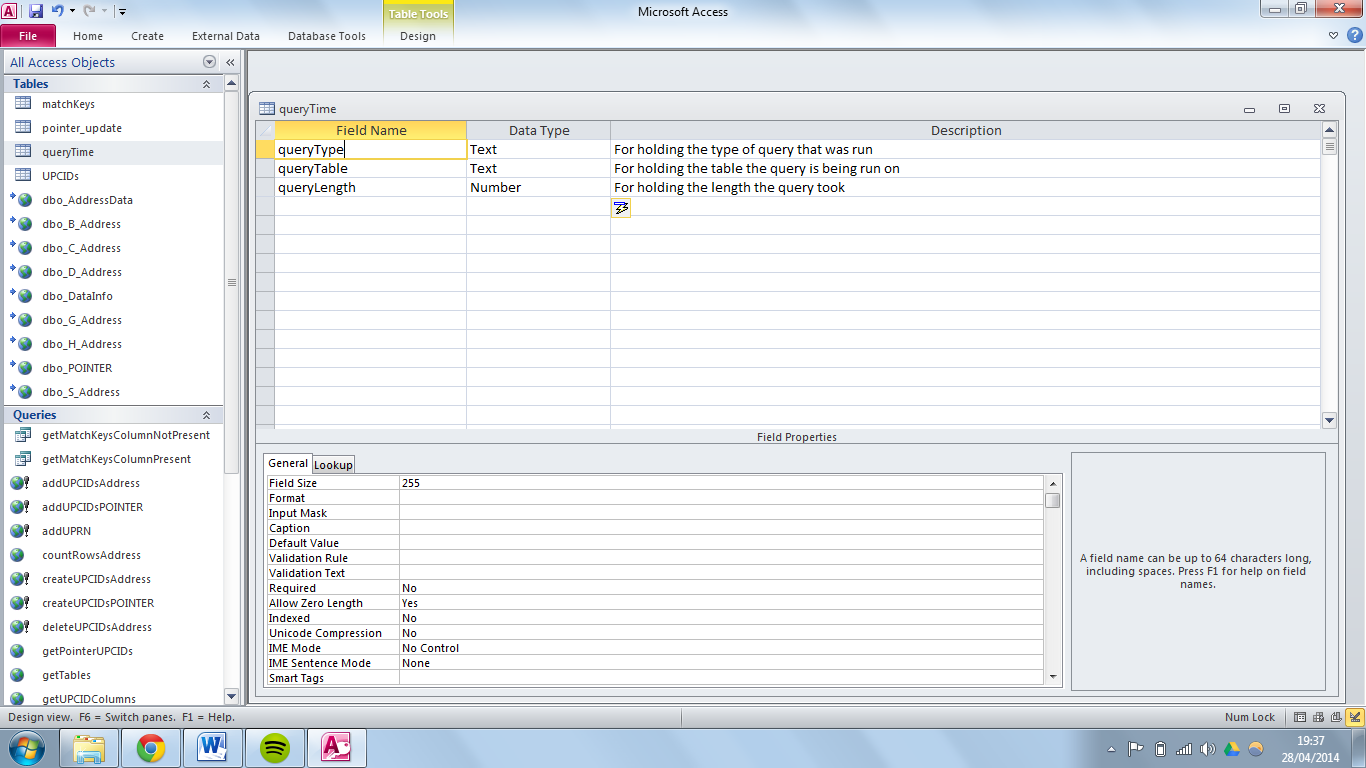
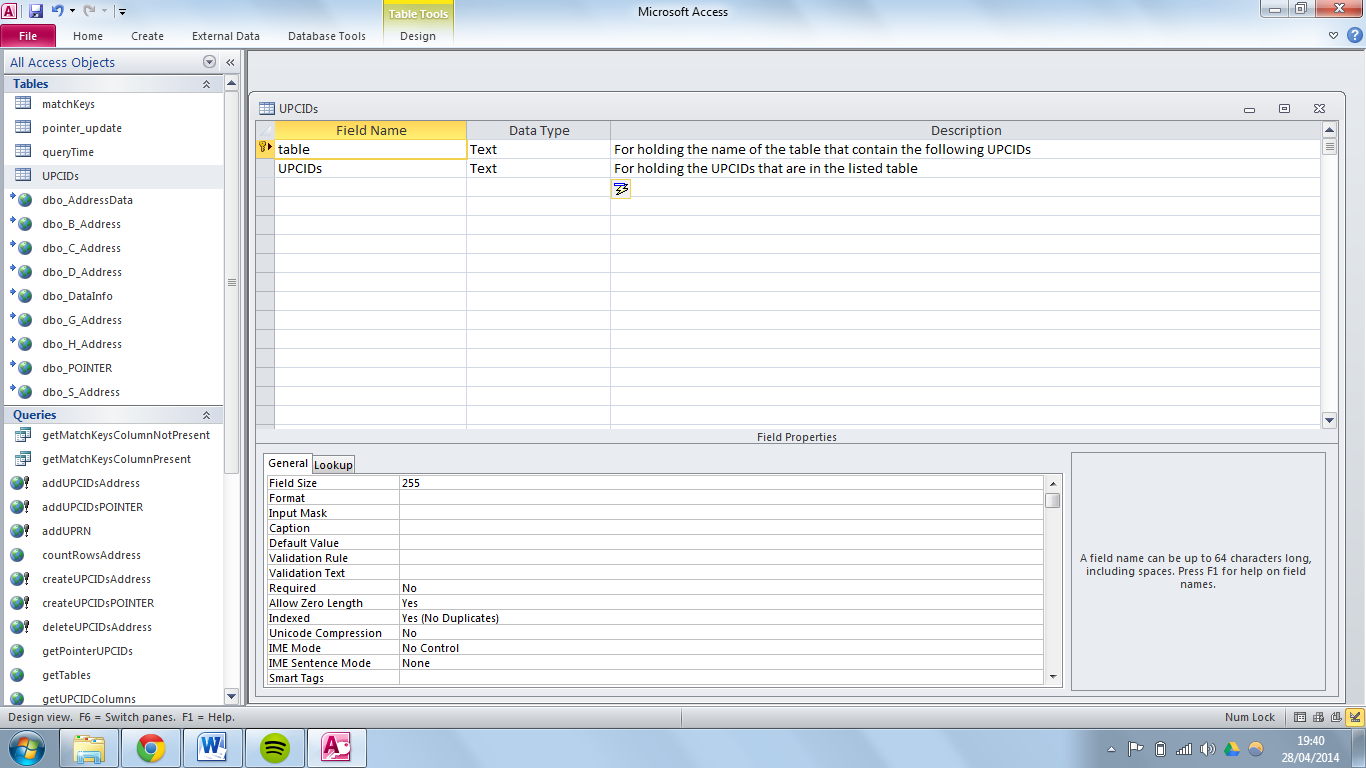


Figure 41 showing the architecture of the UPCIDs table

Figure 40 showing the architecture of the queryTime table

As the application involves the use of tables within itself as well as external tables, figure 42 was made to help understand the communication between the application and tables, queries and functions.

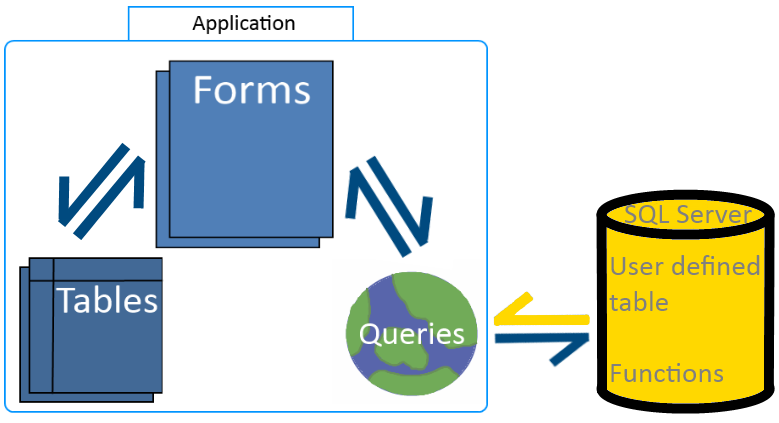


Figure 42 showing the communication between the application and its components.

# 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 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.

Within the SQL statements found in appendix 5 that populate the UPCID fields 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:

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

Figure 43 Extract Charatcers (dbo.fn\_extract\_chars)

This function is called using the keyword dbo.fn\_extract\_chars (x, y)(figure 43) 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.

Figure 44 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()(figure 44) 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.

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

Figure 45 Unwanted (dbo.unwanted)

This function again uses the keyword dbo.unwanted (figure 45) 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 code for the matching algorithm can be seen below; it was based off the pseudo code mentioned earlier in the report. The code in figure 46 is only for matching one UPCID but this code will be the same for all UPCID matches.

select a.UPCID\_N\_WP as addUPCID\_N\_WP, p.UPCID\_N\_WP as PUPCID\_N\_WP, p.UPRN

into tempPointer

from A\_address a

inner join POINTER p

on a.UPCID\_N\_WP = p.UPCID\_N\_WP

select addUPCID\_N\_WP, COUNT(\*) as counter

into tempAddress

from tempPointer

group by addUPCID\_N\_WP

having COUNT(\*) >= 1

Update main

Set main.uprn = tp.uprn

from A\_address main

inner join tempPointer tp

on main.UPCID\_N\_WP = tp.PUPCID\_N\_WP

inner join tempAddress ta

on main.UPCID\_N\_WP = ta.addUPCID\_N\_WP

where main.uprn Is Null and len(UPCID\_N\_WP) > 1

drop table tempPointer

drop table tempAddress

Figure 46 Matching algorithm

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. Therefore 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 and the second contains all of the UPCIDs that are selected by the user from the first list box. The code for the add (figure 47) and remove (figure 48) buttons are as follows:

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

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

Figure 48 Remove from listbox

Figure 47 Add to listbox

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.

The above query (figure 49) 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.

Figure 49 Populate right list box

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

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 (figure 50).

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

Figure 50 Change query statement

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 (figure 51) and this is done by inserting the values of the right hand list box into an array.

Figure 51 Declaring 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 (figure 52).

Figure 52 Declaring and populating variable for SQL

A for loop is then used (figure 53) 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, ''), ' ', ''))"

Dim forSQL As String

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

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, ''), ' ', ''))"

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(figure 53).

Figure 53 Case Statement inside for loop for listbox UPCIDs

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

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.

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. They had no previous knowledge of the software or process whereas NISRA did not mention this as they are familiar with the process.

The results for all of the tests can be found in appendix 6.

With the help of testing the applications functionality is working as intended and the results showed some changes or minor adjustments that would like to be implemented in future development and these can be read about later in the report. The testing process was very beneficial to the developer to help them evaluate the success of their application.

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 project 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 (figure 54) 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 |  |
| If a user does not have a username and password for MS SQL Server they should not be able to use the application. | 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

Figure 54 Showing how the project met the requirements

The problem that is being investigated during this project is stated because “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 aim of the project was to develop an application that can carry out the same process as the software that NISRA currently use. The current software handles addresses held within a dataset and tests the validity of these by running them together with defined matching algorithms against an already cleansed address dataset. The development of an application that can do the above will save NISRA substantial expenses in both the software and training as NISRA currently have access to the pre-cleansed dataset. The developed software must achieve at least the same, if not more accurate percentage of valid addresses as well as having an easy to use interface so that all the current members of NISRA staff, as well as future staff, can use the software effectively. The software needed to be able to run on multiple computers so that if it was needed by more than one member of staff it can be done. The application must be able to be updated as and when a new updated release of the pre-cleansed dataset becomes available.

The project succeed in being able to carry out the process of the now previously used application named Experian QAS Batch with a slight increase in address validation rates. The matching process created by the developer proved to be a successful and unique algorithm that is now sought after across the Civil Service due to its high and effect match rates. With the use of this application within NISRA it is confirmed that, when fully implemented, will save NISRA over £2000 per year on software licensing alone which can be invested into other projects that require it.

The previous software was said by members of NISRA staff to be complicated and difficult to use due to the vast amount of features that were not needed by NISRA. This developed application process of matching is now said to be very easy to follow and has all the features that NISRA require with the ability to insert more features when needed. The ability of the application to be used on several computers at the one time now allows users in different sections of NISRA to carry out testing the validity of address on their own datasets without having to wait a single user to finish, this will save NISRA staff time in which can be spent on other projects.

It has been said that the user interface of the developed application helps the staff of NISRA to use the application in that the interface is laid out very intuitively. The user interface is similar to that of an installation process which every user of a computer is familiar with. As the application is organised in steps, the users found it easy to navigate around the application without being overwhelmed by many features and were able to carry out the process quite efficiently.

As this project was one of the largest projects the developer has been involved in, a lot has been learned especially in the aspect of planning. Agile management was the development model used in this project and this helped the timing of the project development so that the application was able to be delivered in the time frame that was given. Without the use of this model the application would not have been finished on time, but by using agile management a prototype was developed early on in the process and this was able to be shown to NISRA to gather feedback. Then this feedback was used to help build the application that NISRA wanted. It was seen that the requirements that were agreed during the analysis of the problem developed over time and this was also due to the development of the prototype. Not all of the requirements were met at the end of the project which was due to poor time management as not enough time was allowed for the implementation of all the features. The features that were not able to be implemented were not critical features to the project but would have been useful in the finished application.

If the developer was to develop this application again, more time would be spent learning a more advanced language that could be used to create a more personal user front end that would allow the application to be more portable to the users. This would also allow the incorporation of a portable data warehouse within the application which would enable the application to be run without installation so that it could be used on any computer, internally or externally.

Further Development

Version control was used during the development of the application and this will allow further development of the application to add features that were not able to be implemented due to the lack of time. The features that will be added in future development are as follows:

* The ability to enable more detailed outputs of the finished process were the user can see which match keys were able to perform the best in which datasets.
* The ability to take the data from the pre-cleansed dataset and update the ‘dirty’ data in the user defined dataset.
* The ability to add more UPCID MatchKeys into the application by users.
* Making the application more portable so that it can be taken off site when attending other branches.
* Automatically downloading and updating the latest Pointer dataset into the database.

The major achievement of the project was the development of the matching algorithm. It was the most time consuming part of the project due to the extensive research and testing of the matching rates. Since the algorithm was so successful there are a number of Civil Service departments that are interested in the use of the application as this process is carried out in many of the departments that have the use of address data.

The project was a new and challenging experience to the developer as it provided an opportunity to develop a solution to a problem by creating an application from that they are in charge of. The ability to create an application for an organisation in which it will be used is a great encouragement to the developer’s confidence and allowed them to expand their knowledge of project management and professional skills by organising their time efficiently and effectively in order to complete the project on time. Having experience in both MS Access and MS SQL Server was beneficial during the development of the project as it allowed the implementation of the majority of the features to run smoothly and when errors arose the developer knew how to deal with them effectively.

The overall aim of the project was met as the application is easy to use by users of all abilities as well as the application itself is effective in its process. NISRA are very happy with the finished application and it is extremely satisfying to know that the application will be used in the future and will be helpful to them and future statistics.

# References

AddressDoctor, 2014. *Address format for GBR+United Kingdom.* [Online]   
Available at: http://www.addressdoctor.com/en/countries-data/address-formats.html  
[Accessed 16 04 2014].

Allies, 2014. *UK address cleansing and correction software | PostCoder Batch | Allies.* [Online]   
Available at: http://www.alliescomputing.com/products/address-cleaning  
[Accessed 26 04 2014].

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

Data8, 2014. *Correct addresses using Data8's PAF Cleansing.* [Online]   
Available at: https://www.data-8.co.uk/en/services/data-cleansing/online-data-cleansing-services/paf-cleansing  
[Accessed 25 04 2014].

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].

helpIT Systems, 2014. *Address Validation.* [Online]   
Available at: http://www.helpit.com/us/solutions/address-validation  
[Accessed 25 04 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].

PostcodeAnywhere, 2013. *Address cleansing - data cleansing services for address data cleansing and suppression.* [Online]   
Available at: http://www.postcodeanywhere.co.uk/data-cleansing/  
[Accessed 26 04 2014].

Royal Mail Group Ltd., 2014. *Clear addressing | Royal Mail Group Ltd..* [Online]   
Available at: http://www.royalmail.com/personal/help-and-support/How-do-I-address-my-mail-correctly  
[Accessed 23 04 2014].

Wahi, M. M., Parks, D. V., Skeate, R. C. & Goldin, S. B., 2008. Reducing Errors from the Electronic Transcription of Data Collected on Paper Forms: A Research Data Case Study. *Journal of the American Medical Informatics Association,* 15(3), p. 386–389.

# 

# Appendices

## 1. Match Rates

|  |  |  |  |
| --- | --- | --- | --- |
| **FROM BSO - April 2012 Download** | | | **%**  Figure 55 Total Records in real dataset |
|  |  |  |  |
| **Total Records** |  | 680471 |  |
|  |  |  |  |
| **UPCID** | | | |
| Total UPRNS | | No UPRNS |  |
| *641517* | | *38954* | |
| **94.28%** | | **5.72%** | |
|  |  |  |  |
| **QAS** | | | |
| Total UPRNS | | No UPRNS |  |
| *637184* | | *43287* | |
| **93.64%** | | **6.36%** | |

Figure 57 Records results for Existing Solution (QAS)

Figure 56 Records results for Developed Solution

Figure 55 shows the number of records that were contained in a real NISRA dataset that the developed solution was tested on.

Figure 56 shows the results of matches that the developed solution achieved and figure 57 shows the results of matches that the existing solution achieved. Note that the developed solution not only matched the number of matched records but exceeded that by around 3000 records.

## 

## 2. Letter of commendation from NISRA





## 

## 3. QAS Email

|  |  |
| --- | --- |
| From: | **May, H** (\*\*\*\*\*\*\*@experian.com) |
| Sent: | 22 August 2013 11:11:50 |
| To: | J\*\*\*\*\*\*@hotmail.co.uk |
|  | |  |  | | --- | --- | | Attachments: | 2 attachments | |  | BatchWithSuppression-4pp-Ext-Mar13.pdf (350.6 KB) , DataSetGuide-45pp-Ext-Mar13.pdf (351.6 KB) | |

Hi J\*\*\*\*,

Thank you for taking the time to speak with me today.

I have attached some further information regarding the Batch software and our pointer data (this can be found on page 6 of our dataset guide) along with confirmation of the pricing we discussed.

**Pricing**

|  |  |
| --- | --- |
| **Solution** | **Price** |
| Batch without Suppression inc Pointer Data | £2,000 |
| Batch with Suppression inc Pointer Data | £3,300 |

The pricing includes -

  12month license for 1 user

  1 days on site training & integration assistance (2days for Batch with Suppression)

  Quarterly electronic data updates

  On-going technical support

  UPRN & Longitude and Latitude co-ordinates

The suppressions are calculated on a per suppression basis and would be in addition to the above pricing, I have outlined the costs for these below -

|  |  |  |
| --- | --- | --- |
| **TRACKING** | **Absolue Contacts Tracking** | **£0.29** |
| **NCOA Tracking** | **£0.34** |
| **GONEAWAYS** | **Absolute Movers Permanent** | **£0.29** |
| Absolute Movers One-Off | **£0.19** |
| **USS Permanent** | **£0.29** |
| USS One-Off | **£0.19** |
| **GAS Permanent** | **£0.60** |
| GAS One-Off | **£0.20** |
| **DECEASED** | **Mortalities Permanent** | **£0.45** |
| Mortalities One-Off | **£0.24** |
| **Experian Mortality Permanent** | **£0.28** |
| Experian Mortality One-Off | **£0.24** |
| **TBR Permanent** | **£0.60** |
| TBR One-Off | **£0.20** |

**How will QAS Batch help your business?**

  Reduces expenditure by up to 19% by updating records that are incorrect reducing the amount of returned mail.

  Increase marketing conversion by not communicating with people that cannot or will not respond.

We are able to arrange a free webex of the software which will enable you to see the product working in a live enviroment and show you the additional funcationality available with the solution.

I will give you a call on Tuesday to confirm you have received my email and to answer any questions you may have, but if you have any in the meantime please let me know on 0207 \*\*\*\*\*\*\*.

**Kind Regards**

**H May   
Internal Sales Team Lead**

Experian QAS  |  George West House  |  2-3 Clapham Common North Side  |  London SW4 0QL  
T: +44 (0)207 \*\*\*\*\*\*\*\*\*  /  F: +44 (0)207 \*\*\*\*\*\*\*\*\*  
[www.qas.co.uk](http://www.qas.co.uk/)

## 

## 4. Questionnaire

## 5. Project Code

The code below (figure 58) is found behind the next button on the select options form. This code detects what stage of the process the user is at and determines which part of the code to run. Some of the code requires functions that can be found earlier in the report.

Private Sub Command6\_Click()

Dim strSQL As String

Dim UPCIDsSplit() As String

Dim ListBoxUPCIDs() As String

Dim UnwantedListBoxUPCIDs() As String

Dim hasUPRN As String

Dim queryLength As Single

Dim queryLengthMin As Single

Dim length As Single

Dim listCount As Integer

On Error GoTo ErrorHandler:

UPCIDsSplit() = Split(UPCIDsExisting, ", ")

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

UnwantedListBoxUPCIDs() = Split(Replace(List0.RowSource, ";", ", "), ", ")

strSQL = "SELECT COLUMN\_NAME FROM [INFORMATION\_SCHEMA].COLUMNS WHERE COLUMN\_NAME = 'uprn' AND TABLE\_NAME = '" & strTable & "'"

ChangePTStatement "getUPRNColumn", strSQL

strSQL = ""

hasUPRN = Nz(DLookup("COLUMN\_NAME", "getUPRNColumn"), "none")

Dim i As Long

Dim j As Long

Dim wanted As String

Dim unwanted As String

unwanted = ""

listCount = Me.List2.listCount

If Check15.Value = True Then

queryLength = Nz(DLookup("queryLength", "queryTime", "queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'")) + Nz(DLookup("queryLength", "queryTime", "queryType = 'pointer'"))

Else

queryLength = Nz(DLookup("queryLength", "queryTime", "queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'"))

End If

If queryLength > 60 Then

queryLengthMin = Fix(queryLength / 60)

queryLength = Round(queryLength - (queryLengthMin \* 60), 0)

Else

queryLengthMin = 0

End If

If queryLength = 0 Then

DoCmd.OpenForm "frmCustomMSG", , , , , , "This query has not been run before and may take some time, Please wait..."

Else

DoCmd.OpenForm "frmCustomMSG", , , , , , "Estimated time: " & queryLengthMin & " minutes and " & queryLength & " seconds, Please wait..."

End If

Pause (1)

If Text7.Value = "create" Then

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

For j = LBound(UPCIDsSplit) To UBound(UPCIDsSplit)

If ListBoxUPCIDs(i) = UPCIDsSplit(j) Then

ListBoxUPCIDs(i) = "@"

End If

Next j

Next i

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

For j = LBound(UPCIDsSplit) To UBound(UPCIDsSplit)

If UnwantedListBoxUPCIDs(i) = UPCIDsSplit(j) Then

unwanted = unwanted & UnwantedListBoxUPCIDs(i) & ", "

End If

Next j

Next i

wanted = Join(ListBoxUPCIDs, ", ")

wanted = Replace(wanted, "@, ", "")

wanted = Replace(wanted, "@", "")

starttime = Timer

If unwanted <> "" Then

unwanted = Left(unwanted, Len(unwanted) - 2)

strSQL = "ALTER TABLE "

strSQL = strSQL & strTable

strSQL = strSQL & " DROP COLUMN " & unwanted

ChangePTStatement "deleteUPCIDsAddress", strSQL

DoCmd.SetWarnings (False)

DoCmd.OpenQuery ("deleteUPCIDsAddress")

MsgBox "UPCID(s) Deleted.", vbOKOnly, "Complete!"

DoCmd.SetWarnings (True)

End If

List2.RowSource = ""

Dim sqlUPCIDs As String

sqlUPCIDs = Replace(wanted, ", ", " varchar(128), ")

If wanted <> "" Then

sqlUPCIDs = sqlUPCIDs & " varchar(128)"

End If

If wanted <> "" Then

If hasUPRN = "none" Then

strSQL = "ALTER TABLE "

strSQL = strSQL & strTable

strSQL = strSQL & " ADD " & sqlUPCIDs & ", UPRN varchar(128)"

ChangePTStatement "createUPCIDsAddress", strSQL

DoCmd.SetWarnings (False)

DoCmd.OpenQuery ("createUPCIDsAddress")

MsgBox "UPCID(s) Created and UPRN column also added.", vbOKOnly, "Complete!"

DoCmd.SetWarnings (True)

Else

strSQL = "ALTER TABLE "

strSQL = strSQL & strTable

strSQL = strSQL & " ADD " & sqlUPCIDs

ChangePTStatement "createUPCIDsAddress", strSQL

DoCmd.SetWarnings (False)

DoCmd.OpenQuery ("createUPCIDsAddress")

MsgBox "UPCID(s) Created", vbOKOnly, "Complete!"

DoCmd.SetWarnings (True)

End If

End If

length = Fix(Format(Timer - starttime, "fixed")) + 1

DoCmd.SetWarnings (False)

If queryLength = 0 Then

strSQL = "INSERT INTO queryTime VALUES ('" & Text7.Value & listCount & "', '" & strTable & "', " & length & ")"

DoCmd.RunSQL strSQL

Else

strSQL = "UPDATE queryTime SET [queryLength] = " & length & " WHERE queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'"

DoCmd.RunSQL strSQL

End If

DoCmd.SetWarnings (True)

DoCmd.Close acForm, "frmCustomMSG", acSaveNo

Dim strSQLDefault As String

strSQL = "SELECT COLUMN\_NAME"

strSQL = strSQL & " FROM [INFORMATION\_SCHEMA].COLUMNS"

strSQL = strSQL & " WHERE COLUMN\_NAME LIKE 'upcid%' AND TABLE\_NAME = " & "'" & strTable & "'"

'call the pass through function

ChangePTStatement "getUPCIDColumns", strSQL

DoCmd.Close

DoCmd.OpenForm "selectUPCIDs", , , , , , strTable

statusmsg = SysCmd(acSysCmdSetStatus, "Please select from the UPCIDs that you wish to populate. If Pointer needs updated then don't forget to check the relevant box. Click the 'i' for more information.")

Forms!selectUPCIDs!Text7.Value = "add"

Forms!selectUPCIDs!Text11.Value = "Select UPCID Columns to populate in the table " & strTable

Forms!selectUPCIDs!List0.RowSource = Forms!selectUPCIDs!List2.RowSource

Forms!selectUPCIDs!List2.RowSource = ""

Forms!selectUPCIDs!step2.Visible = True

If DLookup("date", "pointer\_update") < (Date - 30) Then

Forms!selectUPCIDs!Check15.Visible = True

Forms!selectUPCIDs!Label16.Visible = True

Forms!selectUPCIDs!Check15.Value = False

Else

Forms!selectUPCIDs!Check15.Visible = True

Forms!selectUPCIDs!Label16.Visible = True

Forms!selectUPCIDs!Check15.Value = False

Forms!selectUPCIDs!Check15.Enabled = False

End If

ElseIf Text7.Value = "add" Then

Dim caseStates() As String

Dim forSQL As String

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

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

Dim sqlDefault As String

Dim sqlstr As String

sqlDefault = "Update " & strTable & " set"

sqlstr = ""

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, '')) + 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

strSQL = sqlstr

'call the pass through function

ChangePTStatement "addUPCIDsAddress", strSQL

If IsNull(DLookup("table", "UPCIDs", "table = '" & strTable & "'")) = True Then

strSQL = "INSERT INTO UPCIDs ([table], UPCIDs) VALUES ('" & strTable & "', '" & forSQL & "')"

DoCmd.SetWarnings (False)

DoCmd.RunSQL strSQL

DoCmd.SetWarnings (True)

Else

DoCmd.SetWarnings (False)

strSQL = "DELETE FROM UPCIDs WHERE [table] = '" & strTable & "'"

DoCmd.RunSQL strSQL

strSQL = "INSERT INTO UPCIDs ([table], UPCIDs) VALUES ('" & strTable & "', '" & forSQL & "')"

DoCmd.RunSQL strSQL

DoCmd.SetWarnings (True)

End If

Dim pointerLength As Single

If Me.Check15.Value = True Then

If DLookup("counter", "getPointerUPCIDs") < 0 Then

DoCmd.SetWarnings (False)

starttime = Timer

DoCmd.OpenQuery ("createUPCIDsPOINTER")

DoCmd.OpenQuery ("addUPCIDsPOINTER")

pointerLength = Fix(Format(Timer - starttime, "fixed")) + 1

strSQL = "UPDATE pointer\_update SET [date] = '" & Date & "' WHERE ID = 1"

DoCmd.RunSQL strSQL

DoCmd.SetWarnings (True)

Else

If MsgBox("Are you sure you want to update Pointer UPCIDs?", vbYesNo, "Continue?") = vbYes Then

DoCmd.SetWarnings (False)

starttime = Timer

DoCmd.OpenQuery ("addUPCIDsPOINTER")

pointerLength = Fix(Format(Timer - starttime, "fixed")) + 1

strSQL = "UPDATE pointer\_update SET [date] = '" & Date & "' WHERE ID = 1"

DoCmd.RunSQL strSQL

DoCmd.SetWarnings (True)

End If

End If

End If

starttime = Timer

DoCmd.SetWarnings (False)

DoCmd.OpenQuery ("addUPCIDsAddress")

MsgBox "UPCIDs Added", vbOKOnly, "Complete"

DoCmd.SetWarnings (True)

length = Fix(Format(Timer - starttime, "fixed")) + 1

DoCmd.SetWarnings (False)

If Check15.Value = True Then

If queryLength = 0 Then

strSQL = "INSERT INTO queryTime VALUES ('" & Text7.Value & listCount & "', '" & strTable & "', " & length & ")"

DoCmd.RunSQL strSQL

strSQL = "INSERT INTO queryTime VALUES ('pointer', 'pointer', " & pointerLength & ")"

DoCmd.RunSQL strSQL

Else

strSQL = "UPDATE queryTime SET [queryLength] = " & length & " WHERE queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'"

DoCmd.RunSQL strSQL

strSQL = "UPDATE queryTime SET [queryLength] = " & pointerLength & " WHERE queryType = 'pointer' and queryTable = 'pointer'"

DoCmd.RunSQL strSQL

End If

Else

If queryLength = 0 Then

strSQL = "INSERT INTO queryTime VALUES ('" & Text7.Value & listCount & "', '" & strTable & "', " & length & ")"

DoCmd.RunSQL strSQL

Else

strSQL = "UPDATE queryTime SET [queryLength] = " & length & " WHERE queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'"

DoCmd.RunSQL strSQL

End If

End If

DoCmd.SetWarnings (True)

DoCmd.Close acForm, "frmCustomMSG", acSaveNo

strSQL = "SELECT COLUMN\_NAME"

strSQL = strSQL & " FROM [INFORMATION\_SCHEMA].COLUMNS"

strSQL = strSQL & " WHERE COLUMN\_NAME LIKE 'upcid%' AND TABLE\_NAME = " & "'" & strTable & "'"

'call the pass through function

ChangePTStatement "getUPCIDColumns", strSQL

DoCmd.Close

DoCmd.OpenForm "selectUPCIDs", , , , , , strTable

Forms!selectUPCIDs!Text7.Value = "match"

statusmsg = SysCmd(acSysCmdSetStatus, "Please order the UPCIDs in the order you wish them to be matched. Click the 'i' for more information.")

Forms!selectUPCIDs!Text11.Value = "Select UPCID Columns from table " & strTable & " you wish to match on"

If IsNull(DLookup("[table]", "UPCIDs", "[table] = '" & strTable & "'")) = False Then

Forms!selectUPCIDs!List0.RowSource = DLookup("UPCIDs", "UPCIDs", "[Table] = '" & strTable & "'")

Else

Forms!selectUPCIDs!List0.RowSource = ""

End If

Forms!selectUPCIDs!List2.RowSource = ""

Forms!selectUPCIDs!Check15.Visible = False

Forms!selectUPCIDs!Label16.Visible = False

Forms!selectUPCIDs!step3.Visible = True

ElseIf Text7.Value = "match" Then

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

sqlstr = ""

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

sqlstr = sqlstr & "select a." & caseStates(i) & " as add" & caseStates(i) & ", p." & caseStates(i) & " as P" & caseStates(i) & ", p.UPRN" & \_

" into tempPointer" & \_

" from " & strTable & " a" & \_

" inner join POINTER p" & \_

" on a." & caseStates(i) & " = p." & caseStates(i) & \_

vbNewLine & \_

"select add" & caseStates(i) & ", COUNT(\*) as counter" & \_

" into tempAddress" & \_

" from tempPointer" & \_

" group by add" & caseStates(i) & \_

" having COUNT(\*) >= 1" & \_

vbNewLine & \_

"Update main" & \_

" Set main.uprn = tp.uprn" & \_

" from " & strTable & " main" & \_

" inner join tempPointer tp" & \_

" on main." & caseStates(i) & " = tp.P" & caseStates(i) & \_

" inner join tempAddress ta" & \_

" on main." & caseStates(i) & " = ta.add" & caseStates(i) & \_

" where main.uprn Is Null and len(" & caseStates(i) & ") > 1" & \_

vbNewLine & \_

"drop table tempPointer" & \_

vbNewLine & \_

"drop table tempAddress" & \_

vbNewLine

Next i

strSQL = sqlstr

'call the pass through function

ChangePTStatement "addUPRN", strSQL

strSQL = "SELECT COUNT(\*) AS col FROM " & strTable & " WHERE uprn IS NULL"

ChangePTStatement "countUPRN", strSQL

starttime = Timer

statusmsg = SysCmd(acSysCmdSetStatus, "Matching process has begun, Please Wait...")

Dim uprnCount As Integer

DoCmd.SetWarnings (False)

uprnCount = DLookup("col", "countUPRN")

DoCmd.OpenQuery ("addUPRN")

MsgBox "UPRNs Added", vbOKOnly, "Complete"

DoCmd.SetWarnings (True)

length = Fix(Format(Timer - starttime, "fixed")) + 1

DoCmd.SetWarnings (False)

If queryLength = 0 Then

strSQL = "INSERT INTO queryTime VALUES ('" & Text7.Value & listCount & "', '" & strTable & "', " & length & ")"

DoCmd.RunSQL strSQL

Else

strSQL = "UPDATE queryTime SET [queryLength] = " & length & " WHERE queryType = '" & Text7.Value & listCount & "' and queryTable = '" & strTable & "'"

DoCmd.RunSQL strSQL

End If

DoCmd.SetWarnings (True)

DoCmd.Close acForm, "frmCustomMSG", acSaveNo

DoCmd.Close

DoCmd.OpenForm "whatNext", , , , , , strTable & "|" & uprnCount

statusmsg = SysCmd(acSysCmdSetStatus, "Select the option to carry out next.")

End If

Exit Sub

ErrorHandler:

DoCmd.Close acForm, "frmCustomMSG", acSaveNo

MsgBox "An error occured - error " & Err.Number & ": " & Err.Description

End Sub

## 5. Test Suite

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Function to be tested | Details/Results expected | Actual result | Pass/Fail |
| 1 | Open application | Application opens with splashscreen and acceptance form appears | Application opened and splashscreen was displayed then acceptance form was displayed | Pass |
| 2 | By pass acceptance form | If close form then cannot continue with application | Application closed when trying to exit acceptance form | Pass |
| 3 | Tickbox to enable and disable continue button | If tickbox is not ticked you cannot continue | Tickbox was ticked and unticked and continue button was disabled | Pass |
| 4 | Accept acceptance page | When tickbox is ticked and continue is clicked the application should open to the select table screen | Application opened to select table screen | Pass |
| 5 | Testing SQL log in | If the user enters a correct user name and password the application should continue to open, if an incorrect username and password is entered the application should not allow progression | When a correct username and password is entered the user can continue with the application, if an incorrect username and password is entered the application does not allow progression. | Pass |
| 6 | Test drop down box population | If the user clicks on the drop down box it should be populated with tables within the database | When the drop down box was clicked tables that were in the database appeared. | Pass |
| 7 | Testing record count when table is selected | When the user selects a table the number of records in the table should appear | When a table is selected the number of records appeared | Pass |
| 8 | Testing the please wait function works correctly | When the user selects a table the please wait form should appear with the length of time the last query took or if the query has not been ran before it should state this. | When a new table was selected the please wait form appeared saying this query had not been ran before, when the same table was selected the length of time it last ran was displayed | Pass |
| 9 | Testing the ‘I’ button | When the ‘i’ button is clicked a form displaying more information should be displayed | When the ‘I’ button was clicked a form displaying more information was displayed | Pass |
| 10 | Testing the next button | When the next button is clicked the select options form should open displaying any UPCIDs that are currently in the selected table | When the next button was clicked the select options form was displayed showing all upcids that were currently in the selected table | Pass |
| 11 | Testing the add and remove buttons | When the add button is pressed the selected UPCID should be added from the left list box to the right list box. When the remove button is pressed the UPCID should be removed from the right list box to the left list box | When the relevant button was pressed the UPCID moved as expected. | Pass |
| 12 | Testing the controlTool tip on both list boxes | When a UPCID is selected and hovered on, text should appear alerting the user as to what components the UPCID is made up of | When different UPCIDs were selected the relevant information was displayed. | Pass |
| 13 | Testing adding UPCID columns to selected tables | When any number of UPCIDs are added to the list box on the right and the next button is clicked, the relevant UPCID columns should be added. | When different numbers of UPCIDs were added to different selected tables the relevant UPCID columns were added to the relevant tables | Pass |
| 14 | Testing the removing of UPCID columns from selected tables | When a UPCID that is already in a selected table is removed from the right list box and the next button is clicked, that UPCID column should be removed from the table | When a UPCID that was already in the selected table was removed from the list box on the right, that UPCID column was removed from the table | Pass |
| 15 | Testing the addition of UPRN columns | If a table has not been selected before and UPCIDs are added, a UPRN column should also be added, but if a table has been selected before, no UPRN column should be added. | When a new table was selected and UPCIDs were added, a UPRN column was added also, when this table was used again, no UPRN column was added. | Pass |
| 16 | Test blank list box on right hand side | If not UPCIDs are added into the right list box a warning should appear. | When no UPCIDs were added into the right list box a warning appeared. | Pass |
| 17 | Testing the different stages | When the next button is pressed only UPCID columns that are present from step one should be available in step two and only the columns populated in step 2 should be available in step three | When moving through the process the relevant UPCIDs appeared in the right hand list box | Pass |
| 18 | Testing pointer update | When the pointer tick box is checked and the next button is clicked the pointer dataset UPCIDs should be either added or updated, then the check box should not be able to be clicked for another 30 days | When the pointer tick box was checked and the next button was clicked the pointer table received UPCIDs. The check box was not able to be ticked until time was falsely advanced 30 days and then the pointer table was able to be updated | Pass |
| 19 | Testing step 2 | On the second step when UPCIDs are added to the right hand list box and the next button is clicked the relevant UPCIDs should be populated | When UPCIDs were selected and the next button was clicked the relevant UPCIDs were populated | Pass |
| 20 | Testing step 3 | On the third step when UPCIDs are added to the right hand list box and the next button is clicked the match process should begin and UPRNs should be added to successfully matched records | When UPCIDs were selected and the next button was clicked UPRNs were added to successfully matched records. | Pass |
| 21 | Testing the whats next form | When the match process is finished the whats next form should open | When the matching process finished the whats next form opened | Pass |
| 22 | Testing the match summary button | When the match summary button is clicked a form displaying details of the match process results should be shown | When the match summary button was clicked a form displaying details of the match process was shown | Pass |
| 23 | Testing the close button on the match summary form | When the close button is click the form should close and return to the whats next form | When the close button was pressed the form closed and the whats next form displayed | Pass |
| 24 | Testing the interactive matching form | When the interactive matching button is pressed a form displaying all the matched records should appear | When the interactive matching button was pressed a form appeared displaying all the matched records | Pass |
| 25 | Testing the start over button | When the start over button is clicked the whats next for should close and the select table form should be displayed | When the start over button was clicked the whats next form closed and the select table form was displayed. | Pass |
| 26 | Testing instructions on status bar | Status bar instructions should change throughout the process to relate to the current stage | When carrying out the process the status bar instructions changed to suit the stage of the process the user is on. |  |